DOCTORAL THESIS

Explainable Artificial Intelligence-Based Intrusion Detection Systems

Rajesh Kalakoti

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2025

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
82/2025

Explainable Artificial Intelligence-Based Intrusion Detection Systems

RAJESH KALAKOTI

TAL TECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

The dissertation was accepted for the defence of the degree of Doctor of Philosophy in Information and Communication Technology on 20th October 2025

Supervisor: Tenured Associate Professor Dr. Sven Nõmm,
Department of Software Science, School of Information Technology,
Tallinn University of Technology,
Tallinn, Estonia

Co-supervisor: Research Professor Dr. Hayretdin Bahşi
Department of Software Science, School of Information Technology,
Tallinn University of Technology,
Tallinn, Estonia

Opponents: Professor Dr. Abdulhamit Subasi,
Department of Information Sciences and Technology,
University at Albany,
New York, United States

Professor Dr. Jianhua Zhang,
Department of Computer Science,
Oslo Metropolitan University,
Oslo, Norway

Defence of the thesis: 31st October 2025, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.

Rajesh Kalakoti _____
signature

Copyright: Rajesh Kalakoti, 2025
ISSN 2585-6898 (publication)
ISBN 978-9916-80-401-8 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9916-80-402-5 (PDF)
DOI https://doi.org/10.23658/taltech.82/2025

Kalakoti, R. (2025). *Explainable Artificial Intelligence-Based Intrusion Detection Systems* [TalTech Press]. https://doi.org/10.23658/taltech.82/2025

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
82/2025

Selgitaval tehisintellektil baseeruvad ründetuvastussüsteemid

RAJESH KALAKOTI

TAL TECH
KIRJASTUS

Contents

List of Publications viii

Author's Contributions to the Publications ix

Abbreviations xi

1 Introduction 1
- 1.1 Explainable AI for IDS 2
  - 1.1.1 Research Gap in Centralized XAI-IDS 5
- 1.2 Explainable AI for Decentralized IDS 5
- 1.3 Research questions 6
- 1.4 Research Gaps & Contributions 7
- 1.5 Thesis Structure 11

2 Background Work 13
- 2.1 Explainable AI 13
- 2.2 Explainable AI methods 14
  - 2.2.1 LIME(Local Interpretable Model-Agnostic Explanations) 14
  - 2.2.2 SHAP (SHapley Additive exPlanations) 14
  - 2.2.3 DeepLift (Deep Learning Important FeaTures) 15
  - 2.2.4 Integrated Gradients (IG) 16
  - 2.2.5 Gradient $\times$ Input 16

3 Federated Learning 17

4 Effective IoT Botnet and IoMT Attack Detection 19
- 4.1 In-depth Feature Selection for efficient IoT Botnet Detection 19
- 4.2 Comprehensive Feature Selection for IoMT Intrusion Detection 26
- 4.3 Chapter Discussions 30

5 Explainable AI for Transparent IoT Botnet Detection 31
- 5.1 Quantitative Evaluation of explainability for IoT Bot Net Detection 31
  - 5.1.1 Feature Selection 32
  - 5.1.2 Post-hoc Explainability for IoT botnet detection Models 32
  - 5.1.3 Faithfulness 35
  - 5.1.4 Monotonicity 36
  - 5.1.5 Low Complexity 36
  - 5.1.6 Maximum Sensitivity 36
- 5.2 Evaluation of XAI Methods for Deep Learning-Based Botnet Attack Detection 39
- 5.3 Chapter Discussions 41

6 XAI for Transparent Alert Classification in NIDS 42
- 6.1 Relevance Rank Accuracy 45
- 6.2 Relevance Mass Accuracy 45
- 6.3 Chapter Discussions 47

7 Explainable Transformer-based Intrusion Detection in IoMT Networks 49
- 7.1 Chapter Discussions 56

8 Enhancing IoT Botnet Detection Through Explainable Active Learning (XAL) Paradigm 57
8.1 Background on Active Learning ... 57
8.2 XAL paradigm for IoT botnet detection ... 59
8.3 Chapter Discussions ... 65

9 Privacy-Preserving Explainability in Federated Learning for IoT Threat Detection . 67
9.1 Explainable Federated model for IoT botnet detection ... 67
9.2 Balancing Privacy and Explainability in Federated Learning for Botnet Detection in IoT Networks ... 72
9.2.1 Explaining Server-Side Black Box model in Federated Learning Settings ... 76
9.2.2 Explaining Client-Side Black Box models in Federated Learning Settings ... 82
9.3 Chapter Discussions ... 87

10 Synthetic Data-Driven Explainability for Federated Learning Based IDS ... 88
10.1 Federated Deep Neural Network for Intrusion Detection Systems ... 90
10.2 Federated Synthetic data Using GAN variants ... 92
10.3 Explaining DNN model in FL using Synthetic Data ... 101
10.4 Chapter Discussions ... 106

11 Conclusions and Future Works ... 108
11.1 Future Work ... 111

List of Figures ... 114

List of Tables ... 115

References ... 116

Acknowledgements ... 127

Abstract ... 128

Kokkuvõte ... 130

Appendix 1 ... 133

Appendix 2 ... 153

Appendix 3 ... 167

Appendix 4 ... 187

Appendix 5 ... 197

Appendix 6 ... 211

Appendix 7 ... 219

Appendix 8 ... 229

Appendix 9 239

Appendix 10 265

Appendix 11 295

Curriculum Vitae 307

Elulookirjeldus 309

List of Publications

The present Ph.D. thesis is based on the following publications that are referred to in the text by Roman numbers.

I R. Kalakoti, S. Nõmm, and H. Bahsi. In-depth feature selection for the statistical machine learning-based botnet detection in iot networks. *IEEE Access*, 10:94518–94535, 2022

II M. U. Rehman, R. Kalakoti, and H. Bahşi. Comprehensive feature selection for machine learning-based intrusion detection in healthcare iomt networks. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 2: ICISSP*, pages 248–259. INSTICC, SciTePress, 2025

III R. Kalakoti, H. Bahsi, and S. Nõmm. Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection. *IEEE Internet of Things Journal*, 11(10):18237–18254, 2024

IV R. Kalakoti, S. Nõmm, and H. Bahsi. Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai). In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 595–601. IEEE, 2023

V R. Kalakoti, R. Vaarandi, H. Bahşi, and S. Nõmm. Evaluating explainable ai for deep learning-based network intrusion detection system alert classification. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, pages 47–58. INSTICC, SciTePress, 2025

VI R. Kalakoti, S. Nõmm, and H. Bahsi. Explainable transformer-based intrusion detection in internet of medical things (iomt) networks. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 1164–1169. IEEE, 2024

VII R. Kalakoti, S. Nõmm, and H. Bahsi. Enhancing iot botnet attack detection in socs with an explainable active learning framework. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 265–272. IEEE, 2024

VIII R. Kalakoti, H. Bahsi, and S. Nõmm. Explainable federated learning for botnet detection in iot networks. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 01–08. IEEE, 2024

IX R. Kalakoti, S. Nõmm, and H. Bahsi. Federated learning of explainable ai (fedxai) for deep learning-based intrusion detection in iot networks. *Computer Networks*, page 111479, 2025

X R. Kalakoti, H. Bahsi, and S. Nõmm. Synthetic data-driven explainability for federated learning-based intrusion detection system. In *IEEE Internet of Things Journal*. IEEE, 2025

Other Publications:

XI M. U. Rehman, R. Kalakoti, and H. Bahşi. Exploring the impact of feature selection on non-stationary intrusion detection models in iot networks. In *Accepted for 22nd Annual International Conference on Privacy, Security, and Trust (PST2025)*, 2025 **(Accepted)**

Author's Contributions to the Publications

I In **Publication I**, I was the first author and was responsible for developing the machine learning framework, implementing the feature selection techniques, creating the visualizations, performing the formal analysis, and writing the manuscript.

II In **Publication II**, I was the second author and contributed to the conceptualization and methodology, co-developed the machine learning framework, implemented the feature selection techniques, created the visualizations, performed the formal analysis, and contributed to writing and revising the manuscript.

III In **Publication III**, I was the first author and was responsible for the conceptualization and methodology of the study, the development of the machine learning framework that integrated feature selection, the implementation of explainable AI (XAI) metrics for evaluating explainability, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

IV In **Publication IV**, I was the first author and was responsible for the conceptualization and methodology of the study, the development of the deep learning framework for IoT botnet detection using XAI, the implementation and evaluation of XAI methods, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

V In **Publication V**, I was the first author and was responsible for the conceptualization and methodology of the study, the implementation and evaluation of XAI methods, the collaboration with SOC analysts for validation, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

VI In **Publication VI**, I was the first author and was responsible for the conceptualization and methodology of the study, the implementation and evaluation of XAI Methods, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

VII In **Publication VII**, I was the first author and was responsible for the conceptualization and methodology of the study, the design and implementation of the explainable active learning framework, the integration and evaluation of explainable AI (XAI) techniques, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

VIII In **Publication VIII**, I was the first author and was responsible for the conceptualization and methodology of the study, the design and implementation of the explainable federated learning framework, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

IX In **Publication IX**, I was the first author and was responsible for the conceptualization and methodology, the development of the federated learning (FL) of explainable AI (FedXAI) framework for deep learning–based intrusion detection systems, the integration and evaluation of XAI methods, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

X In **Publication X**, I was the first author and was responsible for the conceptualization and methodology of the study, the design and implementation of a synthetic data–driven explainability approach for FL-based intrusion detection systems, the

integration and evaluation of XAI methods, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

XI In **Publication XI**, I was the third author and was responsible for the conceptualization and methodology of the study, the creation of visualizations, the formal analysis, and the writing and revision of the manuscript.

Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
IoT	Internet Of Things
IoMT	Internet of Medical Things
IDS	Intrusion Detection System
SOC	Security Operations Center
FL	Federated Learning
LSTM	Long Short-Term Memory
DNN	Deep Neural Networks
RF	Random Forest
DT	Decision Trees
XGBoost	Extreme Gradient Boosting
HTTP	Hypertext Transfer Protocol
XAI	Explainable Artificial Intelligence
SHAP	SHAP (SHapley Additive exPlanations)
LIME	Local Interpretable Model-Agnostic Explanations
IG	Integrated Gradients
DeepLift	Deep Learning Important FeaTures
IID	Tndependent and Identically Distributed
Non-IID	Non- Independent and Identically Distributed (IID)

Where the mind is without fear and the head is held high;
Where knowledge is free;
Where the world has not been broken up into fragments
by narrow domestic walls;
Where words come out from the depth of truth;
Where tireless striving stretches its arms towards perfection;
Where the clear stream of reason has not lost its way
into the dreary desert sand of dead habit;
Where the mind is led forward by thee
into ever-widening thought and action—

Into that heaven of freedom, my Father, let my country awake.

— **Rabindranath Tagore**

1 Introduction

With the accelerated pace of global digitization, the frequency and complexity of cyber attacks have dramatically increased [71] [51]. These attacks not only compromise the data security of individuals and businesses but can also cause substantial economic losses and harm to reputation. The global indicator for the "Estimated Cost of Cybercrime" in the cybersecurity market is projected to continuously rise by a total of 6.4 trillion U.S. dollars between 2024 and 2029 [110], [132]. Given the growing sophistication of these threats, there is an urgent need for more advanced threat detection systems to effectively identify and mitigate them.

Simultaneously, the rapid advancement of the Internet of Things (IoT) has led to the widespread adoption of various applications in smart cities [75]. IoT devices operate in diverse environments to achieve various objectives. Recent innovations in IoT include connected computers, sensors, buildings, and communities, all contributing to the concept of "smartness." [117]. IoT devices are primarily at risk due to the lack of sufficient built-in security measures to defend against threats. This vulnerability arises because the IoT environment is a complex network characterized by constant dynamism, resource limitations, and large data volumes [101]. One Noticeable threat is botnet-based attacks. A botnet network is a sophisticated collection of bots used by cybercriminals to conduct malicious activities over the internet. Botnet-based attacks pose significant challenges for IoT networks. Detecting attacks in IoT networks is particularly challenging due to specific requirements, such as the need for low latency, mobility, and a distributed architecture [90]. Moreover, The main challenge is that attackers continuously develop new tools and techniques to exploit vulnerabilities.

In the context of healthcare as well, Internet of Medical Things (IoMT) devices have improved patient monitoring but have also introduced significant security vulnerabilities that attackers can exploit[10], which poses risks to patient privacy and safety [26]. DoS and DDoS attacks on IoMT devices could disrupt critical medical equipment, potentially endangering the lives of patients [26]. Furthermore, IoMT devices are prime targets for attackers because of the sensitive medical data they collect, and any breaches could lead to the exposure of personal patient information [135], [108]. To combat this, intrusion detection systems (IDS) have become an essential part of network security.

IDS is a tool that monitors system activity or network traffic for patterns that may indicate an attack. There are primarily two types of IDSs: a) Host-based IDS and b) Network-based IDS [142]. A Host-based IDS monitors specific hosts or devices and alerts users if it detects any suspicious activities, such as modifications or deletions of system files, unusual sequences of system calls, or unauthorized changes to system configurations. In contrast, a Network-based IDS is typically positioned at key points in the network, such as gateways and routers. Its primary function is to monitor network traffic for signs of intrusion [95]. NIDS is a network monitoring tool for detecting botnet command and control traffic, cyberattacks, and other unauthorised network behaviour. However, organisational NIDS environments often produce hundreds of thousands of alerts per day, with a significant portion of them having low importance [143]. Detection mechanisms in IDS can be categorized into three types: misuse detection, anomaly detection, and hybrid detection. In the misuse detection approach, an IDS maintains a knowledge base of rules for detecting known attack types. Misuse detection techniques can be broadly categorized into knowledge-based and Machine learning-based techniques. In the knowledge-based technique, network traffic or host audit data (such as system call traces) are compared against predefined rules or attack patterns [109].

Knowledge-based techniques can be categorized into three types: (i) Signature match-

ing, (ii) State transition analysis, and (iii) Rule-based expert systems [109]. Signature matching misuse detection techniques check incoming packets against established patterns, flagging those that match as anomalous [107]. State transition analysis models track known suspicious patterns, with different paths leading to a compromised state [48]. Rule-based expert systems have databases of rules for various intrusion scenarios [105]. However, a disadvantage of knowledge-based IDS is that they require regular maintenance of their knowledge database to remain effective. Additionally, they may fail to detect variants of new attacks. The main limitation of signature-based IDS is that it requires regular updates to add signature rules for new attack definitions [76]. Additionally, it tends to generate more false alarms for evolving attacks that do not have defined signatures [88].

In recent decades, researchers have made significant advancements in ID by introducing traditional machine learning (ML) and data mining methods. The main contribution of this dissertation focuses on the domain of ML-based IDS, which adopt a learning-based approach to detect various classes of cyberattacks based on learned normal and attack behaviour. Supervised learning algorithms, such as decision trees (DT), Naive Bayes, and deep learning(DL) techniques [78], are particularly suitable for misuse detection [127], [148]. However, most ML models (except for interpretable ones, such as DT, in simple IDS settings) function as black boxes, offering no explanations for their decisions to the security experts who manage these systems [118] [12]. Specifically, previous ML-based research work has primarily focused on improving the performance of the model without providing insights into the internal reasoning and behaviour of the IDS. This critical limitation motivates the core research challenge addressed in this dissertation—the urgent need for more transparent, explainable, and trustworthy ML solutions for IDS.

1.1 Explainable AI for IDS

Despite the impressive results achieved by many ML methods in the cybersecurity domain, significant concerns arise due to the inherent lack of interpretability in ML models. This implies that security experts may encounter difficulties trusting the outputs of ML models because they do not fully understand how a model reaches a particular decision or classification. Unlike traditional ML models, which provide clear decision boundaries and feature explanations, DL models often operate as "black box models," making it difficult for experts—such as Security Operations Centre (SOC) analysts, system administrators, and developers—to understand the model and trust the underlying decision-making process. In response to this challenge, new approaches have been developed in recent years aimed at improving the transparency of the black-box nature of ML models, making their outputs more interpretable. This concept is commonly referred to in the literature as explainable artificial intelligence (XAI). Understanding why a classifier categorizes a given input instance in a specific way is becoming increasingly important as AI continues to evolve rapidly and influence many aspects of our daily lives. However, many AI systems in use today often lack transparency. XAI is crucial for users who rely on ML models, as it enables decision-makers to understand the rationale behind the model's recommendations. For example, in cybersecurity, experts need to comprehend why the ML model has flagged a particular flow of network traffic as malicious. Such understanding is essential for effectively identifying and addressing potential threats, ultimately fostering trust in cybersecurity experts [126]. XAI further enhance threat investigation by assisting SOC analysts in understanding how the model predicted a specific network traffic flow data point, providing a precise root cause analysis [155]. Additionally, black box models may often suffer from issues such as biases, misclassifications, or overfitting [46]. XAI assists in improving the detection performance and overall reliability of the models [32]. Several XAI methods

have emerged as important local explainers, offering detailed instance-level explanations by assigning importance scores to the features most responsible for model decisions. As a result, experts can gain insights into the model's output by evaluating whether these features are relevant in the context of the associated cyber incident. XAI explanations can be categorised into two scopes: global and local. The local explanation in the model focuses on explaining individual predictions, whereas the global explanation unveils the overall behaviour of the model. In XAI, models can achieve explanations through an intrinsic (model-specific) approach, in which explainability is integrated into the model during training and is not transferable, or a post hoc (model-agnostic) approach. The post-hoc XAI method is independent of any architecture and can be applied to any trained ML model [46, 141]. Currently, XAI approaches for intrusion detection are still in their early stages, and the available literature is limited [103], [65], [25].

Several recent works ([45, 84, 104, 116]) have begun to integrate explainability within IDS to enhance interpretability and user trust, proposed an abstract design for a human-in-the-loop approach in intelligent ID and emphasised the role of explainability in facilitating analyst decision-making. Most of the work on XAI methods in cybersecurity primarily focuses on visualisation and model verification.

The necessity and value of explanations in security systems were first emphasized by Vigan et al. [146], who introduced the "Six Ws" paradigm to enhance understanding of intelligent system functionality. They underscored the role of explainability in improving system transparency and trust by addressing six fundamental questions:'Who', 'What', 'Where', 'When', 'Why', and 'How'.

Hatma et al. proposed a novel model for detecting botnet Domain Generation Algorithms (DGAs) [137]. The study used five ML models on datasets from 55 different botnet families. Random Forest model attained the highest accuracy of 96.3%. To enhance model transparency and trust, the authors incorporated Open-source Intelligence (OSINT) with XAI techniques, including SHAP [82] and LIME [115]. However, the proposed framework does have limitations, including the high computational complexity due to feature extraction and reduced robustness against Mask botnet attacks [137].

Visualization tools were also employed to provide clearer insights into the rationale behind labeling an account as either a botnet or legitimate. Michele et al. proposed ReTweet-Tweet (RTT), a concise yet informative scatterplot visualization created to facilitate the exploration of a user's retweeting behavior [89]. Although their botnet detection framework, Retweet-Buster (RTbust), relies on unsupervised feature extraction using Variational Autoencoders (VAEs) and LSTM, the RTT visualisation can still be effectively utilised after classification. It aids in interpreting the characteristics of accounts identified as bots, thereby contributing to a more transparent and explainable detection process. In the study by [35], authors proposed an explainable hybrid IDS that incorporates a rule-based method with human expertise and ML, constructing a comprehensive hybrid system. The decision tree, a white box model known for its intrinsic explainability, was employed to provide rule-based explanations for IDS, making it more understandable to domain experts. With the improvement of DL in cyber security, various pilot studies have explored to understand the behaviour of ML-IDSs in botnet network traffic. However, cyber-security stakeholders need help building their faith in the results of current DL models due to bad decisions made by complex neural network models. To deal with such as problem, in paper [69], Kundu et al.[69] conducted experiments by utilising a combination of synthetic and real network traffic constructed by the IXAI breaking point system [50]. 1DCNN model is tested over three kinds of datasets. The synthetic dataset is generated by the IXAI appliance, and the others are the Stratosphere IPS Project dataset [133] and

Kitsune dataset [94] in botnet traffic. The authors have demonstrated that the presented DCNN botnet detection models outperform previous ML models, with advancements of up to 15% for all classification metrics. Additionally, SHAP was employed to elucidate model decisions, enhancing trust among cybersecurity stakeholders. Le et al. [72] utilized SHAP alongside DT and RF models to develop a comprehensive IDS. They implemented a heatmap to visualize the impact of individual features on the overall model. Additionally, a Decision Plot was employed to explain specific instances within the dataset. The study was conducted using the ToN-IoT [98] and BoT IoT [66] datasets.

Wang et al. [151] proposed a framework that utilises SHAP [82] to unify local and global explanations, thus enhancing the explainability of ID decisions [151]. Local explanations elucidate the justification behind predictions for individual instances, while global explanations highlight the prominent features learned by the model, showing the relationships between feature importance and attack types. Also, the study conducts a comparative analysis of explanations generated for two classifier configurations, namely, the one-versus-all classifier and the multiclass models, using the NSL-KDD data [140] as also examined in [151].

Liu et al. present FAIXID [79], a framework that incorporates explainability into IDS at various levels. These levels include data cleaning, elucidating the internal workings of a trained supervised model, providing local explanations for predictions, and displaying results to security analysts through different visualizations tailored to the expertise or role of each analyst [79].

Rao et al. [111] used an isolation forest model trained on the NSL-KDD dataset to distinguish between normal and anomalous samples. To achieve explainability, Authors use SHAP [82] and LIME [115] for extracting and visualising feature-based explanations. Moreover, they auto-generate labels for the attacks, assigning to each anomaly the name of the most important feature to make the prediction [111].

Szczepański et al. proposed the hybrid Oracle Explainer IDS, which combines Artificial Neural Networks (ANNs) and DT while utilizing microaggregation methods to enhance performance [138]. This system aims to achieve high accuracy while providing human-understandable explanations for its decisions. The authors developed an Oracle-based Explainer module that measures the distance between clusters formed from the dataset[124] and the test instances. The closest cluster is then used to generate an explanation for the decision.

Zolanvari et al. [159] developed a model-agnostic xai framework called Transparency Based on Statistical Theory (TRUST) for numerical applications. TRUST uses factor analysis to convert input features into latent features, ranks them using mutual information, and employs a multimodal Gaussian distribution to classify new samples [159] . It has been tested on three datasets: NSL-KDD [140], UNSW [99], and the "WUSTL-IIoT" dataset [158] on intrusion network traffic.Finally, compared to LIME [115], the TRUST XAI model has achieved a success rate 98% in explaining random test samples.

In a paper [86], the authors addressed the challenge of explaining IDS in computer networks using DL. They developed an XAI framework to enhance the transparency of the model. To design the DL-based IDS, the authors utilized the NSL-KDD dataset [140]. They employed four XAI methods—SHAP [82], LIME [115], ProtoDash [42], and Contrastive Explanations Method(CEM) [34]—to elucidate the internal workings of the DL-based-IDS model.

1.1.1 Research Gap in Centralized XAI-IDS

While research on XAI in cybersecurity is increasing, However, current XAI techniques may produce unreliable explanations in real-world scenarios due to their limited qualitative evaluation[29]. Despite the growing recognition of the importance of explainability in ML models, researchers face challenges in establishing universal and objective criteria for developing and validating explanations [93]. These challenges need to be addressed to promote the advancement of XAI. Key issues include: (i) reaching a consensus on the appropriate definition of model explainability, (ii) identifying and formalizing explainability tasks from the perspectives of various stakeholders, and (iii) designing measures to evaluate the effectiveness of explainability techniques [39]. Another challenge is finding models that balance both high performance and transparency. Generally, more accurate models tend to be complex and difficult to understand. This trade-off is particularly significant in IDS systems, where users typically expect both high performance and clear explanations to foster a strong sense of trustworthiness [9]. Evaluating explainability is crucial to ensure that the employed XAI methods can be effectively deployed in real-world scenarios and contribute to a better understanding of model outputs. Therefore, designing an XAI system that incorporates robust qualitative and quantitative evaluation procedures for real-world application is critically important for the effective adoption of XAI-based IDSs.

Despite considerable advancements in recent times, significant gaps remain in the existing literature concerning XAI-based ID systems, necessitating a more in-depth investigation. A critical gap that stands out is the absence of standardized evaluation metrics to evaluate the effectiveness and utility of XAI methods within the context of ID domain. Current research in cyber security relies primarily on qualitative assessments of XAI approaches. Developing a comprehensive suite of benchmarks to measure the domain-specific facets of explainability would enable more robust comparisons across diverse XAI methodologies and streamline identifying optimal practices. While related works mentioned in Section 1.1 can be viewed as initial steps toward introducing explainability into IDS systems, to the best of our current understanding, there still needs to be a more quantitative evaluation of XAI methods in the literature pertaining to cybersecurity. This highlights a need for more rigorously evaluating the quality of generated explanations, an essential requirement to establishing confidence in the explanation outputs of AI systems built upon XAI principles [4, 5, 7, 16, 20, 28, 35, 45, 49, 52, 64, 69, 72, 79, 84–86, 89, 96, 100, 104, 106, 111, 116, 119, 125, 128, 137, 138, 146, 149–151, 159].

1.2 Explainable AI for Decentralized IDS

ML is increasingly recognised as a potential approach for detecting and mitigating attacks on IoT devices, which are becoming more sophisticated. Numerous studies have investigated how the effectiveness of traditional IDS can be improved by utilising ML to analyse network traffic and detect potential attacks. While ML increases attack detection accuracy in IDS, most of these ML-enabled systems remain centralised. In centralized systems, a single entity (the server) collects network traffic data from multiple devices to train the model. As a result, the server has access to all network traffic generated by the communication among various devices participating in the training process and the local data from those devices, which could lead to privacy issues [24]. Therefore, decentralized data management solutions are crucial for preserving data privacy during the model training process [47]. As ML is applied to an increasing number of applications, users are becoming more concerned about the privacy of their data. To protect their personal privacy, users may hesitate to share their data. Numerous laws on data privacy and security have been implemented to enhance user privacy and security, such as the EU's General Data Protec-

tion Regulation (GDPR) [37] and China's Cybersecurity Law [153]. These strict regulations prevent the direct exchange or central collection of data from multiple parties to avoid privacy leaks. Therefore, It is essential to find ways to use data from multiple parties for training ML models while ensuring compliance with data privacy and security regulations.

To address the privacy issues associated with traditional centralized ML approaches, Federated Learning (FL) [91] was proposed in 2016 as a collaborative learning method. In FL, end devices (often referred to as clients or parties) do not share their data. Instead, they only transmit partial updates of a global model, which are then aggregated by a central entity. This approach is designed to enhance user privacy, as the data from users' devices remains confidential and is not shared with other entities. Generally, a FL design system involves a large number of client devices with varying amounts and distributions of data. In real-life situations, this data is often non-independent and identically distributed (Non-IID) [156]. For instance, in an IDS deployed on a particular network, some target devices may experience traffic associated with various types of attacks (such as DoS or port scanning), while other devices might only exhibit traffic related to their intended operations[3].

Recently, the concept of Fed-XAI, which integrates FL with XAI paradigms, has emerged. The synergy between these paradigms is crucial for achieving trustworthiness in ML models, as it enables the simultaneous pursuit of transparency and privacy preservation requirements. In recent years, the scientific community has focused significantly on developing XAI [21] [1] [80], but relatively little attention has been dedicated to addressing interpretability issues in FL [18] [38]. In an IDS operating within a FL framework, security experts who analyze intrusions on IoT devices using server-side models need to understand how these models make their decisions. Achieving both privacy and explainability in this context presents a significant challenge. The integration of XAI methods into FL has not received much attention and introduces additional challenges due to the complex, distributed nature of FL, where the model is trained across multiple IoT devices. Post-hoc XAI methods, such as feature importance explainers (eg: LIME or SHAP) and model interpretability techniques, typically require access to the complete training dataset or input reference data, which can pose a privacy risk. Therefore, achieving explainability for a black-box model on the server in FL without providing data to the server remains a challenge that needs to be addressed.

1.3 Research questions

Research Objective: This PhD dissertation research aims to enhance the effectiveness, transparency, and privacy of machine learning-based intrusion detection systems (IDS). It emphasizes the critical role of explainability and its quantitative evaluation in improving the detection of IoT botnets, identifying attacks in IoMT networks, and classifying alerts generated by network intrusion detection systems (NIDS). The research further explores the role of explainable AI in an active learning loop for IoT botnet detection. To advocate for responsible AI practices while preserving data privacy in decentralized environments, this research aims to develop transparent and trustworthy IDS models within federated learning settings, without revealing sensitive client data.

To achieve the above research objective, we have formulated four main research questions that guide our investigation into improving IDS through explainable and privacy-preserving artificial intelligence. These questions are designed to explore the potential of XAI in both centralized and decentralized settings, with the aim of building more transparent, trustworthy, and effective ML-IDS models that are better suited for real-world deployments. The following four research questions are addressed in this dissertation.

- **RQ1:** What minimal feature subsets are most influential for achieving high detection performance criteria in IoT botnet and healthcare IoMT cyber attack detection?
- **RQ2:** How can the quality of explainable ai methods be quantitatively evaluated for intrusion detection across IoT botnet detection, IoMT attack detection, and NIDS alert classification?
- **RQ3:** How do local explanations impact annotation quality and training outcomes in active learning for IoT botnet detection?
- **RQ4:** How can explainable AI be integrated into federated learning for effective intrusion detection while ensuring transparency and data privacy?

In this dissertation,**RQ1** addresses the feature selection problem in intrusion detection systems (IDS), with the goal of identifying optimal subsets of features using various selection methods tailored to different classification scenarios, particularly for detecting IoT botnet and IoMT attacks. To achieve transparency in the black-box nature of ML models for IDS, we applied explainable AI using various methods. To overcome the previously mentioned challenges regarding the scarcity of evaluating explainability, **RQ2** emphasizes the importance of considering the quality of explainability through quantitative evaluation, alongside the detection performance of the models. We advocate for including this evaluation of explainability as a significant criterion for building models that security experts can trust in XAI-based IDS. **RQ3** investigates how local explanations impact the annotation quality and model training within an active learning loop for IoT botnet detection. We advocate for the integration of responsible AI practices by prioritizing transparency, accountability, and ethical compliance in threat detection processes. While post-hoc explainability methods typically require access to real client data to generate explanations, federated learning (FL) settings pose unique challenges due to privacy constraints that prevent server-side access to client data. To address this issue, **RQ4** propose multiple Fed-XAI frameworks designed to enhance the explainability of global IDS models in federated settings, ensuring transparency without compromising the privacy of participating clients.

1.4 Research Gaps & Contributions

This PhD thesis's contribution is based on a compilation of peer-reviewed scientific publications that have been published in reputable journals and international conferences. The contributions advance the field of intrusion detection by promoting the development of effective, transparent, and trustworthy machine learning-based IDS models in both centralized and decentralized settings. These models are validated through extensive experimental studies to ensure their applicability, scalability, and reliability in real-world deployment scenarios. This PhD deseeration systematically addresses five key Research gaps:**RG-A, RG-B, RG-C, RG-D, & RG-E**, presenting corresponding contributions **A1, A2, B1, B2, C1, D1, E1, F1 & F2**.

RG-A. Overlooked Role of Feature Selection in IoT Botnet and IoMT Intrusion Detection: While ML have demonstrated high performance in classifying malicious traffic detection problems, feature selection, an important step in the ML workflow, has not been fully addressed [23]. One significant concern is the curse of dimensionality, which can negatively impact detection performance by causing overfitting when classifiers are trained with a large number of features [74]. Furthermore, a high-dimensional feature space may require more computing resources when models

are deployed in a real-world environment. Most intrusion detection systems must handle large volumes of network traffic, so optimizing resource usage is crucial, especially in IoT networks. Therefore, reducing the size of the feature set can enhance the performance of ML models in several ways. However, existing studies do not examine the impact of feature selection methods on different binary and multi-class classification approaches that can be applied for intrusion detection at various stages of the botnet life cycle. Moreover, feature selection aids in achieving a deeper understanding of the underlying processes that generated the data, as fewer features are easier for experts to interpret. To address this gap, this dissertation emphasizes the critical role of robust feature selection by conducting an in-depth evaluation of feature selection methods for IoT botnet and IoMT attack detection, thereby contributing to the advancement of more robust and efficient ML-based Intrusion detection systems.

A1. In-depth feature selection for IoT botnet detection: In Publication I, to address the above research gap in IoT botnet detection, an in-depth feature selection analysis was conducted by evaluating filter and wrapper methods across various binary and multi-class classification formulations based on the IoT botnet life cycle. This study identifies optimal minimal feature subsets that enhance model detection performance and improve the efficiency of IDS.

A2. Comprehensive Feature Selection for IoMT Intrusion Detection: In Publication II, a comprehensive feature selection study was conducted for healthcare IoMT IDS using two network traffic datasets for detecting cyber attacks in IoMT networks. By addressing the neglected role of feature selection, this study emphasises its significance in enhancing detection capabilities and finding minimal feature subsets using filter-based methods (Fisher Score, Mutual Information, and Information Gain) for both binary and multi-class ID, contributing to the development of efficient and robust IDS.

RG-B. Lack of Quantitative Evaluation of Explainability in ML-based IoT Botnet Detection: Although numerous current works have achieved high detection performance in ML-based IDS, most research has primarily focused on improving classification performance, while neglecting the evaluation of explanation quality. As XAI tools are increasingly being integrated into network security to assist analysts in making informed decisions, a key challenge remains in validating these tools, assessing their explanation quality, and evaluating relevant security metrics. This lack of rigorous evaluation reduces trust in deploying XAI methods for real-world IDS applications. Moreover, many previous works have not conducted comprehensive assessments of XAI methods across different intrusion datasets and diverse ML models, limiting the generalizability of their findings.

B1. Quantitative Evaluation of Explainability in IoT Botnet Detection: To enhance the transparency, trust, and explainability of IDS systems, we developed frameworks that quantitatively evaluate XAI methods for IoT botnet detection, addressing a critical gap in the rigorous evaluation of explanation quality in black-box IDS models. In Publication III, statistical ML models were evaluated, while in Publication IV, deep learning models were assessed across both binary and multi-class classification tasks. The quantitative evaluation was conducted using key metrics such as faithfulness, complexity, and robustness of the generated explanations.

RG-C. Missing Ground-Truth-Based Evaluation of XAI Methods for NIDS Alert Classification: Evaluating explainability in cybersecurity, especially in NIDS alert classification, poses a significant challenge due to the absence of clearly defined ground-truth explanations. Unlike image classification, where humans can directly annotate important sub-images or pixels—creating human-annotated ground-truth masks for evaluating the explanations —NIDS operates on abstract network traffic features are often difficult even for experts to interpret. This lack of clear, human-recognizable ground truth makes it challenging to evaluate whether an explanation accurately reflects the reasoning behind the predictions of black-box models. However, previous research has largely neglected the integration of expert-driven ground truth in evaluating explanation quality, which is essential for ensuring the trustworthiness of XAI-assisted alert prioritization systems in real-world SOC environments.

- **C1. Evaluation of Explainability for NIDS Alert Classification:** In Publication V, Developed a transparent deep learning-based model for classifying real-world NIDS alerts, utilizing a dataset collected from an operational SOC. Collaborated directly with SOC analysts to define high-priority alerts and used their domain expertise as ground truth to evaluate the reliability of XAI outputs, alongside key explainability criteria, including faithfulness, complexity, and robustness.

RG-D. Lack of Explainability Evaluation for Transformer-based IDS in IoMT Attack Detection: Although many previous studies have adopted Transformer architectures to improve IDS performance, they have largely neglected the explainability of Transformer models. While explainability has been explored for traditional ML models or in non-IoMT domains, there remains a significant gap in systematically evaluating XAI methods specifically for Transformer-based IDS operating on healthcare-specific IoMT network traffic.

- **D1. Explainable Transformer-based IDS for IoMT Attack Detection:** In Publication VI Designed a Transformer architecture and adapted it for network traffic analysis to enable effective intrusion detection for IoMT cyber attack detection. To foster trust and transparency in Transformer-based decision-making for critical healthcare security environments, explainable AI techniques were integrated to facilitate the understanding of feature contributions in Transformer model predictions, and the quality of explanations was rigorously evaluated.

RG-E. Limited Integration of Explainability in Active Learning for IDS: While multiple previous work have explored XAI methods and active learning independently for enhancing ID, very few works have addressed their combined application in real-world IDS settings. In particular, existing IDS research primarily focuses on improving model detection performance but often neglects the role of XAI in helping SOC analysts during the labeling process in active learning cycles. Moreover, prior studies rarely evaluate the quality of explanations produced by XAI methods within active learning loops using quantitative evaluation metrics.

- **E1.** Explainable Active Learning (XAL) Framework for IoT Botnet Detection:In Publication VII, We proposed an Explainable Active Learning (XAL) framework to enhance both model performance and transparency in IoT botnet detection. The framework demonstrates how active learning–based query strategies, such as uncertainty sampling and query-by-committee, can be improved through the

integration of explainable AI methods and the quantitative evaluation of explanation quality to support the selection of the most informative instances for annotation.

RG-F. Challenges in Integration of Explainable AI in Federated Learning: While several recent works have investigated incorporating XAI into FL frameworks for ID, notable challenges have still not been addressed. Previous studies have mainly explored applying post-hoc explainability methods either to individual client models or to the global server model FL. Nevertheless, most of these approaches disregard the privacy risks associated with explaining server-side models, as many post-hoc explainers require access to training client data or reference data to generate explanations.

F1. FedXAI-based IDS: In Publication VIII & IX, We proposed a FedXAI framework that integrates explainable AI into FL to enable privacy-preserving intrusion detection of IoT botnets across both binary and multi-class classification tasks in decentralized settings. To address the challenge that post-hoc explainability methods (e.g., SHAP) typically require access to reference data (Training data), the framework employs a secure aggregation mechanism based on secret sharing to compute server-side SHAP explanations without compromising client data privacy. This framework ensures that the server achieves transparency and explainability of the global model while preserving individual client data. The reliability of the aggregated SHAP explanations was evaluated by comparing them against server-side SHAP explanations generated using real client data, demonstrating that client-side SHAP aggregation can effectively approximate real data-based explanations, thereby achieving both transparency and privacy in federated IDS environments.

F2. Synthetic data-driven explainability for Federated IDS model: We propose a novel framework called FEDXAI to address the challenges of achieving explainability for the black-box nature of the global model on the server side in FL. This framework leverages synthetic data generated by variants of Generative Adversarial Networks (GANs) to provide transparency and explainability of global model in FL without accessing client data. We evaluated the quality of global model explanations using synthetic data by comparing them to explanations derived from real client data. Experimental results demonstrated synthetic data–based explanations closely approximate with explanations based on real data, validating the effectiveness and feasibility of privacy-preserving explainability in decentralized IDS environments.

Table 1: Mapping of dissertation chapters to research questions, corresponding publications, and key contributions

Chapter	Research Question	Research Gap	Contributions	Publications
4	**RQ1**	RG-A	A1, A2	1, 2
5	**RQ2**	RG-B	B1	3, 4
6		RG-C	C1	5
7		RG-D	D1	6
8	**RQ3**	RG-E	E1	7
9	**RQ4**	RG-F	F1	8, 9
10			F2	10

1.5 Thesis Structure

The main content of the dissertation is organised into chapters that address the key research questions outlined in the previous subsection, with each chapter contributing to specific theoretical and practical advantages in transparency and privacy serving IDS in both centralized and decentralized settings.

Table 1 provides a clear mapping of each chapter of this dissertation to its corresponding research question, associated publications, and specific contributions. This mapping is presented for the sake of clarity and to highlight how each chapter addresses the core objective of the research. A brief description of the content of each chapter is provided as follows.

- **Chapter 1:** Introduction emphasizes the importance of intrusion detection in the context of evolving cybersecurity threats. It defines the research problem, highlights existing research gaps, outlines the objectives of the study, formulates the key research questions, and presents the major contributions of the dissertation.

- **Chapter 2:** This chapter provides background information on Explainable AI (XAI) and the various methods used in this dissertation to achieve model explainability.

- **Chapter 3:** Provides background on Federated Learning, including its architecture, privacy challenges, and relevance to intrusion detection systems.

- **Chapter 4:** addresses **RQ1** for the evaluation of the feature selection methods to identify the minimal number of features for improving the robustness and efficiency of ML-based IDS models for IoT botnet and IoMT attack detection.

- **Chapter 5:** Addresses **RQ2** by evaluating the quality of explanations generated by post-hoc explainable AI (XAI) methods for IoT botnet detection in both binary and multi-class classification tasks. The evaluation is based on key criteria including the explanation of faithfulness, complexity, and robustness.

- **Chapter 6:** Focuses on deep learning-based NIDS alert classification using real-world data collected from a SoC. It includes direct feedback from SOC analysts to assess the quality and reliability of explanations generated by post-hoc XAI methods, addressing important aspects of **RQ2**.

- **Chapter 7:** discusses a ChatGPT-inspired transformer architecture tailored for IoMT attack detection, which focuses on enhancing the transparency of transformers using XAI methods to explain transformer model decision-making in IoMT cyber attack prediction, addressing **RQ2**.

- **Chapter 8:** Addresses **RQ3** by proposing an Explainable Active Learning (XAL) framework that integrates explainable AI into the active learning loop to enhance annotation quality and improve model performance in IoT botnet detection.

- **Chapter 9:** addresses **RQ4** by developing the FedXAI framework, which introduces secure aggregation of client-side SHAP explanations to provide transparency into the global black-box model on the server in a federated learning-based intrusion detection system without requiring direct access to client data, thereby achieving explainability while preserving privacy.

- **Chapter 10:** discusses **RQ4** further by proposing synthetic data-driven explainability using GAN-variant–based synthetic data to explain the global model on the server in a federated learning setting, achieving transparency in intrusion detection systems that operate in federated learning without accessing client data.
- **Chapter 11:** Presents the main conclusions of the research, highlights its limitations, and outlines possible directions for future work based on the findings and contributions of the dissertation.

2 Background Work

To situate this research within the broader scientific landscape, the following section provides a detailed overview of the background on explainable AI and methods used in this dissertation for various publication studies.

2.1 Explainable AI

Using advanced machine-learning models trained on large datasets can lead to decision-making systems that we don't fully understand. This lack of understanding raises concerns about ethics, accountability, safety, and industrial liability. Explainability and interpretability methods help understand how machine learning (ML) detection systems make decisions, a concept known as Explainable Artificial Intelligence (XAI). In 2016, the Defense Advanced Research Projects Agency (DARPA) initiated the "Explainable AI (XAI) Program" to develop models that are easier to understand while achieving high performance in learning and prediction. The program aims to enhance user trust and management of future artificial intelligence partners. The goal of XAI is to build trust among experts by clarifying the reasons behind ML model predictions, identifying potential biases, and ensuring compliance with legal requirements in areas like the healthcare sector and cybersecurity domains.

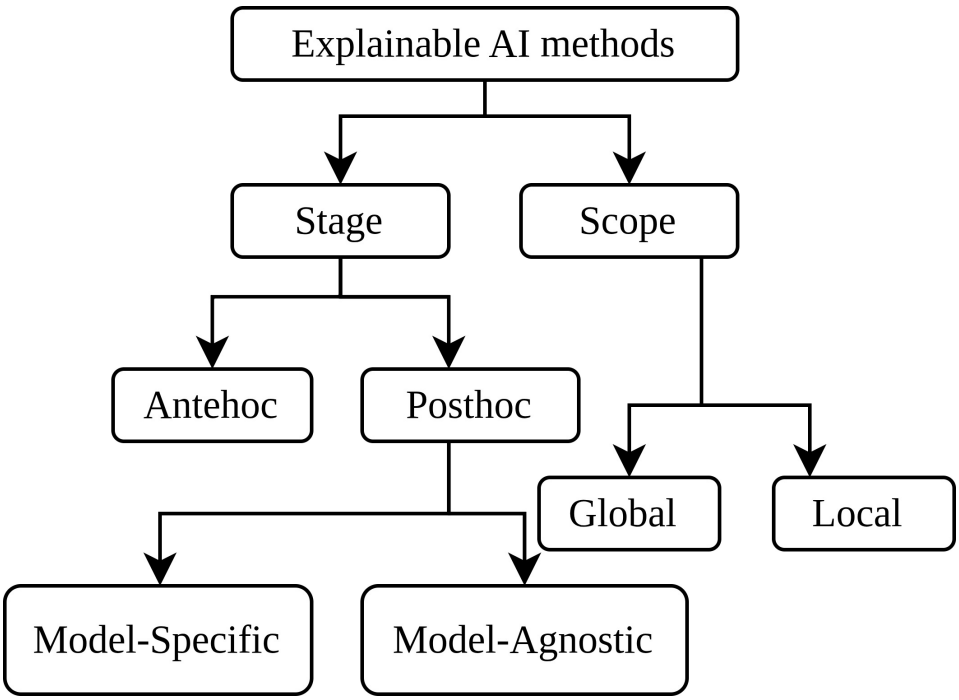


Figure 1: Explainable ai methods Taxonomy

Figure 1 presents a taxonomy of explanation AI methods.It's essential to differentiate between ante-hoc explainability, which involves directly training explainable models, and post-hoc explainability, which focuses on explaining pre-trained models. Examples of ante-hoc explainable models include linear regression, decision trees, k-nearest neighbors, and Bayesian learners. Many believe that ante-hoc explainable models under-

perform, resulting in the use of opaque models. These complex models are often black boxes, which are complex to understand. Post hoc explanation methods generate explanations for already trained models. They can be divided into model-agnostic methods, which work with any model, and model-specific methods, which are tailored for specific models. A scope of explanations can be achieved through both local and global methods. Local explanations provide insights into individual predictions of the model, while global explanations describe the overall behavior of the model. For instance, global explanations address the question of "how does the ML model make predictions?" Local explanations answer "why is this instance given this model prediction?". In this dissertation, various Posthoc explanation methods were used for the publications addressing the problem of intrusion detection systems. A description of these XAI methods is provided below.

2.2 Explainable AI methods

For data set D, input $x \in \mathbb{R}^d$, where d is the dimensionality of the feature set, and the black box model M maps the input to an output $M(x) \in Y$. Denote $\mathscr{D} = \{(x^i, y^i)\}$ as the collection of all input-output pairs in the dataset. A post hoc explanation, denoted as g as an explanation mapping that for predictor $\mathbf{M}$ and point of interest x returns an importance score $\mathbf{g}(M,x) = \varphi_x \in \mathbb{R}^d$ for all features. The evaluation criterion μ is the mapping that takes predictor M, explainer g and the point of interest x as arguments and returns a scalar value for g.

2.2.1 LIME(Local Interpretable Model-Agnostic Explanations)

LIME is an XAI method that provides interpretable local explanations for black-box models by approximating the model's behaviour in the local region around a specific instance [115]. Given an instance $x \in \mathbb{R}^d$ and an explanation of the model $g \in G$ where G is a set of interpretable models (e.g.: Linear models). provides explanations $\varphi(x)$ obtained by below equation:

$$\varphi(x) = \underset{g \in G}{\operatorname{argmin}} \{\mathscr{L}(M, g, \omega_x) + \Omega(g)\} \tag{1}$$

In the case of a classification model M, ω_x is a proximity measure or weight between the actual instance and the new instances. A higher value of ω_x signifies a stronger similarity between the new and original instances. $\mathscr{L}$ is a loss function used to measure the proximity between the predictions made by the explanation model and the original model and $\Omega(g)$ quantifies the complexity of model g. Hence, LIME minimises $\mathscr{L}(M, g, \omega_x) + \Omega(g)$ to create a locally interpretable model, which is then used to predict the instance via the explanation model $\omega(x)$.

2.2.2 SHAP (SHapley Additive exPlanations)

This method is also another popular method to interpret the output of ML models [82]. It is based on the concept of Shapley values from cooperative game theory [123] and explains each feature's contribution to the model prediction [134].

SHAP is typically applied to tabular data and exhibits the following properties: local accuracy, handling of missing data, and consistency [81]. Local accuracy ensures that the explanation model matches the original model. Missingness ensures that missing features in the original input do not have an impact. Consistency, where increasing the impact of a feature in the model should also result in a higher SHAP value for that feature, regardless of other features.

SHAP creates simplified inputs z by mapping x to z through $x = h_x(z)$. The original model $M(x)$ can be approximated using binary variables with a linear function:

$$M(x) = g(z) = \varphi_0 + \sum_{i=1}^{d} \varphi_i Z_i \tag{2}$$

where $z = \{0,1\}^d$, d represents the number of input features, $\varphi_0 = M(h_x(0))$, and φ_i denotes the feature attribution value

$$\varphi_i = \sum_{S \in F \setminus \{i\}} \frac{|S|!(d-|S|!-1)!}{|d|!} [M_x(S \cup i) - M(S)] \tag{3}$$

$$M_x(S) = M(h_x^{-1}(z)) = E[[M(x)]|x_s] \tag{4}$$

Where F is the non-zero input set in z, S is the subset of F excluding the i^{th} feature from F, and φ_i is the SHAP value, a unified measure of additive feature attributions.

Since computing $E[[M(x)]|x_s]$ is particularly difficult, many approximation methods have been created, including Kernel SHAP, Deep SHAP, and Tree SHAP. In this dissertation, Tree SHAP was used when explaining the predictions of tree-based models (e.g., XGBoost, Random Forest), while Deep SHAP was utilized to explain predictions of deep learning models (e.g., LSTM, deep neural networks, Transformers).

2.2.3 DeepLift (Deep Learning Important FeaTures)

DeepLift is an additive feature attribution method that satisfies local precision and recursively explains the results of the deep learning black box model [129]. This method employs a linear composition technique to linearize the non-linear components of the black-box model [130].

DeepLIFT method quantifies the influence of an input neuron on the difference between the activation of a neuron corresponding to class c and the input data matrix x when compared to a reference input x'. This is represented as follows:

$$\Delta y_0 = y_{u0} - y'_{u0} \tag{5}$$

Thus, the deep learning model is effectively reverse-propagated. Here, y_{u0} represents the neural activation $u0$ of a specific layer with respect to the input data x. Similarly, y'_{u0} gives the neural activation for the respective reference input x'. The neurons in the previous layer are denoted as u_i, where $i = [1,n] \in N$. n indicating the total number of neurons in the layer under consideration. The ratio $C_{\Delta u_i, \Delta y_0}$ of a neuron u_i at Δy_0 is used in the following calculations.

$$\Delta y_i = y_{u_i} - y'_{u_i} \tag{6}$$

In this context, $C_{\Delta u_i, \Delta y_0}$ represents the ratio of the difference between the input values $u_1, \ldots, u_n$ of a neuron to the difference of the output value u_0 of a neuron. This formulation applies equally to both the original and reference datasets. For all n neurons, this can be expressed as follows:

$$\Delta y_0 = \sum_{i=1}^{n} C_{\Delta u_i, \Delta y_0} \tag{7}$$

Based on these interim results, the DeepLIFT method derives certain multipliers, defined as the ratio of $C_{\Delta u, \Delta y_0}$ to the difference Δu

$$m_{\Delta_u \Delta_{y_0}} = \frac{C_{\Delta_u \Delta_{y_0}}}{\Delta_u} \tag{8}$$

For a deep neural network, these individual multipliers are concatenated as follows:

$$m_{\Delta e_i \Delta y_0} = \sum_j m_{\Delta e_i \Delta y_j} \cdot m_{\Delta u_j \Delta y_0} \tag{9}$$

This formulation describes the impact of an input neuron $m_{\Delta e_i \Delta y_0}$ on the importance statement of DeepLIFT. Here, j represents an index that iterates across the hidden layer neurons connected to a specific input neuron. In this context, e_i denotes the input neurons, u_i refers to the hidden layer neurons, and u_c indicates the output neuron for a particular class of the deep neural network model.

However, for the commonly used Rectified Linear Units (ReLU) activation function, there is an exception to the chain rule for multipliers from equation 9. Therefore, in the DeepLift method, the rescale rule is applied. In this regard, Δf^+ and Δf^- are expressed as follows:

$$\Delta f^+ = \frac{\Delta f}{\Delta y} \Delta y^+ \quad \text{and} \quad \Delta f^- = \frac{\Delta f}{\Delta y} \Delta y^- \tag{10}$$

Taking Equation (8) into account, the multiplier can be written as:

$$m_{\Delta y^+ \Delta f^+} = \frac{\Delta f}{\Delta y} \tag{11}$$

Lundberg and Lee [82] have made modifications to the DeepLIFT method. They replaced the original estimation of a neuron's impact on the activation difference of another neuron with a procedure based on Shapley Values. As a result, they have named their new approach DeepSHAP. A description of SHAP can be found in Section 2.2.2. In this dissertation, TreeSHAP was used for tree-based models, while DeepSHAP was employed to explain deep learning models.

2.2.4 Integrated Gradients (IG)

Another XAI method we used in this research is Integrated Gradients (IG) [136], which determine the relevance R_i of the input variable x_i by approximating the integral below using the Riemann sum.

$$R_i = (x_i - \bar{x}_i) \cdot \int_0^1 \frac{\partial f_c\big(\bar{\mathbf{x}} + \alpha \cdot (\mathbf{x} - \bar{\mathbf{x}})\big)}{\partial x_i} \, d\alpha, \tag{12}$$

The variable $\bar{x}_i$ represents a baseline input that must be chosen when applying this method. The authors define this baseline input as indicating the absence of a feature input x[136]. The original authors applied a zero-value baseline to the image. In the experiments, this zero-valued baseline was used for the tabular data set in the publications of this dissertation.

2.2.5 Gradient $\times$ Input

Gradient $\times$ Input computes the importance of each feature by multiplying the model's partial derivative with respect to that feature by the input feature value [129]. The formula for Gradient * Input, denoted as $g(M,x)$, is given by:

$$g(M,x) = \frac{\partial M(x)}{\partial x} \times x \tag{13}$$

3 Federated Learning

This section presents background information on the fundamentals of federated learning

FL trains ML models on smart devices without sharing local data. Devices share parameters and gradients with a global model hosted on a server. The global model aggregates local model updates by averaging individual model parameters or gradients, allowing each model to learn collaboratively from the global model. This ensures that sensitive raw data remains securely stored on client devices.

Let N be the number of clients, denoted as $c_1, \ldots, c_N$. Each client c_i has a set of samples denoted by U_{c_i}, the feature space by X_{c_i}, the label space by Y_{c_i}, and the data set by $D_{c_i} = \{u_{c_i}^{(j)}, x_{c_i}^{(j)}, y_{c_i}^{(j)}\}_{j=1}^{|D_{c_i}|}$. Here, the data point $(u_{c_i}^{(j)}, x_{c_i}^{(j)}, y_{c_i}^{(j)})$ indicates that the client c_i owns the sample $u_{c_i}^{(j)}$ with features $x_{c_i}^{(j)}$ and a label $y_{c_i}^{(j)}$. For example, in IoT networks, U refers to the set of all IoT devices, X denotes features such as Network Traffic and Communication Protocols, and Y indicates whether an IoT device is infected with Malware (for example Dos and DDos) or not. Based on data partition (X, Y, U) among clients, FL can be categorised into three types by Yang et al. [154]:

- Horizontal Federated Learning (HFL): In HFL, clients share the same feature and label space ($X_i = X_j, Y_i = Y_j$) but have different sample spaces ($U_i \neq U_j$).
- Vertical Federated Learning (VFL): In VFL, some common data samples are shared among clients ($U_i \cap U_j \neq \emptyset$), but the space between features and labels differs ($X_i \neq X_j, Y_i \neq Y_j$).
- Federated Transfer Learning (FTL): FTL does not impose restrictions on the sample, feature, and label space, allowing arbitrary differences.

In addition, Federated learning can be categorised as cross-device FL (business-to-consumer) and cross-silo FL (business-to-business). Cross-device FL involves many participants with low computational resources, while cross-silo FL consists of companies or institutions with stronger communication and computational capacities. The client-server architecture is the most popular HFL architecture and the Federated Averaging (FedAvg) algorithm [91] is based on it. The optimisation objective of HFL is as follows:

$$\min_{\theta} L(\theta) = \sum_{i=1}^{N} L_{c_i}(\theta) = \sum_{i=1}^{N} \frac{1}{|D_{c_i}|} \sum_{j=1}^{D_{c_i}} l(x_{c_i}^{(j)}, y_{c_i}^{(j)}; \theta) \tag{14}$$

Where θ represents the model parameters, $L(\theta)$ is the global optimisation objective. $L_{c_i}(\theta) = \frac{1}{|D_{c_i}|} \sum_{j=1}^{D_{c_i}} l(x_{c_i}^{(j)}, y_{c_i}^{(j)}; \theta)$ is the client's optimisation objective c_i based on its local data D_{c_i}, with $l(x, y; \theta)$ denoting the loss function, such as cross-entropy or mean squared loss. SGD (stochastic gradient descent) is commonly used to optimise the equation 14. The HFL training process is as follows:

1. Each client samples a batch of data B, and calculates gradients $g_{c_i} = 1/|B| \nabla_\theta \sum_{x,y \in B} l(x, y, ; \theta)$ and sends g_{c_i} to a server.
2. The server aggregates the gradients (e.g. averaging $g = 1/N \sum_{i=1}^{N} g_{c_i}$) and then distributes the aggregated gradient to all clients.
3. Each client updates the model by gradient descent based on the aggregated gradient g.

The process is executed iteratively until the loss function $L(\theta)$ converges. After training finishes, all clients share the same model parameters θ.

HFL has a high communication overhead, especially in CDFL. FedAvg reduces communication costs by increasing local computation to converge the model faster in HFL. Instead of transmitting gradients for every batch, FedAvg transmits gradients for every $\mathscr{E}$ epoch. Assuming current model parameters $\theta^{(t)}$, batch size b, and learning rate η, the FedAvg algorithm follows these steps.

1. Each client divides the local data D_{c_i} into batches of size b.
2. For each batch B, the clients perform parameter updates $\theta_{c_i}^{(t)} \leftarrow \theta_{c_i}^{(t)} - \eta/b \sum_{x,y \in B} \nabla_\theta l(x, y; \theta_{c_i}^{(t)})$ until the local dataset D_{c_i} is iterated $\mathscr{E}$ epochs.
3. Each client sends the updated parameters $\theta_{c_i}^{t}$ to aggregation server.
4. The aggregation server computes a weighted average of the model parameters $\theta^{t+1} = \sum_{c_i=1}^{N} w_{c_i} \theta_{c_i}^{(t)}$, where w_{c_i} is calculated from the size of each local dataset
5. the aggregation server distributes $\theta^{(t+1)}$ to the clients.

4 Effective IoT Botnet and IoMT Attack Detection

This chapter addresses **RQ1**, which focuses on identifying the minimal subset of features that impact achieving high detection performance in ML models for IoT botnet detection and IoMT cyber attack detection. To accomplish this, we conduct an in-depth evaluation of feature selection methods using filter and wrapper techniques. The findings presented in this chapter highlight the identification of optimal features across various stages of the IoT botnet lifecycle, as well as for IoMT attack detection, with the goal of improving the performance of IDS models across different classification tasks. This work is contextualized in relation to Publication I and Publication II.

4.1 In-depth Feature Selection for efficient IoT Botnet Detection

The widespread use and weak security of IoT devices make them attractive targets for cyber attackers, allowing compromised devices to be added to a botnet. Major IoT botnets execute significant cyber attacks, such as spam campaigns and DDoS attacks, resulting in substantial financial losses for companies by disrupting their access to servers and services. Consequently, most research related to IoT botnets focuses on attack detection, which is a critical but often late stage in the botnet life cycle.

An IoT botnet is a specific type of botnet in which the devices involved are IoT devices, rather than traditional computers typical of regular botnets. Regardless of the type, all botnets go through a similar series of phases in their existence, which are collectively known as the botnet life cycle. In this context, the botnet lifecycle comprises four stages: formation, command and control (C&C), attack, and post-attack.

1. Formation phase: A device is compromised and infected by a master, making it part of a botnet controlled by a botmaster (also called Bot Herder, Attacker). This phase is known as spreading or injection.

2. The Command and Control (C&C): C&C channel is used by the bot-master to communicate with the bots. This channel can be executed using various protocols and applications, such as HTTP, P2P, or IRC. Commands are sent through this channel to instruct the bots on the actions they need to take, such as launching attacks.

3. Attack Phase: This phase occurs after the botnet receives an instruction and involves the execution of the attack by its members. The primary goal of a botnet is to carry out large-scale distributed attacks. This phase is also known as the application phase.

4. Post-Attack Phase: Following the attack and exposure to the defender, some bots may be removed from the infection (for example, if a vulnerability is patched). As a result, it becomes necessary to recruit new members to maintain or expand the size and capabilities of the botnet. To achieve this, scanning attacks are conducted. The newly recruited bots may be combined with both non-exposed bots and those that are still operational to form a new botnet. This new botnet will then receive instructions to carry out attacks through the command and control (C&C) channel, repeating the cycle.

The first three steps can be seen as the core components of the botnet lifecycle. The final step, however, restarts the formation process with the goal of addressing the challenges that emerged after the attack phase, thereby increasing the botnet population.

Due to the significant and detrimental consequences of IoT botnet-based attacks, much of the related research concentrates on attack detection as a crucial first step in mitigating these threats. In this context, Publication I examines the application of an in-depth feature selection model to improve the capabilities of botnet detection.

Feature selection is a crucial step in the ML workflow. By decreasing the dimensionality of the data and determining the most discriminative subset of features, we can improve classification performance while also minimizing computational needs [74]. This process avoids problems associated with the curse of dimensionality and improves the model's explainability by simplifying its complexity. Additionally, feature selection leads to faster training times and benefits to mitigate overhitting issues.

Feature selection methods are classified predominantly into filter, wrapper, and embedded [74]. Filter methods rank features using statistical methods, serving as an initial step before model building. Wrapper methods evaluate feature sets based on their effects on model performance. On the other hand, embedded methods combine feature selection with model training, which makes them more efficient than pure filter or wrapper methods [43].

Publication I analyzes the impact of filter and wrapper selection techniques on the detection accuracy of machine learning models used to identify IoT botnet attacks. Two feature selection types (Filter & Wrapper) were applied to two datasets: N-BaIoT and Med-BioT, to determine the impact of network category for IoT botnet prediction. These datasets include network traffic activities representing different stages of the botnet life cycle in IoT networks. The N-BaIoT dataset contains instances related to the Attack phase, while the Med-BioT dataset encompasses the post-attack and C&C phases. To identify the optimal subset of features, we utilized filter and wrapper feature selection methods for various classification tasks, as detailed in Table 2, which can be applied to detect IoT botnet attacks.

N-BaIoT [92] and Med-BIoT [41] datasets contain 115 and 100 features, respectively, extracted from network traffic generated by bots in controlled testing environments. Each data point reflects aggregated statistics of raw network streams across five time windows: 100 ms, 500 ms, 1.5 s, 10 s, and 1 min, coded as L5, L3, L1, L0.1, and L0.01. There are five main network traffic categories feature categories are: Host-IP (Traffic from a specific IP address). Host-MAC and IP (Traffic from the same MAC and IP), Channel(Traffic between specific hosts), Socket(Traffic between particular hosts and ports) and Network Jitter(Time intervals between packets in communication).

Four feature selection techniques were employed in the case of filter methods: Pearson Correlation, Fisher score [2], Mutual Information (MI) [36], and the ANOVA F-test.

- Pearson Correlation utilizes the linear correlation between features
- Fisher Score is based on the relationship between inter-class and intra-class separation.
- Mutual Information uses entropy to measure the amount of information that one variable contains about another.
- ANOVA F-test is based on the analysis of variance.

Unlike filter methods, wrapper methods are classifier-agnostic, selecting the best set of functions for a specific classifier. Classification algorithms are used to determine the significance of different features and evaluate their performance. The wrapped method assesses feature subsets based on predictive accuracy from test data. The feature set is constructed iteratively by adding features using forward selection or removing them using

Table 2: Classification problems addressed for IoT botnet detection using feature selection [57]

Dataset	Classification Task	Class Name	Description of the Class name
N-BaIoT	Binary	Benign	Legitimate Network Traffic
		Attack	Malicious Network Traffic (Mirai, Gafgyt)
	3-class	Mirai	Mirai malware-infected network traffic
		Benign	Legitimate Network Traffic
		Gafgyt	Gafgyt malware-infected network traffic
	9-class	ACK	Gafgyt malware Sending Spam data
		Benign	Legitimate Network Traffic
		COMBO	Gafgyt malware Sending spam data and opening a connection to IP, port
		JUNK	Mirai Malware ACK-Flooding
		SCAN	Scans the network devices for vulnerabilities,(Mirai &Gafgyt)
		SYN	Mirai Malware SYN-Flooding
		TCP	Gafgyt malware TCP Flooding
		UDP	UDP flooding (Mirai & Gafgyt)
		UPDPLAIN	Mirai malwar UDP flooding with Less of an option for higher packet per second
MedBIoT	Binary	Benign	Legitimate Network Traffic
		Attack	Malicious Network Traffic (Mirai, Bashlite, Torii)
	3-class	Benign	Legitimate Network Traffic
		C&C	network traffic for C&C
		Spread	Spread Attack network traffic
	4-class	Bashlite	Bashlite malware-infected network traffic
		Benign	Legitimate Network Traffic
		Mirai	Mirai malware-infected network traffic
		Torii	Torii malware-infected network traffic

backward elimination. Methods differ in evaluating feature significance, model performance criteria, and the number of features added or removed. Three feature selections from wrapped methods were employed: Recursive feature selection (RFE) [44], Sequential Backward selection (SBS) [67], and Sequential Forward selection (SFS) [67].

- RFE begins with a complete set of features and then removes the least relevant features one by one to identify the most significant ones.
- SBS starts with the entire feature set and iteratively removes the least important features until the model performance criteria are achieved.
- SFS starts with an empty feature set and adds more important features one by one until the model performance criteria are achieved.

The computational experiments followed a classical ML workflow. The initial datasets were large enough to provide balanced samples with respect to all features across malware types, attack types, and device types. After preprocessing, the dataset was split into

training and testing subsets in an 80/20 ratio. k-nearest neighbours (kNN), decision trees (DT), random forests (RF), and extremely randomized trees (ET) were used to evaluate feature sets.

Table. 3 & 4 show feature sets from filter and wrapper methods, respectively. The feature sets were selected based on top-performing features from both methods. All these feature sets provided the best performance of the F1-score for both N-BaIoT and MedBIoT datasets.

Table 3: Publication. I - Filter Method Feature Sets for the N-BaIoT and MedBIoT Dataset

Dataset	Classification Type	Pearson Correlation	Fisher Score	Mutual Information	ANOVA
N-BaIoT	Binary	33	5	3	3
	3-class	33	6	3	5
	9-class	33	68	28	59
MedBIoT	Binary	34	51	36	85
	3-class	34	42	38	49
	4-class	34	46	41	52

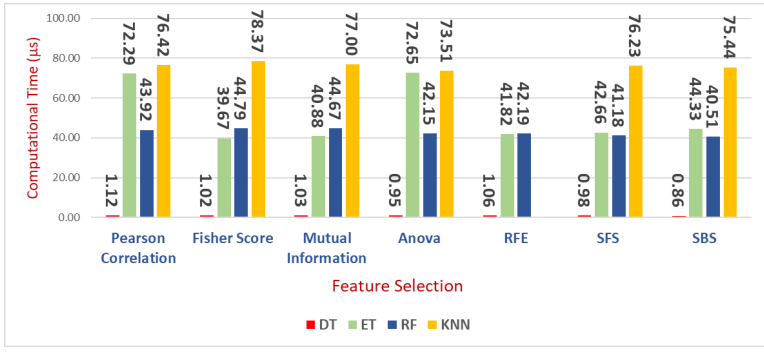
Table 4: Publication. I - Wrapper Methods Feature Sets for N-BaIoT and MedBIoT

Dataset	Classification Type	Wrapper Method	DT	RF	ET	KNN
N-BaIoT	Binary	RFE	3	4	4	
		SFS	3	3	3	3
		SBS	3	3	3	3
	3-class	RFE	3	4	4	
		SFS	3	3	3	3
		SBS	3	3	3	3
	9-class	RFE	28	23	25	
		SFS	3	3	3	3
		SBS	3	3	3	3
MedBIoT	Binary	RFE	29	27	24	
		SFS	7	7	7	7
		SBS	7	7	7	7
	3-class	RFE	26	27	24	
		SFS	7	7	7	7
		SBS	7	7	7	7
	4-class	RFE	29	24	22	
		SFS	7	7	7	7
		SBS	7	7	7	7

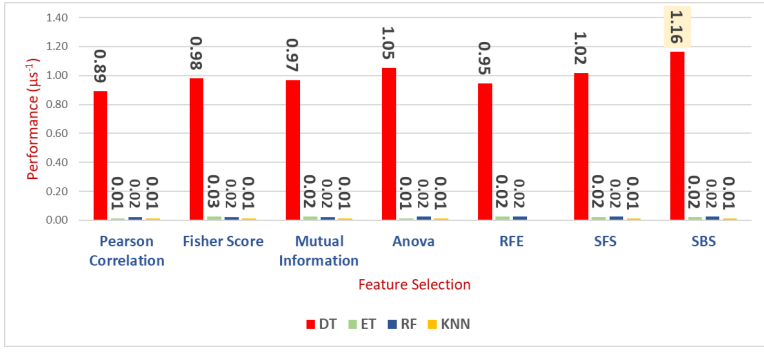
In the experiments, a three-step evaluation process was utilised to evaluate the distinct subsets of features from feature selection methods in both datasets. First, the F1 score metric was used to evaluate each set of features. Second, computational time was

Table 5: Publication. I - F1 scores for binary classification models using feature subsets (represented in Table 3 and 4) of feature selection algorithms in the N-BaIoT dataset

Binary N-BaIoT					
FS Method	**Approach**	**DT**	**ET**	**RF**	**KNN**
Filter	Pearson Correlation	0.9987	0.9983	0.9983	0.9957
	Fisher Score	0.9997	0.9997	1.0000	0.9847
	Mutual Information	0.9990	0.9990	0.9990	0.9977
	Anova	0.9973	0.9970	0.9970	0.9970
Wrapper	RFE	0.9983	0.9983	0.9987	
	SFS	0.9996	1.0000	0.9999	0.9994
	SBS	0.9998	0.9999	0.9997	0.9966



(a) Computational Time required to classify a sample by binary classification models on N-BaIoT dataset using feature sets (see in Table 3 & 4) of feature selection methods from Publicaton [57]



(b) Performance achieved by binary classification models over the N-BaIoT dataset using feature sets (see in Table 3 & 4) of feature selection methods from Publication [57]

Figure 2: Publication. I - Computational Time and Performance of feature sets using Filter and Wrapper feature selection for Binary classification models in N-BaIoT dataset

measured, which refers to the total time a computer with a specific processor takes to complete a task. Computational time was measured to calculate the cost of classifying a sample (only test samples). Lastly, the performance of the feature set was calculated by determining the ratio between the F1 score and the computational time. Figure 2 illustrates the computational time and performance of feature sets for binary classification

models in the N-BaIoT dataset.

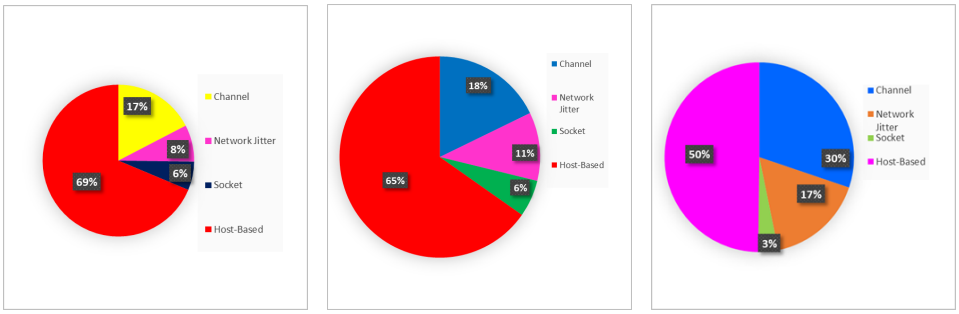
The computational time for the models, particularly the performance during testing, was calculated after feature selection using various filter and wrapper methods. Consequently, the time required for feature selection was not included, as testing time was considered more important than training time.

Table 5 shows the F1-scores for the performance of models based on the selected feature sets from both Filter and Wrapper methods. Figure 2a illustrates the computational time required to classify a simple using binary classification models on the N-BaIoT dataset using different feature sets. To choose the optimal feature set and best model, we evaluated their performance by calculating the ratio between the F1 score and the computational time. Figure 2b shows the performance achieved by binary classification models on the N-BaIoT dataset using feature sets from the feature selection methods detailed in Tables 3 and 4.DT with SBS (SBS-DT) achieved the highest performance metric using three features for binary classification in the N-BaIoT dataset, as illustrated in Figure 2b.

After identifying the optimal feature subsets for binary classification, we performed a frequency analysis to determine which feature categories and time windows were most utilized in feature sets, as shown in Figure 3. Host-based features were crucial for distinguishing malicious from benign traffic, while network jitter and socket features were less significant in binary classification (see in Figure. 3a).

The feature set with three features from the SBS-DT model performed similarly to binary classification by demonstrating a lower computation time for classifying test samples. It achieved the best performance, measured by the ratio of the F1-score to computational time, for both 3-class and 9-class classification tasks in the N-BaIoT dataset.

Figure 3 presents a frequency analysis of the feature category distributions from selected feature sets using both Filter and Wrapper methods. It illustrates that in both the 3-class (see in Figure 3b) and 9-class (see in Figure 3c) classification tasks on the N-BaIoT dataset, host-based features play a crucial role in predicting botnet types for the 3-class models and identifying botnet attacks for the 9-class models.



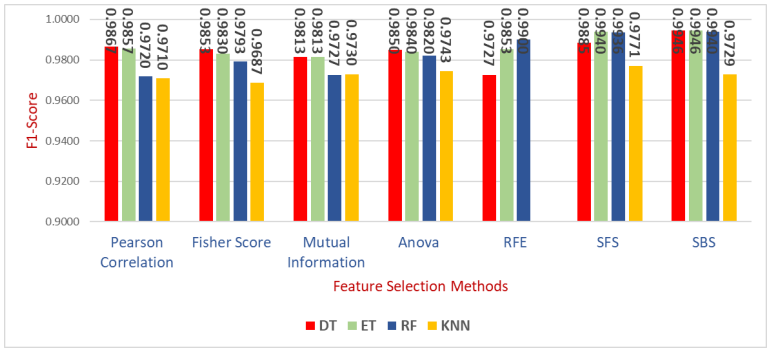
(a) Network Category-wise feature contribution for Binary classification in N-BaIoT dataset.

(b) Network Category-wise feature contribution for 3-class classification in N-BaIoT dataset.

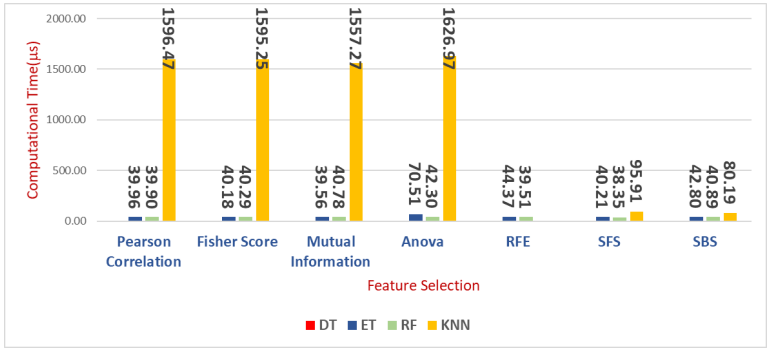
(c) Network Category-wise feature contribution for 9-class classification in N-BaIoT dataset.

Figure 3: Publication. I - Contribution of Network Category-wise features for Binary, 3-class, and 9-class classification in the N-BaIoT dataset

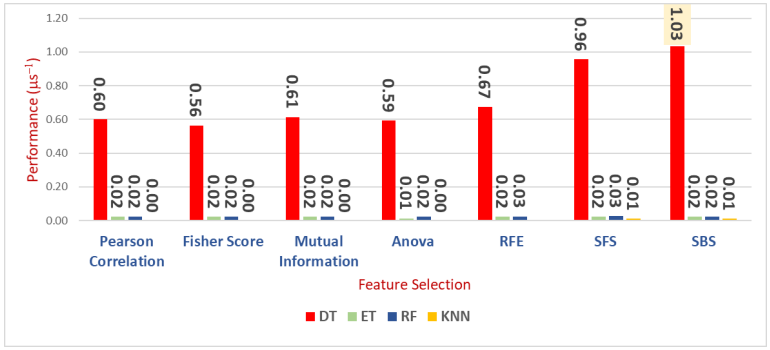
MedBIoT dataset contains malicious network traffic from the Mirai, BashLite, and Torii botnet malware, which were deployed on 83 real or emulated IoT devices. The experimental results of the binary, 3-class, and 4-class classification models generated using this dataset are detailed in Table 2, which outlines the classification formulations. Feature sets for filter methods and wrapper methods are presented in Table 3 and Table 4,



(a) F1-scores for binary classification models in the MedBIoT dataset using feature subsets (see in Table 3 & 4) of feature selection algorithms from Publication [57].



(b) Computational time required to classify a sample by binary classification models over the MedBIoT dataset using feature sets (see in Table 3 & 4) of feature selection methods from Publication [57].



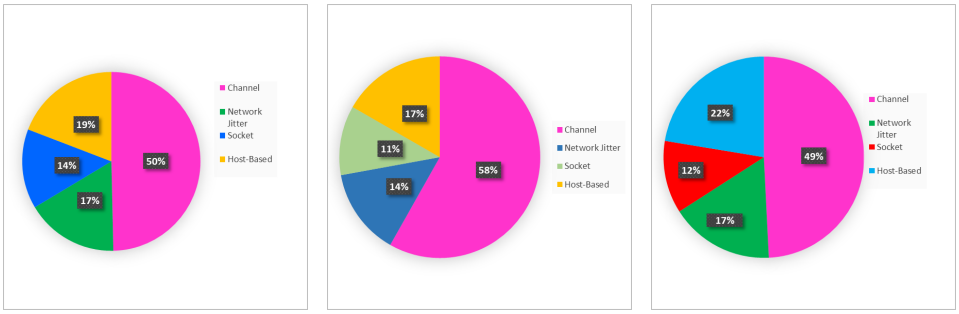
(c) Performance achieved by binary classification models over the MedBIoT dataset using feature sets (see in Table 3 & 4) of feature selection methods from Publication [57].

Figure 4: Publication. I - F1-score of models, Computational time and performance of feature sets using filter and wrapper feature selection for binary classification models in the MedBIoT dataset from Publication

respectively, for all classification problems in the MedBIoT dataset.

Figure. 4a illustrates the F1-score performance of various models (DT, RF, KNN, ET) using feature sets derived from both feature selection methods for a binary classification task on the MedBIoTdataset. Figure. 4b displays the computational time required to classify samples from models using feature sets, while Figure. 4c presents the performance

evaluation by showing the ratio between the F1 score and computational time. SBT-DT feature set, consisting of seven features, achieved the best performance among all feature sets for binary classification on MedBIoT dataset. Likewise, in the 3-class and 4-class classification tasks, the SBS-DT feature set with seven features also demonstrated the highest performance.



(a) Network Category-wise feature contribution for Binary classification in MedBIoT dataset.

(b) Network Category-wise feature contribution for 3-class classification in MedBIoT dataset.

(c) Network Category-wise feature contribution for 4-class classification in MedBIoT dataset.

Figure 5: Publication. I - Contribution of Network Category-wise features for Binary, 3-class, and 4-class classification in the MedBIoT dataset from Publication

Figure. 5 shows the frequency analysis of feature categories for classification tasks in the MedBIoT dataset. For binary classification (see Figure. 5a), the channel category is more dominant than the host-based category. MedBIoT focuses on detecting malicious activities during the C&C and formation phases of the botnet lifecycle, emphasizing the significance of host-to-host communication features. In contrast, N-BaIoT targets the attack phase and identifies malicious activities using host-based features. For the 3-class (in Figure. 5b) and 9-class (in Figure. 5c) classification tasks, channel-based features were more useful than other network categories in achieving the highest performance. Compared to binary classification, the prevalence of channel features were more frequent.

Using feature selection approaches for IoT botnet detection, tree-based models such as DT, ET, and RF achieved the best results for all classification tasks across both datasets, particularly in multiclass classification scenarios. In contrast, the k-NN classifier was unsuitable for multiclass classification and required significantly more computational time to classify samples compared to the tree-based models. A key finding from our analysis is that host-based features were more influential for the N-BaIoT dataset, while channel-based features were better at distinguishing activities in the MedBIoT dataset. This dataset focuses on the spread and C&C activities of IoT malware, making it essential to track communications between network nodes to differentiate malicious actions from benign ones and identify the type of threat.

4.2 Comprehensive Feature Selection for IoMT Intrusion Detection

IoT is an emerging technology in industry that has introduced the Internet of Medical Things (IoMT), significantly transforming healthcare. IoMT-enabled healthcare applications are gaining significant attention. The integration of sensor technology enables remote health monitoring through various wearable sensors, such as blood pressure, temperature, and pulse rate. Security of IoMT is vital in healthcare, as interconnected devices handle sensitive patient data. These devices are often targeted by cyberattacks, threaten-

ing patient safety and data privacy. IDS are critical for monitoring and identifying malicious activities to protect these networks. ML is essential for IDS in the IoMT, as it can recognize complex attack patterns and adapt to changing threats. In IoMT networks, Data is sent to local gateway devices for processing and feature selection, then transferred to remote devices for disease diagnosis or prediction using ML/DL models.

Publication II focuses on analyzing the impact of feature selection methods from filter methods on the intrusion detection of ML models for IoMT attack detection. Similar to Publication I, Publication II also involved a ML workflow that consisted of data preprocessing, feature selection, and model training/testing. The datasets used for Publication II were the CICIoMT2024 [31] and IoT-traffic data [11], which focus on IoMT devices in the healthcare sector. These datasets are designed to evaluate and improve the cybersecurity of IoMT devices through ID.

CICIoMT2024 dataset [31] includes traffic generated from 40 devices (25 actual and 15 simulated) across multiple protocols such as Wi-Fi, MQTT, and Bluetooth. 18 cyberattacks were simulated, which are categorized into five main categories: DDoS, DoS, Recon, MQTT, and Spoofing. In the CICIoMT2024 dataset, three types of classifications were developed: Binary, category-based (6-class), and attack-based (19-class) classifications.

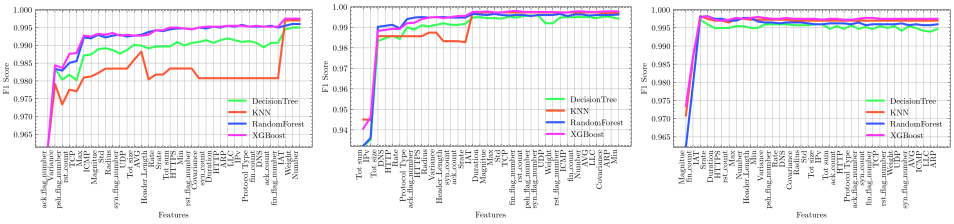
- Binary classification was utilized to distinguish between benign and attack traffic.
- In the category-based classification, six categories of network traffic were identified: benign, MQTT attacks, DDoS, DoS, reconnaissance, and ARP spoofing attacks.
- In the attack-based classification, there were 19 distinct classes, encompassing various types of attacks including Normal traffic such as ARP spoofing, ping sweep scan, reconnaissance, vulnerability scan, OS scan, port scan, malformed data packets, and different forms of denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks, including connect flood (DoS), publish flood (DDoS and DoS), TCP (DoS and DDoS), ICMP (DoS and DDoS), UDP (DoS and DDoS), and SYN (DoS and DDoS).

IoMT-TrafficData dataset [11] is a collection of network traffic data featuring both benign and malicious traffic from eight types of cyberattacks. Two types of classifications were developed from IoMT-TrafficData.

- Binary classification was developed to distinguish between benign traffic and attack traffic.
- Multi-class classification was used to classify the attack traffic with Normal Traffic: DoS, DDoS, ARP spoofing, CAM table overflow, MQTT malware, network scanning, Bluetooth reconnaissance, and Bluetooth injection.

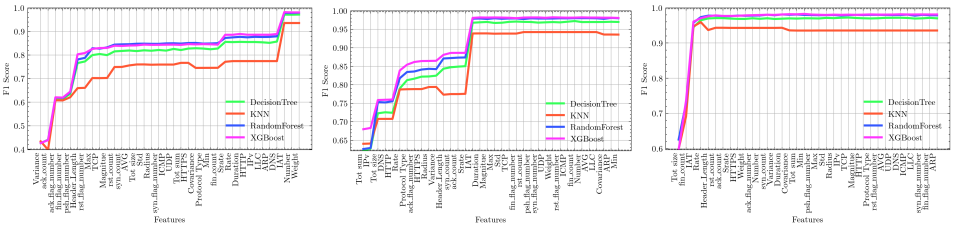
In the ML for IoMT attack detection, Pearson's linear correlation coefficient was utilized as a data preprocessing step to eliminate redundant and irrelevant features. Any feature that was highly correlated with another feature ($|r| > 0.80$) was removed, retaining only one of the correlated features. As a result, out of the initial set of 44 features used to describe each sample in the dataset, 36 features remained in the final feature set of the ciciIoMT dataset, while 21 features were removed from the IoMT traffic dataset.

In the feature selection step, three filter methods were used: Fisher score [2], Mutual Information (MI) [36], and Information Gain(IG) [63]. An iterative, stepwise approach was employed to evaluate each method by training ML model. Starting with the highest-ranked feature, added one feature at a time to the training set. For instance, if the features were ranked as $f = \{f_1, f_2, \ldots, f_n\}$, the model was first trained with $\{f_1\}$, then



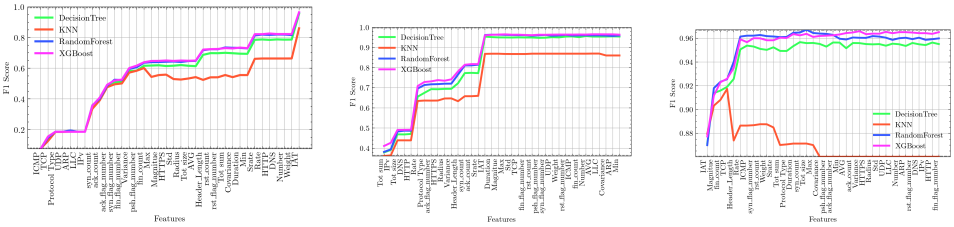
(a) Fisher Score *(b) Mutual Information* *(c) Information Gain*

Figure 6: Publication. II - Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for Binary Classification.



(a) Fisher Score *(b) Mutual Information* *(c) Information Gain*

Figure 7: Publication. II - Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for 6-class Classification.

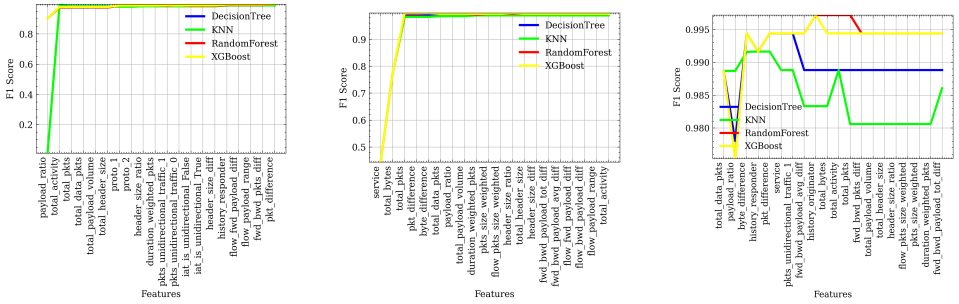


(a) Fisher Score *(b) Mutual Information* *(c) Information Gain*

Figure 8: Publication. II - Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for 19-class Classification.

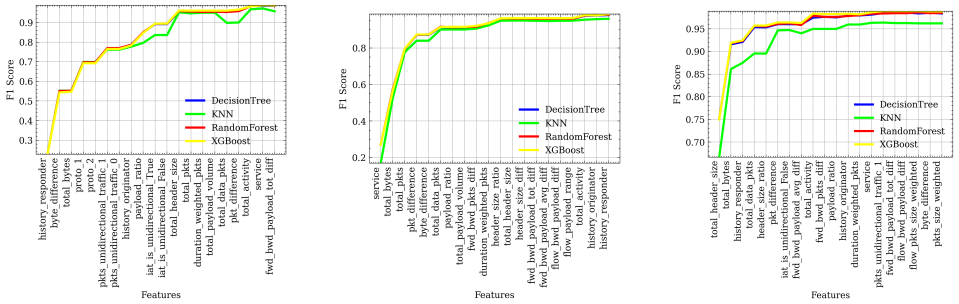
with $\{f_1, f_2\}$, and so on up to $\{f_1, f_2, \ldots, f_n\}$. This process was repeated for all n ranked features in each method across both datasets to evaluate the impact of each feature on model performance. F1-score was used to evaluate feature sets with four classification algorithms: Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), and XGBoost (XGB).

Figure 6 illustrates the performance of various feature selection methods on the CICIoMT2024 dataset for binary classification. Classifiers showed significant improvement with initial features, with most achieving high F1 scores (over 0.99) using just 5 to 10 features. Notably, XGB and RF achieved near-optimal performance using fewer than five features. In both 6-class (Figure 7) and 19-class (Figure 8) classifications as well, the performance was nearly the same compared to binary classification. XGB and RF consistently performed the best, especially when a limited number of features were utilized.



(a) Fisher Score Features *(b) Mutual Information* *(c) Information Gain*

Figure 9: Publication. II - Comparison of algorithms performance using Feature selection methods over IoMT-TrafficData dataset for Binary Classification.



(a) Fisher Score *(b) Mutual Information* *(c) Information Gain*

Figure 10: Publication. II - Comparison of algorithms performance using Feature selection methods over IoMT-TrafficData dataset for Multi-class classification (9 classes).

Figure 9 compares the performance of the algorithm using Filter methods feature selection methods on the IoMT-TrafficData dataset for binary classification. Mutual Information demonstrates significant performance improvements with the initial features, while Information Gain effectively ranks features. Additionally, it achieved the best performance with a limited number of features when using the XGB model compared to other classifiers.

Figure 10 illustrates the performance comparison of feature sets for a 9-class classification (multiclass classification) in IoMT traffic data. The performance of the models gradually improved with a higher number of features when using the Fisher Score. In contrast, the models achieved higher performance earlier (after four features) when utilizing MI & IG. For multiclass classification in IoMT traffic data, the performance was comparable across different feature selection methods, except for KNN, which showed lower performance when using the IG feature selection method.

By examining the important features from feature selection in both datasets, the key network characteristics for attack detection in IoMT traffic were identified. Publication II demonstrates that 3 to 4 features can achieve optimal F1-score and accuracy in binary classification for IoMT attack detection, whereas 7 to 8 features provided reasonable performance in most multi-class classification tasks across both datasets. Features from Information Gain using XGBoost with 15 features achieved excellent results in both binary and multi-class classification settings. Protocol types, traffic metrics, temporal patterns,

and statistical measures were identified as essential network characteristics for accurate IoMT attack classification.

4.3 Chapter Discussions

This chapter discusses the evaluation of feature selection methods to improve model performance in detecting IoT botnet and IoMT attacks. The response to research question RQ1 was provided in this section. The research question aims to identify which minimal subsets of features are most influential in achieving high detection performance in both IoT botnet and healthcare IoMT cyber attack detection.

Previous academic studies have utilized feature selection techniques to enhance the detection scores of existing ML classifiers. However, these studies have not examined the impact of feature selection methods on various binary and multi-class classification formulations for intrusion detection throughout different stages of the botnet life cycle. Specifically, the set of features that is effective for detecting malicious traffic at one stage may not be suitable at another. In Publication I, a reduced set of features was proposed to detect and classify some of the most popular botnet malware with high F1-score metrics. Various classification studies (Binary and Multi-class) have been proposed that are related to the IoT botnet lifecycle. In Publication I on botnet detection, both filter and wrapper methods achieve high detection rates with a limited number of features. Wrapper methods provide optimal feature sets, while filter methods perform less effectively. Especially, High performance is achieved with more than 28 features using filter methods for the 9-class classification of the N-BaIoT dataset and for all classifications using the MedBIoT dataset. In contrast, the wrapper methods, specifically SFS and SBS, identify optimal sets of 3 features for the N-BaIoT dataset and 7 features for the MedBIoT dataset, applicable to their respective binary and multi-class classifications. Channel-based features are preferred for post-attack detection and C&C stages, whereas host-based features are better for identifying bot attacks from originating bots. The approach outlined in this study is more efficient in terms of time and achieves comparable or better performance than other methods used for classifying these botnet malware.

Regarding IoMT attack detection, Publication II addresses a significant research gap by conducting a comprehensive cross-dataset feature selection analysis for ID in healthcare IoMT networks, an area that has rarely been explored in the existing literature. Publication II aims to identify key features across both binary and multiclass classification tasks using two well-established IoMT datasets in IDS tasks. Publication II reveals that 3 to 4 features are sufficient to achieve optimal F1 scores and accuracy in binary classification, while 7 to 8 features perform well for multi-class classification. Additionally, features selected using Information Gain with XGBoost (15 features) demonstrated excellent performance for IoMT attack detection.

5 Explainable AI for Transparent IoT Botnet Detection

This chapter describes the dissertation's contribution towards achieving the transparency of ML-based IDS models in centralized settings, addressing **RQ2**. It specifically explores the integration, and quantitative evaluation of post-hoc Explainable AI (XAI) methods for both machine learning and deep learning models used in IDS for IoT botnet detection problems. Publication III explores the quantitative evaluation of explainable AI methods applied to statistical machine learning models for detecting IoT botnets. Publication IV examines the evaluation of explainable methods used in deep learning-based detection of IoT botnet attacks.

5.1 Quantitative Evaluation of explainability for IoT Bot Net Detection

A botnet is a network of Internet-connected devices controlled by a single entity, the botmaster. They can be used for malicious activities like DDoS attacks, spamming, and cryptocurrency mining. Due to the proliferation of IoT devices, these botnets can be much larger than traditional ones, increasing the threat to Internet security.

Detecting IoT botnets is challenging due to their distributed nature and diverse infected devices. Traditional methods, like signature-based and anomaly-based approaches, are inadequate since attackers can quickly change their behaviour to elude detection. Machine learning (ML) has become an effective tool for detecting IoT botnets over the past two decades. It is capable of processing large datasets and identifying complex patterns. However, despite the impressive achievements of ML models in the field of cybersecurity, their black-box nature presents challenges. The lack of transparency can hinder security experts' trust in ML models, as they may not fully understand how these models make specific decisions or classifications.

Post hoc explainability methods, which are model-agnostic, have gained popularity in research due to their broad applicability. Post hoc explainability methods are designed to provide reasoning behind individual predictions for predicted class labels in classification, which can help increase confidence in the outcomes. However, the quality of these explainability methods has raised concerns, as they rely on additional tools (such as linear models at the point of interest or game theory concepts to identify importance scores), which may introduce errors in the explanations [77]. As a result, many current XAI methods can produce unreliable explanations in real-world scenarios due to insufficient qualitative evaluation [29]. Addressing these issues is essential for IDS in IoT networks, particularly those that might expose sensitive device activity patterns. As XAI-based solutions for IDS grow in usage, it is crucial to evaluate their explainability components. This evaluation ensures that the XAI methods can be effectively implemented in real-world scenarios and enhance understanding of model outputs.

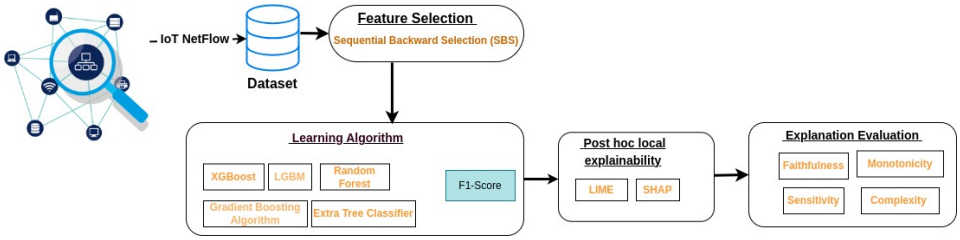


Figure 11: Quantitative Evaluation of Explainability for IoT botnet detection Framework [55]

Publication III highlights the crucial role of evaluating the quality of explainability

through quantitative evaluation, alongside the model's detection performance, in the context of IoT botnet detection. Figure 11 presents the framework developed for Publication III, which focuses on the quantitative evaluation of explainability for IoT botnet detection. In this framework, Sequential Backward Selection (SBS) was employed for feature selection across three IoT botnet datasets, and then learning models were created. The datasets utilized in this research for evaluating the framework include N-BaIoT[92], MedBIoT[41], and the BoT-IoT [66] network traffic datasets. A brief description of the N-BaIoT and MedBioT datasets is available in Chapter 4.

The Bot IoT dataset was generated using Ostinato, a tool that simulates realistic network traffic. The dataset comprises five IoT scenarios: a weather station, smart refrigerator, motion-activated lights, remotely activated garbage door, and smart thermostat. The dataset includes network traffic data showcasing various attacks, such as UDP, TCP, OS fingerprinting, service scans, HTTP attacks, keylogging, and data exfiltration. According to the source paper of the dataset, the top ten features were selected from the BoT-IoT dataset.

In Publication III, a quantitative evaluation of explainability was conducted for both binary and multiclass models. For botnet detection using binary classification, the study utilized the N-BaIoT, MedBIoT, and BotIoT datasets to distinguish between normal and malware traffic. Several classification algorithms were tested, including Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LGBM), Gradient Boosting Classifier (GBC), Random Forests (RF), and Extremely Randomized Trees (ET). To explain the black-box nature of these models, two popular post-hoc explainable AI methods, LIME and SHAP, were utilized.

5.1.1 Feature Selection

Sequential Backward Selection was applied using classification algorithms during the feature selection step. This step resulted in selecting the following feature counts: F1-scores exceeding 99% were achieved for three datasets. Specifically, there were three optimal k-feature subsets for the N-BaIoT dataset, seven optimal k-feature subsets for the MedBIoT dataset, and four optimal k-feature subsets for the BoT IoT dataset in binary classification.

5.1.2 Post-hoc Explainability for IoT botnet detection Models

After training the models, two model-agnostic feature importance XAI methods were used to explain the outcomes of black-box models: 1) LIME and 2) SHAP. A description of the LIME and SHAP XAI methods can be found in Chapter 2.

The LIME method explains the reasoning behind assigning probabilities to each class by comparing these probability values with the actual class of a given data point. To illustrate the local explanations provided by these explainers, a single actual malware data point was selected (see in Table 6) from each of the three network categories: Host-based, network jitter, and socket-based. The optimal features obtained from the SBS-XGB method were statistical features of host-based network traffic. Features derived from the SBS-LGBM method were based on network jitter, while features obtained from the SBS-GBC method were related to socket-based traffic.

LIME explanations for the selected data points in the network categories are presented in Figure 12. In the subfigures of Figure 12, the green bars represent the features that contribute to classifying a data point as malware, while the red bars indicate the features that contribute to classifying a data point as benign. Figure 12 presents the LIME local explanations for the N-BaIoT dataset. For instance, in Figure 12a, the LIME explanations for the features in the Host network category indicate that the XGB model accurately pre-

Table 6: Actual Data Points of N-BaIoT Dataset Across Three Network Categories[55]

Network Category	Model	Features	Data points
Host based	XGB	MI_dir_L5_weight	108.087471
		MI_dir_L1_weight	266.816511
		MI_dir_L0.01_weight	5843.19767
Net work jitter	LGBM	HH_jit_L5_mean	1.50591E+09
		HH_jit_L0.1_weight	1.00000E+00
		HH_jit_L0.1_mean	1.50591E+09
Socket Based	GBC	HH_L0.01_radius	4.00513E+01
		HH_L0.01_weight	3.73916E+04
		HH_L1_mean	5.53999E+02

dicted the malware class with 100% accuracy for the actual class label of the datapoint (mentioned in Table 6).

The following explanations include:

- 102.43 < MI_dir_L0.01_weight <= 19356.77: If the packet count for the host-based (MI) feature within a 1-minute window (L0.01) is between 102.43 and 19356.77, the XGB model is more likely to classify it as malware, indicating a potential compromise of the host.

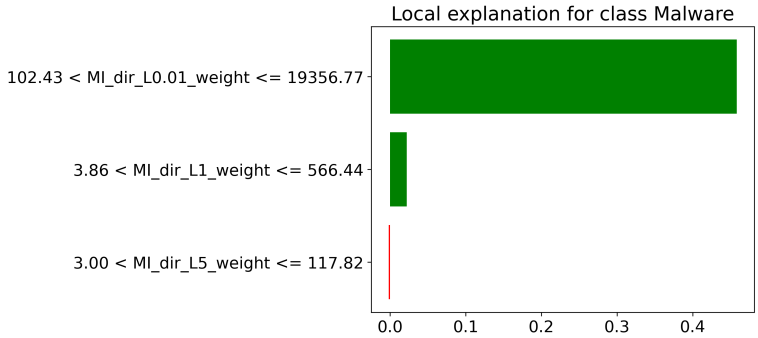
- 3.86 < MI_dir_L1_weight <= 566.44: If the packet count (weight) of the host-based (MI) feature in a 1.5-second time window (L1) is between 3.86 and 566.44, the XGB model indicates a potential IoT botnet malware presence.

- 3.00 < MI_dir_L5_weight <= 117.82 : If the packet count for the host-based feature exceeds 117.28 within 100 microseconds, the XGB model classifies it as benign, suggesting no malware is present. This indicates normal, non-malicious network traffic in very short time intervals.

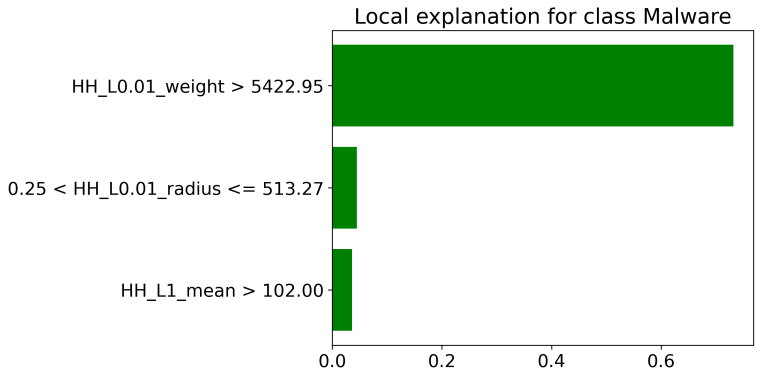
Like wise, LIME explanations for a data-point in Table 6 are shown in Figure 12b and 12c for different network categories. The GBC model predicts malware for the socket-based feature set, while the LGBM model predicts malware for network jitter features in the N-BaIoT dataset.

Another method used in Publication III for model explanations was SHAP (SHapley Additive exPlanations). SHAP is commonly employed to provide both local and global explanations. In local explanations, a specific data point is chosen, and the model's prediction is clarified by highlighting the contribution of each feature. SHAP computes Shapley values to reveal the contribution of features to model predictions. Figure 13 provides local explanations for a malware data point (see Table 6) from the N-BaIoT data set. SHAP force plots were used to show local explanations for the XGB, GBC, and LGBM models, emphasizing the contribution of each feature to the predictions.

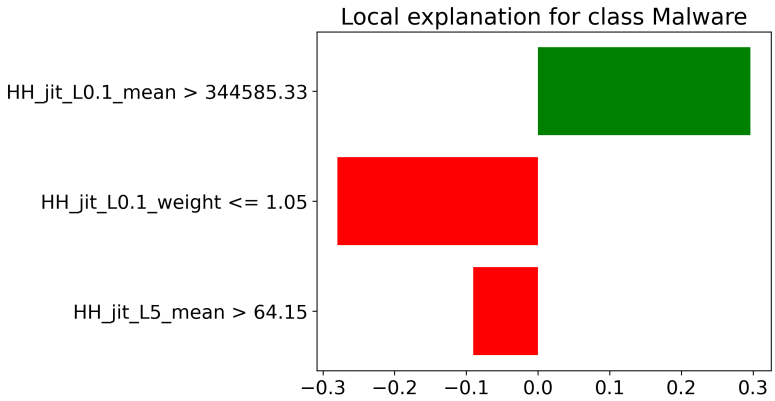
The plot displays the base value, with features that have a positive influence on the prediction shown in red and those with a negative influence shown in blue. The base value represents the average of all prediction values. Each strip in the plot illustrates how the features affect the predicted value, either bringing it closer to or pushing it further away from the base value. Red strips indicate features that increase the predicted value, while



(a) LIME Explanations for XGB model over Host-Based category features



(b) LIME Explanations for GBC model over socket features



(c) LIME Explanations for LGBM over Network-jitter category features

Figure 12: LIME Local explanations of Malware instance of N-BaIoT dataset for Binary Classifiers [55]

blue strips indicate features that decrease it. Features with wider strips contribute more significantly to the prediction.

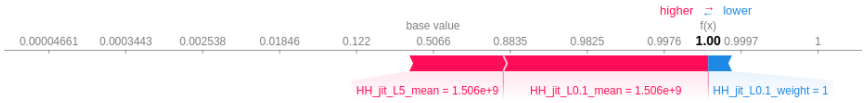
Figure 12c shows the SHAP local explanations for the LGBM model related to network jitter features for a specific data point (see Table III) in the N-BaIoT dataset. The base value is 0.566. The features HH_jit_L0.1_mean and HH_jit_L5_mean positively contribute to the predicted value, while HH_jit_L0.1_weight negatively impacts it. HH_jit_L0.1_mean is the



(a) SHAP Explanations for XGB model over data point of Host-Based category features



(b) SHAP Explanations for GBC model over data point of socket category features



(c) SHAP Explanations for LGBM model over data point of Network-jitter

Figure 13: SHAP Local explanations of Malware instance of N-BaIoT dataset for Binary Classifiers [55]

most significant feature due to its greater contribution. The total positive contributions exceed the negative ones, resulting in a final predicted value higher than the base value, leading to the prediction of malware.

Similarly, In Figure. 13a, XGB model using host-based features shows a base value of 0.4932, with a predicted value of 1 for the malware data point listed in Table III. Among the features analyzed, MI_dir_L0.01_weight exhibits the broadest range and is identified as the most important feature. In Figure 13b, the GBC model utilizing socket features reveals a base value of 0.5979. In this case, HH_L0.01_weight is the most significant feature, contributing to a predicted value of 1 for the malware class within the N-BaIoT dataset.

In response to the growing demand for objective evaluation of XAI methods, Research field has shifted toward developing quantitative metrics to assess the quality and reliability of explanations. Various metrics have been introduced to evaluate the effectiveness of these explainability methods. XAI evaluation is divided into three groups: user-focused, application-focused, and functionality-focused evaluations [29]. The first two categories are part of the human-centred evaluation and consist of both subjective and objective measures.

To evaluate the explanations provided by LIME and SHAP for botnet prediction models in publication 3, this study employed four metrics: high faithfulness, monotonicity, low complexity, and maximum sensitivity. These criteria were deemed suitable for assessing the local explanations generated by LIME and SHAP. The following metrics were used to evaluate explainers and provide a detailed description of each metric.

5.1.3 Faithfulness

The xai algorithm $\mathbf{g}$ should imitate the model's behaviour. $\mathbf{g}(M, \mathbf{x}) \approx M(\mathbf{x})$. Faithfulness criteria measure the consistency between the prediction model M and explanation g. The Faithfulness correlation [19] and Monotonocity [83] metrics were used to evaluate the Faithfulness of explanations.

Faithfulness metric $\mu_F(M, g; x)$ calculates how well the feature importance scores rendered by the explanation function $\mathbf{g}$ reflect the actual importance of the features in the black-box model $\mathbf{M}$ for input x. This property is best calculated using Pearson's correla-

tion coefficient between the sum of feature attributions assigned to the optimal line value and the corresponding difference in the output values. Let B be the set of indices where features are set to a baseline value. The faithfulness metric is then computed as follows:

$$\mu_F(M,g;x) = \underset{B\in\binom{|d|}{|B|}}{\rho}\left(\sum_{i\in B} g(M,x)_i, M(x) - M(x_B)\right) \tag{15}$$

where $x_B = x_i | i \in B\}$, Zero base line values were used.

5.1.4 Monotonicity

Let $x, x' \in R^d$ be two input points such that $x_i \leq x'_i$ for all $i \in 1,2,\ldots,d$. M and g are said to be monotonic if the following condition holds: For any subset $S \subseteq 1,2,\ldots,d$, the sum of the attributions of the features in S should be nonnegative when moving from x to x', that is,

$$\sum_{i\in S} g(M,x)i \leq \sum_{i\in S} g(M,x')_i$$

imply

$$M(x) - M(x_{[x_s=\bar{x}_s]}) \leq M(x') - M(x'[x'_s = \bar{x}_s])$$

5.1.5 Low Complexity

Explanations that use fewer features are preferred. It is assumed that explanations involving many features are harder for users to understand. $\min \|\mathbf{g}(M,\mathbf{x})\|_0$

Low complexity [19] metric calculates the entropy of each feature's fractional contribution to the total attribution magnitude individually. A complex explanation utilizes all d features to identify which aspects of x are important to the model M. However, this approach can be less interpretable, particularly when d is large. To improve clarity, we define a fractional contribution distribution, where $|\cdot|$ represents the absolute value.

$$P_g(i) = \frac{|g(M,x)_i|}{\sum\limits_{j\in|d|}|g(M,x)_j|}; \quad P_g = \{P_g(1),\ldots,P_g(d)\} \tag{16}$$

Note that P_g represents a valid probability distribution. Let $P_g(i)$ denote the fractional contribution of the feature x_i to the total magnitude of the attribution. If every feature had the same attribution value, the explanation would be complex, even if it remained faithful. In contrast, the simplest explanation would focus on a single feature. Complexity is defined as the entropy of P_g.

Given a prediction $M(x)$, an explanation function g, and a point x, the complexity of g at x is:

$$\mu_C(M,g;x) = -\sum_{i=1}^{d} P_g(i)\log P_g(i) \tag{17}$$

5.1.6 Maximum Sensitivity

Robustness indicates that similar inputs should produce similar explanations. $\mathbf{g}(M,\mathbf{x}) \approx \mathbf{g}(M,\mathbf{x}+\varepsilon)$ for small ε. Maximum Sensitivity was used to derive the robustness of explanations.

Maximum Sensitivity [19] is used to ensure that nearby inputs with similar model output have similar explanations, it is desirable for the explanation function g to have a low

sensitivity in the region surrounding the point of interest x, assuming the differentiability of the predictor function M. Maximum sensitivity of an explanation function g at a point of interest x in its neighbourhood is defined as follows: Consider a neighbourhood N_r of points within a radius r of x, denoted by $N_r = z \in D_x | p(x,z) \leq r, M(x) = M(x)(z)$, where D is the distance metric, and p is the proximity function. Given a predictor $M(x)$, a distance metric D, a proximity function p, a radius r, and a point x, we define the maximum sensitivity of g at x as follows:

$$\mu_M(M, g, r; x) = \max_{z \in N_r} D\left(g(M(x), x), g(M(x), z) \right) \tag{18}$$

Table 7: Results of Evaluating the quality of LIME&SHAP using Faithfulness (μ_f), Monotonicity (μ_m), Complexity (μ_c), Sensitivity(μ_s) for IoT malware detection binary classification models over three datasets: N-BaIoT, MedBIoT, BoT-IoT.

Dataset	Model	ET		LGBM		GBC		RF		XGB	
	XAI Metric\XAI method	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP
N-BaIoT	μ_f	0.28 ±0.75	0.34±0.73	0.65 ±0.65	0.84 ±0.21	0.46 ±0.67	0.48 ±0.67	0.36 ±0.65	0.91 ±0.23	0.77 ±0.57	**0.99 ±0.13**
	μ_m	94.80%	95.38%	91.55%	99.25%	57.33%	73.08%	75.23%	79.33%	97.70%	**98.15%**
	μ_c	0.97 ±0.13	0.80 ±0.21	0.90 ±0.24	0.99 ±0.16	1.14 ±0.14	0.85 ±0.23	0.97 ±0.11	0.91 ±0.14	0.72 ±0.24	**0.50 ±0.25**
	μ_s	1.71 ±1.54	0.02 ±0.01	2.42 ±2.22	0.12 ±0.07	7.88 ±10.35	0.09 ±0.06	0.04 ±0.02	0.01 ±0.02	0.05 ±0.02	**0.001 ±0.001**
MedBIoT	μ_f	0.11 ±0.84	0.14 ±0.93	0.80 ±0.39	0.87 ±0.40	0.08 ±0.85	0.68 ±0.48	0.71 ±0.57	0.87 ±0.31	0.81 ±0.40	**0.95 ±0.12**
	μ_m	92.28%	94.34%	72.16%	84.76%	84.16%	87.70%	75.18%	78.28%	94.10%	**94.34%**
	μ_c	0.92 ±0.09	0.87 ±0.13	0.93 ±0.09	0.73 ±0.13	0.82 ±0.17	0.79 ±0.08	1.09 ±0.01	0.76 ±0.05	0.75 ±0.09	**0.60 ±0.19**
	μ_s	0.28 ±0.71	0.03 ±0.05	1.06 ±3.33	0.03 ±0.02	3.53 ±9.03	0.10 ±0.18	0.03 ±0.05	0.01 ±0.04	0.02 ±0.01	**0.01 ±0.01**
BoT-IoT	μ_f	0.43 ±0.49	0.46 ±0.53	0.06 ±0.76	0.53 ±0.37	-0.13 ±0.48	0.20 ±0.48	0.35 ±0.37	0.59 ±0.26	0.64 ±0.38	**0.73 ±0.24**
	μ_m	62.95%	63.98%	26.60%	37.41%	27.03%	30.30%	52.02%	60.40%	86.10%	**89.01%**
	μ_c	1.54 ±0.21	1.34 ±0.26	1.60 ±0.12	1.46 ±0.17	1.52 ±0.09	1.50 ±0.18	1.65 ±0.09	1.55 ±0.12	1.18 ±0.12	**0.91 ±0.28**
	μ_s	0.55±0.99	0.44±0.03	0.10±0.11	0.50±0.71	2.07±2.41	0.12±0.15	0.03±0.05	0.01±0.02	0.01±0.05	**0.01±0.01**

Faithfulness (μ_f) computes correlation between the importance that an XAI method assigns to features and their actual impact on the model's prediction probabilities. A high faithfulness correlation value for an XAI method indicates that the importance assigned to each feature closely aligns with its effect on the model's predictions. This alignment ensures that the explanations provided are accurate and trustworthy.

Monotonicity (μ_m) evaluates how individual features influence model prediction probabilities by examining the changes in prediction probability as each feature is added sequentially in order of their importance. When features are added, the model's prediction probability consistently increases, leading to a monotonic increase in probabilities. A high monotonicity score indicates that the explanations provided by the xai method align well with the model's predictions for the given input.

Low complexity(μ_c) metric used to compute the entropy of feature attribution obtained from the XAI method. A lower entropy indicates that fewer features have high attribution scores.

The sensitivity metric(μ_s) evaluates the robustness of explanations by ensuring that nearby inputs in the feature space generate similar explanation outputs when the sensitivity value is low. To evaluate this metric, the nearest neighbor points were identified based on the prediction label of the explanation score using Euclidean distance with a radius of 0.1. which helps to locate data points in the feature space that are closest to the instance and provide comparable explanations for the predicted label.

LIME and SHAP explanations were evaluated using 2000 test points, as shown in Table 7. This analysis covers binary classifiers (ET, LGBM, GBC, RF, and XGB) on the N-BaIoT, Med-BaIoT, and BoT IoT datasets.

In the N-BaIoT dataset, SHAP consistently demonstrated superior performance compared to LIME when evaluating their explanations using faithfulness. It shows higher mean faithfulness correlation values, especially for the LGBM and XGB models. Remarkably,

the XGB model, when explained using SHAP, achieved an outstanding faithfulness score ($\mu_f = 0.99 \pm 0.13$), which indicates that SHAP provides highly accurate and reliable explanations for this model.

Furthermore, by evaluating the monotonicity of the explainer—which indicates how consistently the explanations change in relation to the input features—both LIME and SHAP show high monotonicity values. Notably, SHAP explanations for LGBM and XGB models achieve monotonicity scores exceeding 99%.

Complexity measures how concise the explanations given by the explainer are. Lower complexity values signify simpler and more interpretable explanations. Among all models and explainers, the SHAP explainer for XGBoost produced the most concise explanations, achieving the lowest complexity with a mean value $(\mu_c = 0.50 \pm 0.25)$.

Sensitivity assesses how stable the explanations are for data points that are close to each other. Lower sensitivity values indicate more stable explanations. In this context, the XGB model explained using SHAP exhibited the lowest sensitivity $(\mu_s = 0.001 \pm 0.001)$. This suggests that it provides highly stable and reliable explanations for nearby data points within the same feature space utilized by the XGB model.

Similarly, SHAP explanations consistently outperformed LIME for binary classifiers on both the MedBIoT and BoT IoT datasets (see Table 7). Notably, when applied to the XGB model, SHAP provided explanations with higher fidelity, greater consistency, lower complexity, and greater robustness than those generated by LIME.

Table 8: Results of Evaluating the quality of LIME&SHAP Botnet malware type (multiclass) on two datasets: N-BaIoT, MedBIoT

Dataset	Model	ET		LGBM		GBC		RF		XGB	
	XAI Metric	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP
N-BaIoT	Faithfulness	-0.026±0.757	-0.060±0.852	0.395±0.641	0.239±0.573	-0.124±0.502	0.268±0.033	0.130±0.615	0.152±0.608	0.898±0.129	**0.907±0.382**
	Monotonocity	74.70%	73.30%	56.90%	66.10%	57.70%	48.70%	58.60%	64.70%	89.00%	**100.00%**
	Complexity	1.017±0.086	0.994±0.079	0.772±0.195	0.753±0.109	0.926±0.542	0.887±0.215	0.990±0.092	0.885±0.155	0.710±0.236	**0.686±0.236**
	Sensitivity	0.021±0.012	0.054±0.137	0.028±0.023	0.034±0.015	0.789±1.995	0.878±0.324	0.025±0.012	0.082±0.204	0.042±0.043	**0.002±0.001**
Med-BaIoT	Faithfulness	0.11±0.38	0.46±0.43	0.05±0.32	0.29±0.45	0.12±0.59	0.20±0.49	0.31±0.38	0.57±0.25	0.62±0.27	**0.81±0.23**
	Monotonocity	36.00%	45.00%	35.00%	41.00%	28.00%	49.00%	27.00%	58.00%	78.00%	**84.00%**
	Complexity	1.45±0.23	1.41±0.30	1.59±0.15	1.57±0.14	1.68±0.15	1.92±0.22	1.64±0.11	1.48±0.20	1.09±0.31	**0.91±0.19**
	Sensitivity	0.02±0.01	0.00±0.01	0.82±1.46	0.05±0.02	0.74±0.23	0.04±0.02	0.03±0.03	0.01±0.03	0.02±0.09	**0.01±0.01**

Similar to binary classification, in Publication III, the quality of explanations obtained from LIME and SHAP were evaluated on multiclass models, explicitly focusing on the detection of botnet malware types.

In the study of multiclass classification for botnet type detection, the N-BaIoT and MedBIoT datasets were utilized. The detection of botnet malware types involves three classes for the N-BaIoT dataset—Mirai, Gafgyt, and Benign—and four classes for the Med-BaIoT dataset—Mirai, BashLite, Torii, and Benign.

During the feature selection stage of botnet type detection using a Sequential Backward Selection (SBS) approach, the k-best feature set for the N-BaIoT dataset included three selected features, while the Med-BaIoT dataset comprised seven selected features.

After performing feature selection, LIME and SHAP explainers were used to interpret the trained models for botnet detection. Table 4 displays the quality of the explanations produced by both LIME and SHAP across various models. The evaluation of explanation quality has shown varying performance across different models when using explainers for both data sets.

5.2 Evaluation of XAI Methods for Deep Learning-Based Botnet Attack Detection

Early AI frameworks, such as Decision Trees and K-Nearest Neighbors (KNN), were transparent and easy to understand. In contrast, deep neural network (DNN) models, despite their complexity and high resource requirements, have achieved significant success due to efficient algorithms and a wide range of parameters. However, DNNs often lack interpretability, making their decision-making processes difficult to comprehend. Publication IV is the extension of Publication III, which focuses on the evaluation of explainers for a Deep Neural Network (DNN) model designed to detect IoT botnet attacks and summarizes key findings. Publication IV uses the N-BaIoT dataset to classify eight attack types, along with legitimate network traffic. The categories include ACK, Benign, COMBO, JUNK, SCAN, SYN, TCP, UDP, and PLAIN. Detailed descriptions of these attacks are found in Table 2 in Chapter 4.

The dataset was divided into training and testing sets with an 80:20 ratio. Min-max normalization was applied to scale the feature values between 0 and 1. A deep neural network (DNN) for predicting botnet attacks consists of 3 hidden layers, each containing 9 hidden units. Random search hyperparameter tuning was utilized to obtain the best hyperparameters using Ray Tune. Hyperparameters are shown in the Table 9

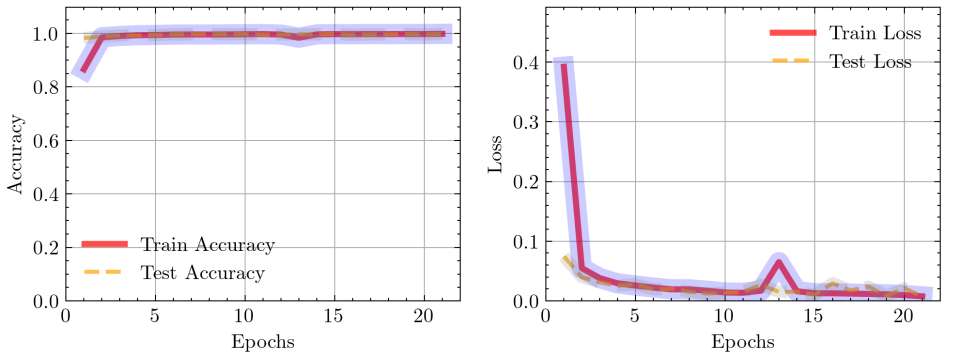
The input layer of DNN model receives a 115-dimensional feature vector, representing aggregated statistics from five different time windows. This input is passed through the first fully connected hidden layer with 9 units, applying the SELU activation function. The output from the first hidden layer is passed to the second hidden layer, which contains 9 units and uses the SELU activation function. The output layer, a fully connected layer, maps these activations to the output size corresponding to the botnet attack types: ack, benign, combo, junk, scan, syn, tcp, udp, and udpplain. The model was trained over 21 epochs with a batch size of 256. Cross-Entropy Loss function was used for computing the loss which is ideal for multi-class classification tasks. DNN model prediction probabilities of class labels were obtained using the Softmax activation function.

The performance of the DNN model for detecting botnet attack types was evaluated using accuracy. Figure 14a displays the training and testing accuracy over epochs, while Figure 14b shows the training and testing loss of the model's performance.

Table 9: DNN Model Hyperparameters (Random Search)

Hyperparameter	Value
Hidden Layers	3
Hidden Units	9
Learning Rate	0.01296
Optimizer	RAdam
Activation	SELU
Batch Size	256
Epochs	21

To achieve the goals of XAI, various methods have been developed to generate explanations for understanding model behavior [120, 121]. Post hoc XAI methods are commonly classified into categories such as feature importance-based, saliency-based, and gradient-based approaches, among others. In this publication, seven posthoc explainers, including those based on feature importance and saliency, were used to explain a DNN model for predicting IoT botnet attacks. SHAP, LIME, and Feature Ablation are examples



(a) Accuracy *(b) Loss*

Figure 14: Training and Testing Accuracies and Loss Values of DNN Model for IoT Botnet Attack Detection

Table 10: comparison results of the evaluation of XAI metrics

Explainer/metric	High Faithfulness	Sensitivity	Complexity	Monotonicity
Integrated Gradients	0.44 ± 0.32	0.61 ± 1.44	3.32 ± 0.38	52.00%
Gradient * Input	0.42 ± 0.36	0.61 ± 1.44	3.75 ± 0.29	60.00%
DeepLIFT	$\mathbf{0.71 \pm 0.12}$	$\mathbf{0.48 \pm 0.76}$	$\mathbf{2.82 \pm 0.12}$	**69.19%**
SHAP	0.55 ± 0.18	8.17 ± 8.11	3.30 ± 0.40	44.00%
Feature Ablation	0.48 ± 0.31	0.38 ± 0.76	4.29 ± 0.41	39.00%
Saliency	0.27 ± 0.26	9.79 ± 21.36	4.33 ± 0.06	46.00%
LIME	0.39 ± 0.21	10.62 ± 5.68	4.30 ± 0.31	28.00%

of feature importance-based explainers, while saliency-based explainers include methods like Saliency, Integrated Gradients, Gradient × Input, and DeepLIFT. These explainers provide local interpretations by identifying key features that influence the model's prediction of a specific attack class label. Through saliency-based xai methods, they attribute importance scores to individual features, explaining how the model makes decisions for a given data point.

The evaluation of these explainers was conducted using quantitative metrics such as faithfulness, monotonicity, complexity, and sensitivity. Specifically, the effectiveness of these post-hoc local explanation methods was evaluated and compared when applied to various multiclass attack types for IoT botnet detection in a detailed benchmarking setting. These metrics were used to evaluate the post hoc local explanations generated by the explainers across test data points. Table 10 presents the comparative results of the XAI evaluation metrics in terms of mean and standard deviation values. DeepLIFT was the top performed explainer among the explainers, achieving a mean faithfulness correlation of 0.71 $\pm$0.12. This indicates that its explanations closely align with the DNN model's behaviour, enhancing their trustworthiness. Additionally, DeepLIFT showed a lower sensitivity value of 0.48 $\pm$0.76, indicating consistent explanations for nearby data points in the feature space. DeepLIFT achieved a notable monotonicity score of 69.19%, reflecting strong consistency in feature influence on model predictions.

5.3 Chapter Discussions

This chapter addresses research question **RQ2**, which aims to explore how the quality of XAI methods can be quantitatively evaluated in the context of IoT botnet detection. As XAI has become increasingly important in network security, helping security analysts in understanding the decision-making processes of ML-based IDS, thereby enhancing threat analysis and enabling effective responses. This chapter contributes to addressing the significant challenge of evaluating the performance of XAI by proposing a framework for assessing the quality of local explanations generated by post hoc XAI methods for ML-based IDS in the context of IoT botnet detection.

In Publication III, we assessed the performance of various ML classifiers and their respective feature sets in predicting malware infections in IoT devices using three datasets: 1) N-BaIoT, 2) MedBIoT, and 3) BoT-IoT. We employed LIME and SHAP explainers to provide interpretable explanations of the model predictions. Using Sequential Backward Selection (SBS), we identified key features that improved classifier performance. While the selected features varied by dataset, network traffic features such as host, channel, network jitter, and socket were consistently important for predicting IoT botnet malware and types. Host-based features were critical for the N-BaIoT dataset, whereas channel- and socket-based features were more prominent in the MedBIoT dataset. In the BoT-IoT dataset, the number of inbound connections per source and destination IP were more important features. Overall, the XGBoost model achieved the highest F1 score across all datasets, proving to be an effective choice for malware detection in IoT devices.

The results from Publication III demonstrate that both LIME and SHAP local explanations are highly consistent and explainable for the XGBoost model using the SBS-XGBoost feature set, as indicated by their high monotonicity scores. This suggests that the explanations provided by LIME and SHAP effectively capture the underlying relationships between the feature set and the prediction outcomes (malware and benign) in network traffic classification. Additionally, both LIME and SHAP exhibit high mean values for faithfulness correlation, which implies that security analysts can trust the explanations given for the XGBoost model. The complexity scores are lower, along with the high trustworthiness, which assists analysts in better understanding the model's predictions and making informed decisions based on its outputs. Furthermore, the explanations for the XGBoost model are more robust, as evidenced by lower sensitivity values, which means that similar inputs producing comparable model outputs will yield similar explanations. In conclusion, Publication III shows that the XGBoost classifier is effective for detecting malware in IoT devices using various datasets. The LIME and SHAP explainers provide valuable insights that help security analysts prioritize alerts for critical devices. Although LIME offers lower complexity and higher consistency across different models of IoT botnets, still, SHAP explainer is better than LIME.

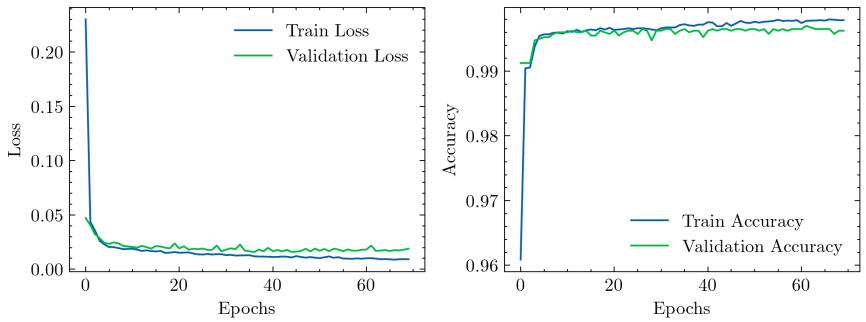
In Publication IV, post-hoc explanations for deep learning-based IoT botnet detection were evaluated using a DNN model for multi-class IoT botnet attacks from the N-BaIoT dataset. Seven post-hoc explanation methods were employed, including feature attribution and saliency-based methods (SHAP, LIME, Feature Ablation, Integrated Gradients, Gradient*Input, and DeepLIFT), to explain the predictions of IoT botnet attacks (ACK, Benign, COMBO, JUNK, SCAN, SYN, TCP, UDP, and PLAIN) using the DNN model. The effectiveness of post-hoc XAI methods was evaluated using quantitative metrics, such as faithfulness, monotonicity, low complexity, and maximum sensitivity. The results of this study highlight that DeepLIFT provided the highest faithfulness and offered more robust explanations, along with lower complexity compared to the other posthoc XAI methods for improving the transparency of DNN–based IDS model in IoT botnet detection.

6 XAI for Transparent Alert Classification in NIDS

Many organizations use open-source NIDS like Suricata and Snort [8], as well as commercial options like Cisco's NGIPS, to detect malicious network traffic. However, NIDS tends to produce a large number of alerts, most of which are of low significance. A recent study [6] on SoCs identified NIDS as one of the key technologies used in SOCs. One of the primary challenges faced by SOC analysts is the high volume of low-priority NIDS alerts. Specifically, there is a significant number of genuine alerts (i.e., not false positives, but alerts that indicate real attacks) that are considered low priority given the specific environment.

Publication V focuses on a comprehensive study of evaluating the local explainability of the LSTM model used in NIDS alerts. Publication V utilizes a dataset from a Suricata NIDS system at the SoC of TalTech, collected over 60 days from January to March 2022 [143]. During this period, Suricata generated alerts for network activity involving 45,339 hosts, including 4,401 from TalTech. The dataset consists of 1,395,324 data points with human-assigned binary labels: "important" and "irrelevant" which indicate the priority level of the NIDS alerts. LSTM model was developed to prioritize alerts as either "important" or "irrelevant" in a binary classification task.

For the experimental setup of training the LSTM model, 10,000 data points were selected for each class label—'irrelevant' and 'important'—resulting in a total of 20,000 samples. The dataset was divided into training and testing sets with an 80-20 ratio. Min-max normalization was then applied to scale the input features to a range between 0 and 1. The performance of the LSTM model was evaluated using a confusion matrix. Accuracy, precision, recall, and F1-score metrics were used.



(a) Loss *(b) Accuracy*

Figure 15: Loss and Accuracy from Best LSTM Performance Model for NIDS alert classification

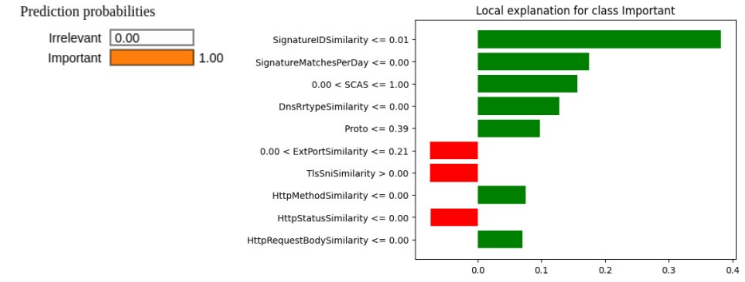
Table 11: Key Features Identified by Taltech SOC Analyst for Determining NIDS Alert Significance

SignatureMatchesPerDay
Similarity
SCAS
SignatureID
SignatureIDSimilarity

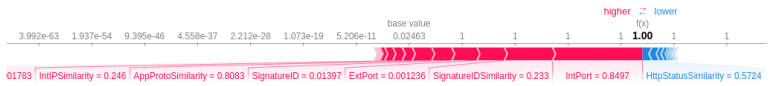
Figure 15a illustrates the training and validation loss over 70 epochs, achieved through random search tuning. Both the training and validation losses of the LSTM model dra-

matically decrease and ultimately approach zero. Figure 15b shows that the training and validation accuracy quickly stabilizes above 99.5%, indicating strong model performance.

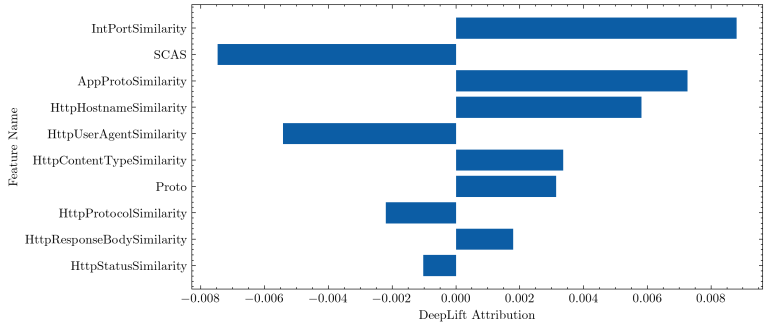
In Publication V, collaboration with SoC analysts from TalTech, Estonia, was conducted to assess the reliability of the post-hoc explanations generated for the decisions made by the black-box model used for alert classification in this work. A detailed description of the TalTech SOC can be found in a Paper [144] Utilizing their expertise in managing NIDS alerts, the SOC team at TalTech identified five key features for determining alert significance, as mentioned in Table 11.



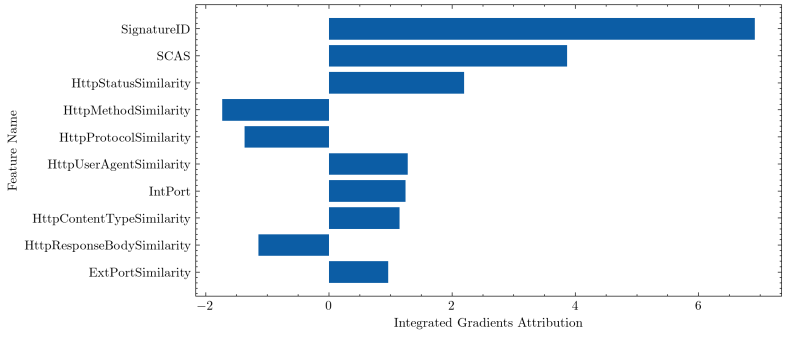
(a) LIME explanations for important NIDS alerts using an LSTM model



(b) SHAP explanations for an important NIDS alert data point using an LSTM model



(c) DeepLIFT feature importance for an important NIDS alert data point using an LSTM model



(d) Integrated Gradients feature importance for an important NIDS alert data point using an LSTM model

Figure 16: Explanations for an important NIDS alert data point using an LSTM model

Four different Explainable AI (XAI) methods—LIME, SHAP, IG, and DeepLift—were utilized to explain the predictions made by the LSTM model on the test data within the context of post-hoc local explanation scenarios.

Figure. 16a shows an local explanation from LIME method for a NIDS alert labeled as "Important.". Left side presents prediction probabilities with a 100% probability for the "Important" class. On the right side it illustrates the impact of features. For instance, when the feature 'SignatureIDSimilarity' is less than or equal to 0.01, it positively affects the "Important" classification of NIDS alert. Additionally, 'SignatureMatchesPerDay' and 'SCAS' being less than or equal to 1.00 also contribute positively. Conversely, 'ExtPortSimilarity' and 'TlsSniSimilarity' have impacts, suggesting that some NIDS alerts may not be relevant. SHAP employs Shapley values to showcase how features influence model predictions in Figure. 16b of force plot, red bar signifies the positive impact while blue bar indicates the negative impact on the model output. Each bar demonstrates whether the features bring the predicted value closer to or farther from the base value of 0.02463. The plot's base value is the average of all prediction values. Each strip in the plot displays the impact of the features on moving the predicted value closer to or farther from the base value. Final prediction is deemed an "important class label", with a value of 1.00 for this NIDS alert. Features, like 'IntPort' (Internal Port) 'SignatureIDSimilarity'. ExtPort' (External Port) along with 'SignatureID' play a role in indicating the importance of NIDS alert. However, the feature 'HttpStatusSimilarity' might suggest that this alert could be a less critical feature to its impact.

DeepLift is a technique used to attribute the output of LSTM model to its input features by comparing neuron activation to a reference activation and assigning contribution scores based on the variance. Figure. 16c illustrates the significance of features using the DeepLift explainer for the 10 features of a NIDS alert data point labeled as "important." The negative attribution of 'SCAS' suggests its influence on classifying as "Important" in NIDS alerts. Additionally 'HttpMethodSimilarity' and 'IntIP' show negative attributions while HttpContentTypeSimilarity has a slight positive impact countering the "Important" classification. IG attribute a LSTM model's prediction its input features by integrating gradients of the model's output with respect to the input along from a baseline to the input. This explanation technique works best for models that use linear activation functions. Figure. 16d showcases feature importance using IG explainer for a data point in the "Important" NIDS alert class label among the 10 features. Features such, as 'SignatureID' 'SCAS,' and 'HttpStatusSimilarity' display attributions. When compared, the features identified by the TalTech SOC analyst closely matched those derived from the explainers for nearly all data points. An example of local post hoc explanations from employed explainers can be found in Publication V.

The quality of explanations provided by XAI methods was evaluated for alert classification in LSTM-based NIDS alerts using a dataset of 2000 data points. The evaluation was based on four criteria: faithfulness, robustness, complexity, and reliability.

- **Faithfulness:** The explanation algorithm $\mathbf{g}$ should replicate the model's behavior. $\mathbf{g}(M,x) \approx M(x)$. To evaluate the faithfulness of explanations, the Faithfulness correlation (Bhatt et al., 2020) and Monotonicity (Luss et al., 2019) metrics were used.

- **Robustness:** Robustness refers to similar inputs should result in similar explanations. $g(M,x) \approx g(M,x+\varepsilon)$ for small ε. For robustness of explanation, the Maximum Sensitivity metric was used.

- **Complexity:** Explanations using a smaller number of features are preferred. It is assumed that explanations using a large number of features are difficult for the

user to understand.$\min \|g(M,x)\|_0$. For evaluating the complexity of explanations a Low complexity metric was used.

- **Reliability** An explanation should be centred around the region of interest, the ground truth GT. $g(M,x) = GT$. For evaluating the reliability of explanations, relevance rank accuracy and relevance mask accuracy were implemented from a Paper [15]. 'Major' parts of an explanation should lie inside the ground truth mask $GT(x)$ for both Relevance Mass Accuracy and Relevance Rank Accuracy metrics used in Publication V, and the Ground truth mask was determined by the features SOC Analysts identified (see Table. 11). $GT(x)$ is represented as a binary vector. Each entry in the vector is set to 1 if the corresponding feature has been identified as relevant by SOC analysts, and 0 if it has not. This binary mask reflects the expertise of domain specialists and was used as a reference mask to evaluate the reliability of post hoc explanation methods.

XAI metrics regarding the criteria of Faithfulness, Robustness, and Complexity are detailed in Chapter 5.1. On the other hand, XAI metrics related to Reliability, such as Relevance rank accuracy (RRA) and Relevance Mass accuracy (RMA) metrics, are described below.

6.1 Relevance Rank Accuracy

Relevance rank accuracy measures how much of the high-intensity relevance lies within the ground truth. Top K values of $g(M,x)$ are sorted in decreasing order $X_{\text{top}K} = \{x_1, ..., x_K \mid g(M,x)_{x_1} > ... > g(M,x)_{x_K}\}$.

$$RRA = \frac{|X_{\text{top}_k} \cap GT(x)|}{|GT(x)|} \tag{19}$$

Here top_k are features Identified by SOC Analyst.

6.2 Relevance Mass Accuracy

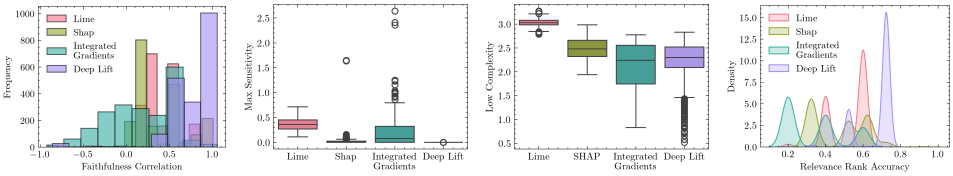
The relevance mass accuracy was calculated as the sum of the explanation values within the ground truth mask divided by the sum of all values.

$$\text{RMA} = \frac{\sum\limits_{i \in GT(x)} g(M,x)_i}{\sum\limits_{i=1}^{d} g(M,x)_i} \tag{20}$$

Here, $g(M,x)_i$ is the explanation score assigned to the i^{th} feature for input x, $GT(x) \subseteq \{1,\ldots,d\}$ is the set of indices corresponding to ground truth relevant features, and d is the total number of features.

Table 12: Evaluation Results of Explainable AI Methods: Mean (μ) and Standard Deviation (σ) Values.

Explanation Criterion	Faithfulness		Robustness	Complexity	Reliability	
Explainer/Metric	High Faithfulnes $\mu \pm \sigma$	Monotonicity μ	Max Sensitvity $\mu \pm \sigma$	Low Complexity $\mu \pm \sigma$	Relevance Mass Accuracy $\mu \pm \sigma$	Relevancy Rank Accuracy $\mu \pm \sigma$
Lime	0.4209 ±0.1835	59.55%	0.3617 ±0.1152	3.0318 ±0.0703	0.6234 ±9.7008	0.5250 ±0.1041
Shap	0.3959 ±0.2928	64.45%	0.0245 ±0.0862	2.4677 ±0.2074	0.6527 ±3.8334	0.4743 ±0.1418
IG	0.1761 ±0.3815	73.70%	0.1774 ±0.2505	**2.1745 ±0.4134**	0.5939 ±0.6840	0.3410 ±0.1545
Deep Lift	**0.7559 ±0.2681**	**78.35%**	**0.0008 ±0.0004**	2.2635 ±0.3299	**0.7812 ±25.2805**	**0.6754 ±0.0897**



(a) High Faithfulness *(b) Maximum Sensitivity* *(c) Low Complexity* *(d) Relevancy Rank Accuracy*

Figure 17: Quality of Explainable AI evaluation metrics distribution for LSTM model based NIDS alerts classification.

Table 12 shows the results regarding the quality of explanations for various XAI methods. The prediction probabilities for the LSTM model were calculated using the Softmax activation function. Mean (μ) and standard deviation (σ) values were selected for the XAI computed metrics based on a test dataset of 2000 points. Among the evaluated methods, DeepLIFT achieved the highest correlation values for faithfulness, with a mean and standard deviation (0.7559 $\pm$0.2681) for the test data points.

The explanation's monotonicity was analyzed to understand how individual features affect model probability by adding each attribute to enhance its importance and observing its influence on the model's probability. DeepLIFT achieved high monotonicity with 78% (μ). To evaluate complexity, we calculated the entropy of feature attribution in the explanations. Among the XAI methods evaluated by a low complexity metric, IG) achieved a lower complexity score (2.174 $\pm$0.413), closely followed by DeepLift (2.264 $\pm$0.330). Deep LIFT achieved Lower sensitivity with Maximum Sensitivity metric (0.0008 $\pm$0.0004).

Two metrics were used to evaluate the reliability of explanations: RMA and RRA. These metrics assessed the explanations by comparing them to a ground truth mask, which was developed based on features identified in collaboration with a SOC analyst. Both the RMA (0.781 $\pm$25.281) and the RRA (0.6754 $\pm$0.089) metrics provided reliable deep lift explanations.

Figure 17 illustrates the distribution of XAI metric results for 2000 data points. It highlights that DeepLIFT's explanations show high faithfulness, lower sensitivity, lower complexity, and more relevance rank accuracy. Additionally, a Wilcoxon signed-rank test [152] was conducted to rigorously evaluate whether the performance differences among the four XAI methods (LIME, SHAP, IG, and DeepLIFT) were statistically significant across the evaluation metrics.

Following standard practices for evaluating XAI methods [53], the Wilcoxon signed-ranks test [152] was employed to assess the statistical significance of differences in XAI metric scores between pairs of explainers (e.g., $explainer_A$, $explainer_B$) for NIDS alert classification. The null hypothesis (H_0) posits that the XAI metric scores of the two explainers are equivalent, meaning there is no significant difference between them (XAI Metric Score($explainer_A$) = XAI Metric Score($explainer_B$)). Conversely, the alternative hypothesis (H_1) asserts that they are not equivalent (XAI Metric Score($explainer_A$) $\neq$ XAI Metric Score($explainer_B$)), indicating a significant difference in their metric scores. In this study, the XAI metrics evaluated include High Faithfulness, Maximum Sensitivity, Low Complexity, Relevance Mass Accuracy, and Relevancy Rank Accuracy. Wilcoxon signed-rank test was conducted separately for each metric to thoroughly assess the performance differences among the explainers.

The analysis in Table 13 shows significant differences among explainers across all metrics, with p-values consistently below 0.05. DeepLift outperforms the others in faithful-

Table 13: Statistical Comparison of Explainers Across Multiple Metrics (p-values)

Metric	Explainer	Shap	IG	Deep Lift
Faithfulness	LIME	L (3.34e-41)	L (1.03e-134)	D (5.61e-185)
	SHAP	-	S (6.22e-91)	D (1.03e-169)
	IG		-	D (1.30e-230)
Maximum Sensitivity	LIME	S (0.00e+00)	I (1.64e-221)	D (0.00e+00)
	SHAP	-	S (1.38e-185)	D (3.54e-126)
	IG		-	D (3.29e-126)
Low Complexity	LIME	S (0.00e+00)	I (0.00e+00)	D (0.00e+00)
	SHAP	-	I (5.45e-146)	D (1.26e-88)
	IG		-	I (1.07e-42)
RMA	LIME	S (5.12e-25)	L (1.97e-80)	D (6.22e-83)
	SHAP	-	S (4.67e-155)	D (6.47e-91)
	IG		-	D (2.82e-54)
RRA	LIME	L (7.61e-39)	L (3.07e-210)	D (0.00e+00)
	SHAP	-	S (5.52e-155)	D (3.97e-253)
	IG		-	D (0.00e+00)

D (Deep Lift), L (LIME), S (SHAP), and I (Integrated Gradients) indicate the better-performing explainer in each pairwise comparison.

- $p > 0.05$ — No significant evidence against H_0; H_0 is not rejected
- $0.01 < p \leq 0.05$ — Significant evidence against H_0; H_1 is accepted at 95% confidence level
- $0.001 < p \leq 0.01$ — Strong evidence against H_0; H_1 is accepted at 99% confidence level
- $p \leq 0.001$ — Very strong evidence against H_0; H_1 is accepted at 99.9% confidence level.

ness, maximum sensitivity, RMA, and RRA, with pairwise comparisons yielding $p < 0.001$. The performance of SHAP, LIME, and IG varies by metric, as shown in Table 13.

A global explanation of the LSTM model's predictions is provided using SHAP values for all the testing data. A higher SHAP value indicates a positive impact on the prediction, while a lower value indicates a negative contribution. Figure 18 displays the global explanation of the LSTM model. The graph illustrates the average influence of each feature on the magnitude of the model's output for the class labels "irrelevant" and "important" classifications. The most impactful features for important network intrusion detection system (NIDS) alerts include SignatureIDSimilarity, SignatureMatchesPerDay, ProtoSimilarity, and SCAS. Notably, these top features align with those identified by human expert SoC analysts. In contrast, lower-ranked features, such as HTTP-related similarities (e.g., HttpHostnameSimilarity and HttpUrlSimilarity) and IP-related features (e.g., ExtIPSimilarity), have a comparatively lesser impact on the model's decisions.

6.3 Chapter Discussions

This chapter provides an answer to **RQ2**. The main challenge of evaluating explainability for the cybersecurity domain in NIDS alert classification is the absence of clearly provided ground truth explanations, unlike image explanations where humans can evidently mark important sub-images or pixels But NIDS operates with abstract network traffic features such as packet size and protocol flags and flow duration and byte counts which experts

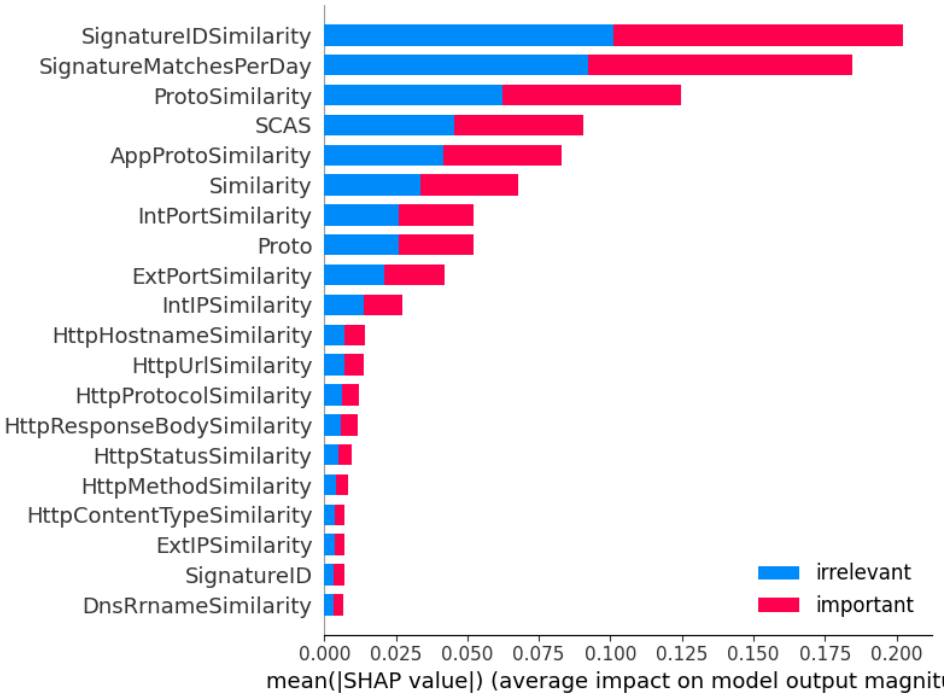


Figure 18: SHAP global explanation for LSTM model in NIDS alert classification

find difficult to interpret, The absence of clear human-recognizable ground truth makes it challenging to determine if an explanation accurately reflects the reasons behind black box model predictions. So Expert domain knowledge from SOC analysts needs to be incorporated to create reference points for evaluating explanation quality. The research presented in this chapter tackles the problem by incorporating SoC analyst-annotated relevance to explanation evaluation through two metrics: Relevance Mass Accuracy and Relevance Rank Accuracy, to evaluate the reliability of explanations. In Publication V, a real-world NIDS alert dataset was employed from the SoC at TalTech (Tallinn University of Technology) in Estonia. A Long Short-Term Memory (LSTM) model was developed to prioritize alerts. To explain the alert prioritization decisions made by the LSTM model, four explainable artificial intelligence (XAI) methods were implemented and compared: LIME, SHAP, Integrated Gradients, and DeepLIFT. Our comprehensive assessment of the XAI framework evaluated the effectiveness of these methods based on criteria such as faithfulness, complexity, robustness, and reliability. We examined how well these XAI techniques can explain NIDS alerts. We investigated how well these XAI techniques explain NIDS alerts. The findings Publication V demonstrate that DeepLIFT consistently outperformed the other XAI methods, providing explanations with high faithfulness, low complexity, robust performance, and strong reliability. In collaboration with SOC analysts, we identified key features essential for effective alert classification. The strong alignment between the features identified by the analysts and those obtained through the XAI methods further validates their effectiveness and enhances the practical applicability of our approach.

7 Explainable Transformer-based Intrusion Detection in IoMT Networks

This chapter summarises the contributions of this dissertation to IoMT cyberattack detection research and contextualises the work presented in Publication VI.

Internet of Medical Things (IoMT) is an advanced network that effectively integrates Internet-connected medical devices and their corresponding software applications to exchange healthcare-related information, thereby facilitating treatment and patient monitoring over the Internet [112]. At the same time, IoMT devices are often resource-constrained. The rapid growth of the market, along with the high value of data and existing security vulnerabilities, makes these devices attractive targets for cybercriminals [26]. DDoS and DoS attacks on IoMT devices can disrupt vital medical equipment, potentially endangering patient lives [68]. Most previous studies have focused on improving detection accuracy in IDS using transformer models but they have not addressed the importance of understanding the decision-making process of these models. Additionally, they have not worked with recent and realistic IoMT datasets in actual IoMT networks. To address this gap, Publication VI proposes an explainable transformer-based model for detecting and categorizing network attacks within IoMT networks. This research investigates the recent and realistic CIC-IoMT2024 dataset [31], which encompasses various types of cyber attacks and normal traffic in real-world IoMT network scenarios. This study employs a sliding window approach to adapt transformer models for analysing network traffic as sequences, thereby enhancing detection accuracy. The aim of this study is to enhance the transparency of Transformer-based IDS in IoMT by integrating XAI methods such as LIME and SHAP.

Transformer model was first presented by Vaswani et al. [145] through self-attention mechanisms which generate sequence representations without requiring recurrence. An attention function is a weighted average of values. It maps a query q, along with a set of keys, K, and their corresponding values V, to produce an output [17]. The output of an attention function is based on a similarity score between the query and each key, which determines the weights for the averaged sum of the values.

$$\text{Attention}(q,K,V) = \sum_i \text{Score}(q,k_i)v_i$$

The attention mechanism captures the relationship between a query and each key in a database, adjusting the values based on these relationships. The "Scaled Dot-Product Attention" function, described in [7], uses the dot product as the similarity measure between the query and keys. Both the query and keys are vectors of the same dimension d_k, while the values are vectors of dimension d_v. If Q represents a matrix of queries, the Scaled Dot-Product Attention is calculated as follows:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

To enhance representational capabilities, the queries, keys, and values are linearly transformed before applying the attention function, with an additional linear transformation applied to the output. These transformations are implemented as layers in a neural network and learned through gradient backpropagation. Additionally, to capture diverse representations, the queries, keys, and values are processed h times in h parallel attention heads, and the outputs are concatenated before the final transformation.

The transformer [145] is a sequence transduction model consisting of two main components: an encoder and a decoder. The encoder takes a sequence of inputs $x = (x_1, \ldots, x_n)$

and converts it into a sequence of lower-dimensional representations $z = (z_1, \ldots, z_n)$, while maintaining the same length. The decoder then uses these representations z as inputs to produce an output sequence of arbitrary length: $y = (y_1, \ldots, y_m)$. Both the encoder and the decoder are constructed as stacks of identical layers.

The encoding layers are composed of the following steps:

- A self-attention step, using the input sequence as the query, along with the set of keys and the set of values.
- A feed-forward step, which applies a linear transformation to its inputs, followed by an activation function.

In contrast, each decoding layer performs the following steps sequentially:

- A masked self-attention step, using the target sequence as the query, the set of keys and the set of values. In order to avoid leakage of information from yet unseen positions of the target sequence during training, this self-attention step is said to be masked: each position of the sequence can only attend to the previous ones, and the forbidden attention values are set to $-\infty$.
- An attention step where the queries and keys are taken from the encoder output, while the input values come from the output of the masked self-attention step.
- A feed-forward step that applies a linear transformation to the output of the encoder-decoder attention layer, followed by an activation function.

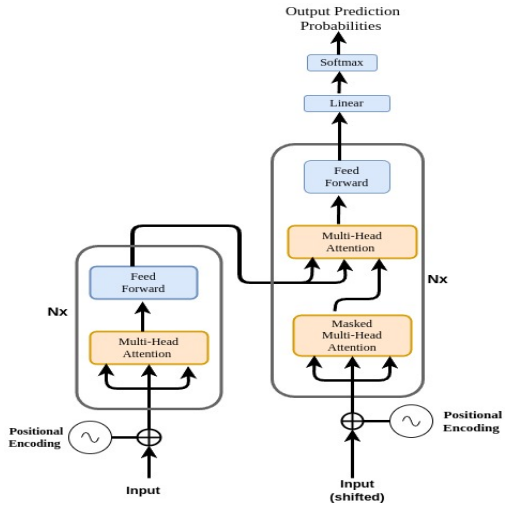


Figure 19: Transformer Architecture for IoMT attack detection

The transformer architecture includes N encoding layers that form the encoder block and N decoding layers that make up the decoder block. The encoder takes a sequence as input, while the decoder receives a version of this sequence missing the last w positions. The decoder's output is then transformed by a final linear layer to reconstruct the original sequence. A simplified diagram of the transformer model based on one presented in a paper [145] can be found in Figure 19.

In this research, as mentioned in Publication VI, the CICIoMT2024 dataset [31], which the Canadian Institute for Cybersecurity developed, was utilized to simulate realistic IoMT environments and collect network traffic for cybersecurity research. It encompasses IoT devices connected through Wi-Fi, MQTT, and Bluetooth Low Energy (BLE). The experimental setup included 25 real devices and 15 simulated devices, generating a variety of network traffic, including data from healthcare devices and cameras connected via Wi-Fi. The attacks executed during the experiments included ARP spoofing, DoS, DDoS, port scans, vulnerability scans, MQTT Connect Floods, MQTT Publish Floods, MQTT malformed data attacks, and disruptions to BLE devices. 44 relevant features were extracted from the network traffic PCAP files for each network flow.

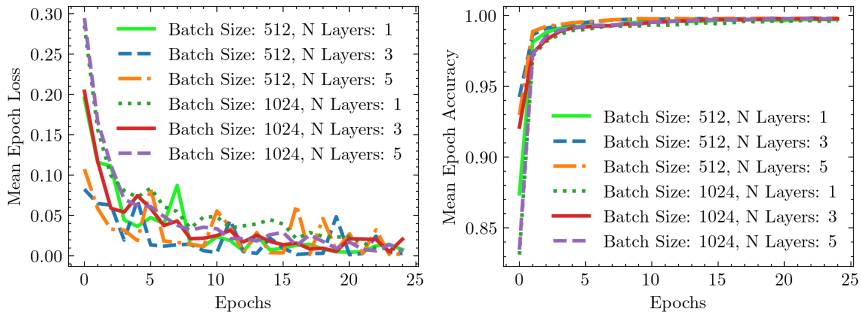
Transformer model was trained to perform intrusion detection on recordings of network flows that are sequentially ordered by their ending time. A sliding-window approach was used for network monitoring, as outlined in the work of Marino et al. [87]. In this method, features extracted from consecutive network flows were grouped into fixed-length windows, denoted as L_w. These windows serve as input sequences for the proposed transformer model. Additionally, each flow undergoes a normalization process through standardization. For a given window $W = \{x_1, \ldots, x_{L_w}\}$ of grouped IoMT Network flows fed into the encoder, a shifted version, W_{shifted}, which omits the last L_s flows, is provided to the decoder. The decoder's task is to reconstruct the original window of length L_w from its shifted version, which has a length of $L_w - L_s$.

Transformer model was fine-tuned for IoMT network flow by adjusting several hyperparameters, including the number of layers (N), window length (L_w), shift length (L_s), number of attention heads (H), epochs (E), and batch size (B). For the experiments, the shift length (L_s) was set to one unit for next-flow prediction, while the window length (L_w) was tested with values ranging from 50 to 100 flows. Stochastic Gradient Descent (SGD) and cross-entropy as the loss function was used for training Trasformer model. A single attention head was used since the CICIOMT dataset consists of a feature space with 44 attributes per flow, which is smaller compared to typical natural language processing tasks. The ReLU activation function was applied in the model.

Two classification studies were developed from the CICIoMT2024 dataset in IoMT attack detection using Transformer in publication 6. The studies focused on: 1) binary classification and 2) multiclass classification.

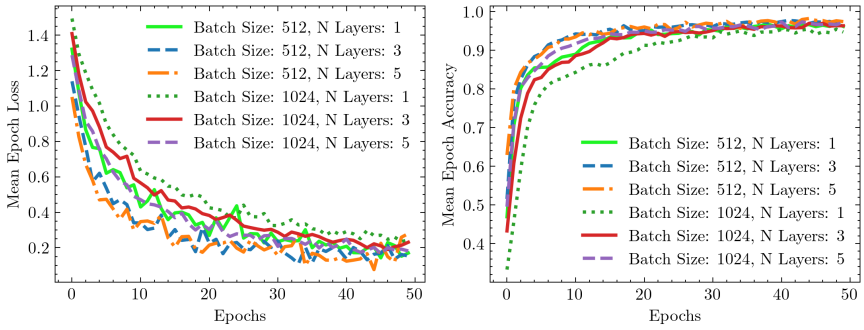
- In the binary classification task, the Transformer model is responsible for distinguishing between benign and attack traffic.
- Multi-class classification study involves categorizing specific types of attacks. The categories for IoMT attack detection include six types of network traffic: benign, MQTT attacks, DDoS, DoS, reconnaissance, and ARP spoofing attacks.

Figures 20 and 21 show the average training loss and accuracy of various transformer model configurations trained on the CIC IoMT dataset 2024 for binary and multi-class classifications. This evaluation involved testing various configurations of the transformer model to optimize its performance. The model configuration and training utilized a window length ($L_w = 100$) for both binary and multi-class classification tasks. The batch sizes were set to $B = [512, 1024]$. The number of encoder/decoder layers was varied with $N = [1, 3, 5]$. Training was conducted for 25 epochs for binary classification and 50 epochs for multi-class classification. Experiments demonstrated that with a window length of $L_w = 100$, models could fit the training data with just one layer. However, adding more layers improved performance, especially for multiclass classification task. Models with



(a) Loss *(b) Accuracy.*

Figure 20: Training loss and accuracy for Binary classification



(a) Loss *(b) Accuracy.*

Figure 21: Training loss and accuracy of Transformer model for Multiclass classification

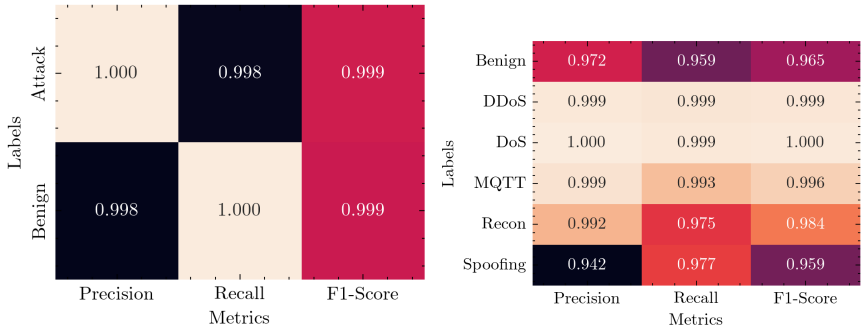
3 and 5 layers achieved lower loss values and higher accuracy for attack types and categories. The testing data was evaluated using a model with $N = 3$ layers and a batch size of $B = 512$.

Figure 20 shows the mean training loss and accuracy for a binary classification task. All model configurations converge quickly, with significant loss reduction within 5 epochs and over 99% accuracy achieved by 10 epochs, showcasing the transformer architecture's effectiveness. As illustrated in Figure 22a, the Transformer model achieves over 99% accuracy, showing perfect precision for benign traffic and nearly perfect recall for detecting attacks, effectively differentiating between normal and malicious network activities.

In multi-class classification, Figure 21 shows that models with 3 and 5 layers achieve lower loss values and higher accuracy more quickly than a single-layer model with a batch size of 1024, which may be under-fitting.

Figure 22b shows the strong performance of the Transformer model, achieving 97% accuracy on test data. Transformed excellently performed for multiclass classification at detecting DDoS and DoS attacks with F1-scores of 0.999 and 1.000, respectively. However, it performs slightly lower on benign traffic and spoofing attacks, with F1 scores of 0.965 and 0.959.

Table 14 compares the proposed transformer model with Decision Tree (DT) and k-



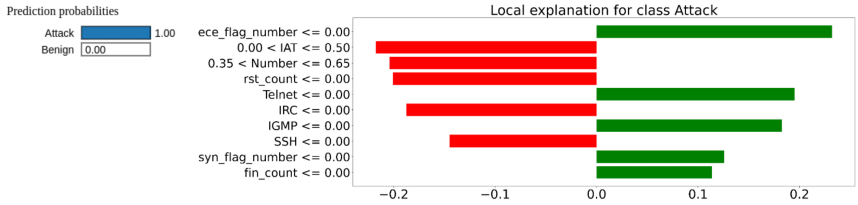
(a) Binary Classification *(b) Multiclass Classification*

Figure 22: Transformer Performance of Classification Report for Binary and Multi-class Classification IoMT attack detection

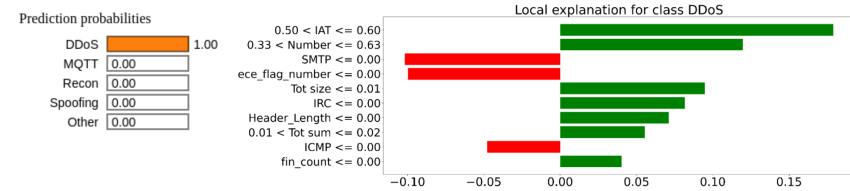
Table 14: Transformer Model Comparison with KNN and DT

Accuracy of Transformer Comparision			
Classification type/model	DT	KNN	Transformer
Binary Classification	0.9955	0.99621	0.99847
Multi-class classification	0.96167	0.88967	0.97426

nearest Neighbors (KNN) algorithms using the accuracy metric. The transformer model outperforms both DT and KNN in accuracy for IoMT binary and multi-class classification.



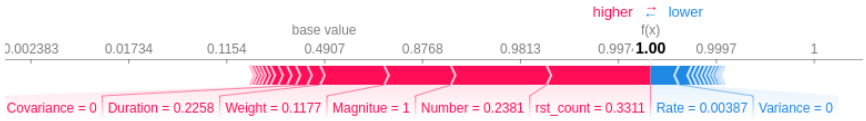
(a) LIME local explanations for Attack class instance



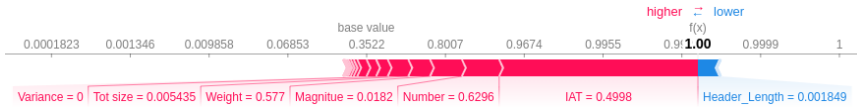
(b) DDoS attack instance LIME local explanations

Figure 23: LIME Local explanations for malware instances in Binary and Multi-class classification settings

To explain the predictions of the transformer model, LIME and SHAP XAI methods were employed in Publication VI. A description of these XAI methods can be found in Chapter 2.



(a) SHAP local explanations for DDoS attack instance



(b) SHAP local explanations for DDoS attack instance

Figure 24: SHAP Local explanations for multi-class classification instances

The output of the transformer model was obtained by applying the softmax activation function to the decoder output, resulting in class-wise prediction probabilities.

To demonstrate the post-hoc local explanations provided by the LIME and SHAP XAI methods, a single data point of test data from each class was selected to illustrate the local explanations offered by the XAI methods for both binary and multi-classification tasks. LIME method explains the reasoning behind the probabilities assigned to each class by comparing these probability values to the actual class of the data point. For example, Figure 23a presents LIME explanations for the "Attack" class of an instance. The left side of the figure displays the probability for each class, while the right side highlights the influential features that contributed to the prediction of this class. The transformer model used for binary classification achieved a 100% accuracy in predicting the "Attack" class label. On the right side of the bar chart, the features that helped predict the instance as belonging to the "Attack" class are shown with green bars. These features include 'Ece flag number', 'Telnet', 'IGMP', 'Syn flag number', and 'fin count'. Conversely, to predict an instance as not being an attack (indicated in red), the negatively influential features are 'IAT', 'Number', 'Rst count', 'IRC', and 'SSH'.

Similarly, Local explanations using LIME were provided for the transformer model prediction of the "DDoS" class label in selected instances of multi-class classification, as illustrated in Fig. $23b$. In this figure, the model predicted the DDoS class label with 100% accuracy. The features represented by the green bar indicate the most influential factors for predicting the 'DDoS' class, while the features shown in the red bar represent the most significant factors for predicting classes other than DDoS.

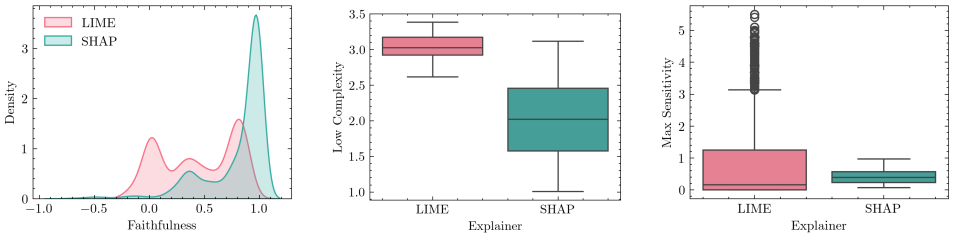
Figure 24a shows the SHAP force plot for the "Attack" class label in the binary classification of the Transformer model. It highlights the positive contributions of features such as 'Rst count', 'Number', 'Magnitude', 'Weight', 'Duration', and 'Covariance', which are shown in red strip. In contrast, the features 'Rate' and 'Variance' are represented in blue strip to indicate their negative contributions. The base value is 0.4907, which is the average of all prediction values. Since the positive contributions outweigh the negative ones, the final predicted value supports the classification as "Attack."

Figure 24b shows posthoc local SHAP explanations for the "DDoS" class label in the multi-class classification task of the Transformer model. Features such as 'Number', 'IAT', and 'Magnitude' positively influence the model output, indicated in red, which raises the prediction from a base value of 0.35 to 100%. In contrast, the 'Header length' feature, shown in blue, negatively impacts the prediction.

The quality of posthoc local explanations provided by the LIME and SHAP XAI methods were evaluated for the Transformer model using 2000 data points across both binary and

Table 15: Quality Evaluation of local explanations results

Metrics	Explainer	Binary Classification $\mu \pm \sigma$	Multi-class $\mu \pm \sigma$
High Faithfulness	LIME	**0.78 ±0.2**	0.44 ±0.34
	SHAP	0.04 ±0.74	**0.79 ±0.3**
Maximum Sensitivity	LIME	**0.63 ±0.18**	1.08 ±1.6
	SHAP	0.88 ±0.93	**0.41 ±0.2**
Low Complexity	LIME	3.00 ±0.11	3.04 ±0.15
	SHAP	**2.51 ±0.18**	**2.00 ±0.5**



(a) High Faithfulness *(b) Low Complexity* *(c) Maximum Sensitivity*

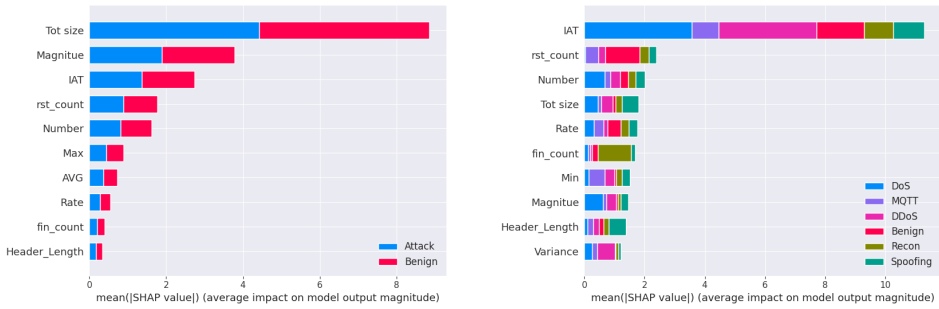
Figure 25: Distribution of Quality Evaluation Metrics for LIME and SHAP in Multi-class classification

multi-class classification scenarios. The evaluation employed three metrics: high faithfulness, maximum sensitivity, and low complexity. Detailed descriptions of these metrics can be found in Chapter 5.1. Table 15 presents the results of quality explanations for LIME and SHAP's local explanations. The mean (μ) and standard deviation (σ) values were calculated for the quantitative evaluation metrics of XAi methods computed on the test data.

In binary classification tasks, LIME outperforms SHAP in terms of the faithfulness metric. LIME scores 0.78 ±0.2, while SHAP only scores 0.04 ±0.74 which demonstrates that LIME explanations align more closely with the model's behavior in predicting outputs. Additionally, LIME demonstrates lower sensitivity at 0.63 ±0.18 compared to SHAP's 0.88 ±0.93, indicating that LIME explanations are generally more reliable for various inputs. Although both methods provide explanations, SHAP's explanations (2.51 ±0.18) are slightly less complex than LIME explanations (3.00 ±0.11).

In the context of multiclass classification, SHAP has demonstrated high faithfulness, lower complexity, and lower sensitivity (see in Table II). Figure 25 illustrates the distribution of evaluations of XAI metrics for test points in multi-class classification. SHAP explanations were considered more reliable because the average values of SHAP explanations for these metrics tend to correlate positively with high faithfulness and negatively with low complexity and maximum sensitivity.

A global explanation using a SHAP summary plot is provided (see Figure. 26) to highlight the most influential features in the transformer mode. This plot illustrates how each feature contributes to the model's output, averaged over the test data for the top 10 features. Figure 26a shows the global feature importance for binary classification, while Figure 26b presents the importance of detecting attack categories in multi-class classification. Key features such as 'IAT', 'Number', 'Rst count', 'Magnitude', and 'Tot size' were



(a) Binary Classification

(b) Multi-class classification

Figure 26: SHAP Global explanations for Transformer Model

found to be highly influential in both classification tasks.

7.1 Chapter Discussions

This chapter presents extension work aimed at addressing **RQ2**, which focuses on making state-of-the-art black-box models, such as transformers, transparent through the use of XAI for intrusion detection in IoMT cyber attack detection. The incorporation of XAI in IDS enhances the transparency of IoMT, helping to bridge the gap between complex ML models and human understanding. This transparency enables experts to understand the reasons behind specific decisions made by the Transformer model, thereby increasing trust and allowing for more informed responses to detected threats.

Two types of classification were studied based on the recently published CICIoMT2024 dataset: 1) Binary classification and 2) Multi-class classification. The proposed transformer model for Publication VI achieved an exceptional accuracy of 99.85% in Binary classification and 97.43% in Multi-class classification on the CICIoMT2024 dataset, outperforming traditional ML models like Decision Trees (DT) and K-Nearest Neighbors (KNN).

LIME and SHAP, as XAI methods, were utilized to provide transparency regarding the decisions made by the transformer model for both classification types. A quantitative evaluation of the employed XAI methods for the transformer model in Binary classification showed that LIME exhibited greater faithfulness, robustness, and lower complexity compared to SHAP explainers. On the other hand, SHAP demonstrated superior performance compared to LIME for multi-class classification when detecting types of IoMT attacks.

8 Enhancing IoT Botnet Detection Through Explainable Active Learning (XAL) Paradigm

This chapter introduces an explainable active learning paradigm that integrates post hoc explainability into the active learning process. This approach aims to improve model performance while promoting transparency in intrusion detection tasks, particularly for detecting IoT botnet attacks. The work presented in this chapter is based on Publication VII.

8.1 Background on Active Learning

In the active learning approach, often viewed as a type of semi-supervised learning, a supervised learning algorithm selects specific samples—typically the most informative ones—from an unlabeled data set. These selected instances are then sent to experts for labeling. The newly labeled instances are utilized to enhance the model's knowledge, which is initially derived from a small set of labeled data samples.

The core concept of active learning (AL) is that a learning algorithm intelligently chooses which instances to label next, allowing it to achieve good performance with significantly less training data [122]. AL approach addresses a significant issue in modern ML: obtaining labeled data can be both time-consuming and costly [157]. Active learning can be applied in various scenarios [27], such as stream-based (from a stream of incoming data) [73] and pool-based (from a large set of unlabeled instances) [122]. To choose the next instance for labeling, various query sampling strategies have been proposed in the literature [30, 33, 73]. The most common approach is the pool-based active learning method. In this approach, active learning assumes the existence of a small set of labeled data and a large pool of unlabeled data. Instances are selected from the unlabeled pool for annotation by an expert. The labeled samples are then incorporated into the training set, which is used to update the model. For Publication VII, Two active learning query strategies were tested: 1. Uncertainty Sampling [122] and 2. Query by Committee Sampling[122].

Uncertainty sampling is a widely used approach for selecting data points to label, where data points are chosen one by one based on a measure of the model's uncertainty. In this approach, the classifier makes label predictions for all unlabeled data points and then selects the one about which it is most uncertain, i.e the point for which the classifier has the lowest confidence score in its prediction. There are various strategies to measure the level of uncertainty associated with a prediction for a data point x and its predicted label y.

- Classification Uncertainty: A classifier identifies the instance x for which it has the least confidence in assigning a label, indicated by a high uncertainty score. This uncertainty is calculated using the least confidence score, defined as:

$$U(x) = 1 - P(\hat{y}|x).$$

  $P(\hat{y}|x)$ is probability of the predicted label given the instance x.

- Classification Margin: Classification margin is a score that calculates the difference between the probabilities of the most likely and the second most likely predictions. The smallest margin represents the greatest uncertainty, so the sample with this smallest difference is selected. The classification margin score is defined as

$$M(x) = P(\hat{y_1}|x) - P(\hat{y}|x)$$

- Classification entropy: Classification entropy (H) is a score that uses entropy to quantify the uncertainty of a classification. The data point selected is the one with the highest entropy score, which is calculated using the following equation:

$$H(x) = \sum_k p_k log(p_k)$$

, In the above, p_k is the probability that a given unlabeled sample belongs to the k^{th} category, based on the current state of knowledge of the classifier.

Another commonly used approach for data point selection in active learning (AL) is known as query-by-committee (QBC). In this method, a group of diverse models, referred to as a committee, is trained on the same labeled data but represents different hypotheses within the hypothesis space. Each model in this group votes on how to classify new examples. The most informative sample is the one that generates the most disagreement among the models regarding its class. Various methods can be employed to measure the level of disagreement, including Vote Entropy (VE), Consensus Entropy (CE), and Maximum Disagreement (ME).

- Vote Entropy: Vote entropy selects the query instance with the highest entropy in the vote distribution. It is calculated as below.

$$VE = \text{argmax}_x \left(-\sum_i \frac{V(y_i)}{C} \log \left(\frac{V(y_i)}{C} \right) \right)$$

  In this equation, y_i is each possible class label, $V(y_i)$ is the count of votes that a particular label receives from the committee for a specific instance x, and C denotes the total number of members in the committee.

- Consensus entropy is determined by first calculating the average predicted probabilities for each class from all classifiers. This average is referred to as the consensus probability (P_{cs}). Next, the entropy of this consensus probability is computed. The sample with the highest consensus entropy is then chosen for labeling.

  The disagreement score is calculated as follows:

$$CE = -\sum_y P_{cs} \log(P_{cs})$$

  where $P_{cs} = \frac{1}{C} \sum_{c=1}^{C} P(y_i)$ is the consensus probability.

- Maximum disagreement: It quantifies the level of disagreement among classifiers by first creating a consensus probability. This consensus is calculated as the average of the class probabilities predicted by each classifier. Instead of using entropy, it employs Kullback-Leibler divergence to assess how much each classifier's predictions deviate from the consensus. The sample that shows the greatest divergence is then chosen as the query instance. The score of disagreement is computed as,

$$MD = \text{argmax}_x - \frac{1}{C} \sum_y D(P_{\theta^{(C)}} || P_{cs})$$

  where $D(P_{\theta^{(C)}} || P_{cs}) = \sum_y P(y_i|X;\theta^{(c)}) log \frac{P(y_i|x;\theta(C)}{P_{CS}}$ is the corresponding Kullback–Leibler divergence. Here $\theta_{(C)}$ denotes a specific classifier model within the committee, and P_{CS} represents consensus probability.

8.2 XAL paradigm for IoT botnet detection

Active learning is especially useful in situations where collecting data is easy and inexpensive, but labeling that data is costly. The intrusion detection problem exemplifies this scenario, as it is relatively simple to gather raw network or host data and process relevant network traffic for machine learning algorithms. However, allocating the necessary resources for labeling tasks is challenging due to the limited number of human experts with sufficient skills to perform these tasks. Moreover, concerns about confidentiality often hinder organizations from sharing any data, whether raw or labeled, which worsens the issue.

For Publication VII, an Explainable active learning (XAL) paradigm was proposed for detecting IoT botnets, as shown in Figure 27. This paradigm integrates active learning (AL) with XAI to boost model performance while ensuring transparency. The process starts with a classifier trained on a limited set of labeled samples, known as the initial seed. The model selects informative instances from an unlabeled pool, which are then analyzed using a post hoc XAI method to provide local explanations. These instances and their explanations are presented to security analysts, who review the information and assign final labels. The reliability of these explanations is evaluated in each active learning cycle through quantitative metrics, including monotonicity, faithfulness and sensitivity. This evaluation assesses the extent to which the explanations accurately represent the model's actual decision-making process across query rounds. The labeled instances are used to retrain the ML model, enhancing its performance and promoting collaboration between humans and machines by building trust. XGBoost (eXtreme Gradient Boosting) classifier was selected for all benchmarking scenarios in the active learning process discussed in Section 8.1.

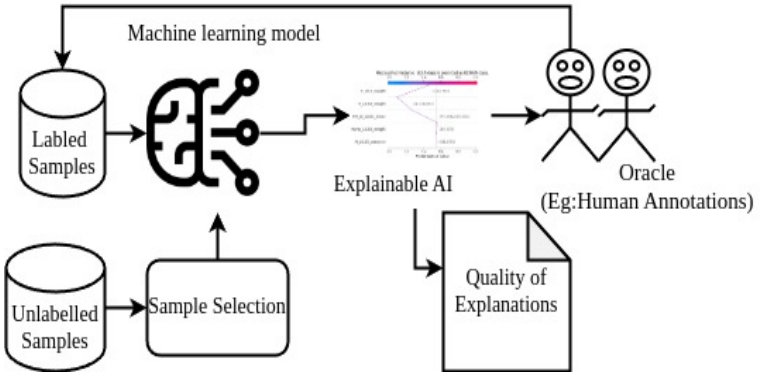


Figure 27: Explainable Active Learning (XAL) Framework for IoT Botnet Attack Detection

For Publication VII, the N-BaIoT dataset was utilized, from which an attack-type multiclass classification was developed. Data points were classified into eight attack types and legitimate network traffic: ACK, Benign, COMBO, JUNK, SCAN, SYN, TCP, UDP, and PLAIN. The N-BaIoT dataset consists of a total of 115 features. A description of these attack class labels can be found in Table 2 in Chapter 4. To eliminate redundant and irrelevant data, Pearson's linear correlation coefficient (r) was used. Feature that exhibited a high correlation ($|r| > 0.80$) with any other feature were removed, retaining only one of the correlated features. Fisher score [2] was used to rank these 33 features. Various subsets of features were tested, including the top 3, 5, 10, 15, and 20. For the evaluation of each feature set, a complete dataset sample of 100,000 instances was utilized. The dataset was divided into

training and testing sets using an 80% to 20% ratio. Performance results for the XGBoost (XGB) model are illustrated in Figure 28. The top 3 and top 5 features achieved the highest detection rates in terms of accuracy, F1 score, recall, and precision. Consequently, for this analysis of XAL paradigm, the top 5 features were selected and are presented in Table 16.

For the experimental setup of active learning, the N-BaIoT dataset was divided into 80% for training and 20% for testing to benchmark different active learning label strategies. Active learning requires a substantial number of unlabeled data points, known as a pool. In Publication VII, this pool consistently contained 100,000 data points drawn from a larger training dataset. The testing dataset was used exclusively to evaluate the learning outcomes using the F1-score metric for XGBoost and was not involved in the training process at any stage of the active learning cycles. Ten iterations were performed for each model during the active learning cycle. In these iterations of the active learning cycle, in addition to evaluating the models' performance, this study also applied explainable AI techniques such as LIME and SHAP to the test data.

Table 16: Top-5 Selected Features for Active Learning-Based IoT Botnet Detection Using XGBoost

Network Category	Feature
Host	H_L0.01_variance
	H_L0.1_weight
	H_L0.01_weight
Socket	HpHp_L0.01_weight
Network-jitter	HH_jit_L0.01_mean

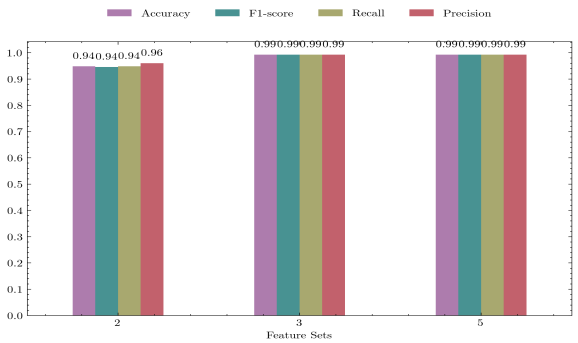
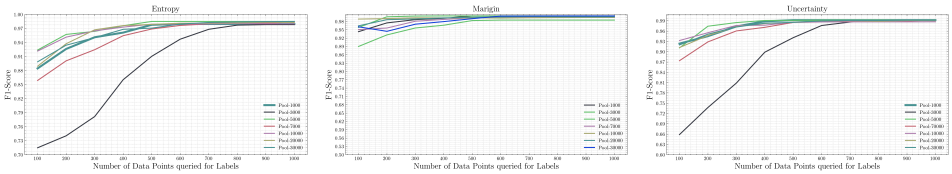


Figure 28: Feature Sets Comparisons with metrics for for Active Learning-Based IoT Botnet Detection.

Figure 29 illustrates the F1-score performance of active learning cycles employing three uncertainty sampling strategies: classification entropy, classification margin, and classification uncertainty. These strategies were applied to active learning in the context of IoT botnet attack detection using the XGBoost model. The experiment commenced with an initial set of 20 labeled data points, and various pool sizes (1000, 2000, 3000, 5000, 7000, 10000, 20000, and 30000) of unlabeled data points were used in the study. The training sets were balanced in all cases. During the active learning training experiment, up to 1000 data points were selected from the pool and presented to a human expert for labeling. The selection of data points for querying was based on the classification uncertainty scores derived from the three uncertainty sampling strategies.

Figure 29a illustrates the F1-score performance of the XGBoost model using Entropy sampling across various pool sizes (ranging from 1000 to 30000) and different amounts of



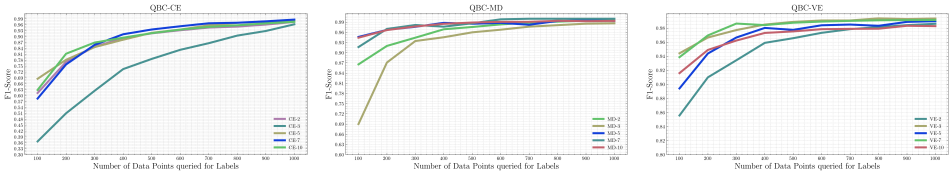
(a) Classification Entropy *(b) Classification Margin* *(c) Classification Uncertainty*

Figure 29: Uncertainty sampling: classification uncertainty results.

data points queried for labels (ranging from 100 to 1000). XGBoost model achieves an F1 score of approximately 0.97 to 0.99 for most pool sizes when querying 1,000 data points for labels, demonstrating high performance in detecting attack labels.

In Margin Sampling (in Figure 29b), F1-score increases with the number of data points queried for labels, similar to Entropy Sampling. When querying 1000 data points for labels, the model achieves F1 scores between 0.95 and 0.98 for most pool sizes.

F1-score of uncertainty shown in Figure 29c aligns well with the results from the previous two sampling techniques. F1 score improves as more data points are queried for their labels, demonstrating the effectiveness of AL. XGBoost achieves F1 scores ranging from 0.96 to 0.99 for most pool sizes when querying 1,000 labeled data points. All three classification uncertainty strategies show that increasing the number of queried data points consistently results in higher F1 scores across all pool sizes, which indicates that model performance improves with more labeled data. Notably, larger pool sizes (e.g., 20000 and 30000) tend to achieve slightly higher F1 scores compared to smaller pool sizes, especially when the number of queried data points is low (between 100 and 400).



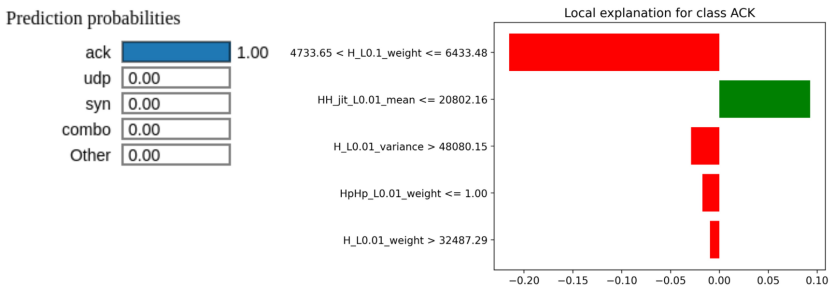
(a) Consensus entropy *(b) Maximum disagreement* *(c) Vote entropy*

Figure 30: Query by committee results

Another active learning query selection strategy used in Publication VII is query by committee. The query-by-committee approach involves a group of independent classifiers that select data points from the unlabeled pool to label queries. Figure 30 shows the performance of the QBC approach during active learning cycles using the XGBoost model. The level of disagreement among the classifiers in the committee was measured using various disagreement scores: the consensus entropy score (QBC-CE) illustrated in Figure 30a, the vote entropy score (QBC-VE) presented in Figure 30c, and the maximum disagreement score (QBC-MD) shown in Figure 30b.

Experiments were conducted using various committee sizes (2, 3, 5, 7, and 10), with each committee member being an independent XGBoost classifier. A fixed dataset of 10,000 data points was utilized, and the seed size was set to 100. The results showed that high F1 scores for all three disagreement scores improved as the number of labeled data points increased. Among the different committee sizes, a committee size of 7 consistently outperformed the others across all disagreement metrics. As illustrated in Fig-

ure. 30b, QBC-MD with a committee size of 7 (QBC-MD-7) achieved the highest F1 scores among the three disagreement metrics which demonstrates the effectiveness of the QBC approach in actively selecting informative data points for labeling, leading to improved model performance.



(a) LIME Explanations



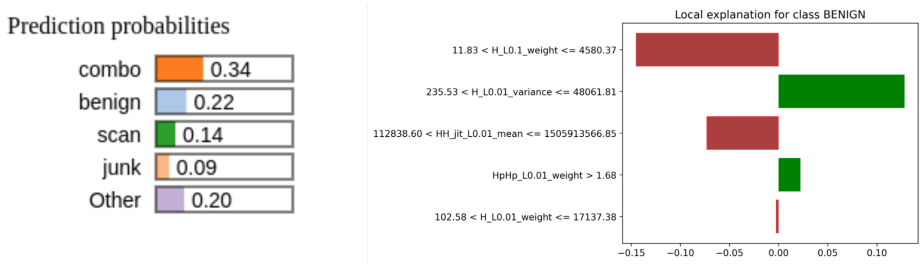
(b) SHAP Local Explanations

Figure 31: LIME & SHAP Local Explanations for Correctly Classified Instance (ACK Attack)

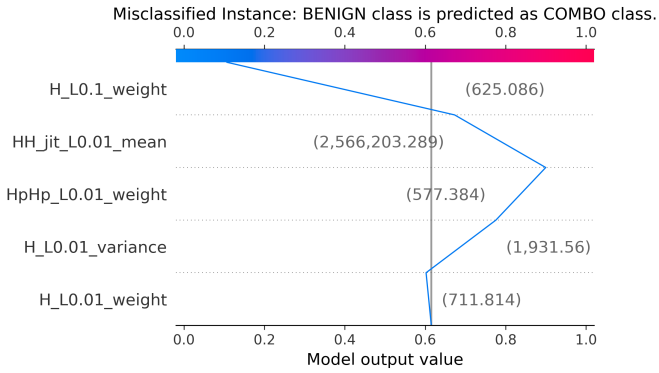
In Publication VII, two post hoc explainability methods were used and compared to understand predictions of IoT botnet attacks within the active learning loop. The oracle, a security analyst, uses these explanations to better understand the decision-making process of the XGBoost model when predicting IoT botnet attacks. By integrating LIME and SHAP with active learning, a strong foundation of trust and transparency is built between the oracle and the learning model. For a security analyst, it is essential to understand why a model made a particular prediction. This understanding is crucial for determining whether to trust the model's output and for providing accurate labels or feedback to improve the model's training. Specifically, post hoc methods like LIME and SHAP explain the model's decisions by assigning an importance value to each feature.

LIME and SHAP explanations for the model's features, particularly regarding a correctly classified sample identified as an ACK attack type, are presented in Fig. 31. In the LIME explanations (see Fig. 31a), red bars represent features that contribute to correctly predicting the data point as an ACK attack label, while green bars indicate features that contribute to predicting the data point as belonging to other classes according to the model. In these visualizations, prediction probabilities are assigned to each class displayed on the left side of the figure. LIME explanations for the features show that the XGBoost Model accurately predicted the ACK class with 100% accuracy for the selected actual class label.

SHAP force plot is a valuable tool for explainability, visually representing the contribution of each feature to a specific prediction. It illustrates both the relative importance and the direction of impact of these features. In the context of local explanations with SHAP, a specific data point is chosen, and the model's prediction is explained by highlighting each feature's contribution, which is quantified using Shapley values. Figure 31b demonstrates



(a) LIME Explanations for Misclassified Instance



(b) SHAP Explanation for Misclassified Instance

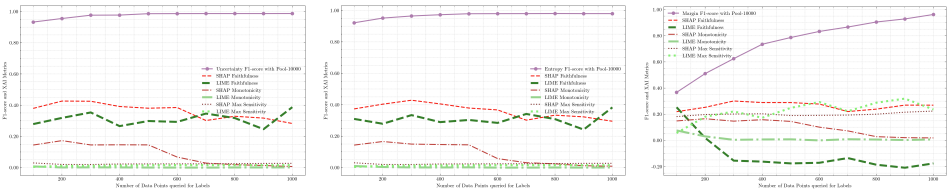
Figure 32: LIME & SHAP Local Explanations for Misclassified Instances

a local explanation for a data point through a SHAP force plot, clearly showcasing how each feature contributes to the overall prediction. The plot illustrates the base value, with features that positively influence the prediction displayed in red, while those that have a negative impact are shown in blue. The base value represented in the plot is the average of all prediction values. Each strip in the plot indicates the influence of the features in pushing the predicted value closer to or farther from the base value. Red strips indicate features that push the value higher, whereas blue strips indicate those that push it lower. Wider strips represent a greater contribution to the prediction.

Based on Figure 31b, the base value for the ACK class attack is 0.5783. The feature "H_L0.01_variance" positively contributes to the prediction value and is the most crucial feature due to its larger contribution range. Overall, the total positive contributions exceed the negative contributions, resulting in a final predicted value higher than the base value. Therefore, the model correctly classifies the instance as an ACK attack.

Figure 32 illustrates how LIME and SHAP can be utilized to explain misclassified instances, particularly in cases where a benign traffic instance is incorrectly classified as a COMBO attack. In the LIME explanation for the misclassified instance (see Figure. 32a), the prediction probabilities indicate that the likelihood of a COMBO attack is 0.34, while the probability of it being benign is 0.22, and the probability of it being classified as a scan attack is 0.14. The feature contributions for this misclassified prediction are shown on the left side of the figure, with red bars representing features that support the prediction of a COMBO attack and green bars indicating features that contribute to the predictions of other classes.

A SHAP explanation (see Figure. 34b) is provided for the same misclassified instance,



(a) Classification Uncertainty *(b) Classification Entropy* *(c) Classification Margin*

Figure 33: XAI metrics results for Uncertainty sampling strategies



(a) Vote entropy *(b) Consensus entropy* *(c) Maximum disagreement*

Figure 34: XAI metrics results for QBC strategies

where a benign class is incorrectly predicted as a COMBO attack. In this explanation, the SHAP contribution line shifts toward the blue line instead of the red red line which indicates that the features are influencing the prediction toward the COMBO class, even though the true label is benign.

Using LIME and SHAP explanations, the oracle can determine whether a model depends on irrelevant features, thus guiding the selection of the most informative instances for the next iteration of model training in active learning. SHAP explanations enable annotators to understand the relative importance and influence of each feature on an individual prediction. These post-hoc local explanations can help identify features that are consistently important across different predictions, assisting in prioritizing which data points should be labelled next. Additionally, the oracle utilizes these explanations to assess the validity of the model's predictions and provide feedback for model improvement.

The quality of post-hoc explanations was evaluated within the active learning loop by applying three quantitative metrics. These metrics were used to determine whether the data points selected from the pool and presented as labeling queries to the human expert were effectively explained by the model. The evaluation of explanation quality was conducted using the following three metrics: 1) high faithfulness, 2) monotonicity, and 3) maximum sensitivity. Descriptions of these metrics are available in Chapter 5.1.

High Faithfulness metrics compute the correlation between the feature importance scores provided by the explainers and the actual model predictions. Monotonicity measures the influence of individual features on model prediction probabilities by evaluating how the prediction probability changes when each attribute is incrementally added in order of increasing importance. As each feature is added, the model's probability consistently increases, resulting in monotonically increasing model prediction probabilities. A high monotonicity score suggests that the explanations by XAI method are consistent with the model's predictions for the given input. Maximum Sensitivity, used to assess the robustness of the explainers, is based on the premise that similar instances in feature space should give similar explanations.

Results of the XAI metrics were presented exclusively for the models that showed

the highest performance across various sampling strategies. Specifically, a pool size of 10,000 yielded the best results for uncertainty sampling. Moreover, combining a pool size of 10,000 with a committee size of seven produced superior outcomes in the QBC approach. The performance results for the XAI metrics, illustrated in the graphs, represented the mean values of these metrics for explainers over the test data throughout the active learning cycles.

Figure. 34 illustrates the XAI metrics results for QBC strategies in the AL loop. Mean values of evaluation explainability metrics were computed for test data using explanations generated by LIME and SHAP explainers throughout the AL cycle. In QBC strategies, SHAP explanations consistently outperform LIME explanations in terms of faithfulness, monotonicity, and maximum sensitivity. The only exception is that neither SHAP nor LIME have consistently achieved better results for the faithfulness of consensus entropy. In the QBC strategies that utilize a maximum agreement strategy with a committee size of 7 (QBC-MD-7) (see Figure. 34c), as the number of data points queried for labels increases, the faithfulness of SHAP explanations remains high and stable, with correlation values consistently above 0.8. On the other hand, the faithfulness of LIME explanations is lower and more variable, with correlation values ranging from 0.4 to 0.6. However, the monotonicity of both explainers improves as more data points are queried, indicating that the explanations become more consistent with the model's predictions as active learning progresses. Using the monotonicity metric, SHAP demonstrates a higher level of consistency, with monotonicity percentages consistently exceeding 60%. In contrast, LIME's monotonicity consistency percentages range from 40% to below 60% in the QBC-MD-7 results. Both explainers show a decrease in maximum sensitivity as more data points are queried, which suggests that the explanations become more robust over time. Lower sensitivity values indicate that similar instances in the feature space yield similar explanations. SHAP's maximum sensitivity values are consistently lower than those of LIME, highlighting that SHAP explanations are more reliable. For the QBC-MD-7 model, which achieved the highest F1 score, SHAP explanations exhibit high faithfulness correlation values (above 0.8), consistent monotonicity percentages (above 60%), and low maximum sensitivity values, all of which indicate robust explanations.

QBC-CE-7 (see Figure 34b) and QBC-VE-7 (see Figure 34a) show similar outcomes in xai metrics. In particular, SHAP explanations outperform LIME in terms of faithfulness, monotonicity, and maximum sensitivity. However, the QBC-MD-7 strategy consistently achieves the highest F1 scores and provides the most reliable explanations among the three query strategies that utilize SHAP.

In the context of uncertainty sampling strategies, as shown in Figure 33, SHAP explanations were more reliable than LIME. However, when comparing the QBC strategy, the metrics for faithfulness and consistency of monotonicity showed fluctuations. This variability highlights that the explanations related to query strategies may not always be dependable.

8.3 Chapter Discussions

This chapter addresses **RQ3** by proposing the XAL (Explainable Active Learning) paradigm adopted for IoT botnet detection. This paradigm integrates post-hoc explainability methods directly within the active learning loop to enhance both model performance and explainability. Though previous research in active learning for IDS has mainly focused on improving performance of the model through active learning query strategies, it has paid little attention to the quality of explanations provided during the labeling process—a crucial aspect in human-in-the-loop systems where SOC analysts rely on interpretable insights

to assign accurate labels and build trust in the model.

Publication VII addresses this critical gap by incorporating LIME and SHAP explainers in each iteration of the active learning cycle and evaluating the quality of explanations using three quantitative metrics: faithfulness, monotonicity, and maximum sensitivity. The findings of this work disclosed that the reliability of explanations varies across different sampling strategies in AL cycle. Among the tested strategies, the Query-by-Committee with Maximum Disagreement (QBC-MD-7) achieved not only the highest F1-score performance but also produced the most consistent and faithful explanations based on the evaluation of XAI metrics. Notably, SHAP explanations performed better than LIME in terms of faithfulness and robustness.

9 Privacy-Preserving Explainability in Federated Learning for IoT Threat Detection

The following chapter discusses the integration of explainable AI methods in federated learning settings for privacy-preserving explainability in botnet detection within IoT networks. This chapter focuses on the main contributions and findings of Publications VIII and Publication IX.

9.1 Explainable Federated model for IoT botnet detection

ML/DL-based Intrusion Detection Systems analyze network traffic, system logs, or host data to detect threats by identifying patterns and anomalies. IDS continuously monitor data, classifies it as normal or malicious, adapts to new threats, and issues alerts for detected intrusions. However, Many IDS approaches rely on centralized ML models that require significant private data from IoT edge devices, which can threaten user privacy. In particular, in sensitive fields like healthcare and finance, transferring confidential data to a centralized entity for training a DL model to detect malicious activities might not be impractical due to privacy concerns. Federated Learning (FL) enables the collaborative training of a global ML model across distributed nodes while keeping local data private. Each node updates its model with its own data and shares the updated model with a central server, which aggregates these models into a global one through an iterative process. This method protects privacy by preventing raw data sharing among clients.

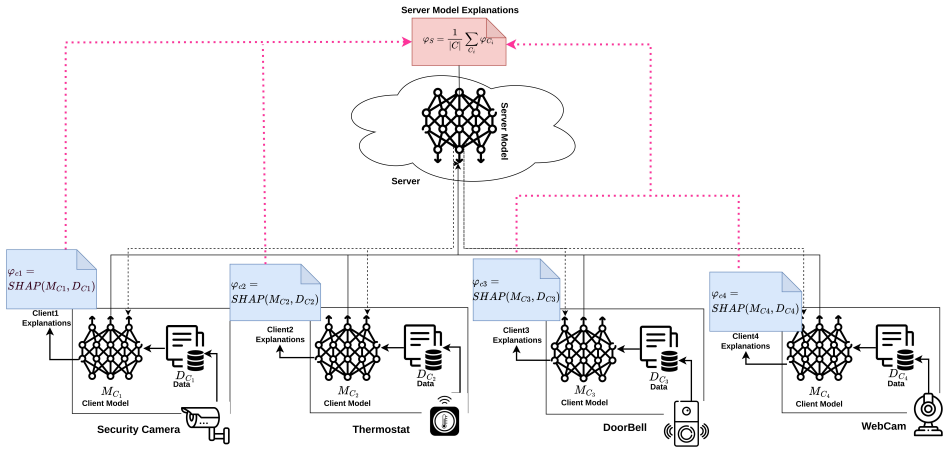


Figure 35: FedXAI (Federated Learning of Explainable AI) paradigm for Client-Server Intrusion Detection Model Explanations in IoT Network Devices

Trustworthy AI is becoming increasingly important in cybersecurity, particularly for enhancing the security and reliability of the black box nature of DL-based IDS. Recent advancements have led to the development of Federated Explainable Artificial Intelligence (FEDXAI), which integrates FL with XAI. The combination of these frameworks is crucial for building trust in AI-based IDS, as it allows for transparency while maintaining privacy. Two key paradigms that enhance trustworthy IDS with DL are FL and XAI. FL prioritizes data privacy through collaborative learning on decentralized data. On the other hand, XAI enhances transparency, accountability, and trust in FL-based IDS by providing clear explanations of its predictions and decisions, helping security analysts verify detected threats. In an IDS operating in a FL setting, security experts responsible for managing the server-

side models must comprehend how these models make decisions regarding intrusions affecting client nodes. In these situations, finding a balance between the need for privacy and the need for explainability presents a significant challenge. Integrating Posthoc XAI methods into FL has not received much attention and presents additional challenges due to FL's complex, distributed nature, where the model is trained across multiple client nodes such as IoT devices and sensors. Additionally, Post-hoc XAI methods, such as LIME or SHAP, for model interpretability require full access to the training dataset and model parameters, which can pose privacy risks.

Publication VIII explores the use of post-hoc XAI in FL for detecting botnets in IoT networks. The goal of this publication is to enhance the explainability and transparency of decision-making processes in FL-based IDS, all while maintaining data privacy. Figure 35 illustrates the FedXAI-based IDS framework for providing client-server model explanations in IoT network devices. HFL (Hierarchical Federated Learning) architecture was proposed for IDS in IoT networks connects various devices across different locations. These devices monitor their network traffic using tools and employ local host-based IDS. Selected IoT devices share their DL model parameters with a central server, which aggregates these to create an improved intrusion detection model. This collaborative learning approach enhances learning by enabling devices to detect intrusions based on behaviour patterns learnt from other participating devices.

Table 17: Number of Botnet malware & benign samples for each device over N-BaIoT dataset [92]

IoT Devcie Name	Short Name	Deployed BoTnet Samples		
		Mirai	Gafgyt	Benign
Danmini Doorbell	Doorbell1	652100	316650	49548
Philips B120N10 Baby_Monitor	BabyMonitor	610714	312723	175240
Provision PT 737E Security_Camera	SecurityCam1	436010	330096	62154
SimpleHome XCS7 1003 WHT Security Camera	SecurityCam2	514860	316438	19528
Ecobee Thermostat	Thermostat	512133	310630	13113
Provision PT 838 Security Camera	SecurityCam3	429337	309040	98514
SimpleHome XCS7 1002 WHT Security Camera	SecurityCam4	513248	303223	46585
Ennio Doorbell	Doorbell2	-	316400	39100
Samsung SNH 1011N Webcam	Webcam	-	323072	52150
Total		3668402 (57.9%)	2198800 (34.7%)	464682 (7.4/%)

The N-BaIoT dataset is a suitable dataset that enables us to evaluate privacy-preserving collaborative training for IoT botnet and B5G applications. This dataset contains network traffic data from nine different IoT devices that have been infected with the Mirai and Gafgyt malware, as well as legitimate traffic data. All devices generated traffic while non-corrupted (benign samples) and when infected by Mirai and BASHLITE. However, the Ennio doorbell and the webcam did not generate traffic while infected by Mirai. Table 17 displays the number of benign and botnet malware samples for each device, along with the total count.

In FL for IoT botnet detection, a deep neural network (DNN) was proposed and trained using data distributed across multiple IoT devices to identify IoT botnets in the N-BaIoT dataset by leveraging network traffic features. Each client's data is split into training and validation sets using an 80:20 ratio. In each round of federated training, the validation data from each client is tested on its respective client model. After processing all clients, the combined validation data from all clients is evaluated on the server model. To prepare

the data for training the DNN model, Min-Max normalization was applied to scale the feature values between 0 and 1 for both the training and validation data of each client.

The DNN model includes an input layer with 115 neurons, which corresponds to the number of network traffic features in the dataset. The model was trained using the Rectified Adam (RAdam) optimizer with a learning rate of 0.000669451. After each hidden layer, the Scaled Exponential Linear Unit (SELU) activation function was applied to introduce non-linearity into the model. The output layer of the model was a fully connected layer that maps activations from the final hidden layer to the output size, which corresponds to the number of classes for botnet malware type classification: Benign, Mira, and Gafgyt. During the training phase, each client model underwent 90 epochs with a batch size of 512 for the data of each client in every communication round. The Cross Entropy Loss function was used to compute the loss during training, making it well-suited for this multi-class classification task.

The proposal for Publication VIII aims to use explainable AI to interpret a DNN model trained with the FedAvg algorithm in a HFL setup for IoT botnet detection. Publication VIII focuses on utilizing the N-BaIoT dataset, which includes nine IoT devices that have been attacked by botnets such as Gafgyt and Mirai. Since The Mirai botnet did not impact the 'Doorbell2' and 'Webcam' devices, therefore, these two were excluded from the experiments to maintain consistency in the label space and input feature space during federated training using the FedAvg algorithm, which follows a horizontal approach to FL.

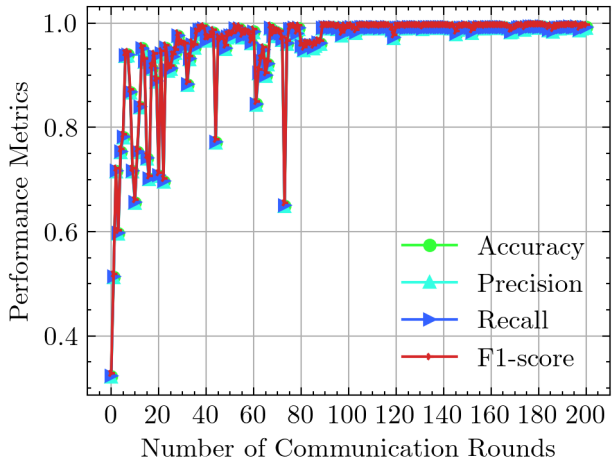
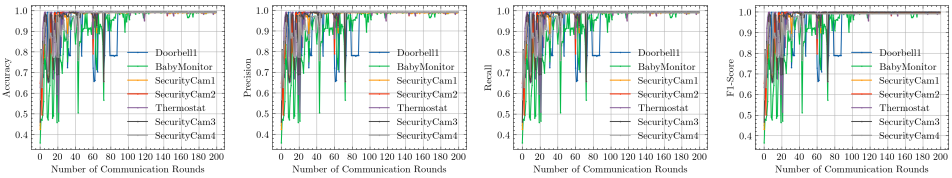


Figure 36: Evaluation of DNN Model Performance on Server side

Figure 36 shows the performance of the federated DNN model in terms of accuracy, precision, recall, and F1-score for server model. The evaluation used test data from 7 IoT devices over 200 communication rounds with the FedAvg algorithm in a HFL setup. For each round, at least 50% of clients were randomly selected, ensuring diverse updates and robust generalization. All seven clients were required for the evaluation round to ensure reliable model performance estimates. In Figure 36, the evaluation of the server model shows that the accuracy, precision, recall, and F1 score metrics initially sat at around 0.4. After 200 communication rounds, these metrics significantly improved, reaching approximately 0.99.

Figure 37 illustrates the classification performance metrics of client DNN models using their own parameters on validation data. As communication rounds increased, accuracy,



(a) Accuracy *(b) Precision* *(c) Recall* *(d) F1-score*

Figure 37: Evaluation of DNN model performance on client's Side

precision, recall, and F1-score improved for each client, indicating successful learning. Although there was some variation in performance among clients in the initial communication rounds, nearly all achieved high classification performance by the end, demonstrating the model's strong effectiveness on the validation data.

A post hoc explanation, denoted as g, maps a black box model predictor M and a point of interest x to an importance score $g(M,x) = \varphi_x \in \mathbb{R}^d$ for all features. SHAP (SHapley Additive exPlanations) is defined as $g(M,x) = \phi_0 + \sum_{j=1}^{M} \phi_j$, where ϕ_j is the feature attribution for feature j. SHAP values can provide both local explanations for individual instances and global explanations for the model as a whole.

SHAP (SHapley Additive exPlanations) is defined as $g(M,x) = \phi_0 + \sum_{j=1}^{M} \phi_j$, where ϕ_j is the feature attribution for feature j. SHAP values can provide both local explanations for individual instances and global explanations for the model as a whole. In publication 8, global explanations for server model were derived without sacrificing client data which is important because post hoc explanations like SHAP and LIME require input reference data or training data, or background data to achieve feature importance-based explanations. Consequently, In server-based FL settings, explaining the learned global model requires the server to have access to the complete training dataset (D^{Train}) from its clients. Alternatively, the server should be able to compute the centroids of a dataset created by combining the training sets of all clients. However, sharing client data or aggregated information raises privacy concerns, as sending client data to the server contradicts the fundamental principle of decentralized learning in FL. Thus, In publication VIII, an explanation of the server model was proposed by aggregating all client SHAP explanations, thus avoiding data sharing with the server model.

After completing the training of FL, each client c_i receives its trained model M_{c_i}, which is then sent to the server. The server creates its own model M_s by averaging the weights of the received models using the FedAvg algorithm. Each client c_i can then obtain a SHAP explanation $g_{c_i}(M_{c_i},x)$ for its data points based on its model M_{c_i}, which strongly relies on the training data.

To derive explanations from the server's perspective, the additive nature of SHAP values was leveraged which allows for the generation of explanations for the server model M_s as an aggregation of explanations from the individual client models M_c within the set U. Where U is a set of participating clients in the federation. Therefore, for a point of interest x, an explanation of the prediction made by the server model M_s can be obtained using the following equation.

$$\mathbf{g}_s(\mathbf{M}_s,x) = E_{local} = \frac{1}{|N|} \sum_{c_i \in N} \mathbf{g}_{c_i}(\mathbf{M}_{c_i},x). \tag{21}$$

To evaluate whether the local client explanations E_{local} were sufficient for the server model, the server model M_s was provided with data from all clients N to generate explana-

tions. These explanations were then aggregated by combining the training data and using the SHAP explainer, resulting in a global explanation E_{global}. Finally, to assess the accuracy of the explanations, the difference (d_{g-l}) between E_{global} and E_{local} was computed.

$$d_{(g-l)} = E_{global} - E_{local}. \tag{22}$$

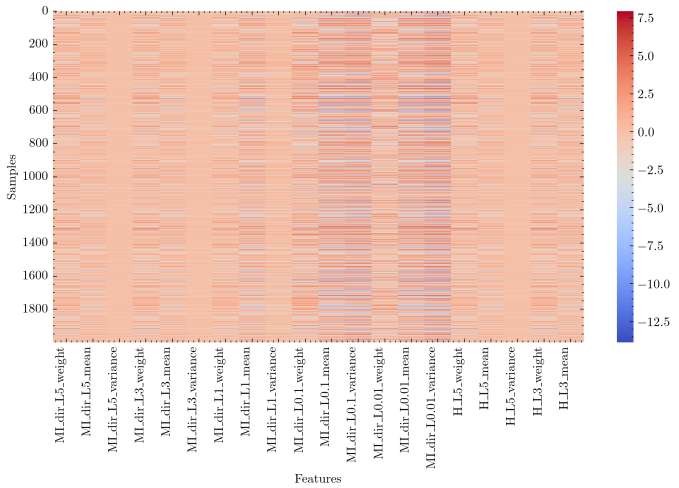
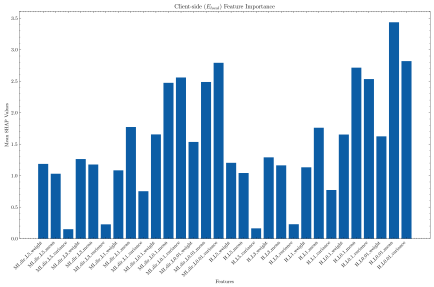
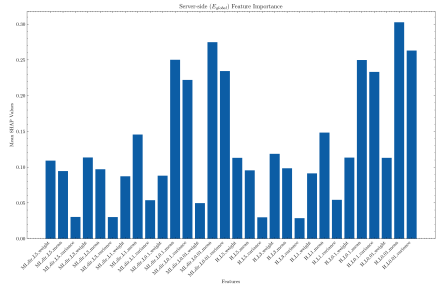


Figure 38: Heatmap for Difference Between aggregation of client's models Explanations and server model explanations



(a) SHAP values feature importance over Aggregation of Client's Side Explanation



(b) SHAP Value feature importance for side Server model

Figure 39: SHAP value feature importance from both Client's side models and Server model

The difference between E_{global} and E_{local} represents the divergence between server-based explanations (obtained by directly accessing the clients' data) and the average client's explanations (acquired by aggregating individual client explanations without accessing their data). This calculation of divergence helps assess the quality and reliability of client explanation aggregation (E_{local}) compared to the server-based explanations (E_{global}) which enables explainability in FL-settings while preserving data privacy.

SHAP values were computed using validation data. Each client's (IoT device) SHAP value explanations formed an $m \times d$ matrix, where m represents the number of data points from the test set and d stands for the number of features. To aggregate the client-based explanations, we calculated the element-wise average of the individual client SHAP value

explanations, resulting in E_{local}, which is also an $m \times d$ matrix. Similarly, when the server model was applied to the test data from clients, it produced server-based explanations denoted as E_{global}, which again forms an $m \times d$ matrix. The difference $d_{(g-l)}$ between E_{local} and E_{global} was calculated by taking the element-wise difference between the two matrices, resulting in another matrix of size $m \times d$.

SHAP values were calculated for over 2000 test data points from each client. By aggregating these values, we generated client-based explanations (E_{local}). A comparative analysis assessed the effectiveness of E_local in representing the server model's behavior. To create server-based explanations E_{global}, the server accessed each client's data, and SHAP values were recalculated and aggregated, matching the size of E_local. Finally, the difference (d_{g-l}) between E_{global} and E_{local} was computed to highlight the divergence between the two sides of explanations.

Figure 38 presents a heatmap that visualizes the differences in magnitude between server-based explanations and client-based explanations for each sample and feature. The heatmap highlights the discrepancies for 20 features, selected from a total of 115 for the sake of readability. From the heatmap, it can be observed that the overall divergence between the server-based and client-based explanations is relatively small. However, there were a few notable instances where the divergence was more significant. For example, some samples display darker shades for certain features, indicating a larger discrepancy between server-based and client-based explanations for those specific instances.

To conduct a more thorough analysis of the features, both client-based and server-based explanations were evaluated, emphasizing their differences. The average SHAP values were computed for the client-based and the aggregated server-based explanations (refer to Figure 39). It can be observed that the SHAP values of most features are comparable in both Figure 39a and Figure 39b, implying that the aggregation of client-based explanations closely approximates server-based explanations. Thus, Client-based SHAP explanations for global interpretations in terms of feature importance closely approximated server-based explanations, achieving similar levels of explainability for the server model (global model) without compromising data privacy.

9.2 Balancing Privacy and Explainability in Federated Learning for Botnet Detection in IoT Networks

In Chapter 2.1, Publication VIII proposed a framework for detecting botnets in IoT networks utilizing a Deep Neural Network (DNN) model. To explain the global model on the server side, the SHAP (Shapley Additive Explanations) explainer was used. It was demonstrated that aggregated SHAP value explanations from clients were sufficient to achieve explanations of the global server model. These client-aggregated SHAP value explanations closely approximated the explanations of SHAP values when the server accessed client data. The dynamic nature of IoT networks integrated with the evolution of more advanced botnet attacks requires more robust and explainable detection solutions.

In Publication IX, a FEDXAI framework was proposed to enhance privacy guarantees in FL settings by ensuring the privacy of post hoc local explanations generated for the server model. The Long Short-Term Memory (LSTM) model, which was trained in this publication, operates in a horizontal federated learning (HFL) setting using the FedAvg Aggregation algorithm for both binary and multiclass classification tasks. Additionally, this publication presents to explain the predictions made by the global LSTM (the server model based on FedAvg aggregation) by aggregating explanations from participating clients while maintaining the privacy of their data. This is achieved through secure aggregation of SHAP (SHapley Additive ExPlanations) value explanations from individual client models based

on IoT devices, allowing for an approximation of the server model's explanations without direct access to client data.

Publication IX presents three studies conducted in federated learning settings using the N-BaIoT dataset: 1) Binary Classification, 2) Botnet-type Classification, and 3) Attack-type Classification.

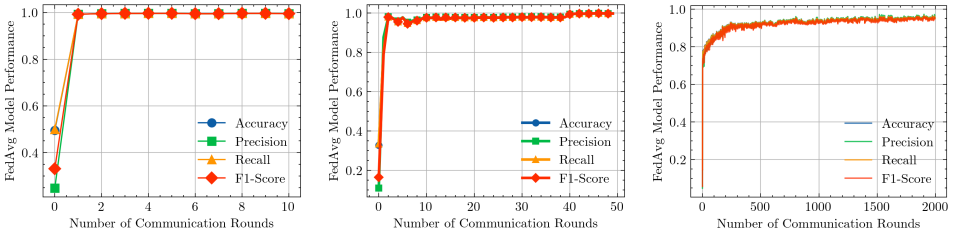
- In the binary classification study, the objective was to distinguish between malicious and benign traffic.
- In the Botnet-type detection study, the aim was to classify different types of malware. The classes used for botnet-type detections included Mirai, Gafgyt, and Benign.
- In the Attack-type classification study, data points were classified into various attack types, including ACK, benign, junk, scan, SYN, combo, TCP, UDP, and plain UDP.

In the experimental settings for FL, an LSTM model was trained using data from multiple IoT devices, ensuring that a fixed number of samples were obtained from each device. This approach minimized the impact of varying training instances and maintained a consistent dataset size across all clients while keeping the same class proportions. The number of communication rounds required for training the model was determined through iterative experimentation. This process began with a random selection of initial rounds, which were then adjusted based on the observed performance of the model. For instance, the binary classification task converged within 10 rounds, while 50 rounds were sufficient for detecting botnet types. In contrast, 2000 rounds were tested for classifying different types of attacks. In the evaluation of LSTM learning models in federated learning, a validation dataset was utilized exclusively on IoT devices. Metrics such as Accuracy, Precision, Recall, and F1-score were employed to evaluate both the client and server models in federated learning across all classification settings.

The classification performance metrics of client models were obtained from the evaluation results using their specific parameters on each client's validation data. To assess the FedAvg algorithm on the server side, test data was used exclusively from each IoT device-based client for evaluating its own model. In every round of training and evaluation during the federated training process, at least 50% of the clients were randomly selected.

All 9 IoT device-based clients participated in each evaluation round to provide a reliable estimate of the model's performance for binary classification. However, in the multiclass classification setting, only seven clients were involved. The 'Doorbell2' and 'Webcam' devices were not infected by the Mirai botnet, so these two devices were excluded from the experiments for Botnet-type and Attack-type detections. This exclusion was required to maintain consistency in both the input feature space and label space during federated training using the FedAvg algorithm, considering the horizontal nature of the FL approach.

Figure 40 illustrates the performance metrics of the LSTM model on the server side across multiple communication rounds in HFL settings, using the FedAvg algorithm for three classification types: binary classification, botnet-type detection, and attack-type detection. The performance metrics of the server-side LSTM model were evaluated starting from the initial (0th) communication round, which serves as a baseline for the model's performance prior to any client communication or parameter updates. This baseline performance was included to observe how the global model improves as it updates parameters from client models in the subsequent communication rounds.



(a) Malware Detection (Binary) *(b) Botnet-type Detection* *(c) Attack-type Detection*

Figure 40: Performance evaluation of FedAvg LSTM model on the server side across different detection tasks: (a) Malware Detection (Binary), (b) Botnet-type Detection, and (c) Attack-type Detection.

In Figure 40a, it is evident that the LSTM model utilizing FedAvg achieved early convergence with fewer communication rounds for binary classification. All performance metrics—accuracy, precision, recall, and F1-score—quickly reached optimal levels with minimal communication overhead. The client-side LSTM models were evaluated using IoT device-specific data after each training communication round. For binary classification, as shown in Figure 41, all LSTM models based on device-specific data achieved nearly 99% performance, with metrics including accuracy (Figure 41a), precision (Figure 41b), recall (Figure 41c), and F-score (Figure 41d) reaching these levels in just a few rounds which demonstrates the models' quick convergence.

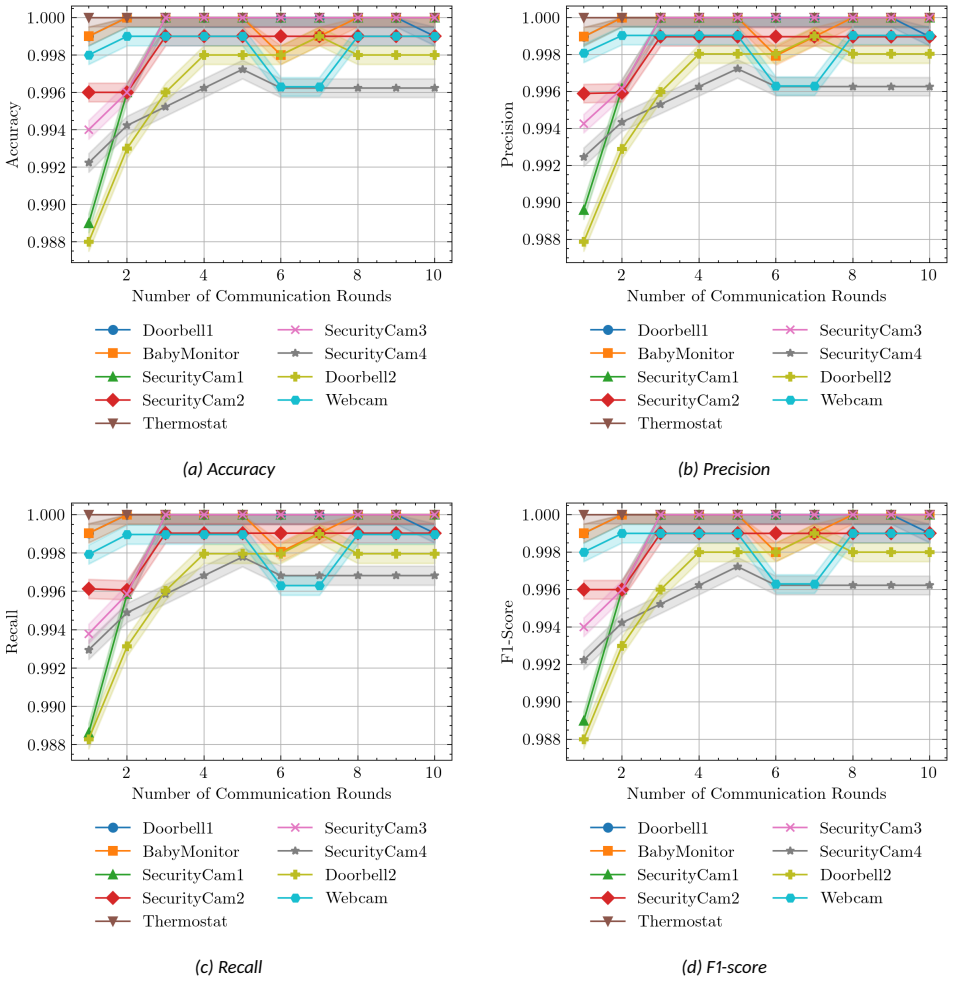
The shaded areas in Figure 41 represent the confidence intervals surrounding the classification metrics (accuracy, precision, recall, and F1-score). Fixed probabilistic error rates were utilized to estimate potential variability in the classification metrics. For binary classification, the error rate was set to 0.05%. These error rates were also used to calculate confidence intervals.

- Lower Bound = metric $-$ (metric $\times$ error percentage)
- Upper Bound = metric $+$ (metric $\times$ error percentage)

The convergence for botnet-type detection occurred after two communication rounds, similar to binary detection. Figure 40b illustrates the server-side evaluation of the LSTM model's performance in identifying botnet types in IoT devices. The figure shows a rapid improvement in all metrics, stabilizing above 0.95 after two communication rounds. By the end of 50 communication rounds, the LSTM model's performance in the server-side evaluation of the FedAvg algorithm for botnet-type detection closed to 1 in classification metrics.

Figure 42 presents the performance evaluation results for the client-side LSTM models, where the error rate was set at 0.01%. The evaluation metrics include accuracy, precision, recall, and F1-score for individual IoT devices participating in FL for botnet-type detection. The results demonstrated consistently high performance across the devices. The thermostat device-based client model showed more variability in the initial rounds but stabilized after 15 rounds of communication. In contrast, the BabyMonitor device model began with lower performance metrics but gradually improved to exceed 0.99 after 50 communication rounds.

In contrast to binary classification and botnet-type detection, a significantly larger number of communication rounds (2000) were necessary to achieve optimal performance in attack-type detection. The metric results for the attack type of the LSTM model are il-



(a) Accuracy *(b) Precision*

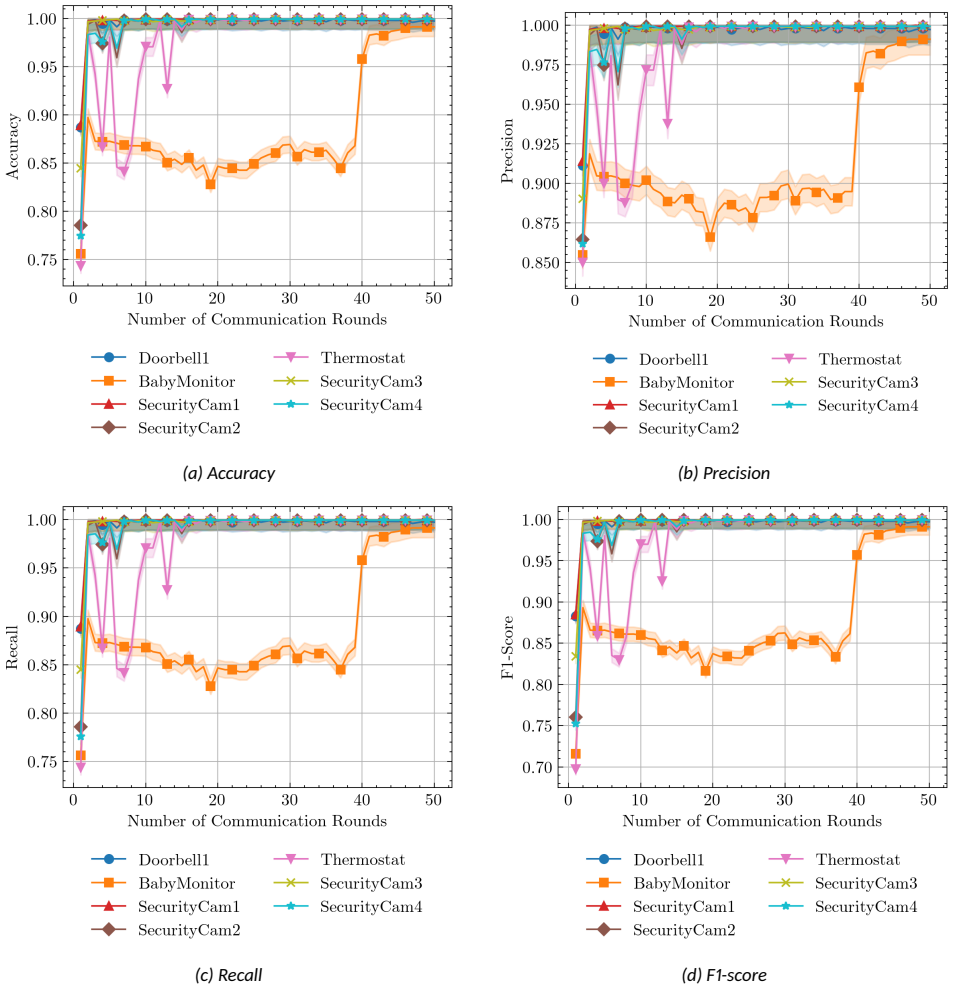
(c) Recall *(d) F1-score*

Figure 41: Device-specific client-side evaluation of LSTM model performance for Binary classification

lustrated in Figures 40c and 43, which present the server-side and client-side LSTM model evaluations, respectively.

Figure 40c shows the evaluation metrics for the LSTM model using FedAvg on the server side, including accuracy, precision, recall, and F1 score. The model improved quickly initially but stabilized after about 1,500 communication rounds, indicating that additional rounds were necessary for consistent performance.

Figure 43 displays the performance metrics for client-side LSTM models across seven IoT devices over 2000 communication rounds, with an error rate of 0.02%. Most models, including Doorbell1, BabyMonitor, SecurityCam1, SecurityCam2, and SecurityCam3, maintained performance above 0.95 after 500 rounds. In contrast, the Thermostat model showed fluctuations, ranging from 0.85 to 0.95. The SecurityCam4 model initially performed lower but improved and stabilized around the 1,000th round. These variations illustrate the complexities of detecting attack types and the need for more rounds for some devices to achieve convergence.



(a) Accuracy *(b) Precision*

(c) Recall *(d) F1-score*

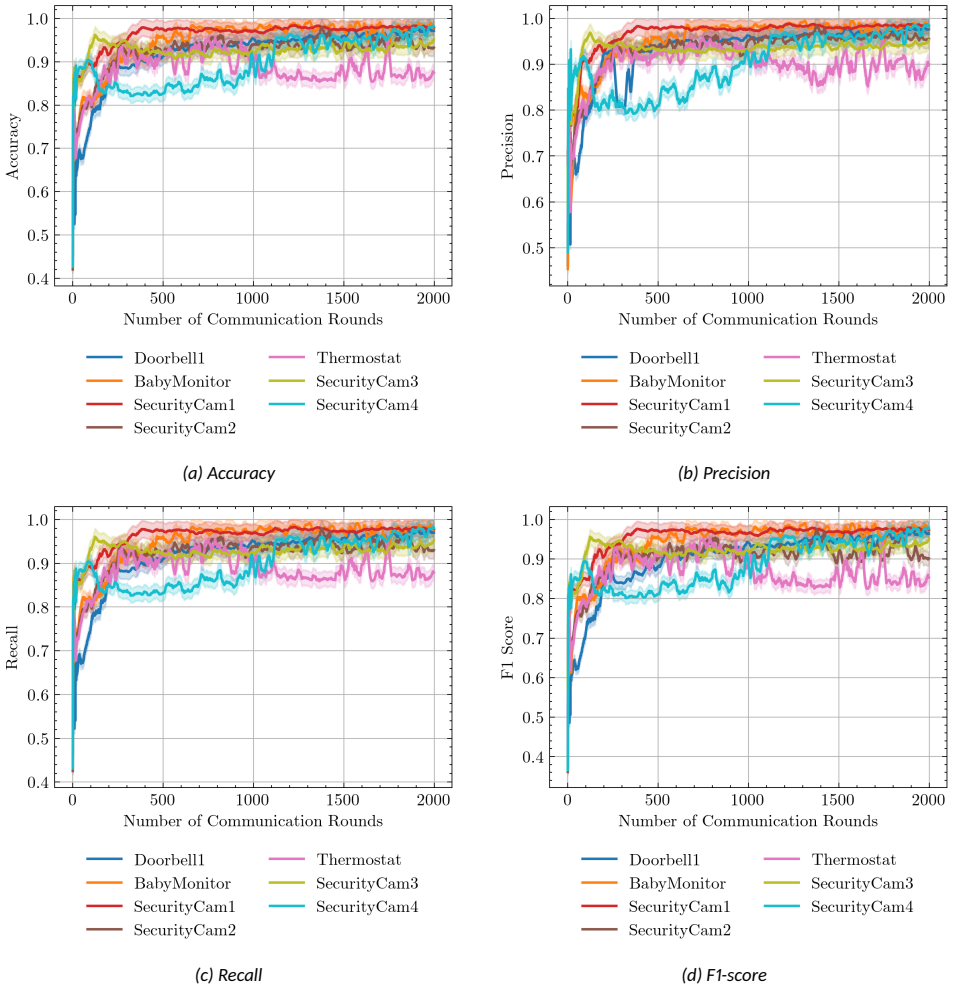
Figure 42: Device-specific client-side evaluation of LSTM model performance for Botnet-type Detection

9.2.1 Explaining Server-Side Black Box model in Federated Learning Settings

In FL, explaining the participated client black box model is straightforward since each client keeps its own data private. Each client uses its training data as a reference for post hoc explainable AI (XAI) methods, ensuring model explainability. Post-hoc explainers, such as the SHAP explainer, require access to both the training data and the trained model in order to compute SHAP values for the test data. On the other hand, Publication IX aimed to explore the significance of features in the server model without sharing any data with the server.

For Publication IX as well, SHAP values of client-based models were aggregated to explain the server model in FL. For a point of interest x, the explanation of the prediction made by the server model M_s is obtained using the following equation.

$$\varphi_s(x) = \frac{1}{|N|} \sum_{c_i \in N} \varphi_{c_i}(x_j) \tag{23}$$



(a) Accuracy *(b) Precision*

(c) Recall *(d) F1-score*

Figure 43: Device-specific client-side evaluation of LSTM model performance for Attack-type Detection

Sharing the SHAP values generated by clients with the server could compromise privacy, as the server might gain excessive knowledge about the parameters of client models which raises significant vulnerabilities in post hoc explanation techniques that can be exploited by adversaries to create classifiers with controllable post hoc explanations [131]. Specifically, an attacker can deceive target classifiers and the methods used for explanation while maintaining the classifier's output consistency [70]. To protect the privacy of SHAP explanations from client models, a secure multiparty computation (SMPC) protocol [22] based on Secret Sharing was implemented which enables the secure aggregation of SHAP values from client models without revealing individual client SHAP values to the server or other clients.

SMPC, or Secure Multi-Party Computation, is a cryptographic protocol that enables N parties (clients and a server) to compute an aggregate function $f(x_1, x_2, \ldots, x_N)$ using private inputs $x_1, x_2, \ldots, x_N$ without revealing them. This ensures client data privacy during secure aggregation. SMPC was applied to securely aggregate SHAP values from client-side

models. The protocol allows the server to compute the aggregated SHAP values $\sum_{c_i \in N} \varphi_{c_i}$ without knowing the individual SHAP values φ_{c_i}.

Algorithm 1 illustrates the steps for secure aggregation of SHAP values which enables the server to obtain explanations for its model M_s while preserving the privacy of individual clients' SHAP values.

To determine if the secure aggregation of client-side SHAP values, denoted as E_{local}, is adequate for the server model M_s, the server model was provided with data from all clients N to generate explanations. The server-based SHAP values were aggregated for all clients by using the combined training data with the SHAP explainer, resulting in the global explanation E_{global}. Finally, to verify the accuracy of the explanations, the difference d_{g-l} between E_{global} and E_{local} was calculated, as shown in Equation 22.

Algorithm 1: Secure Aggregation of Client-side SHAP Values

Input: SHAP values $\varphi_{c_i} \in \mathbb{R}^d$ for each client c_i,
Number of clients N,
Large prime number P

Result: Secure Aggregated SHAP values $\sum_{i=1}^{N} \varphi_{c_i}$

Step 1: Secret Sharing:

for *each client* c_i **do**

Split φ_{c_i} into N shares using a (t, N) threshold secret sharing scheme;

Distribute shares to all other clients and the server;

Step 2: Masking:

for *each client pair* (i, j) *where* $i \neq j$ **do**

Generate random masking vector $S_{i,j}$;

for *each client* c_i **do**

Compute masked SHAP value:;

$$z_i = \varphi_{c_i} + \sum_{j>i} S_{i,j} - \sum_{j<i} S_{j,i} \mod P$$

Send z_i to the server;

Step 3: Aggregation at Server:

Compute aggregated value:;

$$\sum_{i=1}^{N} z_i = \sum_{i=1}^{N} \left(\varphi_{c_i} + \sum_{j>i} S_{i,j} - \sum_{j<i} S_{j,i} \right) \mod P$$

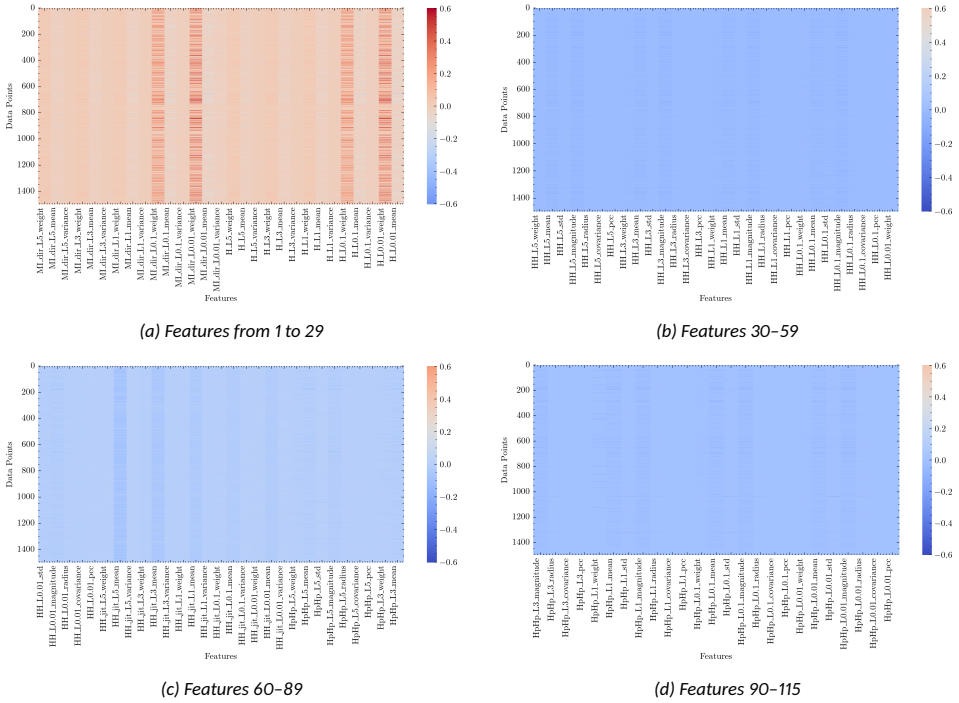
Simplify to:;

$$\sum_{i=1}^{N} z_i = \sum_{i=1}^{N} \varphi_{c_i} \mod P$$

Output: Secure Aggregated SHAP values $\sum_{i=1}^{N} \varphi_{c_i}$

SHAP values were calculated for each validation dataset of the IoT device-based client model to estimate the contribution of each feature to the model's output. This output specifically refers to the probability distribution over class labels generated using the Softmax activation function in the LSTM model. After completing federated training, all participating client models and the server model from the last communication round were used

to generate SHAP explanations. For binary classifications, models from the 10th round were used, while for botnet-type classifications, models from the 50th round were utilized, and for attack-type classifications, models from the 2000th round were selected to generate the SHAP explanations.



(a) Features from 1 to 29

(b) Features 30–59

(c) Features 60–89

(d) Features 90–115

Figure 44: Heatmap for the difference between the secure aggregation of client models' explanations and server model explanations for binary classification

Figures 44, 45, and 46 present heatmap graphs for three detection types: Binary, Botnet-type, and Attack-type. To ensure the clarity of this paper, a heatmap has been provided for all 115 features across four sub-figures.

In Figure 44, the differences between server-based explanations and client-based explanations were visualized for each sample and feature. Each cell in the heatmap represents the difference for a specific combination of sample and feature. The colour scale on the right side of the heatmap provides a reference for interpreting the magnitude of these differences. Data points close to zero indicate a slight difference between server-based and client-based explanations, indicating that the secure aggregation of client-based SHAP values closely approximates the server model explanations when the server has access to the data. Conversely, instances further from zero reveal a more significant discrepancy between client-based and server-based explanations.

Figure 44 shows a heatmap for binary detection. The differences between server-based and client-based explanations were minor, as most data points were close to zero. In Figure 44a (which covers features 1-29), there are several notable data points where the differences are more apparent, indicated by values deviating from zero. Specifically, for features such as 'MI_dir_L0.1_weight', 'MI_dir_L0.01_weight', 'H_L0.1_weight', and 'H_L0.01_weight', these variations suggest slight discrepancies in the secure aggregation of Shapley values compared to the server-side explanation.

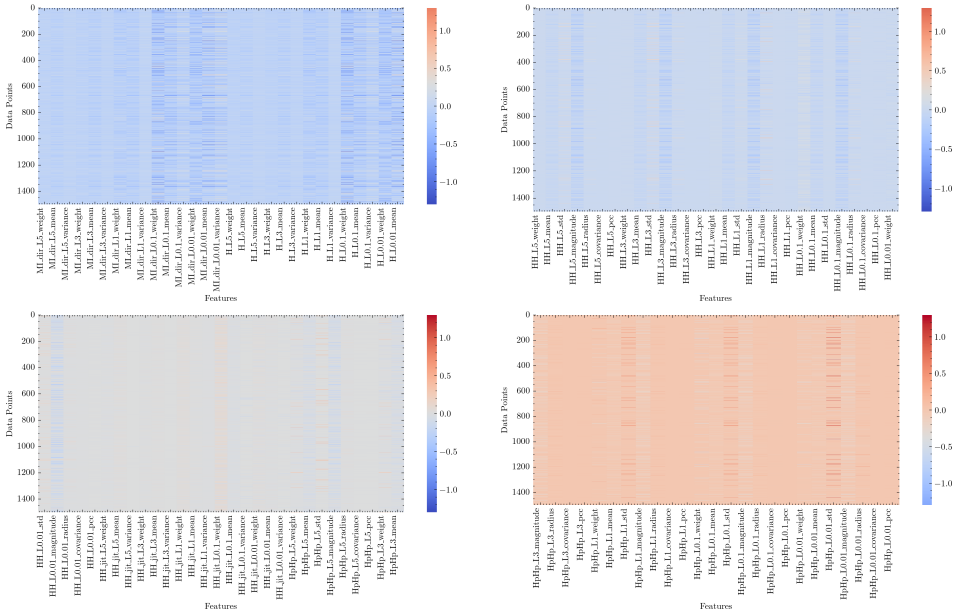


Figure 45: Heatmap for Difference Between Secure aggregation of client's models Explanations and server model explanations for Botnet-type detection

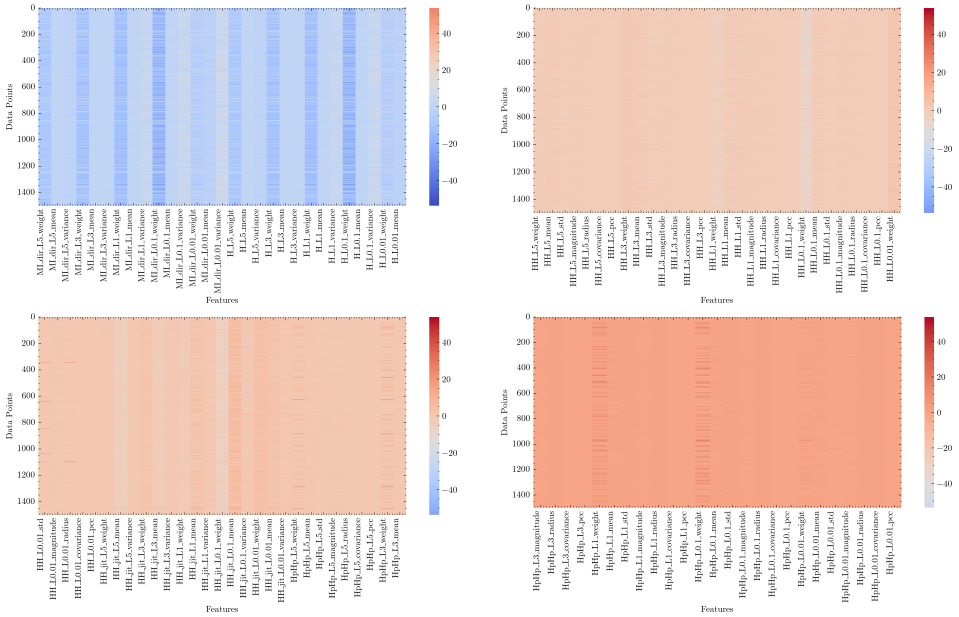
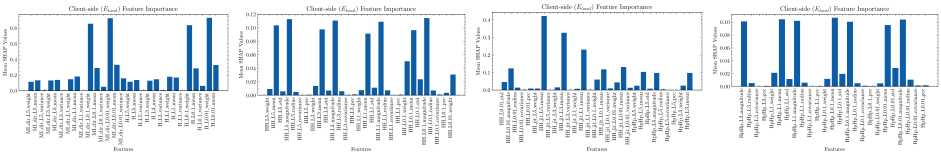


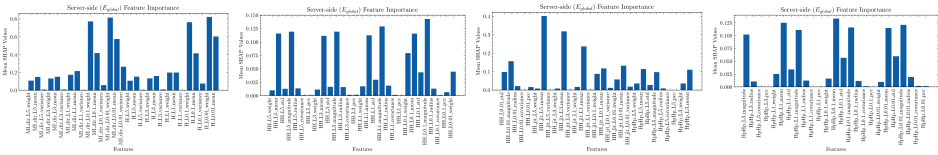
Figure 46: Heatmap for Difference Between Secure Aggregation of client-side models Explanations and server-side explanations for Attack-type Detection

Similarly, Heatmap graphs for detecting Botnet types, as shown in Figure 45, demonstrate a high similarity between the explanations generated by server-based and client-based models, with most instances being close to zero. These instances indicate specific

features and cases where the client-based explanations diverge from those of the server-based models. Similarly, the heatmap for attack-type detection in Figure 46 shows almost the same level of similarity, with most data points near zero which indicates a strong alignment between the server-based SHAP value explanations and the client-based secure aggregation of SHAP explanations. However, there are some instances with more significant discrepancies that are further away from zero. Across all three detection types, the secure aggregation of client-based SHAP value explanations closely approximates the server-side model explanations, even without granting the server direct access to the training and testing data from IoT device-based clients.

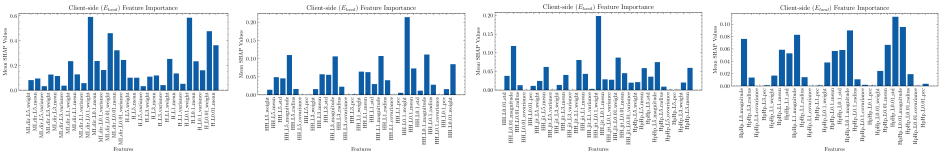


(a) Client-side absolute mean feature importance (Features 1-115)

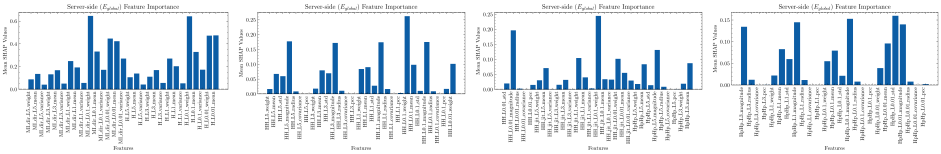


(b) Server-side absolute mean feature importance (Features 1-115)

Figure 47: Mean SHAP values of Secure Aggregation Matrix of Client-Side and Server-Side Models for Binary-type classification



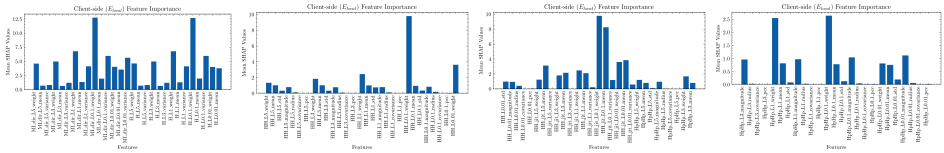
(a) Client-side absolute mean feature importance (Features 1-115)



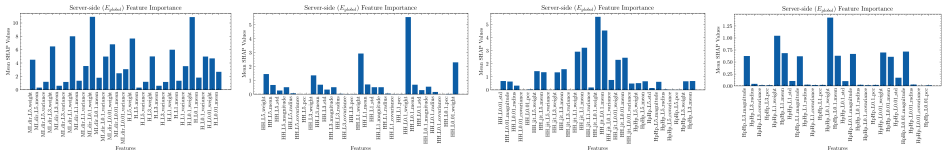
(b) Server-side absolute mean feature importance (Features 1-115)

Figure 48: Mean SHAP values of Secure Aggregation Matrix of Client-Side and Server-Side Models for Botnet-type detection

A detailed analysis was conducted across 115 features by examining both client-side (E_{local}) and server-side (E_{global}) explanations and comparing their differences. The average SHAP values were computed for both types of explanations. Figure 47 illustrates that the feature importance, as indicated by the mean SHAP values from the client-side (Figure 47a) and server-side (Figure 47b), is quite similar. Figure 47a presents the mean SHAP values derived from the secure aggregation of client-side models, while Figure 47b



(a) Client-side absolute mean feature importance (Features 1-115)



(b) Server-side absolute mean feature importance (Features 1-115)

Figure 49: Mean SHAP values of Secure Aggregation Matrix of Client-Side and Server-Side Models for Attack-type detection

displays the mean SHAP values from the server model, which has access to training data from IoT device-based clients.

By comparing these two sides of mean SHAP values, it is evident that the top features influencing the model's predictions remain consistent between the client-side and server-side analyses.

Figure 48 showcases the average SHAP values across 115 features for both client-side (Figure 48a) and server-side (Figure 48b) explanations related to botnet detection. These figures demonstrate a strong consistency in feature importance between the two explanations. Similarly, Figure 49 compares the mean SHAP values for attack type detection, highlighting the client-side feature importance in Figure 49a and the server-side feature importance in Figure 49b. This comparison analysis demonstrated that the key features contributing to model predictions closely align between the client side and the server side. Therefore, securely aggregating SHAP values in client-based explanations is sufficient, eliminating the need to transmit data from to the server for explanations of the server model.

9.2.2 Explaining Client-Side Black Box models in Federated Learning Settings

To explain the outcomes of the LSTM model predictions for client-side models, three post-hoc feature importance techniques of XAI were used. A comprehensive quantitative evaluation was conducted to assess the effectiveness of three prominent XAI methods: LIME, Integrated Gradients (IG), and SHAP, across all participating clients in FL settings. This evaluation focused on four key metrics: High Faithfulness, Monotonicity, Low Complexity, and Maximum Sensitivity. Local explanations generated by LIME, IG, and SHAP were evaluated across 2000 test points. Table 18 displays the results of the XAI metrics, including the mean and standard deviations for both LIME and SHAP explanations. These evaluations were conducted on 2000 data points from all participating clients in FL across three classification categories: Binary, Botnet type, and Attack type.

In binary classification, the SHAP explainer demonstrates higher mean values for faithful explanations compared to LIME and IG explainers across all test instances from participating clients. For instance, SHAP achieved a faithfulness score of 0.34 $\pm$0.05 for the BabyMonitor model, whereas IG scored 0.31 $\pm$0.04 and LIME scored 0.24 $\pm$0.30 which indicates that SHAP offers more reliable feature importance that is closely aligned with its

effect on model prediction probabilities.

Table 18: Results of evaluating Quality of LIME, IG & SHAP explanations using High Faithfulness (μ_f), Low Complexity (μ_c), Maximum Sensitivity (μ_s) & Monotonicity (μ_m) for participated client-side models in FL for three classification types.

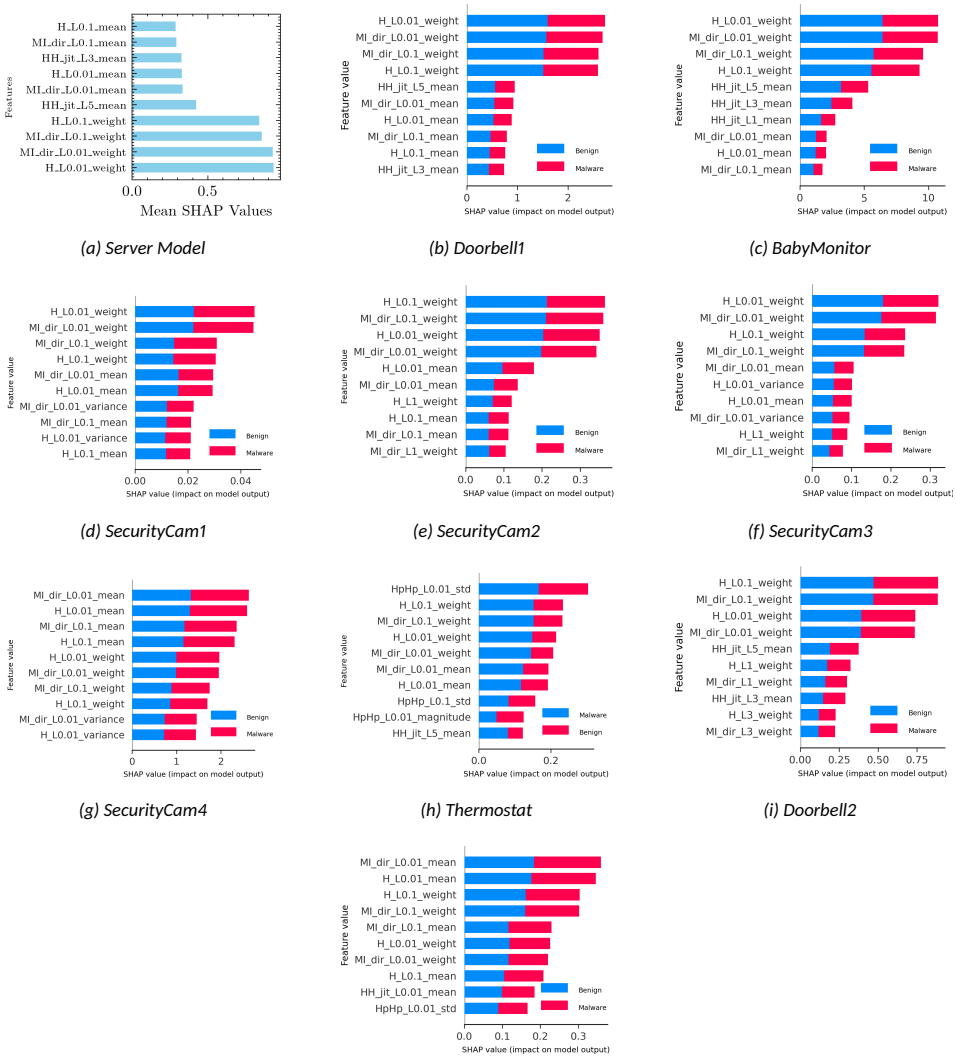
Client	Metric \explainer	Binary			Botnet Type			Attack-Type		
		Lime	IG	SHAP	Lime	IG	SHAP	LIME	IG	SHAP
BabyMonitor	μ_f	0.24 ±0.30	0.01±0.25	0.34 ±0.23	0.47 ±0.34	-0.03±0.24	0.79 ±0.15	0.43 ±0.28	0.00±0.22	0.62 ±0.32
	μ_c	3.95 ±0.07	3.29±0.34	3.55 ±0.08	4.06 ±0.11	3.25±0.39	1.26 ±0.31	4.01 ±0.31	3.39±0.39	1.39 ±0.75
	μ_f	0.14 ±0.10	0.26±0.11	0.04 ±0.05	0.41 ±0.18	0.08±0.23	0.09 ±0.23	0.26 ±0.11	0.11±0.70	0.09 ±0.15
	μ_m	31.07%	1.00%	46.00%	97.00%	0.03%	99.00%	99.00%	0.00%	96.00%
Doorbell1	μ_f	0.38 ±0.29	-0.01±0.23	0.55 ±0.21	0.14 ±0.09	0.03±0.28	0.25 ±0.31	0.32 ±0.33	0.00±0.23	0.56 ±0.32
	μ_c	4.03 ±0.05	3.74±0.36	3.55 ±0.10	3.99 ±0.11	2.99±0.49	1.42 ±0.35	3.89 ±0.26	3.23±0.48	1.59 ±0.58
	μ_f	0.20 ±0.10	0.08±0.00	0.02 ±0.02	0.65 ±0.20	0.01±0.00	0.00 ±0.00	0.36 ±0.13	0.01±0.00	0.06 ±0.10
	μ_m	41.40%	1%	66.54%	74%	1%	85%	91%	6%	92%
SecurityCam1	μ_f	0.31 ±0.34	-0.00±0.25	0.51 ±0.32	0.36 ±0.21	-0.01±0.29	0.59 ±0.22	0.39 ±0.39	0.00±0.23	0.70 ±0.25
	μ_c	3.89 ±0.07	3.88±0.29	3.35 ±0.23	4.13 ±0.17	3.16±0.49	1.31 ±0.32	3.94 ±0.43	3.23±0.48	1.42 ±0.44
	μ_f	0.10 ±0.05	0.35±0.12	0.02 ±0.04	0.39 ±0.17	0.10±0.08	0.09 ±0.21	0.23 ±0.09	0.30±0.14	0.05 ±0.09
	μ_m	40.41%	5%	39.67%	96%	1%	100%	97%	20%	96%
SecurityCam2	μ_f	0.34 ±0.37	0.01±0.29	0.45 ±0.37	0.21 ±0.17	0.01±0.19	0.36 ±0.26	0.34 ±0.29	-0.00±0.17	0.66 ±0.31
	μ_c	3.95 ±0.05	3.86±0.52	3.35 ±0.07	4.13 ±0.09	3.45±0.40	1.76 ±0.22	3.95 ±0.20	3.50±0.40	1.39 ±0.73
	μ_f	0.18 ±0.10	0.06±0.10	0.02 ±0.02	0.53 ±0.21	0.36±0.22	0.07 ±0.14	0.47 ±0.15	0.01±0.00	0.06 ±0.12
	μ_m	36.55%	4%	42.45%	70%	1%	95%	98%	2%	98%
SecurityCam3	μ_f	0.38 ±0.34	-0.01±0.19	0.59 ±0.26	0.18 ±0.33	-0.04±0.26	0.51 ±0.24	0.36 ±0.31	-0.00±0.22	0.62 ±0.30
	μ_c	3.89 ±0.08	3.50±0.38	3.41 ±0.24	4.02 ±0.23	3.20±0.48	1.40 ±0.20	4.01 ±0.26	3.22±0.47	1.32 ±0.72
	μ_f	0.08 ±0.03	0.42±0.15	0.01 ±0.02	0.35 ±0.22	0.42±0.28	0.04 ±0.14	0.23 ±0.05	0.26±0.21	0.07 ±0.13
	μ_m	42.85%	1%	40.77%	100%	1%	100%	94%	0%	90%
SecurityCam4	μ_f	0.33 ±0.28	0.03±0.27	0.47 ±0.27	0.46 ±0.14	0.02±0.19	0.63 ±0.14	0.48 ±0.36	0.01±0.24	0.71 ±0.37
	μ_c	4.00 ±0.07	3.65±0.44	3.49 ±0.17	4.07 ±0.19	3.39±0.42	1.43 ±0.23	4.04 ±0.25	3.41±0.43	1.36 ±0.71
	μ_f	0.11 ±0.07	0.01±0.22	0.01 ±0.02	0.37 ±0.10	0.12±0.05	0.02 ±0.08	0.26 ±0.10	0.40±0.00	0.05 ±0.09
	μ_m	44.20%	9%	37.65%	100%	8%	100%	92%	1%	99%
Thermostat	μ_f	0.37 ±0.38	0.03±0.26	0.56 ±0.33	0.39 ±0.13	-0.01±0.19	0.61 ±0.09	0.34 ±0.26	-0.00±0.19	0.66 ±0.26
	μ_c	3.87 ±0.05	3.45±0.36	3.37 ±0.11	4.10 ±0.20	3.48±0.40	1.52 ±0.27	4.05 ±0.14	3.48±0.41	1.29 ±0.65
	μ_f	0.10 ±0.06	0.06±0.12	0.01 ±0.02	0.38 ±0.19	0.01±0.00	0.06 ±0.16	0.31 ±0.11	0.01±0.00	0.05 ±0.09
	μ_m	34.49%	3%	37.46%	90%	1%	100%	98%	3%	100%
Doorbell2	μ_f	0.24 ±0.17	-0.01±0.20	0.43 ±0.15						
	μ_c	4.19 ±0.16	3.57±0.42	3.64 ±0.07						
	μ_f	0.09 ±0.03	0.04±0.08	0.09 ±0.09						
	μ_m	58.41%	8%	53.98%						
Webcam	μ_f	0.46 ±0.42	0.01±0.25	0.51 ±0.39						
	μ_c	3.88 ±0.10	3.35±0.40	3.14 ±0.10						
	μ_f	0.12 ±0.02	0.04±0.09	0.01 ±0.02						
	μ_m	57.21%	9%	69.23%						

When evaluating local explanations using low complexity metrics, SHAP nearly achieved lower mean values than LIME and IG across all clients. However, for the Client models BabyMonitor and Doorbell2, the IG explainer outperformed the other explainers using these metrics.

Sensitivity refers to the degree to which explanations remain stable for nearby data points. Lower sensitivity values indicate more stable explanations. On average, SHAP models, when explained using SHAP, demonstrate lower sensitivity values compared to LIME and IG explanations across all clients. However, for the Doorbell2 model, IG provides more robust explanations than both LIME and SHAP. This suggests that SHAP offers more stable and reliable explanations for nearby data points within the same feature space used by its client model.

Overall, SHAP explanations were more consistent in terms of monotonicity compared to LIME and IG when evaluated using a monotonicity metric across all client models. However, for the SecurityCam1, SecurityCam3, SecurityCam4, and Doorbell2 models, the LIME explanations displayed greater monotonicity, with percentages of 40.41%, 42.85%, 44.20%, and 58.41%, respectively, compared to SHAP explanations.

Similar to binary classification, SHAP is more reliable than LIME and IG (Integrated Gradients) for both botnet types and attack types, as shown in Table 18. SHAP demonstrates a strong correlation between the importance of features calculated by its explainer and the prediction probabilities of the client's model. Additionally, SHAP demonstrates lower complexity and sensitivity compared to LIME and IG. However, for the Doorbell1 and Se-



(a) Server Model *(b) Doorbell1* *(c) BabyMonitor*

(d) SecurityCam1 *(e) SecurityCam2* *(f) SecurityCam3*

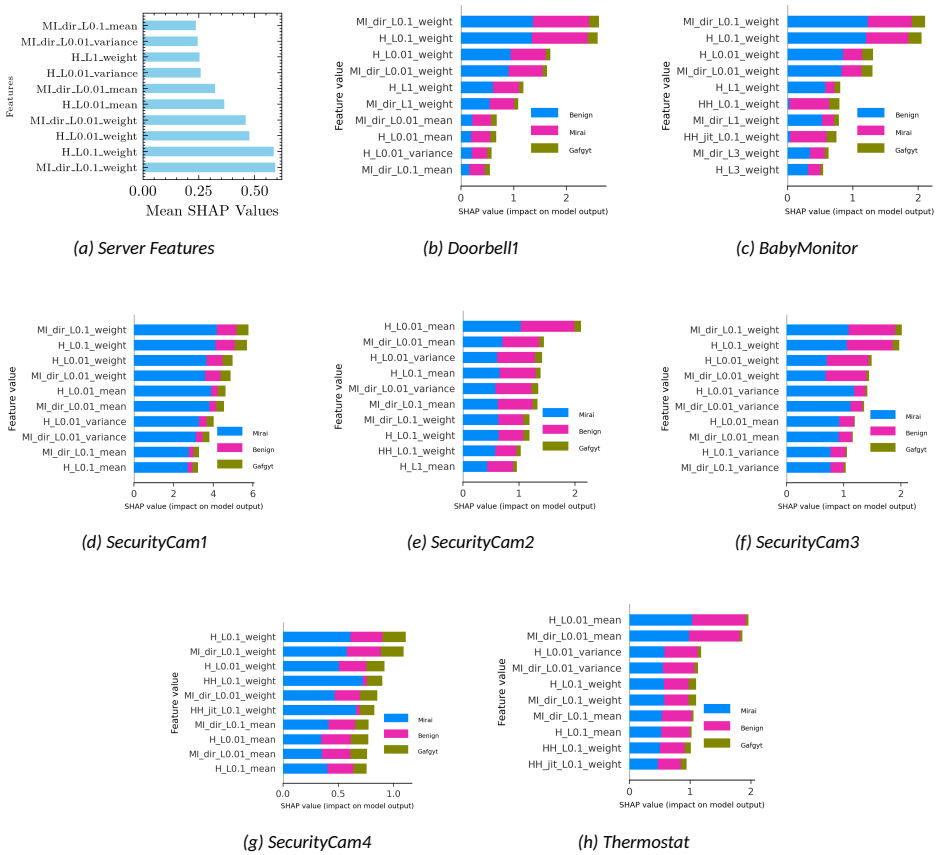
(g) SecurityCam4 *(h) Thermostat* *(i) Doorbell2*

(j) Webcam

Figure 50: Global explanations for server model feature importance and participated clients' IoT device models feature importance in Binary classification type

curityCam2 models, IG demonstrates to be more robust than both LIME and SHAP when using the maximum sensitivity metric. When considering the monotonicity metric, SHAP shows greater monotonicity compared to LIME and IG for both botnet-type and attack-type detection across all clients. However, In the attack-type detection for the BabyMonitor, the SecurityCam1 and SecurityCam3 models, the LIME explainer performs slightly better than SHAP concerning monotonicity metrics values (see Table 18).

Global explanations for server and client models were provided for those participating in FL, using the SHAP explainer. In these global explanations, clarify the model's predictions by detailing the contribution of each feature. Figure 50 illustrates the global feature importance explanations for both the server model and the IoT device models of par-



(a) Server Features *(b) Doorbell1* *(c) BabyMonitor*

(d) SecurityCam1 *(e) SecurityCam2* *(f) SecurityCam3*

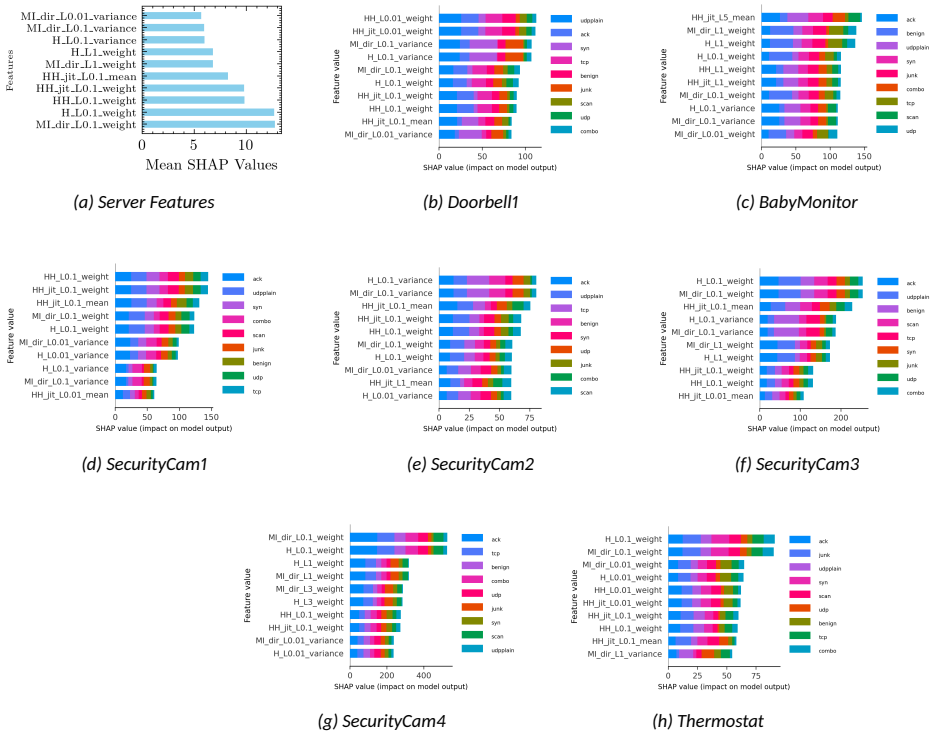
(g) SecurityCam4 *(h) Thermostat*

Figure 51: Global Explanations for Server Model Feature Importance and Participated Clients' IoT Devices Feature Importance for Botnet-type Detection

ticipating clients in a binary classification context. It highlights the top ten features that significantly influence the model's predictions. In the client-based model of global explanation in FL, each participating client used its own training data and trained model to calculate the Shapley values based on the test data, which shows the overall impact of each feature's contribution to the model's predictions.

Figs. 50b–50jprovide global explanations from the SHAP explainer regarding the top ten features for client-based models participating in FL for binary classification. Figure 50a displays the top 10 features of the server model for binary classification, obtained through the secure aggregation of SHAP value explanations from clients. It is evident that features related to host IP (H), network jitter (HH_Jit), and host MAC & IP (MI) are the most impactful categories for predictions in binary classification. Notably, the features H_L0.1_weight and MI_dir_L0.01_weight are the most significant across clients. Additionally, the count of network traffic packets (weight) plays a crucial role in distinguishing between benign and malicious network traffic. Moreover, shorter time windows, specifically L0.1 (500 μs) and L1 (1.5 seconds), are particularly influential in analyzing network time windows.

Figure 51 presents global explanations for botnet type detection using the SHAP explainer , highlights the ten most important features influencing the detection of various botnet types across multiple LSTM client-side IoT device models and the server model.



(a) Server Features *(b) Doorbell1* *(c) BabyMonitor*

(d) SecurityCam1 *(e) SecurityCam2* *(f) SecurityCam3*

(g) SecurityCam4 *(h) Thermostat*

Figure 52: Global explanations for Server Model Feature importance and participated clients IoT devices feature importance for Attack-type Detection

Figs. 51b–51h provide SHAP's global explanations for participating client IoT device models in FL. Figure 51a displays the top ten most important features for the server model, based on secure aggregated SHAP values derived from the explanations of the participating client models. In the context of detecting different types of botnets, the most effective features included Host IP (H), Host MAC & IP (MI), and Network Jitter (HH_jitt), particularly when analyzed within short to moderate time window frames. Additionally, network statistical values, especially the packet count (weight) along with their mean and variance, are crucial for analyzing network traffic features in botnet-type detection within federated IoT networks.

Similarly, Figure 52 illustrates the global explanations for both the server model. (Fig. 52a) and participated client-side models (Figs. 52b–52h) used in attack type detection. For the attack-type detection, the Host-MAC and IP (MI) features, as well as the Host-IP (H) features, were found to be more influential than other network categories among the top 10 selected features. In contrast, for botnet type detection, the HH_jit features were considered less impactful. Additionally, channel and socket-based features showed no significant influence. Regarding the time window, shorter durations, such as (100 μs) and L0.01 (500 μs), were more influential according to the SHAP explainer in the global assessments. Furthermore, packet count features had a greater impact in terms of network traffic statistics.

9.3 Chapter Discussions

This chapter addresses **RQ4**, which aims to explore how to integrate explainability into FL settings without compromising the privacy of client data while explaining the client and server models. Post-hoc XAI methods, such as feature importance explainers like LIME or SHAP, generally need access to the entire training dataset (input reference data) and the parameters of the trained model. This requirement can create privacy risks when using these methods to interpret a model hosted on a server in FL settings. To address this, Publications VIII and IX proposed the aggregation of client-based SHAP model explanations to explain the server model. This approach aims to enhance the transparency of the server-side model using XAI methods without relying on client data.

In Publication VIII, a Deep Neural Network (DNN) model was trained in a horizontal federated learning (HFL) setting using FedAvg algorithm. The DNN model achieved high detection rates, demonstrating accuracy, precision, recall, and F1-score in the botnet detection task, focusing on multiclass classification for both IoT device-based client-side models and the server-side model. Additionally, we analyzed the importance of features contributing to IoT botnet detection by utilizing the generated SHAP explanations. The results indicate that the aggregation of client-based SHAP explanations closely approximates the server-based explanations, achieving similar levels of explainability without compromising data privacy.

In Publication IX, we developed a high-performance LSTM-based model in an FL setting for binary classification, botnet-type detection, and attack-type detection. Our model achieved 99.90% accuracy in binary classification, 99.28% in botnet-type detection, and 94.89% in attack-type detection, outperforming existing FL-based intrusion detection systems. It also demonstrated high precision, recall, and F1 scores, effectively distinguishing between benign and malicious traffic, various botnet types, and multiple attack types. In Publication IX as well, proposed a framework for aggregating SHAP values from client-side models to explain the server model. To secure the aggregation of SHAP values and protect against adversarial attacks, secure multiparty computation was used to ensure the security of the explanations derived from these SHAP values. Our framework securely aggregates SHAP values on the client side, demonstrating that these securely aggregated explanations effectively approximate the feature attributions of the server model when the server accesses data from the client. We evaluated LIME, SHAP, and Integrated Gradients (IG) as post-hoc explanation methods for explaining client models that participated in FL. Our findings demonstrated that SHAP provided high faithfulness, lower complexity and was more robust, showing lower maximum sensitivity for stable attributions across similar inputs, as well as higher monotonicity.

10 Synthetic Data-Driven Explainability for Federated Learning Based IDS

This chapter presents the contributions of Publication X, which extends the explainability of FL-based IDS by proposing a synthetic data-driven explainability framework. While Chapter 9 (Publication VIII) demonstrated that aggregated client SHAP explanations can make the server model transparent, it required direct computation on client data, thereby raising concerns about scalability. Publication X addresses this limitation by introducing synthetic sample generation using federated generative models to approximate client data distributions. These synthetic samples are used as input references for post-hoc explainers like SHAP at the server, allowing privacy-preserving yet faithful explanations of the global black-box model. Thus, this chapter builds on prior work by eliminating dependency on real data for explanation, promoting both transparency and privacy in FL-based IDS.

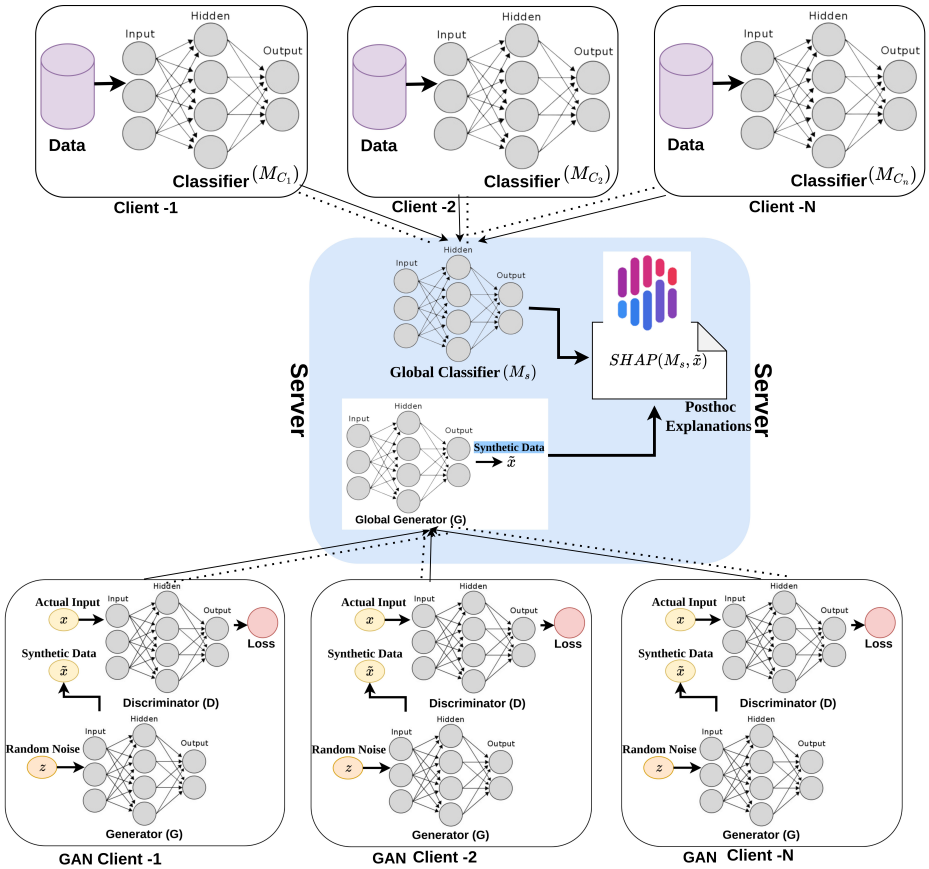


Figure 53: Synthetic Data-Driven Explainability Framework for Explaining Blackbox Classifier in FL settings

In Chapter 9, Publication VIII presented a framework for detecting botnets in IoT networks using a Deep Neural Network (DNN) model. The SHAP (Shapley Additive Explanations) explainer was used to provide insights into the global server model by aggregating SHAP value explanations from clients. These aggregated values closely matched the

SHAP values obtained from direct access to client data. However, this method is computationally expensive, as it requires calculating SHAP values for each client, which can be resource-intensive, especially with a growing number of IoT clients. The issue is more noticeable when using SHAP explainer types, such as DeepExplainer or KernelExplainer, for deep learning models.

To address the above problem, Publication X proposed synthetic data-driven explanations for the global model on the server side, specifically for FL-based IDS tasks, which aimed to explain the black box models using post-hoc explainability methods. Particularly, post-hoc explainability methods require training data as input reference samples to provide explanations for black-box models. The framework proposed in Publication X incorporates several Generative Adversarial Network (GAN) variants within FL to generate synthetic data. This synthetic data is then utilized as input reference baseline data for a post-hoc explainability method, which explains the decisions made by the FL-DNN classifier, ultimately enhancing the transparency of the black box model in FL settings. These synthetic data-driven explanations ensure that the FL-DNN model for explanations do not rely on real client data, thereby reducing potential privacy risks while still providing effective explanations in FL contexts.

In Publication X, a dual FL framework was proposed, as illustrated in Figure 53. This framework integrates two key components: 1) an FL-based Intrusion Detection System (FL-IDS) using Deep Neural Networks (FL-DNN) and 2) FL-based synthetic data generation. To evaluate this framework, 4 datasets were used: 1) NSL-KDD, and 2) UNSW-NB15. 3). CIC-IoT2023 4). CIC-IoMT2024.

FL-IDS (FL-DNN) is designed to classify network traffic into attack and benign classes while preserving privacy. Each client trains a local DNN model using its own network traffic data, and the FedAvg algorithm is employed to aggregate these local models into a global FL-DNN model at the server.

FL-DNN is designed to classify network traffic into benign or attack categories while preserving client data privacy. Each client trains a local DNN on its traffic data, and model updates are aggregated at the server using the FedAvg algorithm to construct a global FL-DNN.

For explainability, federated generative models are employed to generate synthetic data at the server. Each client participates in training a GAN-based model, and their updates are used to build a global generator. The synthetic data generated by this global generator serves as reference input for post-hoc explainability methods such as SHAP, which are applied at the server to interpret the decisions made by the global DNN model without accessing real client data.

The description of the CIC-IoMT2024 dataset is provided in Chapter 4. Below, the descriptions of the remaining three datasets are presented.

- NSL-KDD [140] dataset is an enhanced version of the KDD Cup 1999 dataset, commonly used for IDS task. This dataset categorizes network traffic into five classes: Denial of Service (DoS), Probe, Remote to Local (R2L), User to Root (U2R), and Normal. Network traffic is collected at fixed intervals as time series data, with each unit containing multiple packets and a total of 41 features.

- UNSW-NB15 dataset [99], produced at the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) using the IXIA PerfectStorm tool, contains raw network packets that mix real-world activity with synthetic attacks. It features nine attack types, including Denial of Service (DoS) and reconnaissance, and includes 46 attributes extracted using Argus and Bro-IDS tools.

- CIC-IoT2023 dataset [102], introduced in 2023, is the largest publicly available dataset for IoT ID. Collected in a controlled smart environment, it includes network traffic from 105 real-world IoT devices, such as smart TVs, sensors, cameras, and home automation appliances. Featuring 33 distinct attack types, this dataset serves as a comprehensive resource for evaluating IDS models in IoT contexts.

10.1 Federated Deep Neural Network for Intrusion Detection Systems

FL-IDS (FL-DNN) was designed to classify network traffic into attack and benign categories while ensuring data privacy, as outlined in Algorithm 2. Each client trains a local DNN model using its own network traffic data, and the Federated Averaging (FedAvg) algorithm was employed to aggregate these local models into a global FL-DNN model at the server. FL-IDS is composed of a central global model on a server and multiple distributed IDS clients, which can include IoT edge devices, network nodes, or sensors. Unlike traditional centralized IDS systems that require data to be gathered and stored on a central server, FL-IDS allows clients to maintain their own data, thereby enhancing privacy. The central server manages a DN model that operates across these various IDS clients.

In our work, a federated DNN model is proposed for classifying network traffic as malicious or benign in an HFL setup, as described in Algorithm 2. The server initializes the global DNN model M_s with parameters $\theta^{(0)}$ and distributes them to N clients. Each client c_i receives the parameters $\theta^{(r-1)}$ at round r, samples a mini-batch $B \subset \mathbb{D}_{c_i}$, and trains a local DNN model M_{c_i} over E epochs using the Adam optimizer. The locally updated parameters $\theta_{c_i}^{(r)}$ are then sent back to the server. The server aggregates the updates using the FedAvg algorithm [91], computing a weighted average to obtain $\theta^{(r)}$, which is then sent back to all N. This process is repeated for R communication rounds until convergence.

Algorithm 2: FedAvg for FL-DNN based IDS

Input: N, $\theta^{(0)}$, η, $\mathbb{D}_{c_i}$, $\mathscr{E}$
Output: $\theta^{(R)}$
Initialize: $\theta^{(0)} \rightarrow \{c_1, c_2, \ldots, c_N\}$;
for $r = 1$ ***to*** R **do**
  for $c_i \in \{c_1, \ldots, c_N\}$ **do**
    Receive $\theta^{(r-1)}$;
    for $e = 1$ ***to*** $\mathscr{E}$ **do**
      Sample $B \subset \mathbb{D}_{c_i}$;
      `/* Local optimization */`
      $\theta_{c_i}^{(r)} \leftarrow \theta_{c_i}^{(r-1)} - \frac{\eta}{|B|} \sum_{(x,y)\in B} \nabla_\theta l(x, y; \theta_{c_i}^{(r-1)})$;
    Send $\theta_{c_i}^{(r)}$ to server;
  Server aggregation:
  $\theta^{(r)} \leftarrow \sum_{i=1}^{N} w_{c_i} \theta_{c_i}^{(r)}$ where $w_{c_i} = \frac{|\mathbb{D}_{c_i}|}{\sum_{j=1}^{N} |\mathbb{D}_{c_j}|}$;
  Distribute $\theta^{(r)} \rightarrow \{c_1, c_2, \ldots, c_N\}$;
return $\theta^{(R)}$

To classify network traffic characteristics—using flow-based, packet-based, and protocol-specific features—from the NSL-KDD and UNSW-NB15 datasets, an FL-based deep neural network (FL-DNN) was developed. The model includes an input layer, several densely con-

nected hidden layers, and a single-output layer neuron. The input layer has $d = 41$ features for the NSL-KDD dataset, $d = 46$ for the UNSW-NB15 dataset, $d = 39$ for CIC-IoT2023 and $d = 46$ for CIC-IoMT2024. The structure of the hidden layers is optimized through random experimentation for the best performance. The rectified linear unit (ReLU) activation function was used in all hidden layers to improve training efficiency. In the output layer, a single neuron assigns class labels for benign and attack traffic, utilizing the softmax function to provide output probabilities for accurate classification.

The number of communication rounds for training the model in FL was determined through a process of iterative experimentation. The process began with a random number of initial rounds which were adjusted based on the observed performance of the model. For instance, when training FL-DNN classifiers for binary classification, 50 communication rounds were utilized. Experimental results were presented and evaluated using two network traffic-based IDS datasets: NSLKDD and UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024. These datasets contain characteristics of network flows, including both packet-level and flow-based statistics for detecting attacks. To evaluate the effectiveness of the FL-DNN classification method, experiments were conducted to perform binary classification, distinguishing between benign and malicious network traffic. Each client's data was transformed and normalized using min-max scaling, standardizing it within the range of [0, 1] to improve the model's accuracy and performance.

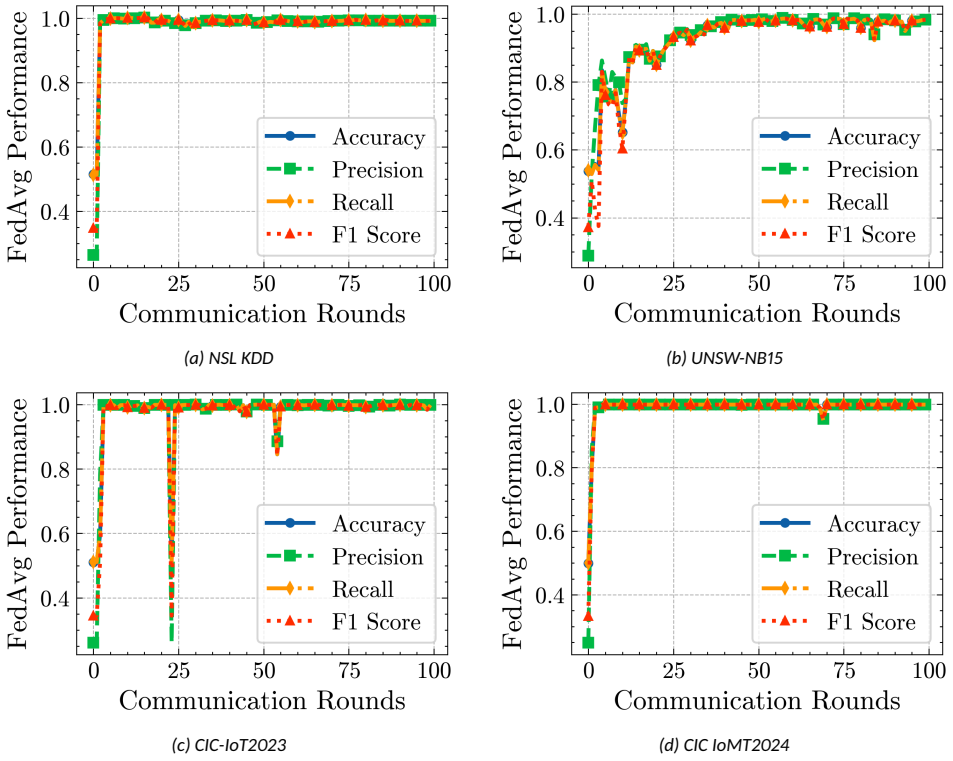


Figure 54: Global DNN model performance using FedAvg aggregation across communication rounds for NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets

Experiments were conducted with N = {5, 10, 15} clients in a non-IID setting. However, the results are specifically presented for the case with N = 5 clients, as similar performance

was observed across the other configurations for both the FL-DNN and GAN variants of FL models.

For all four datasets, the global DNN model displayed rapid convergence, with considerable improvements in classification performance metrics within the first ten communication rounds, followed by stabilization. For example, in Fig. 54a on the NSL KDD dataset, accuracy and recall exceeded 99% after round 15 with minimal fluctuations in subsequent rounds. Besides, as shown in Fig. 54b, the UNSW NB15 dataset demonstrated stable performance, with the DNN model attaining precision and recall values above 98% within 20 rounds. Similar performance was observed in CIC IoT2022 (Fig. 54c and CIC IoMT2024 (Fig. 54d) datasets, where the global DNN model consistently achieved high performance with classification metrics.

10.2 Federated Synthetic data Using GAN variants

In federated learning (FL) settings, explaining a client's model using post-hoc XIA methods is straightforward since clients have access to their own data. However, explaining the model on the server side is challenging because client data is never shared. Post-hoc methods require training or reference data, along with the trained model, for explanation. To address this issue, a synthetic data-driven approach was proposed in Publication X to explain the FL-DNN model (Global model on the server Side). For this purpose, various generative adversarial network (GAN) variants were used in a federated manner to generate synthetic data. This synthetic data served as input reference data for a post-hoc explainer, helping to explain the server model.

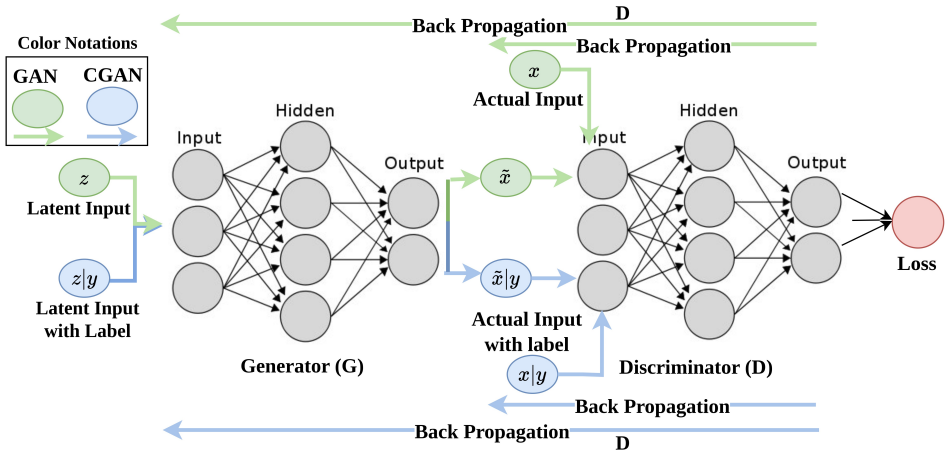


Figure 55: Architecture of Basic GAN and CGAN

Generative Adversarial Networks (GANs), introduced by Goodfellow et al. [40] in 2015, are deep generative models that function as a two-player game. They consist of two neural networks that adjust their weights to improve both data generation and discrimination capabilities.

Let z represent the random noise input, and $z \mid y$ denote the generator's input conditioned on the label y. Similarly, x and $x \mid y$ refer to the actual training data and the actual training data conditioned on the label y, respectively. The symbol $\tilde{x}$ represents the data generated by a standard GAN, while $\tilde{x} \mid y$ indicates the data generated by a conditional GAN conditioned on the label y. Fig. 55 shows an overview of both the standard GAN and

the conditional GAN (CGAN).

Weight updates are performed through backpropagation, with the discriminator fixed during generator updates. The key difference between Wasserstein GANs (WGAN) and standard GANs is in loss calculation and weight updates. For Publication. X, Four different types of generative model variants in FL settings were developed.

a) **GAN & CGAN** The Generative Adversarial Network (GAN) training procedure involves a game played between two competing neural networks: the generator network G and the discriminator network D. The generator G transforms a source of random noise into samples that resemble the input data space. Meanwhile, the discriminator D aims to differentiate between samples that come from the actual data distribution and those generated by G

These two networks engage in a two-player minimax game with the following equation

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \tag{24}$$

Here, P_r is the true data distribution, and P_g is the generative data distribution, which is implicitly defined by $x = G(z)$ where z is sampled from a simple noise distribution $p(z)$ (such as uniform, normal, or Gaussian distribution). The discriminator D is optimized to maximize the probability of correctly classifying both training samples and samples generated by G. Conversely, the generator G is optimized to minimize $\mathbb{E}_{x\sim P_g}[\log(1-D(x))]$ or $\mathbb{E}_{x\sim P_g}[-\log D(x)]$.

The Conditional Generative Adversarial Network (CGAN) is a variation of the GAN that incorporates additional information, y, into both the generator and discriminator. This information y can represent a class label or any other form of auxiliary data. Furthermore, the training process for CGAN is the same as that used in GAN.

Formally, the objective function for the generator G and the discriminator D is defined by the following minimax equation:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_r}[\log D(x|y)] + \mathbb{E}_{x\sim P_g}[\log(1-D(x|y))] \tag{25}$$

In this equation, P_r and P_g retain the same meanings as in GAN, and y is combined with the prior noise as inputs to the hidden layer. The optimization process in CGAN is almost similar to that of GAN.

Since CGANs are an extension of the original GAN variants, they share some of the same challenges, such as mode collapse and unstable training due to vanishing gradients, among other issues. Additionally, researchers have noted that when both GANs and CGANs utilize the Jensen–Shannon (JS) divergence as a metric for generative samples, they are limited to generating only continuous data, rather than discrete data [13].

$$\mathrm{JS}(P_r,P_g) = \mathrm{KL}(P_r\|P_m) + \mathrm{KL}(P_g\|P_m) \tag{26}$$

where KL denotes the Kullback–Leibler divergence, and P_m is the mixture distribution defined as $P_m = \frac{1}{2}(P_r + P_g)$.

b) **WGAN & WGAN-GP**

Unlike GAN and CGAN, WGAN and WGAN-GP utilize the Earth-Mover (EM) distance, also known as Wasserstein-1, instead of the JS divergence to measure the distance between the true data distribution and the generative data distribution, which is because the EM distance offers better smoothness compared to the JS divergence. Theoretically, WGAN addresses the vanishing gradient problem commonly observed in GAN and CGAN. Moreover, studies have demonstrated [13, 14] that replacing the JS divergence with the EM distance also helps mitigate the issue of mode collapse.

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{27}$$

Where $\Pi(P_r, P_g)$ is the complete set of feasible joint distributions $\gamma(x, y)$ for the true data distribution P_r and the generative data distribution P_g. The term $W(P_r, P_g)$ is defined as the minimum cost required to transport mass in order to transform the distribution P_r into the distribution P_g. Additionally, under mild assumptions, $W(P_r, P_g)$ is continuous and differentiable almost everywhere. However, Equation 27 is highly intractable; therefore, the EM distance can be reconstructed using the Kantorovich-Rubinstein duality [147].

$$W(P_r, P_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)] \tag{28}$$

Here, the supremum is over all K-Lipschitz functions $f : x \to \mathbb{R}$, with K being the Lipschitz constant (set to 1 for the original WGAN). This forms the minimax objective for the generator and critic shwon in Equation 27

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{x\sim P_g}[D(x)] \tag{29}$$

WGAN often struggles with producing high-quality samples and can fail to converge due to weight clipping, which is an ineffective method for enforcing a Lipschitz constraint on the discriminator. To solve this, Gulrajani et al. proposed WGAN-GP, which uses a penalty on the gradient norm of the discriminator instead of weight clipping. Their results showed that WGAN-GP outperforms the standard WGAN and allows stable training of various GAN architectures with minimal hyperparameter tuning, represented by the minimax formulation in Equation 30

$$\begin{aligned} \min_G \max_D V(D, G) = {} & \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x})] \\ & - \lambda\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1 \right)^2 \right] \end{aligned} \tag{30}$$

where λ is the gradient penalty coefficient, and $\hat{x}$ is sampled along straight lines between the true data distribution P_r and the generative data distribution P_g: $\hat{x} = \varepsilon x + (1 - \varepsilon)\tilde{x}$, where $\varepsilon \sim \text{Uniform}[0, 1]$, $x \sim P_r$, and $\tilde{x} \sim P_g$.

c) **WCGAN-GP** The development of WGAN-GP can extend to WCGAN-GP by conditioning both the discriminator and generator on auxiliary information y, which represents class labels in this study.

In the discriminator, we concatenate the real data distribution P_r and the generated data distribution P_g with y to form a joint hidden representation. The generator also concatenates y with $p(z)$ in the same way. The objective function for the generator and discriminator is defined as a minimax problem below

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_r}[D(x|y)] - \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x}|y)] - \lambda\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x}|y)\|_2 - 1\right)^2\right] \tag{31}$$

Here, λ denotes the gradient penalty coefficient, and the sampling strategy for $\hat{x}$ follows that of WGAN-GP. The discriminator and generator loss functions are defined as:

$$L(D) = -\mathbb{E}_{x\sim P_r}[D(x|y)] + \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x}|y)] + \lambda\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x}|y)\|_2 - 1\right)^2\right] \tag{32}$$

$$L(G) = -\mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x}|y)] \tag{33}$$

To enable explainability without accessing real client data in FL settings, we use synthetic data generated through federated generative models. Each client maintains a local model that generates synthetic network traffic samples based on its dataset. These local model parameters are sent to a central server and aggregated using FedAvg to form a global generative model. This global model generates synthetic data on the server, which is then used as reference input for a post-hoc XAI method to explain the global FL DNN model.

In Algorithm 3, the server begins by initializing the global generator $\theta_{G_s}^{(0)}$ with parameters θ_0, and distributes them to all N participating clients. At each communication round r, each client c_i receives the latest global generator parameters $\theta_{G_s}^{(r-1)}$. Locally, each client updates its generator G_{c_i} and discriminator D_{c_i} using its local client dataset $\mathbb{D}_{c_i}$. Specifically, a mini-batch of real samples $x \subset \mathbb{D}_{c_i}$ is drawn, along with a corresponding batch of latent noise vectors $z \sim p(z)$. The discriminator D_{c_i} is trained to maximize its ability to distinguish between real and generated samples, while the generator G_{c_i} is trained to minimize the discriminator's ability to do so, thereby improving the realism of the generated data. This min-max training procedure is performed locally over $\mathscr{E}$ epochs using the Adam optimizer.

After local training, only the updated generator parameters $\theta_{G_{c_i}}^{(r)}$ are sent back to the server. The server performs weighted aggregation using the FedAvg algorithm to compute the updated global generator parameters $\theta_{G_s}^{(r)}$, where the weights w_{c_i} are proportional to the size of each client's dataset, i.e., $w_{c_i} = \frac{|\mathbb{D}_{c_i}|}{\sum_{j=1}^{N} |\mathbb{D}_{c_j}|}$. Importantly, after each aggregation step, the global generator G_s with parameters $\theta_{G_s}^{(r)}$ is used to generate synthetic samples

Algorithm 3: Federated generative model

Input: N, $\mathbb{D}_{c_i}$, G_{c_i}, $\theta_{G_s}^{(0)}$, D_{c_i}, η_G, η_D, R, $\mathscr{E}$

Output: $\theta_{G_s}^{(R)}$

Initialize: $\theta_{G_s}^{(0)} \rightarrow \{c_1, c_2, \ldots, c_N\}$;

for $r = 1$ ***to*** R **do**

  for $c_i \in \{c_1, \ldots, c_N\}$ **do**

    $\theta_{G_{c_i}}^{(r)} \leftarrow \theta_{G_s}^{(r-1)}$;

    for $e = 1$ ***to*** $\mathscr{E}$ **do**

      Sample $\{x\} \subset \mathbb{D}_{c_i}$;

      $z \sim p(z)$;

      `/* Local discriminator update */`

      $\theta_{D_{c_i}} \leftarrow \theta_{D_{c_i}} - \eta_D \nabla_{\theta_D} \mathscr{L}_D(x, z)$;

      `/* Local generator update */`

      $\theta_{G_{c_i}}^{(r)} \leftarrow \theta_{G_{c_i}}^{(r)} - \eta_G \nabla_{\theta_G} \mathscr{L}_G(z)$;

    Send $\theta_{G_{c_i}}^{(r)}$ to server;

  Server aggregation:

  $\theta_{G_s}^{(r)} \leftarrow \sum_{i=1}^{N} w_{c_i} \theta_{G_{c_i}}^{(r)}$, where $w_{c_i} = \frac{|\mathbb{D}_{c_i}|}{\sum_{j=1}^{N} |\mathbb{D}_{c_j}|}$;

  `/* Generate synthetic samples on server */`

  $z \sim p(z)$;

  $\tilde{x}_s^{(r)} \sim G_s(z; \theta_{G_s}^{(r)})$;

return $\theta_{G_s}^{(R)}$

$\tilde{x}_s^{(r)} \sim G_s(z; \theta_{G_s}^{(r)})$, where the latent noise $z \sim p(z)$ is sampled from a predefined prior distribution. This process is repeated for R communication rounds.

In this study, we explore four types of generative model variants within the FL framework: standard GAN, CGAN, WGAN-GP, and WCGAN-GP. Algorithm 3 illustrates the federated training procedure for the standard GAN case, referred to as FL-GAN. For other variants (FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP), the training follows the same procedure, with only the loss functions modified according to the specific generative model variant used.

We implemented four generative model variants: FL-GAN, FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP, all trained under Non-IID conditions using the Federated Averaging (FedAvg) algorithm. During training, only the generator parameters were aggregated on the server, while the discriminator parameters remained local to each client, ensuring data privacy.

To assess performance, we used Wasserstein Distance (W_D) as a metric to measure the discrepancy between the distributions of real and synthetic data at server side. A W_D value close to zero indicates that the distributions are similar, while a larger value indicates greater disparity. We computed W_D between the real test data and synthetic data generated by the global model across $R = 1000$ communication rounds, using test data at each round for an unbiased evaluation of synthetic data quality.

Figure. 56a illustrates the W_D performance of four global generative model variants

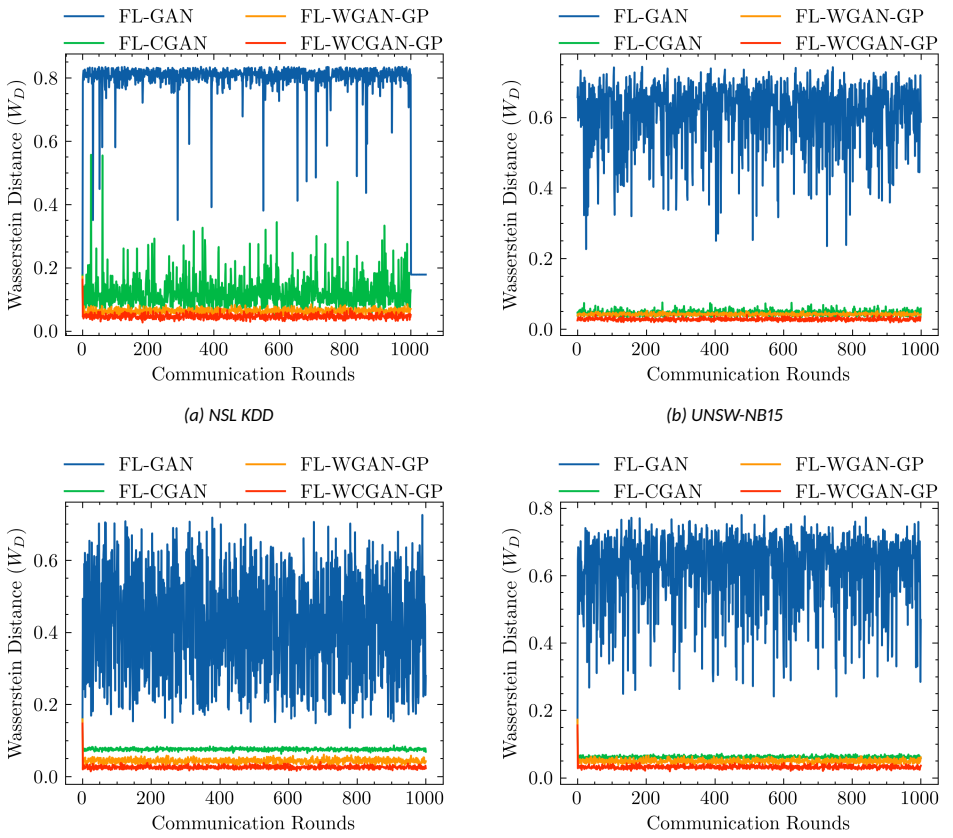


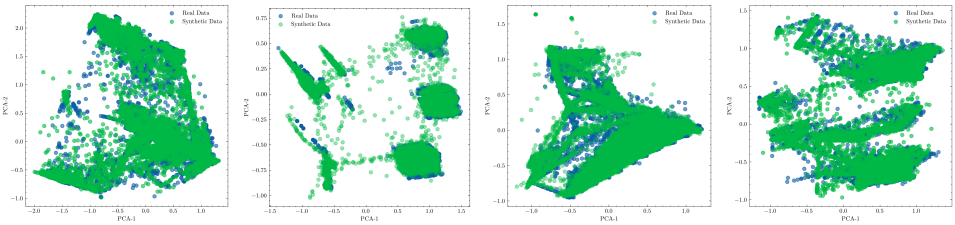
Figure 56: Performance of federated generative models across communication rounds for the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets.

on the server in FL using FedAvg on the NSL-KDD dataset. Basic FL-GAN demonstrates the highest W_D values, fluctuating between 0.6 and 0.8, indicating a significant discrepancy between the real and synthetic data distributions over several communication rounds. FL-CGAN model displays better stability over the rounds, but noticeable fluctuations were still shown, especially in the early rounds. In contrast, the Wasserstein-based generative models in FL perform better in generating synthetic data that was much closer to the real data distribution. Both FL-WGAN-GP and FL-WCGAN-GP achieved lower W_D values across the communication rounds, demonstrating their effectiveness.

Similarly, Figure. 56b shows W_D performance of four GAN variants in FL using FedAvg on the UNSW-NB15 dataset over 1000 communication rounds. Basic FL-GAN has the highest W_D values, whereas the FL-CGAN consistently performs better with lower W_D values, despite some minor fluctuations over rounds. Wasserstein GAN variants (FL-WGAN & FL-WCGAN-GP) outperform other GAN variants in producing high-quality synthetic data. Both FL-WGAN-GP and FL-WCGAN-GP consistently exhibit lower W_D values throughout the communication rounds. Notably, FL-WCGAN-GP achieved the lowest W_D value, successfully generating high-quality synthetic samples from the global generator for the UNSW-NB15 dataset.

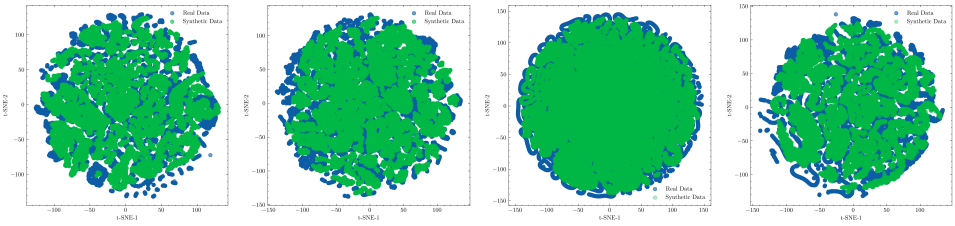
Similar W_D performance was observed across generative models for the CIC-IoT2023 (Figure. 56c) and CIC-IoMT2024 datasets (Figure. 56d). Among the federated generative models, the FL-GAN model produced the least accurate synthetic data, as indicated by its higher W_D values. While the FL-CGAN model showed improved performance, it showed noticeable fluctuations in W_D across communication rounds on both datasets. In contrast, the Wasserstein-based models, FL-WGAN-GP and FL-WCGAN-GP, consistently outperformed the other variants. Notably, FL-WCGAN-GP achieved the lowest W_D values and demonstrated high stability, approaching zero, which highlights its effectiveness in producing high-fidelity synthetic data in federated settings.

Among the evaluated generative models (FL-GAN, FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP), FL-WCGAN-GP demonstrated superior performance in minimizing the Wasserstein Distance (W_D) across communication rounds for all four datasets. The FL-WCGAN-GP model consistently achieved the lowest W_D values, close to zero, demonstrating its effectiveness in generating high-quality synthetic data that closely approximates the distribution of real data. Based on these results, we selected the FL-WCGAN-GP global generative model to produce synthetic data on the server side. This synthetic data was subsequently used as the reference input for the post-hoc XAI method employed to explain the global FL-DNN classifier.



(a) NSL KDD *(b) UNSB-nb15* *(c) CIC-IoT2023* *(d) CIC IoMT2024*

Figure 57: 2-D visualization using PCA for real and synthetic data generated by FL-WCGAN-GP for all four datasets

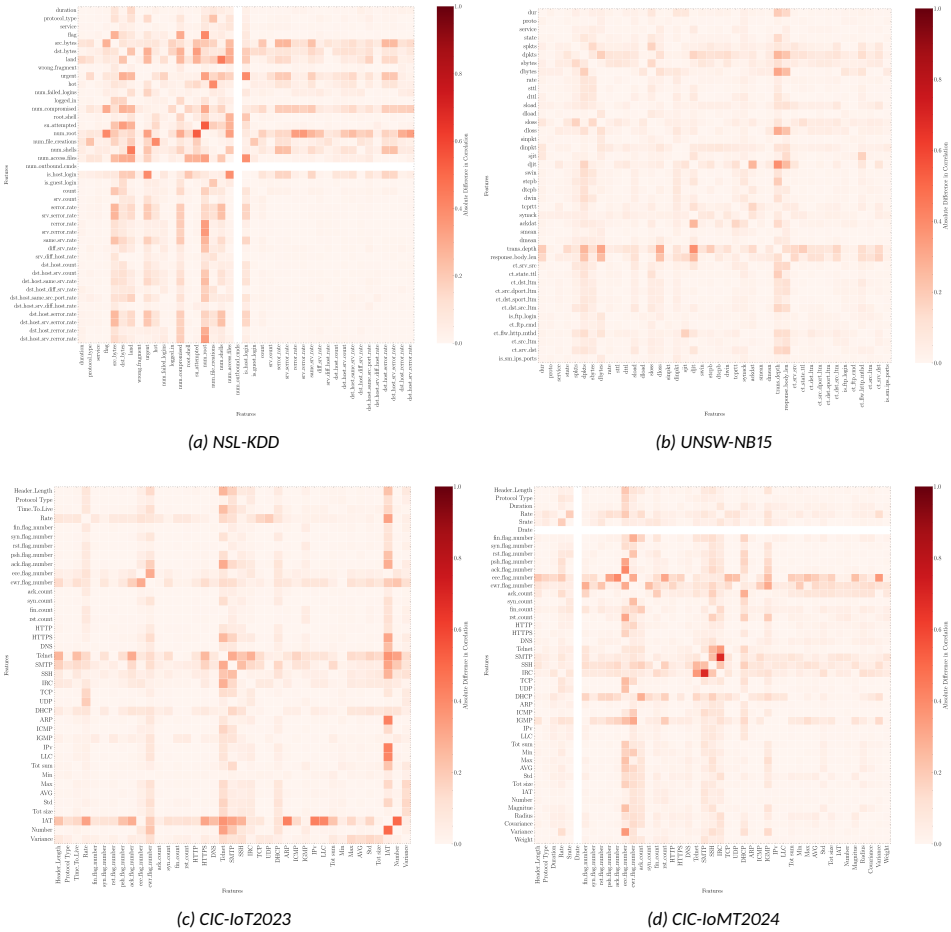


(a) NSL KDD *(b) UNSW-NB15* *(c) CIC-IoT2023* *(d) CIC IoMT2024*

Figure 58: 2-D visualization using t-SNE for real and synthetic data generated by FL-WCGAN-GP for all four datasets

We further examined the differences between synthetic data generated by the FL-WCGAN-GP global model and real data using two-dimensional representations from Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE).

PCA is a statistical method for linear dimensionality reduction that transforms high-dimensional data into a lower-dimensional form while preserving crucial information by identifying principal components along which the data exhibits the greatest variance. In



(a) NSL-KDD

(b) UNSW-NB15

(c) CIC-IoT2023

(d) CIC-IoMT2024

Figure 59: Feature-pairwise Absolute differences in Pearson correlations between real and synthetic data generated by the FL-WCGAN-GP model for the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets

our study, we reduced the dimensionality from 41 features (NSL-KDD), 46 features (UNSW-NB15), 39 features (CIC-IoT2023), and 45 features (CIC-IoMT2024) to two dimensions for visualization using PCA and t-SNE.

Figure 57 displays the 2D PCA projections for all four datasets. The synthetic data generated by the global FL-WCGAN-GP model closely aligns with the distribution of the real test data in the principal component space. FL-WCGAN-GP effectively captures the variance structure of the original datasets, resulting in significant overlap with the synthetic data, despite minor deviations in high-density areas. This consistency highlights the FL-WCGAN-GP's ability to generate high-quality synthetic data.

t-SNE is a nonlinear dimensionality reduction method designed for visualizing high-dimensional data in lower-dimensional spaces while preserving the data's structure. Figure 58 illustrates the 2-D representations of real data and synthetic data generated by the FL-WCGAN-GP model across four datasets. The synthetic samples closely overlap with the real test data, indicating effective preservation of local data structures. While minor discrepancies appear at the edges of dense clusters, the overall synthetic data aligns well

with the real data.

To further evaluate the quality of the synthetic data generated by the FL-WCGAN-GP model in the federated learning setup, we analyzed the Pearson correlation values of the real and synthetic data. We computed these correlations for both datasets and calculated the absolute differences for each feature pair, as shown in Figure. 59 across all four datasets. The absolute difference values range from $[0, 1]$; values closer to zero indicate better alignment of feature relationships between the synthetic and real data, while values farther from zero represent greater discrepancies.

Figure 59a displays a heatmap showing the absolute differences in Pearson correlation coefficients between real and synthetic data for the NSL-KDD dataset, which contains 41 features. Most feature pairs exhibit minimal differences, indicating that the FL-WCGAN-GP model effectively generates synthetic data resembling the real data. However, some feature pairs show larger discrepancies.

Figure 59b presents similar findings for the UNSW-NB15 dataset, which consists of 45 features. Most feature pairs demonstrate low absolute differences, suggesting that the model retains key correlations. Nonetheless, a few feature pairs reveal higher discrepancies from the real data.

Figures 59c and 59d show the absolute differences in Pearson correlations for the CIC-IoT2023 and CIC-IoMT2024 datasets, containing 39 and 45 features, respectively. Here too, most feature pairs exhibit low absolute differences, indicating the model's effectiveness in generating synthetic data that closely resembles the real test data, despite some moderate deviations in a few feature pairs.

Table 19: Performance of global FL-DNN model on Real Data and Synthetic data generated by FL-WCGAN-GP generative model

Data Set	Data-type	Accuracy	Precision	Recall	F1-score	FPR
NSL-KDD	Real	99.31%	99.50%	98.88%	99.18%	1.12%
	Synthetic	96.32%	96.27%	93.11%	94.57%	6.89%
UNSW-nb15	Real	98.46%	98.45%	98.48%	98.46%	1.52%
	Synthetic	95.12%	90.81%	96.10%	93.09%	3.90%
CiC-IoT2023	Real	99.24%	99.25%	99.24%	99.24%	0.76%
	Synthetic	97.11%	97.20%	97.11%	95.90%	2.89%
CIC-IoMT2024	Real	99.94%	99.94%	99.94%	99.94%	0.06%
	Synthetic	98.86%	98.88%	98.85%	98.86%	1.15%

Table 19 shows the classification performance of the global FL-DNN model on real and synthetic test data from four datasets: NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024. The evaluation includes metrics such as accuracy, precision, recall, F1-score, and FPR (the rate of benign samples incorrectly classified as malicious).

FL-DNN model consistently achieves high performance on real test data, with accuracy and F1-scores above 98% and low FPR values. Its best performance is on the CIC-IoMT2024 dataset, with nearly perfect accuracy, an F1-score of 99.94%, and an FPR of just 0.06%. Strong results are also noted for the CIC-IoT2023 and UNSW-NB15 datasets.

FL-DNN classifier demonstrates strong performance on synthetic data generated by the FL-WCGAN-GP model. While there is a slight drop in performance compared to real data, accuracy, precision, and F1-scores remain notable, with F1-scores of 98.86% for CIC-IoMT2024 and 95.90% for CIC-IoT2023. The false positive rate is slightly higher for syn-

thetic data, reflecting minor differences in distribution. Overall, the quality of the synthetic data is sufficient for model evaluation and explainability without needing access to real client data.

10.3 Explaining DNN model in FL using Synthetic Data

In this Publication. X, we propose to utilize the post hoc SHAP XAI method to explain the black box nature of the DL classification model trained using the FedAvg algorithm. We note that SHAP requires access to either the training data $\mathbb{D}^{\text{train}}$ or a "reference set" that is similar to the training set used by the model. This is necessary to create records (q) that determine the impact of each feature value on the final prediction. To expedite the explanation process, a medoid of the dataset is sometimes used, or a small set of centroids [139] is utilized to represent $\mathbb{D}^{\text{train}}$, capturing the main characteristics with a few records of feature importance [97]. Consequently, in server-based FL settings, explaining the learned global model requires the server to have access to the complete set of training data $\mathscr{D}^{\text{Train}}$ from its clients. Alternatively, the server should be able to compute the centroids of a dataset formed by combining the training sets of all clients. However, this approach depends on access to client training data, making it unsuitable in scenarios where privacy regulations prevent data sharing with the server.

After completing the training process in the FL setting, each client c_i holds its local model M_{c_i}, which is then transmitted to the server. The server aggregates the received models using the FedAvg algorithm to construct the global model M_s. Traditionally, explanations for a given instance x_j are computed locally by each c_i using a post-hoc explanation method $\mathbf{g}$, yielding attribution scores $\varphi_{c_i}(x_j) = \mathbf{g}(M_{c_i}, x_j)$, which reflect the contribution of each feature toward the prediction made by M_{c_i}.

To enable post-hoc explainability in FL without compromising data privacy, we propose an approach that leverages synthetic data generated by the global federated generator G_s to interpret the server-side black-box model M_s. For each communication round r, the server updates the global generator G_s to generate synthetic samples $\tilde{x}_s^{(r)} \sim G_s(z; \theta_{G_s}^{(r)})$.

After all communication rounds, the final global generator $G_s^{(R)}$ defines a synthetic distribution P_g^s that closely approximates the true data distribution P_r. This synthetic data is used as the input reference to interpret the black-box classifier M_s without relying on client data. We use a model-agnostic post-hoc explainer $\mathbf{g}$ to obtain the feature attributions $\varphi_s(\tilde{x}) \in \mathbb{R}^d$ for any synthetic instance $\tilde{x} \in \mathbb{R}^d$, defined as:

$$\varphi_s(\tilde{x}) = \mathbf{g}(M_s, \tilde{x}), \quad \tilde{x} \sim P_g^{s,(R)} \tag{34}$$

In our study, we use two post-hoc XAI methods, LIME and SHAP, to explain the global DNN model using synthetic data without relying on real client data.

Analytical Methodology: In our experiments, we aim to evaluate whether synthetic data-driven SHAP explanations are sufficient to approximate real data-based explanations for the global model M_s in a federated setting. To this end, we propose an analytical methodology that compares explanations derived from two settings:

(i) *Real data-based explanation:* The server accesses the real training data $\mathbb{D}_{c_i}^{\text{train}}$ from each client and computes SHAP values.

(ii) *Synthetic data-based explanation:* The server computes SHAP values using synthetic data generated by the global generator G_s, without access to real client data.

To perform the comparison, the following steps are carried out using each client's test data $\mathbb{D}_{c_i}^{\text{test}}$:

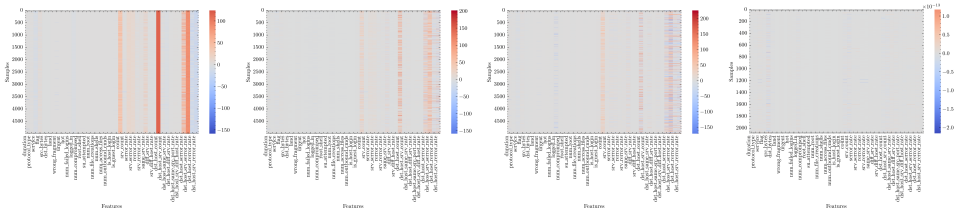
- For each client c_i, Server side SHAP explainer takes test data $\mathbb{D}_{c_i}^{\text{test}}$ and server model M_s, SHAP values are computed, and computes explanation matrices $E_{\text{real}}^{(c_i)} \in \mathbb{R}^{m \times d}$, where m is the number of instances and d the number of features.

- A global explanation matrix $E_{\text{real}} \in \mathbb{R}^{m \times d}$ is derived by averaging SHAP values across all clients:

$$e_{ij}^{\text{real}} = \frac{1}{N} \sum_{c_i=1}^{N} e_{ij}^{(c_i)}.$$

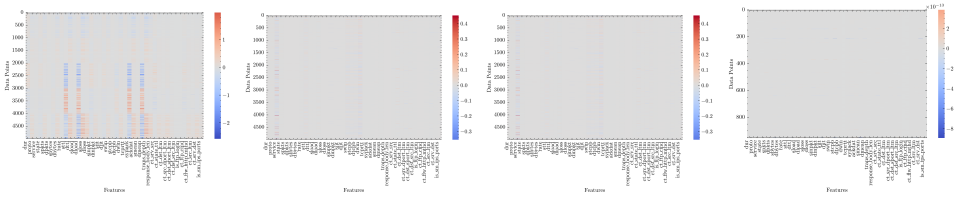
- Similarly, The server samples m synthetic instances from the distribution P_g^s using the final global generator $G_s^{(R)}$, and computes SHAP values for the global model M_s, computing the synthetic-data based SHAP explanation matrix $E_{\text{syn}} \in \mathbb{R}^{m \times d}$.

- A difference matrix $\Delta = E_{\text{syn}} - E_{\text{real}}$ is computed between the synthetic data-based and real data-based explanation matrices. If $\Delta_{ij} \approx 0$ for all i, j, this indicates that the synthetic data-driven explanations are sufficiently close to real data-based explanations, for explaining the global classifier M_s.



(a) FL-GAN *(b) FL-CGAN* *(c) FL-WGAN* *(d) FL-WCGAN*

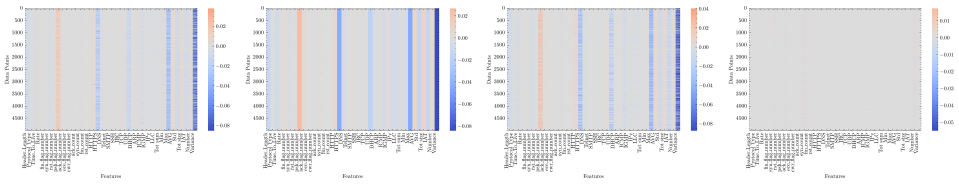
Figure 60: Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on NSL-KDD dataset.



(a) FL-GAN *(b) FL-CGAN* *(c) FL-WGAN* *(d) FL-WCGAN*

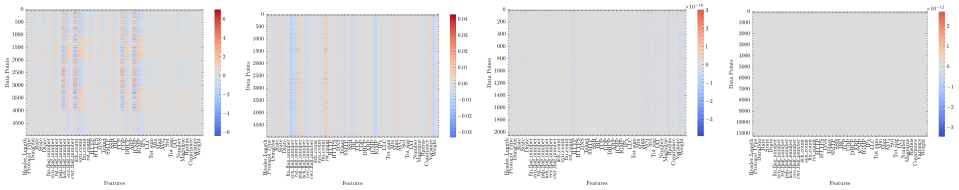
Figure 61: Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the UNSW-NB15 dataset.

To explain the global DNN model on the server side, we utilized the SHAP explainer with synthetic data generated by different federated generative models. SHAP calculates the Shapley value, which indicates the influence of features on model predictions. Specifically, SHAP was applied to the global DNN model obtained at the 100th communication round, using synthetic data as the input reference to compute Shapley values. To assess whether synthetic data is sufficient for constructing reliable explanations for a global DNN model, the server accessed real client data to compute SHAP values again on the same



(a) FL-GAN *(b) FL-CGAN* *(c) FL-WGAN* *(d) FL-WCGAN*

Figure 62: Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the CIC-IoT2023 dataset



(a) FL-GAN *(b) FL-CGAN* *(c) FL-WGAN* *(d) FL-WCGAN*

Figure 63: Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the CIC-IoMT dataset

model. We then calculated the difference $\Delta = E_{\text{real}} - E_{\text{syn}}$, where E_{real} is the SHAP values derived from real data-based explanations, E_{syn} is synthetic data-based SHAP explanations.

Figs. 60–63 present SHAP delta matrices as heatmaps for the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT datasets. Each heatmap visualizes the difference (Δ) in feature attributions between synthetic data-based explanations and real data-based explanations across all features. When the difference closes to zero ($\Delta \approx 0$), this indicates a close alignment between E_{real} and E_{syn}, implying that synthetic data can effectively approximate real data for explaining the global DNN model. Conversely, when Δ values deviate significantly away from zero, it implies that the synthetic data is insufficient to produce reliable explanations.

SHAP values difference (Δ) shown in Figs. 60–63 reveal significant variation in explanation fidelity across different generative models in four datasets. FL-GAN, FL-CGAN, and FL-WGAN-GP show varying levels of divergence between E_{real} and E_{syn}. In particular, FL-GAN shows the highest discrepancy, with SHAP values in difference (Δ) in the heatmaps deviating substantially from zero, displaying noticeable variability in feature attributions, for all four datasets.

FL-CGAN and FL-WGAN-GP show moderate improvements, with several features exhibiting smaller differences in Δ closer to zero, which suggests a better alignment between synthetic data-based SHAP values and real data-based SHAP explanations. Nevertheless, considerable variation remains, particularly in CIC-IoT2023 and CIC-IoMT2024, where feature attribution differences are still evident.

Among all the evaluated federated generative models, the FL-WCGAN-GP generative model on the server side consistently produced the smallest differences in Δ , which are close to zero in all four datasets. This implies that the synthetic data-based SHAP explanations are closely aligned with explanations derived from real data, highlighting FL-WCGAN-GP's effectiveness in generating high-fidelity synthetic data suitable for server-

side explainability of the global FL-DNN model.

Conditional GAN models (CGANs and WCGANs) require both a noise vector and a class label as inputs. This ensures that the generator produces synthetic samples aligned with the specified target label, which is vital for explainability methods that rely on class-specific data. To ensure reliable explanations of the global DNN model, we selected high-fidelity synthetic samples closely resembling real data. We calculated pairwise Euclidean distances and applied a 0.1 threshold to exclude outliers from the synthetic data distribution.



(a) Real data point

(b) Synthetic data point

Figure 64: SHAP local explanations for the global DNN model on an attack instance from the NSL-KDD dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.

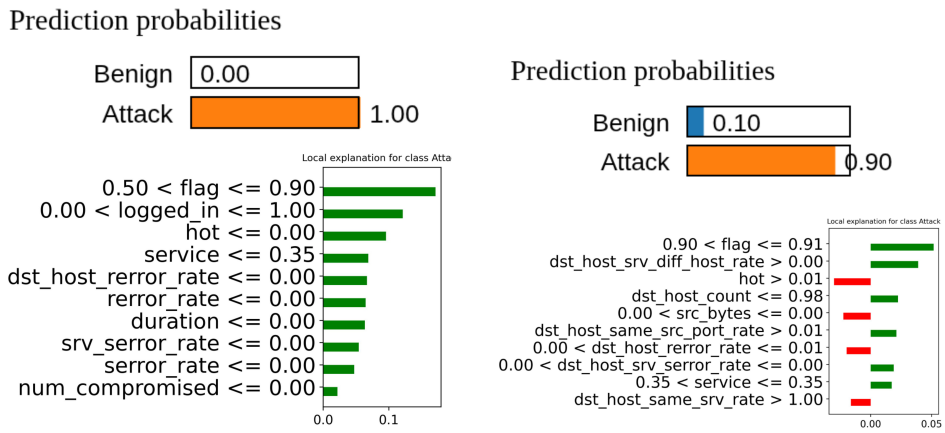
To demonstrate the use case of SHAP for local explainability of a global DNN model on a server, we randomly selected a real data-based instance and a synthetic data-based instance from the NSL-KDD dataset, both of which were classified as "attack." To obtain prediction probabilities for each class label, we applied the softmax activation function at the output layer of the DNN model. This function is essential for interpreting the model's confidence, as it transforms output logits into a normalized probability distribution over the class labels (attack and benign).

Fig. 64 presents SHAP local explanation plots for a single instance classified as an attack in the NSL-KDD dataset, using both real and synthetic data to illustrate each feature's contribution to the prediction of the global DNN model on the server. The plot illustrates the base value, with features that positively influence the prediction represented in red and those that negatively impact the predictions shown in blue. The base value is the average of all prediction values. Each strip in the plot illustrates how different features impact the predicted value, either moving it closer to or further away from the base value. Red strips indicate features that push the predicted value higher, while blue strips show features that push it lower. The width of each strip reflects the strength of the feature's contribution have wider strips indicate a greater impact on the predicted value.

For example, Fig. 64a illustrates the SHAP local explanation for a real data point of an attack class from the NSL-KDD dataset. The global DNN model predicts this instance with a score of 1.0. The base value is 0.4988, and key features such as user login status (logged_in), connection status flag (flag), network service on the destination (service), percentage of connections to the same host that have "REJ" errors (dst_host_rerror_rate), and number of connections to the same service as the current connection in the past 100 (dst_host_srv_count) positively contribute to the final prediction by pushing the output

away from the base value toward the attack class.

Similarly, Fig. 64b presents the SHAP local explanation for a synthetic data point of the attack class, generated by the global FL-WCGAN-GP model. The predicted value of the global DNN model using this synthetic data point is 0.90. Notably, the same influential features (including logged_in, flag, dst_host_rerror_rate, and dst_host_srv_count) appear again, contributing positively to the attack prediction. However, the feature number of bytes sent from source to destination (src_bytes) negatively contributes to the prediction in the synthetic case.



(a) Real Data point *(b) Synthetic Data point*

Figure 65: LIME local explanations for the global DNN model on an attack instance from the NSL-KDD dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.

We have also provided local explanations using the LIME explainer. LIME method explains the rationale behind assigning probabilities to each class by comparing the probability values with the actual class of the data point. As with SHAP, to illustrate local explanations for the global DNN model using LIME, we selected both a real data point and a synthetic data point generated by the FL-WCGAN-GP model for all four datasets.

Fig. 65 illustrates the local explanations generated by LIME for the global DNN model on an attack instance from the NSL-KDD dataset. In these visualizations, green bars represent features that contribute positively toward predicting the data point as belonging to the attack class, while red bars indicate features that influence the prediction toward the benign class label.

In Fig. 65a, the LIME explanations for the global DNN model indicate that the real data point is predicted to be an attack with 100% confidence. Features such as "flag," "logged_in," "hot," "service," "dst_host_rerror_rate," and "rerror_rate" are among the top contributors, as represented by the green bars, highlighting their significant influence on the prediction of the attack class.

In Fig. 65b, the synthetic data point generated by the FL-WCGAN-GP model yields a 90% prediction for the attack class and 10% for the benign class. features shown in green bar color in like flag, dst_host_srv_dff_host_rate , dst_host_count and so on contributes to predicting the class attack, on other hand features shown in red bars like hot, src_bytes, dst_host_rerror_rate, dst_host_same_srv_rate contribute to predicting the Benign class label.

Table 20: Evaluation of SHAP and LIME local explanations on synthetic data for the global DNN model on server using Faithfulness (μ_f) and Max Sensitivity (μ_s) metrics across four datasets.

Dataset	Metric	LIME	SHAP
NSL-KDD	μ_f	0.2087 ± 0.1731	0.6628 ± 0.3006
	μ_s	0.1723 ± 0.0509	0.0170 ± 0.0341
UNSW-nb15	μ_f	0.2853 ± 0.3931	0.5949 ± 0.4048
	μ_f	0.0389 ± 0.0107	0.0206 ± 0.0697
CIC-IoT2023	μ_f	0.3372 ± 0.3682	0.6113 ± 0.1802
	μ_s	0.1314 ± 0.0397	0.0512 ± 0.0455
CIC-IoT2024	μ_f	0.2635 ± 0.2165	0.7884 ± 0.3415
	μ_s	0.2052 ± 0.0560	0.0406 ± 0.0395

Table 20 presents the quantitative evaluation of post-hoc local explanations generated using SHAP and LIME for the global DNN model on the server, based on synthetic data points generated by a global generative FL-WCGAN-GP model. For all four datasets, the mean and standard deviation were computed over 1000 synthetic instances of post hoc local explanations using two XAI metrics, high faithfulness (μ_f) and max sensitivity (μ_s). The better-performing XAI method for each metric and dataset is highlighted in gray in the table.

For the NSL-KDD dataset, SHAP explanations demonstrate significantly higher faithfulness compared to LIME. SHAP achieves the highest mean faithfulness score of $\mu_f = 0.6628 \pm 0.3006$, indicating that the feature attributions produced by SHAP are closely aligned with the actual influence of input features on global DNN model's predictions. This high correlation indicates that SHAP explanations are highly faithful when applied to the global DNN model using synthetic data, making them more accurate and trustworthy than LIME explanations. In terms of robustness for NSL-KDD , SHAP outperforms LIME. It yields the lowest max sensitivity value of $\mu_s = 0.0170 \pm 0.0341$, which means its explanations are more stable when small changes are made to the input data. This low sensitivity suggests that SHAP generates consistent explanations for nearby synthetic samples.

Similarly, in Table 20, for the UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets, SHAP explanations consistently outperform LIME, exhibiting greater fidelity and more robustness in local explanations for the global DNN model using synthetic data generated by the global FL-WCGAN-GP generative model.

10.4 Chapter Discussions

The response to **RQ4** is provided in this section. The research question aimed to investigate how post-hoc explainability can be incorporated into FL environments while preserving the privacy concerns. Previous research has limitations, primarily focusing on achieving the server model, which largely depends on client data and contradicts the core principles of FL. To address this issue, we propose an a Synthetic Data-Driven Explainability Framework that enhances transparency in FL-based IDS models while preserving client data privacy. Our approach utilizes a variant of GANs trained in a federated manner to generate synthetic data. This synthetic data serves as a reference for post-hoc XAI methods, enabling the explainability of the server-side global model without requiring access to real client data, thereby ensuring privacy preservation in FL-based IDS. The experiments were conducted on two widely-used datasets, NSL-KDD, UNSW-NB15, CIC-IoT2023 & CIC-IoMT2024.

For federated synthetic data generation purposes, we explored multiple GAN variants within FL, including FL-GAN, FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP, all trained using the FedAvg algorithm. Among these, FL-WCGAN consistently outperformed the other variants, achieving the lowest Wasserstein distance (W_D) values across communication rounds. This indicates that FL-WCGAN-GP generated synthetic data that closely resembles real data, making it the most suitable GAN variant for explainability in FL-based IDS. To further assess the quality of the synthetic data, we evaluated the similarity between synthetic and real data distributions using Pearson correlation, PCA and t-SNE. The results confirmed that the federated synthetic data generated by global model of FL-WCGAN closely aligns with real data, demonstrating its effectiveness in preserving data while maintaining privacy.

FL-DNN model was trained for binary classification (attack vs. benign) and demonstrated strong performance, achieving 99% accuracy, precision, recall, and F1-score across both the NSL-KDD , UNSW-NB15, CIC-IoT2023 & CIC-IoMT2024 datasets. We used synthetic data as an input reference for the SHAP explainer to explain the global FL-DNN model. Specifically, we used synthetic data generated by the FL-WCGAN model in a federated manner. The SHAP-based local and global explanations for the FL-DNN model using synthetic data closely aligned with those derived from real data. The most influential features for attack detection remained consistent between real and synthetic data-based explanations, demonstrating that the synthetic data effectively preserved key feature attributions in the FL-DNN model.

11 Conclusions and Future Works

This doctoral dissertation addresses the challenges and research gaps related to the design and effectiveness of machine learning-based intrusion detection systems. It specifically emphasizes the impact of feature selection on the detection of IoT botnets and IoMT attacks. Furthermore, the dissertation discusses achieving the transparency of black box models through post hoc explainability and evaluates the local explainability for intrusion detection systems, particularly in the contexts of IoT botnet detection, network intrusion detection alert classification, and IoMT attack detection. Additionally, this dissertation contributes to the Explainable Active Learning paradigm, which integrates post hoc explainability methods and evaluates these methods within the active learning loop for IoT botnet detection that focuses on selectivity and labelling of the most informative data points in active learning to improve performance of the model while ensuring transparency throughout the process. Another proposed approach, FEDXAI for IDS task, integrates FEDXAI to ensure trustworthy black box models. This enables the simultaneous pursuit of transparency and privacy preservation in intrusion detection tasks. The main findings of the Publications correspond to the topics presented in each chapter (from Chapter 4 to Chapter 10) and focus on enhancing the effectiveness of ML-based IDSs.

Botnets have become the preferred modus operandi for hackers orchestrating large-scale cyber-attacks in IoT networks. Small and mid-sized businesses are particularly vulnerable to these attacks due to their limited security resources. IoT botnet attacks can have dire consequences for both individuals and companies, leading to financial losses, damage to servers, and negative impacts on reputation, which can result in a loss of trust from current and potential customers. Due to the many risks of network attacks (Eg: DoS, DDoS) in IoT environments, several threat detection solutions have been created to defend against these threats. Notably, Intrusion detection systems (IDSs) help networks resist external attacks. The goal of IDSs is to provide a defence against attacks on computer systems connected to the Internet. Recent research has significantly focused on machine learning for IDS tasks, with most studies focusing on attack detection. In this context of IoT botnets, the Research Question (**RQ1**) explores the benefits of feature selection for identifying botnet attack systems throughout the botnet lifecycle, using various classification types studied. Publication I contextualizes **RQ1** within the process of feature selection for botnet detection, emphasizing the importance of minimizing feature sets for machine learning tasks in IDS. The study addressed six different binary and multi-class classification problems based on the stages of the botnet life cycle. Specifically, it utilized filter and wrapper methods in conjunction with selected machine learning algorithms to identify optimal feature sets for each classification problem. The experimental of this study showed that it is possible to achieve very high detection rates using a limited number of features. Some wrapper methods guarantee an optimal feature set regardless of how the problem is formulated. However, filter methods do not always reach this level of optimal performance. The feature selection methods found that channel-based features were particularly effective for detection during the post-attack, C&C stages, while host-based features played a more significant role in identifying attacks originating from bots.

Additionally, DDoS and DoS attacks in IoMT networks could disrupt critical medical equipment, endangering patient lives. Publication II further explored RQ1, emphasizing the importance of feature selection for IoMT attack detection within the IDS task. This Publication II examined filter-based feature selection methods used for both binary and multi-class classification studies on two IoMT network flow traffic datasets. Using the XGBoost model evaluated on filter methods, 15 features selected through information gain feature selection achieved optimal performance for both binary and multi-class classifi-

cation settings across two well-developed healthcare IoMT network datasets.

Many ML models, despite their black-box nature, have demonstrated impressive results in cybersecurity. However, their lack of transparency raises concerns among security experts, who often find it difficult to trust the outputs of ML models since the decision-making processes are not clear to understand. To address this issue, explainable AI emerges as a solution aimed at improving the transparency of ML models and making ML model outputs more interpretable. Although post hoc explainability methods in XAI, which are model-agnostic, have gained popularity in research due to their wide applicability, there have been rising concerns regarding the quality of these methods as these xai methods often rely on additional tools like linear models and game theory, which may introduce errors. Consequently, many current XAI techniques may provide unreliable explanations in real-world scenarios due to insufficient qualitative evaluations. To overcome these challenges, Publications III, IV, V, and VI address **RQ2** by emphasizing the importance of considering the quality of explainability by using quantitative evaluations alongside the detection performance of models, particularly for IDS tasks, including IoT botnet detection, IoMT attack detection, and NIDS alert classification.

Publication III highlights the importance of XAI methods in enhancing the interpretability and transparency of the black nature of ML models for detecting IoT botnets. The study employed three datasets for both binary and multi-class classification of IoT botnets, using Sequential Backward Selection (SBS) as the feature selection technique. Local explanations were generated using LIME and SHAP, which help in understanding the behaviour of the botnet detection classification models. The effectiveness of the XAI methods was evaluated using metrics like faithfulness, monotonicity, complexity, and sensitivity. Results of Publication III showed that XAI techniques significantly improve ML-based IoT botnet detection models, with XGBoost achieving optimal performance of explanations. Notably, SHAP outperformed LIME across all metrics. In Publication IV also, Post-hoc XAI were evaluated for enhancing transparency in a deep neural network (DNN) model used to detect multi-class IoT botnet attacks. Seven post-hoc explanation techniques were analyzed to understand the decision-making process of the DNN model, and these explanations were evaluated using the same four specific explainability metrics utilized in Publication III. Deep-lift outperformed other explainers, exhibiting high faithfulness, consistency, and lower complexity and sensitivity.

NIDS alert classification approach using LSTM (long short-term memory) proposed in Publication V is able to highlight NIDS alert data points of high importance. In this Publication, a real-world NIDS alert dataset was used from SoC at TalTech (Tallinn University of Technology) in Estonia to develop an LSTM model for NIDS alert prioritization. To explain predictions of LSTM model decision-making, four XAI methods were employed: LIME, SHAP, Integrated Gradients, and DeepLIFT and these methods were evaluated based on faithfulness, complexity, robustness, and reliability. The findings of this publication demonstrate that DeepLIFT outperforms the other xai methods in terms of high faithfulness, low complexity, and strong reliability. Collaborating with SoC analysts, key features for effective NIDS alert classification were identified, revealing a strong alignment between the analysts' insights and those obtained from the XAI methods.

In Publication VII, an active learning framework was proposed to address **RQ3**, which integrates explainable artificial intelligence (XAI) methods for detecting IoT botnet attacks in SoCs during the labeling process of the active learning loop. Specifically, this framework incorporates post-hoc explainability methods, namely LIME and SHAP, to enhance transparency in the labeling procedures of active learning loop.

In an IDS running federated learning (FL) settings, security experts still need to un-

derstand how server-side models make decisions about intrusions on client nodes. Using post-hoc explainability methods in FL poses challenges in balancing privacy and model transparency, as explainers like LIME or SHAP require access to the complete training dataset and model parameters, which can compromise the privacy of clients in FL. To address this issue, Publications VIII, IX, and X investigate Research Question RQ4, which focuses on achieving Trustworthy AI in a black-box model designed to enhance security and reliability, particularly in Privacy-Preserving IDS systems.

Publication VIII proposed an approach for aggregating client-based SHAP explanations, providing a solution for generating explanations for server models in FL settings while maintaining the data privacy of participating clients in an IoT network. DNN model presented in Publication VIII achieved over 99% accuracy in horizontal federated learning settings when detecting botnets in the IoT network. To enhance the transparency of the DNN model in FL- settings, the approach aggregates explanations from participating clients to generate global DNN model explanations without requiring direct access to their training data, and it analyzes the importance of features that contribute to IoT botnet detection using the generated SHAP explanations. The aggregation of client-based SHAP explanations was evaluated by sending the data to the server. SHAP explanations accessed by the server were found to be approximately equivalent to the aggregated client-based SHAP explanations.

In Publication IX, a similar approach of aggregating SHAP explanations was used to explain the server model using SHAP explainer. To improve the security of SHAP explanation aggregation, a secure multiparty computation protocol was proposed. Three classification tasks were investigated in Publication IX: 1) Binary classification, 2) botnet-type detection, and 3) attack-type detection. LSTM model achieved 99% accuracy for both binary and botnet detection, while it achieved more than 94% accuracy for attack-type detection. To generate explanations for each participating client in FL-settings for Publication 8, post-hoc local explanations were created using Integrated Gradients, LIME, and SHAP, which were then evaluated. It was found that SHAP provided better post-hoc local explanations compared to LIME and Integrated Gradients.

The main disadvantage of the approaches in Publications VIII and IX, which proposed the aggregation of client-based SHAP explanations for the server model, is that computing the SHAP value for each participating client in a federated learning setting is computationally expensive. This issue becomes more noticeable as the number of participating clients in federated learning increases. To address this problem, Publication X proposed a synthetic data-driven explanation approach to achieve transparency of the server model in FL settings for IDS. In this framework, a deep neural network (DNN)-based IDS is developed within the FL setting, where each client retains its private data and independently trains a local model. Client-side explanations are generated using post-hoc XAI methods applied directly on the local data and trained models. However, for server-side explainability, without compromising client data privacy, we propose generating federated synthetic data. Specifically, various generative adversarial network (GAN) variants are trained under the FL setup to generate synthetic data that approximates the client data distribution. These synthetic samples are then used to explain the global server model using post-hoc XAI methods. Our experimental results demonstrate that the SHAP explanations produced using synthetic data generated from the federated Wasserstein Conditional GAN (WC-GAN) closely match those derived from real client data. Evaluation further confirms that server-side SHAP explanations based on synthetic data are approximately equivalent to explanations that would have been obtained using actual client data, thereby validating the effectiveness of the synthetic data-driven explainability approach.

11.1 Future Work

The research conducted in this doctoral dissertation investigated explainable and privacy-preserving intrusion detection systems for centralized and decentralized (federated) environments while addressing intricate challenges in dynamic and evolving data scenarios. Most existing research has treated explainability and privacy as independent elements within federated intrusion detection systems. My future objective involves creating a unified explainability framework for decentralized intrusion detection which will deliver transparency throughout both client and server operations in federated learning.

The recent developments in large language models (LLMs) provide promising ways to improve explainability in ML-based IDS. The traditional post-hoc explanation methods are often limited to feature importance scores, which may not always be intuitive for security analysts, especially when dealing with complex network traffic features. In contrast, LLMs have the potential to generate natural language explanations that translate complex model decisions into human-understandable narratives. Future work will explore the integration of LLM-based explanation frameworks that can Automatically generate narrative explanations for IDS decisions by interpreting feature contributions, attack types, and contextual network information.

List of Figures

1 Explainable ai methods Taxonomy 13
2 Publication. I - Computational Time and Performance of feature sets using Filter and Wrapper feature selection for Binary classification models in N-BaIoT dataset 23
3 Publication. I - Contribution of Network Category-wise features for Binary, 3-class, and 9-class classification in the N-BaIoT dataset..... 24
4 Publication. I - F1-score of models, Computational time and performance of feature sets using filter and wrapper feature selection for binary classification models in the MedBIoT dataset from Publication 25
5 Publication. I - Contribution of Network Category-wise features for Binary, 3-class, and 4-class classification in the MedBIoT dataset from Publication . 26
6 Publication. II - Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for Binary Classification..... 28
7 Publication. II - Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for 6-class Classification. 28
8 Publication. II - Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for 19-class Classification. 28
9 Publication. II - Comparison of algorithms performance using Feature selection methods over IoMT-TrafficData dataset for Binary Classification. 29
10 Publication. II - Comparison of algorithms performance using Feature selection methods over IoMT-TrafficData dataset for Multi-class classification (9 classes)..... 29
11 Quantitative Evaluation of Explainability for IoT botnet detection Framework [55]..... 31
12 LIME Local explanations of Malware instance of N-BaIoT dataset for Binary Classifiers [55] 34
13 SHAP Local explanations of Malware instance of N-BaIoT dataset for Binary Classifiers [55] 35
14 Training and Testing Accuracies and Loss Values of DNN Model for IoT Botnet Attack Detection 40
15 Loss and Accuracy from Best LSTM Performance Model for NIDS alert classification 42
16 Explanations for an important NIDS alert data point using an LSTM model.. 43
17 Quality of Explainable AI evaluation metrics distribution for LSTM model based NIDS alerts classification..... 46
18 SHAP global explanation for LSTM model in NIDS alert classification 48
19 Transformer Architecture for IoMT attack detection 50
20 Training loss and accuracy for Binary classification 52
21 Training loss and accuracy of Transformer model for Multiclass classification 52
22 Transformer Performance of Classification Report for Binary and Multiclass Classification IoMT attack detection 53
23 LIME Local explanations for malware instances in Binary and Multi-class classification settings..... 53
24 SHAP Local explanations for multi-class classification instances..... 54
25 Distribution of Quality Evaluation Metrics for LIME and SHAP in Multi-class classification 55
26 SHAP Global explanations for Transformer Model..... 56

27 Explainable Active Learning (XAL) Framework for IoT Botnet Attack Detection 59
28 Feature Sets Comparisons with metrics for for Active Learning-Based IoT Botnet Detection. 60
29 Uncertainty sampling: classification uncertainty results. 61
30 Query by committee results 61
31 LIME & SHAP Local Explanations for Correctly Classified Instance (ACK Attack) 62
32 LIME & SHAP Local Explanations for Misclassified Instances..... 63
33 XAI metrics results for Uncertainty sampling strategies..... 64
34 XAI metrics results for QBC strategies..... 64
35 FedXAI (Federated Learning of Explainable AI) paradigm for Client-Server Intrusion Detection Model Explanations in IoT Network Devices..... 67
36 Evaluation of DNN Model Performance on Server side 69
37 Evaluation of DNN model performance on client's Side..... 70
38 Heatmap for Difference Between aggregation of client's models Explanations and server model explanations..... 71
39 SHAP value feature importance from both Client's side models and Server model 71
40 Performance evaluation of FedAvg LSTM model on the server side across different detection tasks: (a) Malware Detection (Binary), (b) Botnet-type Detection, and (c) Attack-type Detection..... 74
41 Device-specific client-side evaluation of LSTM model performance for Binary classification 75
42 Device-specific client-side evaluation of LSTM model performance for Botnet-type Detection..... 76
43 Device-specific client-side evaluation of LSTM model performance for Attack-type Detection..... 77
44 Heatmap for the difference between the secure aggregation of client models' explanations and server model explanations for binary classification ... 79
45 Heatmap for Difference Between Secure aggregation of client's models Explanations and server model explanations for Botnet-type detection 80
46 Heatmap for Difference Between Secure Aggregation of client-side models Explanations and server-side explanations for Attack-type Detection..... 80
47 Mean SHAP values of Secure Aggregation Matrix of Client-Side and Server-Side Models for Binary-type classification 81
48 Mean SHAP values of Secure Aggregation Matrix of Client-Side and Server-Side Models for Botnet-type detection 81
49 Mean SHAP values of Secure Aggregation Matrix of Client-Side and Server-Side Models for Attack-type detection 82
50 Global explanations for server model feature importance and participated clients' IoT device models feature importance in Binary classification type . 84
51 Global Explanations for Server Model Feature Importance and Participated Clients' IoT Devices Feature Importance for Botnet-type Detection..... 85
52 Global explanations for Server Model Feature importance and participated clients IoT devices feature importance for Attack-type Detection 86
53 Synthetic Data-Driven Explainability Framework for Explaining Blackbox Classifier in FL settings 88
54 Global DNN model performance using FedAvg aggregation across communication rounds for NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets..... 91

55 Architecture of Basic GAN and CGAN 92
56 Performance of federated generative models across communication rounds for the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets.... 97
57 2-D visualization using PCA for real and synthetic data generated by FL-WCGAN-GP for all four datasets 98
58 2-D visualization using t-SNE for real and synthetic data generated by FL-WCGAN-GP for all four datasets 98
59 Feature-pairwise Absolute differences in Pearson correlations between real and synthetic data generated by the FL-WCGAN-GP model for the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets 99
60 Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on NSL-KDD dataset..... 102
61 Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the UNSW-NB15 dataset..... 102
62 Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the CIC-IoT2023 dataset 103
63 Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the CIC-IoMT dataset 103
64 SHAP local explanations for the global DNN model on an attack instance from the NSL-KDD dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model..... 104
65 LIME local explanations for the global DNN model on an attack instance from the NSL-KDD dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model..... 105

List of Tables

1 Mapping of dissertation chapters to research questions, corresponding publications, and key contributions 10
2 Classification problems addressed for IoT botnet detection using feature selection [57] 21
3 Publication. I - Filter Method Feature Sets for the N-BaIoT and MedBIoT Dataset 22
4 Publication. I - Wrapper Methods Feature Sets for N-BaIoT and MedBIoT .. 22
5 Publication. I - F1 scores for binary classification models using feature subsets (represented in Table 3 and 4) of feature selection algorithms in the N-BaIoT dataset 23
6 Actual Data Points of N-BaIoT Dataset Across Three Network Categories[55] 33
7 Results of Evaluating the quality of LIME&SHAP using Faithfulness (μ_f), Monotonicity (μ_m), Complexity (μ_c), Sensitivity(μ_s) for IoT malware detection binary classification models over three datasets: N-BaIoT, MedBIoT, BoT-IoT. 37
8 Results of Evaluating the quality of LIME&SHAP Botnet malware type (multiclass) on two datasets: N-BaIoT, MedBIoT 38
9 DNN Model Hyperparameters (Random Search) 39
10 comparison results of the evaluation of XAI metrics 40
11 Key Features Identified by Taltech SOC Analyst for Determining NIDS Alert Significance 42
12 Evaluation Results of Explainable AI Methods: Mean (μ) and Standard Deviation (σ) Values. 45
13 Statistical Comparison of Explainers Across Multiple Metrics (p-values) 47
14 Transformer Model Comparison with KNN and DT 53
15 Quality Evaluation of local explanations results 55
16 Top-5 Selected Features for Active Learning-Based IoT Botnet Detection Using XGBoost 60
17 Number of Botnet malware & benign samples for each device over N-BaIoT dataset [92] 68
18 Results of evaluating Quality of LIME, IG & SHAP explanations using High Faithfulness (μ_f), Low Complexity (μ_c), Maximum Sensitivity (μ_s) & Monotonicity (μ_m) for participated client-side models in FL for three classification types. 83
19 Performance of global FL-DNN model on Real Data and Synthetic data generated by FL-WCGAN-GP generative model 100
20 Evaluation of SHAP and LIME local explanations on synthetic data for the global DNN model on server using Faithfulness (μ_f) and Max Sensitivity (μ_s) metrics across four datasets. 106

References

[1] A. Adadi and M. Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[2] C. C. Aggarwal. *Data Mining: The Textbook*. Springer, Cham, 1 edition, 2015. Hardcover ISBN 978-3-319-14141-1; eBook ISBN 978-3-319-14142-8.

[3] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu. Federated learning for intrusion detection system: Concepts, challenges and future directions. *Computer Communications*, 195:346–361, 2022.

[4] U. Ahmed, Z. Jiangbin, S. Khan, and M. T. Sadiq. Consensus hybrid ensemble machine learning for intrusion detection with explainable ai. *Journal of Network and Computer Applications*, 235:104091, 2025.

[5] S. I. Ahsan, P. Legg, and S. I. Alam. An explainable ensemble-based intrusion detection system for software-defined vehicle ad-hoc networks. *Cyber Security and Applications*, 3:100090, 2025.

[6] B. A. Alahmadi, L. Axon, and I. Martinovic. 99% false positives: a qualitative study of {SOC} analysts' perspectives on security alarms. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 2783–2800, 2022.

[7] M. M. Alani and E. Damiani. Xrecon: An explainbale iot reconnaissance attack detection system based on ensemble learning. *Sensors*, 23(11):5298, 2023.

[8] E. Albin and N. C. Rowe. A realistic experimental comparison of the suricata and snort intrusion-detection systems. In *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, pages 122–127. IEEE, 2012.

[9] S. R. Alotaibi, H. K. Alkahtani, M. Aljebreen, A. Alshuhail, M. K. Saeed, S. A. Ebad, W. S. Almukadi, and M. Alotaibi. Explainable artificial intelligence in web phishing classification on secure iot with cloud-based cyber-physical systems. *Alexandria Engineering Journal*, 110:490–505, 2025.

[10] F. Alsubaei, A. Abuhussein, V. Shandilya, and S. Shiva. Iomt-saf: Internet of medical things security assessment framework. *Internet of Things*, 8:100123, 2019.

[11] J. Areia, I. Bispo, L. Santos, and R. L. d. C. Costa. Iomt-trafficdata: Dataset and tools for benchmarking intrusion detection in internet of medical things. *IEEE Access*, 2024.

[12] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and M. Guizani. A survey on iot intrusion detection: Federated learning, game theory, social psychology, and explainable ai as future directions. *IEEE Internet of Things Journal*, 10(5):4059–4092, 2022.

[13] M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[14] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[15] L. Arras, A. Osman, and W. Samek. Clevr-xai: A benchmark dataset for the ground truth evaluation of neural network explanations. *Information Fusion*, 81:14–40, 2022.

[16] O. Arreche, T. Guntur, and M. Abdallah. Xai-ids: Toward proposing an explainable artificial intelligence framework for enhancing network intrusion detection systems. *Applied Sciences*, 14(10):4170, 2024.

[17] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[18] J. L. C. Bárcena, M. Daole, P. Ducange, F. Marcelloni, A. Renda, F. Ruffini, and A. Schiavo. Fed-xai: Federated learning of explainable artificial intelligence models. In *XAI. it@ AI* IA*, pages 104–117. Udine, 2022.

[19] U. Bhatt, A. Weller, and J. M. Moura. Evaluating and aggregating feature-based model explanations. *arXiv preprint arXiv:2005.00631*, 2020.

[20] S. A. Birahim, A. Paul, F. Rahman, Y. Islam, T. Roy, M. A. Hasan, F. Haque, and M. E. Chowdhury. Intrusion detection for wireless sensor network using particle swarm optimization based explainable ensemble machine learning approach. *IEEE Access*, 2025.

[21] F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, and S. Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, 37(5):1719–1778, 2023.

[22] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1175–1191, 2017.

[23] A. L. Buczak and E. Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.

[24] E. M. Campos, P. F. Saura, A. González-Vidal, J. L. Hernández-Ramos, J. B. Bernabé, G. Baldini, and A. Skarmeta. Evaluating federated learning for intrusion detection in internet of things: Review and challenges. *Computer Networks*, 203:108661, 2022.

[25] N. Capuano, G. Fenza, V. Loia, and C. Stanzione. Explainable artificial intelligence in cybersecurity: A survey. *Ieee Access*, 10:93575–93600, 2022.

[26] A. Chacko and T. Hayajneh. Security and privacy issues with iot in healthcare. *EAI Endorsed Transactions on Pervasive Health and Technology*, 4(14), 2018.

[27] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine learning*, 15:201–221, 1994.

[28] P. M. Corea, Y. Liu, J. Wang, S. Niu, and H. Song. Explainable ai for comparative analysis of intrusion detection models. In *2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pages 585–590. IEEE, 2024.

[29] L. Coroama and A. Groza. Evaluation metrics in explainable artificial intelligence (xai). In *International conference on advanced research in technologies, information, innovation and sustainability*, pages 401–413. Springer, 2022.

[30] A. Culotta and A. McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005.

[31] S. Dadkhah, E. C. P. Neto, R. Ferreira, R. C. Molokwu, S. Sadeghi, and A. A. Ghorbani. Ciciomt2024: A benchmark dataset for multi-protocol security assessment in iomt. *Internet of Things*, 28:101351, 2024.

[32] A. Das and P. Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[33] S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215, 2008.

[34] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in neural information processing systems*, 31, 2018.

[35] T. Dias, N. Oliveira, N. Sousa, I. Praça, and O. Sousa. A hybrid approach for an interpretable and explainable intrusion detection system. In *International Conference on Intelligent Systems Design and Applications*, pages 1035–1045. Springer, 2021.

[36] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada. Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201, 2009.

[37] European Union. Art. 22 gdpr – automated individual decision-making, including profiling - general data protection regulation (gdpr), 2025. [Online; accessed 2025-05-10].

[38] J. Fiosina. Explainable federated learning for taxi travel time prediction. In *VEHITS*, pages 670–677, 2021.

[39] K. Gade, S. Geyik, K. Kenthapadi, V. Mithal, and A. Taly. Explainable ai in industry: Practical challenges and lessons learned. In *Companion Proceedings of the Web Conference 2020*, pages 303–304, 2020.

[40] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[41] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm. Medbiot: Generation of an iot botnet dataset in a medium-sized iot network. In *ICISSP*, pages 207–218, 2020.

[42] K. S. Gurumoorthy, A. Dhurandhar, G. Cecchi, and C. Aggarwal. Efficient data representation by selecting prototypes with importance weights. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 260–269. IEEE, 2019.

[43] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[44] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46:389–422, 2002.

[45] D. Han, Z. Wang, W. Chen, K. Wang, R. Yu, S. Wang, H. Zhang, Z. Wang, M. Jin, J. Yang, et al. Anomaly detection in the open world: Normality shift detection, explanation, and adaptation. In *NDSS*, 2023.

[46] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek. Explainable ai methods-a brief overview. In *International workshop on extending explainable AI beyond deep models and classifiers*, pages 13–38. Springer, 2020.

[47] T. Iggena, E. Bin Ilyas, M. Fischer, R. Tönjes, T. Elsaleh, R. Rezvani, N. Pourshahrokhi, S. Bischof, A. Fernbach, J. X. Parreira, et al. Iotcrawler: Challenges and solutions for searching the internet of things. *Sensors*, 21(5):1559, 2021.

[48] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE transactions on software engineering*, 21(3):181–199, 2002.

[49] S. R. Islam, W. Eberle, S. K. Ghafoor, A. Siraj, and M. Rogers. Domain knowledge aided explainable artificial intelligence for intrusion detection and response. *arXiv preprint arXiv:1911.09853*, 2019.

[50] IXIA (Keysight Technologies). Ixia. `https://support.ixiacom.com/sites/default/`, 2022. (Accessed on 12/21/2022).

[51] Y. Javed, M. A. Khayat, A. A. Elghariani, and A. Ghafoor. Prism: a hierarchical intrusion detection architecture for large-scale cyber networks. *IEEE Transactions on Dependable and Secure Computing*, 20(6):5070–5086, 2023.

[52] D. Javeed, T. Gao, P. Kumar, and A. Jolfaei. An explainable and resilient intrusion detection system for industry 5.0. *IEEE Transactions on Consumer Electronics*, 70(1):1342–1350, 2023.

[53] S. Jesus, C. Belém, V. Balayan, J. Bento, P. Saleiro, P. Bizarro, and J. Gama. How can i choose an explainer? an application-grounded evaluation of post-hoc explanations. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 805–815, 2021.

[54] R. Kalakoti, H. Bahsi, and S. Nõmm. Explainable federated learning for botnet detection in iot networks. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 01–08. IEEE, 2024.

[55] R. Kalakoti, H. Bahsi, and S. Nõmm. Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection. *IEEE Internet of Things Journal*, 11(10):18237–18254, 2024.

[56] R. Kalakoti, H. Bahsi, and S. Nõmm. Synthetic data-driven explainability for federated learning-based intrusion detection system. In *IEEE Internet of Things Journal*. IEEE, 2025.

[57] R. Kalakoti, S. Nõmm, and H. Bahsi. In-depth feature selection for the statistical machine learning-based botnet detection in iot networks. *IEEE Access*, 10:94518–94535, 2022.

[58] R. Kalakoti, S. Nõmm, and H. Bahsi. Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai). In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 595–601. IEEE, 2023.

[59] R. Kalakoti, S. Nõmm, and H. Bahsi. Enhancing iot botnet attack detection in socs with an explainable active learning framework. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 265–272. IEEE, 2024.

[60] R. Kalakoti, S. Nõmm, and H. Bahsi. Explainable transformer-based intrusion detection in internet of medical things (iomt) networks. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 1164–1169. IEEE, 2024.

[61] R. Kalakoti, S. Nõmm, and H. Bahsi. Federated learning of explainable ai (fedxai) for deep learning-based intrusion detection in iot networks. *Computer Networks*, page 111479, 2025.

[62] R. Kalakoti, R. Vaarandi, H. Bahşi, and S. Nõmm. Evaluating explainable ai for deep learning-based network intrusion detection system alert classification. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, pages 47–58. INSTICC, SciTePress, 2025.

[63] J. T. Kent. Information gain and a general measure of correlation. *Biometrika*, 70(1):163–173, 1983.

[64] M. Keshk, N. KoronIoTis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya. An explainable deep learning-enabled intrusion detection framework in iot networks. *Information Sciences*, 639:119000, 2023.

[65] I. Kök, F. Y. Okay, Ö. Muyanlı, and S. Özdemir. Explainable artificial intelligence (xai) for internet of things: a survey. *IEEE Internet of Things Journal*, 10(16):14764–14779, 2023.

[66] N. KoronIoTis, N. Moustafa, E. Sitnikova, and B. Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796, 2019.

[67] M. Kudo and J. Sklansky. Comparison of algorithms that select features for pattern classifiers. *Pattern recognition*, 33(1):25–41, 2000.

[68] P. Kumar, G. P. Gupta, and R. Tripathi. An ensemble learning and fog-cloud architecture-driven cyber-attack detection framework for iomt networks. *Computer Communications*, 166:110–124, 2021.

[69] P. P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, and S. G. Teo. Detection and classification of botnet traffic using deep learning with model explanation. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[70] A. Kuppa and N.-A. Le-Khac. Black box attacks on explainable artificial intelligence (xai) methods in cyber security. In *2020 International Joint Conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2020.

[71] M. Landauer, F. Skopik, M. Frank, W. Hotwagner, M. Wurzenberger, and A. Rauber. Maintainable log datasets for evaluation of intrusion detection systems. *IEEE Transactions on Dependable and Secure Computing*, 20(4):3466–3482, 2022.

[72] T.-T.-H. Le, H. Kim, H. Kang, and H. Kim. Classification and explanation for intrusion detection system based on ensemble trees and shap method. *Sensors*, 22(3):1154, 2022.

[73] D. D. Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. *ACM SIGIR Forum*, 29(2):13–19, 1995.

[74] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu. Feature selection: A data perspective. *ACM computing surveys (CSUR)*, 50(6):1–45, 2017.

[75] S. Li, Y. Cao, S. Liu, Y. Lai, Y. Zhu, and N. Ahmad. Hda-ids: A hybrid dos attacks intrusion detection system for iot by using semi-supervised cl-gan. *Expert Systems with Applications*, 238:122198, 2024.

[76] W. Li, S. Tug, W. Meng, and Y. Wang. Designing collaborative blockchained signature-based intrusion detection in iot environments. *Future Generation Computer Systems*, 96:481–489, 2019.

[77] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

[78] H. Liu and B. Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, 9(20):4396, 2019.

[79] H. Liu, C. Zhong, A. Alnusair, and S. R. Islam. Faixid: A framework for enhancing ai explainability of intrusion detection results using data cleaning techniques. *Journal of network and systems management*, 29(4):40, 2021.

[80] L. Longo, R. Goebel, F. Lecue, P. Kieseberg, and A. Holzinger. Explainable artificial intelligence: Concepts, applications, research challenges and visions. In *International cross-domain conference for machine learning and knowledge extraction*, pages 1–16. Springer, 2020.

[81] S. M. Lundberg, G. G. Erion, and S.-I. Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[82] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[83] R. Luss, P.-Y. Chen, A. Dhurandhar, P. Sattigeri, K. Shanmugam, and C.-C. Tu. Generating contrastive explanations with monotonic attribute functions. *arXiv preprint arXiv:1905.12698*, 3, 2019.

[84] B. Mahbooba, M. Timilsina, R. Sahal, and M. Serrano. Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model. *Complexity*, 2021(1):6634811, 2021.

[85] M. M. Mahmoud, Y. O. Youssef, and A. A. Abdel-Hamid. Xi2s-ids: An explainable intelligent 2-stage intrusion detection system. *Future Internet*, 17(1):25, 2025.

[86] S. Mane and D. Rao. Explaining network intrusion detection system using explainable ai framework. *arXiv preprint arXiv:2103.07110*, 2021.

[87] D. L. Marino, C. S. Wickramasinghe, C. Rieger, and M. Manic. Self-supervised and interpretable anomaly detection using network transformers. *arXiv preprint arXiv:2202.12997*, 2022.

[88] M. Masdari and H. Khezri. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Applied Soft Computing*, 92:106301, 2020.

[89] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, and M. Tesconi. Rtbust: Exploiting temporal patterns for botnet detection on twitter. In *Proceedings of the 10th ACM conference on web science*, pages 183–192, 2019.

[90] C. D. McDermott, F. Majdani, and A. V. Petrovski. Botnet detection in the internet of things using deep learning approaches. In *2018 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2018.

[91] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[92] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[93] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[94] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018.

[95] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli. A detailed investigation and analysis of using machine learning techniques for intrusion detection. *IEEE communications surveys & tutorials*, 21(1):686–728, 2018.

[96] S. Mittal, M. Wazid, D. P. Singh, A. K. Das, and M. S. Hossain. A deep learning ensemble approach for malware detection in internet of things utilizing explainable artificial intelligence. *Engineering Applications of Artificial Intelligence*, 139:109560, 2025.

[97] C. Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[98] N. Moustafa. New generations of internet of things datasets for cybersecurity applications based machine learning: Ton_iot datasets. In *Proceedings of the eResearch Australasia Conference, Brisbane, Australia*, pages 21–25, 2019.

[99] N. Moustafa and J. Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[100] H. Nandanwar and R. Katarya. Securing industry 5.0: An explainable deep learning model for intrusion detection in cyber-physical systems. *Computers and Electrical Engineering*, 123:110161, 2025.

[101] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani. Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. *IEEE Communications Surveys & Tutorials*, 21(3):2702–2733, 2019.

[102] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani. Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors*, 23(13):5941, 2023.

[103] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale. Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities. *IEEE Access*, 10:112392–112415, 2022.

[104] C. I. Nwakanma, L. A. C. Ahakonye, J. N. Njoku, J. C. Odirichukwu, S. A. Okolie, C. Uzondu, C. C. Ndubuisi Nweke, and D.-S. Kim. Explainable artificial intelligence (xai) for intrusion detection and mitigation in intelligent connected vehicles: A review. *Applied Sciences*, 13(3):1252, 2023.

[105] A. Ojugo, A. Eboka, O. Okonta, R. Yoro, and F. Aghware. Genetic algorithm rule-based intrusion detection system (gaids). *Journal of Emerging Trends in Computing and Information Sciences*, 3(8):1182–1194, 2012.

[106] A. Oseni, N. Moustafa, G. Creech, N. Sohrabi, A. Strelzoff, Z. Tari, and I. Linkov. An explainable deep learning framework for resilient intrusion detection in iot-enabled transportation networks. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):1000–1014, 2022.

[107] Y. Otoum and A. Nayak. As-ids: Anomaly and signature based ids for the internet of things. *Journal of Network and Systems Management*, 29(3):23, 2021.

[108] M. Papaioannou, M. Karageorgou, G. Mantas, V. Sucasas, I. Essop, J. Rodriguez, and D. Lymberopoulos. A survey on security threats and countermeasures in internet of medical things (iomt). *Transactions on Emerging Telecommunications Technologies*, 33(6):e4049, 2022.

[109] A.-S. K. Pathan. *The state of the art in intrusion prevention and detection*, volume 44. CRC press Hoboken, NJ, USA, 2014.

[110] A. Petrosyan. Global cybercrime estimated cost 2029| statista. [Online; accessed 2025-05-02].

[111] D. Rao and S. Mane. Zero-shot learning approach to adaptive cybersecurity using explainable ai. *arXiv preprint arXiv:2106.14647*, 2021.

[112] S. Razdan and S. Sharma. Internet of medical things (iomt): Overview, emerging technologies, and case studies. *IETE technical review*, 39(4):775–788, 2022.

[113] M. U. Rehman, R. Kalakoti, and H. Bahşi. Comprehensive feature selection for machine learning-based intrusion detection in healthcare iomt networks. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 2: ICISSP*, pages 248–259. INSTICC, SciTePress, 2025.

[114] M. U. Rehman, R. Kalakoti, and H. Bahşi. Exploring the impact of feature selection on non-stationary intrusion detection models in iot networks. In *Accepted for 22nd Annual International Conference on Privacy, Security, and Trust (PST2025)*, 2025.

[115] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[116] E. Roponena, J. Kampars, J. Grabis, and A. Gailitis. Towards a human-in-the-loop intelligent intrusion detection system. In *Doctoral Consortium/Forum@ DB&IS*, pages 71–81, 2022.

[117] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj. Anomaly-based intrusion detection system for iot networks through deep learning model. *Computers and Electrical Engineering*, 99:107810, 2022.

[118] S. I. Sabev. Integrated approach to cyber defence: Human in the loop. technical evaluation report. *Information & Security: An International Journal*, 44:76–92, 2020.

[119] A. Samed and S. Sagiroglu. Explainable artificial intelligence models in intrusion detection systems. *Engineering Applications of Artificial Intelligence*, 144:110145, 2025.

[120] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.

[121] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, 2019.

[122] B. Settles. Active learning literature survey. Technical Report 1648, University of Wisconsin–Madison, Department of Computer Sciences, Madison, WI, 2009. Accessed 2025-08-01.

[123] L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, Volume II*, number 28 in Annals of Mathematics Studies, pages 307–317. Princeton University Press, Princeton, NJ, 1953.

[124] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1(2018):108–116, 2018.

[125] B. Sharma, L. Sharma, C. Lal, and S. Roy. Explainable artificial intelligence for intrusion detection in iot networks: A deep learning based approach. *Expert Systems with Applications*, 238:121751, 2024.

[126] D. K. Sharma, J. Mishra, A. Singh, R. Govil, G. Srivastava, and J. C.-W. Lin. Explainable artificial intelligence for cybersecurity. *Computers and Electrical Engineering*, 103:108356, 2022.

[127] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, 2(1):41–50, 2018.

[128] S. Shoukat, T. Gao, D. Javeed, M. S. Saeed, and M. Adil. Trust my ids: An explainable ai integrated deep learning-based transparent threat detection system for industrial networks. *Computers & Security*, 149:104191, 2025.

[129] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMlR, 2017.

[130] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

[131] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[132] Statista. Statista — the statistics portal for market data, market research and market studies. `https://www.statista.com/`, 2025. [Online; accessed 2025-05-02].

[133] Stratosphere IPS. Datasets overview — stratosphere ips. `https://www.stratosphereips.org/`, 2022. (Accessed on 12/21/2022).

[134] E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41:647–665, 2014.

[135] Y. Sun, F. P.-W. Lo, and B. Lo. Security and privacy for the internet of medical things enabled healthcare systems: A survey. *IEEE Access*, 7:183339–183355, 2019.

[136] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[137] H. Suryotrisongko, Y. Musashi, A. Tsuneda, and K. Sugitani. Robust botnet dga detection: Blending xai and osint for cyber threat intelligence sharing. *IEEE Access*, 10:34613–34624, 2022.

[138] M. Szczepański, M. Choraś, M. Pawlicki, and R. Kozik. Achieving explainability of intrusion detection system by hybrid oracle-explainer approach. In *2020 International Joint Conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2020.

[139] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[140] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

[141] E. Tjoa and C. Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE transactions on neural networks and learning systems*, 32(11):4793–4813, 2020.

[142] A. Torkaman, G. Javadzadeh, and M. Bahrololum. A hybrid intelligent hids model using two-layer genetic algorithm and neural network. In *The 5th Conference on Information and Knowledge Technology*, pages 92–96. IEEE, 2013.

[143] R. Vaarandi and A. Guerra-Manzanares. Stream clustering guided supervised learning for classifying nids alerts. *Future Generation Computer Systems*, 155:231–244, 2024.

[144] R. Vaarandi and S. Mäses. How to build a soc on a budget. In *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 171–177. IEEE, 2022.

[145] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[146] L. Vigano and D. Magazzeni. Explainable security. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 293–300. IEEE, 2020.

[147] C. Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2008.

[148] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman. Deep learning approach for intelligent intrusion detection system. *IEEE access*, 7:41525–41550, 2019.

[149] S. Wali, Y. A. Farrukh, and I. Khan. Explainable ai and random forest based reliable intrusion detection system. *Computers & Security*, page 104542, 2025.

[150] X. Wan, G. Xue, Y. Zhong, and Z. Wang. Separating prediction and explanation: An approach based on explainable artificial intelligence for analyzing network intrusion. *Journal of Network and Systems Management*, 33(1):16, 2025.

[151] M. Wang, K. Zheng, Y. Yang, and X. Wang. An explainable machine learning framework for intrusion detection systems. *IEEE Access*, 8:73127–73141, 2020.

[152] R. F. Woolson. Wilcoxon signed-rank test. *Encyclopedia of biostatistics*, 8, 2005.

[153] F. Yang and J. Xu. Privacy concerns in china's smart city campaign: The deficit of china's cybersecurity law. *Asia & the Pacific Policy Studies*, 5(3):533–543, 2018.

[154] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[155] Z. Zhang, H. Al Hamadi, E. Damiani, C. Y. Yeun, and F. Taher. Explainable artificial intelligence applications in cyber security: State-of-the-art in research. *IEEe Access*, 10:93104–93139, 2022.

[156] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[157] X. J. Zhu. Semi-supervised learning literature survey. Technical Report 1530, University of Wisconsin–Madison, Department of Computer Sciences, Madison, WI, 2005. Accessed 2025-08-01.

[158] M. Zolanvari. WUSTL-IIoT-2021, 2021. Dataset.

[159] M. Zolanvari, Z. Yang, K. Khan, R. Jain, and N. Meskin. Trust xai: Model-agnostic explanations for ai with a case study on iiot security. *IEEE internet of things journal*, 10(4):2967–2978, 2021.

Acknowledgements

This dissertation marks the culmination of a long and often solitary journey. While I often felt like a lonely sailor navigating through the challenges of research, I am deeply aware that I could not have reached this point without the support and encouragement of many people who stood beside me.

First and foremost, I thank God for His guidance, strength, and blessings along this journey. I would like to express my deepest gratitude to my parents, **Subba Rao Kalakoti and Jyothi Kalakoti**. Despite being illiterate and unable to read or write, they always understood the importance of education. They encouraged me wholeheartedly and gave me the freedom to pursue higher education. Their constant belief in me has been my greatest source of strength and motivation. I dedicate this thesis to them, with immense love and gratitude, for being my strongest pillars throughout this long journey. I am also deeply grateful to my brothers, **Suresh Kalakoti and Satish Kalakoti**, for their constant support. I would like to thank my dear friend **Hareesh Reddy Kolli** for his support, encouragement, and friendship throughout this journey.

I would like to thank my main supervisor, **Professor Sven Nõmm**, for his patience and willingness to understand my ideas and working style throughout the process of writing this thesis. He always gave me the freedom to explore my own path and was ready to help in person whenever I encountered any difficulties, not only with research but also with practical matters.

I am also grateful to my second supervisor, **Professor, Hayretdin Bahşi**, who played a pivotal role not only in helping me start my PhD journey but also in seeing it through to the end. Despite the time difference between Estonia and the USA, our many late-night discussions were instrumental in shaping this work. It has been a real pleasure working with him, and we have developed an excellent and collaborative working relationship.

I would also like to thank **Juhan-Peep Ernits**, who ensured Taltech's AI-Lab was up and running whenever compute nodes were down, making sure my experiments were completed on time for my publications.

I would like to thank the Estonian Science foundation ETAG for partially supporting this research through the PRG2100 project

I want to thank **Ülle Lõhmus** and **Kaisa Saarmann** for their support in handling my paperwork, contract details, and conference travel arrangements throughout my PhD journey.

I want to extend a special thanks to **Professor Dr Sanchari Deb**, a special person in my life and the most remarkable woman I have ever known. Her strength, commitment, and unwavering resilience have inspired me from the very beginning. She has been my greatest source of motivation and played a vital role in encouraging me to pursue and complete this PhD journey.

Abstract
Explainable Artificial Intelligence-Based Intrusion Detection Systems

Rapid advancements in network technologies and the increasing volume of data are increasing the complexity of cyber threats, particularly as sophisticated cyberattacks increasingly target the Internet of Things (IoT) and Internet of Medical Things (IoMT) networks. Although machine learning (ML) models have shown promising solutions for detecting malicious activities in such networks, the lack of interpretability and transparency of the models often limits their effectiveness and trustworthiness. This Ph.D. dissertation explores the development of effective, interpretable, transparent, and privacy-preserving machine learning-based intrusion detection systems (IDS) in both centralised and decentralised (federated learning) settings.

Most previous studies in the field of cybersecurity have focused primarily on attack detection in IoT and IoMT networks. In this paper, the role of feature selection is investigated to enhance IDS performance in these environments. For IoT botnet detection, six classification tasks were designed based on the botnet life cycle, where filter and wrapper methods identified minimal feature subsets achieving high detection rates. For IoMT attack detection, filter-based feature selection methods such as information gain, mutual information, and Fisher score were applied to evaluate ML models on two IoMT network traffic datasets. The study demonstrates that using Information Gain, a small subset of characteristics (3–4 for binary classification and 7–8 for multiclass classification) can achieve high detection rates for efficient IDS in IoMT networks.

The growing number of intrusions in networked systems has accelerated research into artificial intelligence (AI) for IDS. A key focus is understanding and explaining ML-based IDS to security analysts who manage these systems to protect their networks. This motivates the integration of explainable AI (XAI) methods to enhance the transparency and trustworthiness of IDS models. Although many studies have incorporated XAI to explain ML-based IDS models, they often lack a systematic evaluation of the quality of the generated explanations. In this paper, the importance of quantitatively evaluating the quality of the explanation is emphasised alongside the detection performance for the detection of IoT botnets. Several post hoc XAI methods were applied, including SHAP (SHapley Additive explanations), LIME(Local Interpretable Model-agnostic Explanations), DeepLIFT(Deep Learning Important FeaTures), Integrated Gradients, and other saliency-based explanation methods, to explain the IoT botnet detection models for both binary and multi-class classification tasks in the IDS context. The quality of these explanations was evaluated using quantitative metrics, including faithfulness, monotonicity, complexity, and sensitivity, which provided a more rigorous evaluation of the XAI methods in IDS. For statistical ML-based models, SHAP and LIME were applied to explain the behaviour of models trained on multiple datasets, with SHAP demonstrating superior performance across explanation metrics. For deep learning-based IDS, a dedicated DNN model was developed for multi-class IoT botnet detection, achieving high classification performance. Seven post hoc XAI methods were applied, including SHAP, LIME, DeepLIFT, IG, Gradient*Input, Feature Ablation, and Saliency, to generate local explanations. The quality of these explanations was rigorously evaluated using quantitative metrics such as faithfulness, monotonicity, complexity, and sensitivity, revealing that DeepLIFT consistently provided the most faithful and stable explanations among the methods evaluated.

In addition to botnet detection, this dissertation investigates the explainability of network intrusion detection system (NIDS) alert classification using real-world data from Tal-

Tech's Security Operations Centre (SOC). An LSTM (Long Short-Term Memory) based model was developed for alert prioritisation, and four XAI methods were applied: LIME, SHAP, Integrated Gradients, and DeepLIFT to explain the model predictions. Expert knowledge was incorporated to assess the quality of the explanation using the precision of the relevance mass and the precision of the relevance rank. Quantitative evaluation showed that DeepLIFT consistently produced the most faithful, robust, and reliable explanations.

Furthermore, this dissertation presents a Transformer-based IDS for cyberattack detection in IoMT networks, which achieves high detection performance for both binary and multiclass classification tasks. To improve transparency transformer IDS, post hoc XAI methods, including SHAP and LIME, were applied and evaluated using metrics such as faithfulness, sensitivity, and complexity, demonstrating that high detection performance and explainability can be achieved simultaneously.

Although IDS based on ML and DL have achieved high classification accuracy, their reliance on centralised data storage raises privacy and security concerns. Federated Learning (FL) addresses these challenges by enabling decentralised, privacy-preserving model training, where data remain local, and only model parameters are shared with the central server. However, explaining the ML model induced in this setting is challenging due to the complex nature of FL, especially using Post hoc XAI methods. Traditional post hoc XAI methods require access to input data for explanations, which violates privacy in FL when explaining the server model. This dissertation addresses the challenge of explainability in federated learning (FL)-based IDS, where privacy constraints limit access to data for post-hoc explanations. To overcome this, a Federated Explainable AI (FedXAI) framework is proposed, incorporating SHAP in a privacy-preserving manner by securely aggregating client-based SHAP values to approximate server model explanations for both binary and multiclass IoT attack detection tasks. In addition, a synthetic data-based explainability approach is introduced that employs various variants of the generative adversarial network (GAN) within FL. The generated synthetic data from FL-Wasserstein Conditional GAN (WC-GAN) enables accurate server-side explanations using post-hoc SHAP, closely matching explanations derived from real data.

Finally, this dissertation proposes an Explainable Active Learning (XAL) paradigm for IoT botnet detection, integrating post hoc explainability methods (SHAP and LIME) directly into the active learning cycle to assist SOC analysts during labelling. The explanation quality is evaluated using faithfulness, monotonicity and sensitivity metrics, demonstrating that Query-by-Committee with Maximum Disagreement (QBC-MD-7) delivers both high model performance and reliable explanations, with SHAP outperforming LIME across all evaluation metrics.

Kokkuvõte
Selgitaval tehisintellektil baseeruvad ründetuvastussüsteemid

Võrgutehnoloogiate kiire areng ja andmemahtude kasv suurendavad küberohtude keerukust, eriti kuna keerukad küberrünnakud on üha enam suunatud asjade interneti (IoT) ja meditsiiniliste asjade interneti (IoMT) võrkudele. Kuigi masinõppe (ML) mudelid on näidanud paljulubavaid lahendusi pahatahtlike tegevuste avastamiseks sellistes võrkudes, piirab mudelite tõlgendatavuse ja läbipaistvuse puudumine sageli nende tõhusust ja usaldusväärsust. Käesolevas doktoritöös uuritakse tõhusate, tõlgendatavate, läbipaistvate ja privaatsust säilitavate masinõppepõhiste sissetungide avastamissüsteemide (IDS) arendamist nii tsentraliseeritud kui ka detsentraliseeritud (föderaalse õpe) keskkondades.

Enamik varasemaid küberjulgeoleku alaseid uuringuid on keskendunud peamiselt rünnakute avastamisele IoT- ja IoMT-võrkudes. Käesolevas artiklis uuritakse tunnuste valiku rolli IDS-i jõudluse parandamisel nendes keskkondades. IoT-botneti tuvastamiseks kavandati botneti elutsükli alusel kuus klassifitseerimisülesannet, kus filtri- ja wrapper-meetoditega määrati kindlaks minimaalne tunnuste alamhulk, mis saavutas kõrge tuvastamismäära. IoMT-rünnakute tuvastamiseks kasutati filtripõhiseid tunnuste valiku meetodeid, nagu informatsiooni kasv, vastastikune informatsioon ja Fisheri skoor, et hinnata ML-mudeleid kahel IoMT-võrguliikluse andmekogul. Uuring näitab, et infokasvu kasutamisel on võimalik väikese tunnuste alamhulgaga (3–4 binaarse klassifitseerimise ja 7–8 mitmeklassilise klassifitseerimise puhul) saavutada kõrge avastamismäär tõhusa IDS-i jaoks IoMT-võrkudes.

Võrgustatud süsteemide sissetungide arvu kasv on kiirendanud tehisintellekti (AI) uurimist IDS-i jaoks. Peamine fookus on ML-põhise IDS-i mõistmine ja selgitamine turvalisuse analüütikutele, kes haldavad neid süsteeme oma võrkude kaitsmiseks. See motiveerib seletatava AI (XAI) meetodite integreerimist, et suurendada IDS-mudelite läbipaistvust ja usaldusväärsust. Kuigi paljudes uuringutes on ML-põhiste IDS-mudelite seletamiseks kasutatud XAI-d, puudub neis sageli süstemaatiline hinnang genereeritud seletuste kvaliteedile. Käesolevas artiklis rõhutatakse seletuste kvaliteedi kvantitatiivse hindamise tähtsust koos IoT-botnetite avastamise tulemuslikkusega. Rakendati mitmeid post hoc XAI meetodeid, sealhulgas SHAP (SHapley Additive explanations), LIME (Local Interpretable Model-agnostic Explanations), DeepLIFT (Deep Learning Important FeaTures), Integrated Gradients ja muud silmapaistvusel põhinevad selgitamismeetodid, et selgitada IoT-botnetite tuvastamise mudeleid nii binaarsete kui ka mitmeklassiliste klassifitseerimisülesannete puhul IDS-kontekstis. Nende selgituste kvaliteeti hinnati kvantitatiivsete mõõdikute abil, sealhulgas usaldusväärsus, monotoonsus, keerukus ja tundlikkus, mis võimaldasid IDS-is XAI-meetodeid rangemalt hinnata. Statistiliste ML-põhiste mudelite puhul kasutati SHAP-i ja LIME-i, et selgitada mitmel andmekogumil treenitud mudelite käitumist, kusjuures SHAP näitas selgituste mõõdikute osas paremat tulemust. Sügavõppe-põhise IDS-i jaoks arendati mitmeklassilise IoT-botneti tuvastamiseks spetsiaalne DNN-mudel, mis saavutas kõrge klassifitseerimise tulemuslikkuse. Kohalike selgituste genereerimiseks rakendati seitset post hoc XAI-meetodit, sealhulgas SHAP, LIME, DeepLIFT, IG, Gradient*Input, Feature Ablation ja Saliency. Nende selgituste kvaliteeti hinnati rangelt kvantitatiivsete näitajate abil, nagu usaldusväärsus, monotoonsus, keerukus ja tundlikkus, mis näitas, et DeepLIFT andis hinnatud meetodite seas järjepidevalt kõige usaldusväärsemad ja stabiilsemad selgitused.

Lisaks botneti tuvastamisele uuritakse käesolevas väitekirjas võrgu sissetungide tuvastamise süsteemi (NIDS) hoiatuste klassifitseerimise selgitavust, kasutades TalTechi turvaoperatsioonide keskuse (SOC) reaalseid andmeid. Häirete prioriseerimiseks arendati LSTM (Long Short-Term Memory) põhinev mudel ja mudeli ennustuste selgitamiseks rakendati nelja XAI meetodit: LIME, SHAP, Integrated Gradients ja DeepLIFT. Selgituste kvaliteedi

hindamiseks kasutati ekspertteadmisi, võttes aluseks asjakohasuse massi täpsuse ja asjakohasuse järjestuse täpsuse. Kvantitatiivne hindamine näitas, et DeepLIFT andis järjepidevalt kõige täpsemaid, robustseimaid ja usaldusväärsemaid selgitusi.

Lisaks tutvustatakse käesolevas väitekirjas Transformer-põhine IDS küberrünnakute avastamiseks IoMT-võrkudes, mis saavutab kõrge avastamismäära nii binaarsete kui ka mitmeklassiliste klassifitseerimisülesannete puhul. Transformer-põhise IDS läbipaistvuse parandamiseks rakendati post hoc XAI meetodeid, sealhulgas SHAP ja LIME, ning neid hinnati selliste näitajate abil nagu täpsus, tundlikkus ja keerukus, mis näitas, et on võimalik saavutada samaaegselt kõrge avastamismäär ja selgitavus.

Kuigi ML- ja DL-põhised IDS-id on saavutanud kõrge klassifitseerimistäpsuse, tekitab nende sõltuvus tsentraliseeritud andmesalvestusest privaatsuse ja turvalisuse probleeme. Federated Learning (FL) lahendab need probleemid, võimaldades detsentraliseeritud, privaatsust säilitavat mudelitreenimist, kus andmed jäävad lokaalsetesse andmebaasidesse ja ainult mudeli parameetrid jagatakse keskse serveriga. Siiski on selles keskkonnas tekkinud ML-mudeli selgitamine keeruline FL-i keeruka olemuse tõttu, eriti post hoc XAI-meetodite kasutamisel. Traditsioonilised post hoc XAI meetodid nõuavad selgituste andmiseks juurdepääsu sisendandmetele, mis rikub FL-i privaatsust serverimudeli selgitamisel. Käesolevas väitekirjas käsitletakse selgitavuse väljakutset föderatiivsel õppel (FL) põhinevas IDS-is, kus privaatsuspiirangud piiravad juurdepääsu andmetele post hoc selgituste andmiseks. Selle ületamiseks pakutakse välja föderatiivse seletatava tehisintellekti (FedXAI) raamistik, mis integreerib SHAP-i privaatsust säilitaval viisil, koondades turvaliselt kliendipõhised SHAP-väärtused, et ligikaudselt arvutada serverimudeli seletused nii binaarsete kui ka mitmeklassiliste IoT-rünnakute tuvastamise ülesannete jaoks. Lisaks tutvustatakse sünteetilistel andmetel põhinevat seletatavuse lähenemist, mis kasutab FL-is generatiivse vastandvõrgustiku (GAN) erinevaid variante. FL-Wasserstein Conditional GAN (WCGAN) abil genereeritud sünteetilised andmed võimaldavad serveripoolseid selgitusi kasutades post-hoc SHAP-i, mis vastavad täpselt tegelikest andmetest saadud selgitustele.

Lõpuks pakub käesolev väitekiri välja selgitava aktiivsõppe (XAL) paradigma IoT-botneti tuvastamiseks, integreerides post hoc selgitavuse meetodid (SHAP ja LIME) otse aktiivsõppe tsüklisse, et aidata SOC-analüütikuid märgistamisel. Selgituste kvaliteeti hinnatakse usaldusväärsuse, monotoonsuse ja tundlikkuse mõõdikute abil, mis näitab, et maksimaalse erimeelsusega komitee poolt esitatud päringud (QBC-MD-7) pakuvad nii kõrget mudeli jõudlust kui ka usaldusväärseid selgitusi, kusjuures SHAP ületab LIME kõigis hindamismõõdikutes.

Appendix 1

I

R. Kalakoti, S. Nõmm, and H. Bahsi. In-depth feature selection for the statistical machine learning-based botnet detection in iot networks. *IEEE Access*, 10:94518–94535, 2022

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 July 2022, accepted 23 August 2022, date of publication 5 September 2022, date of current version 14 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204001

RESEARCH ARTICLE

In-Depth Feature Selection for the Statistical Machine Learning-Based Botnet Detection in IoT Networks

RAJESH KALAKOTI, (Graduate Student Member, IEEE), SVEN NÕMM, AND HAYRETDIN BAHSI

Department of Software Science, School of Information Technology, Tallinn University of Technology (TalTech), 12618 Tallinn, Estonia

Corresponding author: Rajesh Kalakoti (rajesh.kalakoti@taltech.ee)

This work was supported by the European Union through the European Social Fund through the Project Information Communication Technology (ICT) Program.

ABSTRACT Attackers compromise insecure IoT devices to expand their botnets in order to launch more influential attacks against their victims. In various studies, machine learning has been used to detect IoT botnet attacks. In this paper, we focus on the minimization of feature sets for machine learning tasks that are formulated as six different binary and multiclass classification problems based on the stages of the botnet life cycle. More specifically, we applied filter and wrapper methods with selected machine learning methods and derived optimal feature sets for each classification problem. The experimental results show that it is possible to achieve very high detection rates with a very limited number of features. Some wrapper methods guarantee an optimal feature set regardless of the problem formulation, but filter methods do not achieve that in all cases. The feature selection methods prefer channel-based features for detection at post-attack, communication, and control stages, while host-based features are more influential in identifying attacks originating from bots.

INDEX TERMS Feature selection, machine learning, Internet of Things, botnet, intrusion detection.

I. INTRODUCTION

IoT (Internet of Things) is shaping the way we live our human lives [1], from tiny toys to home-made applications to smart cities. IoT is a system of interrelated devices connected to the Internet to transmit and receive data from one device to other parts of the system; it can be an edge device, a cloud server, or another field device. At the same time, the IoT security issue has become more important as an enormous amount of data is associated with IoT networks. Due to the exponential growth of IoT devices [2], hackers and cybercriminals have more opportunities to exploit network vulnerabilities [3], resulting in various IoT-based botnet attacks [4], [5], [6]. The botnet, a large set of compromised machines controlled by attackers, is one of the strongest threats on the Internet to perpetrate cybercrimes, such as launching DDoS attacks [4], stealing sensitive data [7] or distributing malicious spam [8]. As a result, botnets act as a source of spreading malicious activity and usually threaten the availability of networks, in addition to other significant security consequences. It is important to develop security countermeasures against botnet threats.

A typical botnet life cycle has four phases, formation, command and control (C&C), attack and post-attack [9]. Attackers spread malware that helps them recruit new bots (that is, members of botnets) during the formation phase. C&C phase enables them to establish continuous communication with bots to control them for future actions. In the attack phase, attackers carry out malicious operations using bots. The post-attack phase covers activities related to the spread of IoT malware with the purpose of expanding the botnet. IoT networks constitute a lucrative target for botnet

The associate editor coordinating the review of this manuscript and approving it for publication was Chin-Feng Lai.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

owners, as it is possible for them to recruit large numbers of IoT devices, which are usually shipped with various security vulnerabilities.

One of the effective security countermeasures against botnets is to establish security monitoring systems to detect malicious activities. An organization hosting various IoT devices is interested in the identification of devices that are compromised by IoT bot malware; therefore, its focus is much more on detection at formation, C&C or post-attack phases. On the other hand, organizations receiving attacks from IoT bots aim to prevent malicious traffic launched during the attack and post-attack phases. Therefore, it is important to develop a monitoring system that encompasses the entire botnet life cycle. This endeavor requires a more in-depth understanding of malicious actions and their characterization in each phase.

The Internet of Things (IoT) has received great attention in research on network anomalies and intrusion detection [10]. Malicious network traffic has been detected with conventional signature-based solutions such as Snort [11] or Suricata [12]. The drawback of signature-based systems is the inability to detect unknown or previously unidentified attacks, in addition to the obstacles that arise from mismanagement of signatures.

Instead of signature-based solutions, a behavior- or anomaly-based solution goes beyond identifying individual attack signatures to detect and analyze malicious behavior patterns. Machine learning is considered a viable solution that detects new variants of attacks with the elimination of the need for signatures. Although the application of statistical machine learning (ML) techniques has demonstrated highly accurate classification results in malicious traffic detection problems [13], feature selection as an important step in the ML workflow has not been fully addressed. The curse of dimensionality can be a concern that decreases detection performance due to overfitting when classifiers are trained with a large number of features [14]. In addition, a high-dimensional feature space may require more computing resources when the models are deployed in the operational environment. In most cases, intrusion detection systems should handle a large volume of network traffic, so maximizing resource usage is vital. IoT environments bring additional restrictions, so that detection sensors, system components that are responsible for the collection of network traffic and performing the detection function, may run on resource-constrained devices (e.g., edge devices). Therefore, reducing the size of the feature set can improve the performance of ML models in many ways. Additionally, feature selection helps to achieve a deeper understanding of the underlying approaches that rendered the data, since fewer features would be more perceivable by experts.

Various academic works [15], [16], [17], [18], [19], [20], [21] use feature selection techniques to improve the detection scores of existing ML classifiers. However, these studies do not explore the impact of feature selection methods on different binary and multiclass classification formulations that can be performed for intrusion detection at various stages of the botnet life cycle. More specifically, the set of features that is effective in detecting malicious traffic at one stage may not be instrumental at another stage. Furthermore, the performance of models that use different feature selection methods can vary according to the classification formulation.

The crux of this paper is to find the optimal subset of features with the help of filter and wrapper feature selection methods for various classification formulations that can be applied to IoT botnet attack detection. For this purpose, we have induced ML classifiers using the methods, extra tree classifier, random forest, decision tree, and k-nearest neighbor. The optimal feature sets are derived by a 10-fold cross-validation with classifiers from filter and wrapper methods.

In this research, we applied the feature selection methods to two datasets, namely N-BaIoT [22] and MedBIoT [23], which include network activities belonging to different steps of the botnet life cycle in IoT networks. Based on the phases of the botnet life cycle given in [9], we can deduce that N-BaIoT has instances related to the attack phase, while MedBIoT covers post-attack and C&C phases.

In addition to a binary classification, such as discriminating malicious traffic from benign traffic, it is possible to formulate various multiclass classification problems from these datasets. One of such formulations may focus on the detection of the malware type that induces the malicious traffic (e.g., Mirai, Bashlite), which is applicable for both datasets, whereas the second one may deal with the attack type that is conducted by the corresponding malware. For the latter case, N-BaIoT provides labels on the types of attacks that originated from infected devices (e.g., UDP flooding, spam), and MedBIoT has labels on whether the activity belongs to the C&C or post-attack phase. Depending on the situation, security administrators may be interested in different aspects of detection to make more informed operational decisions. For example, identifying the type of malware on the infected device would be necessary to apply the correct malware removal procedures. On the other hand, identifying the type of attack rather than the type of malware would be more essential for organizations that receive botnet attacks, as they need to develop defensive countermeasures to block or redirect network traffic accordingly. In our study, we investigate which feature sets are optimal for each binary and multiclass classification formulation and analyzed whether there exist variations in the optimal feature set that may impact the design considerations of intrusion detection in such different contexts. This contribution is unique because, to our knowledge, there is no study that provides a deeper analysis of the variations in feature sets that are effective in intrusion detection at different stages of the botnet life cycle.

The structure of this research work is described below. In Section II we have mentioned background work and a review of the literature related to botnet detection and feature selection. In Section III, the feature selection methods and experiments are described. Finally, our results are presented in Section IV. Section V gives a discussion of the main

findings of this research work. Conclusions are drawn in Section VI.

II. BACKGROUND AND LITERATURE REVIEW

A. BOTNET DETECTION

Researchers have introduced traditional machine learning and data mining methods for botnet detection in recent decades and made significant advances. BotMiner [24], [25] and BotSniffer [26] used statistical algorithms to detect malicious traffic on an IoT network that is part of a botnet.

The Bayesian optimization Gaussian process (BO-GP) [27] is combined with the decision tree classifier as an optimized ML-based framework to detect botnet attacks on IoT devices. The detection rate for binary classification is improved to 99%.99 when the accuracy, precision, recall, and f1 score metrics are compared to the Decision Tree, SVM, with this optimized DT-BOGP framework. In this work, the Bot-IoT-2018 dataset [28] is used.

Convolutional neural networks (CNN) are used to detect IoT malware. This approach was created for the detection of Linux IoT botnets based on the PSI graph together with the CNN classifier [29]. Experiments were carried out using 4002 labeled IoT botnet datasets provided by the IoT-POT [30] team. These data sets were collected over one year, from October 2016 to October 2017. The detection rates, 92% precision and 94% F1 score, are achieved with the CNN classifier.

Yin et al. proposed the Bot-GAN framework to improve botnet detection performance [31]. Generative adversarial networks are used, where the GAN generator creates fake samples. A 3-layer LSTM network was selected as the generator and a 4-layer neural network architecture was chosen as the detector in the Bot-GAN setup. The ISCX dataset [32] is used for this framework. Of 491,381 training samples, 192,112 (39.10%) are malicious and include seven botnets, while the test set consists of 348,452 testing samples. This test set has 169988 (48.78%) malicious samples that possess 16 botnet types. The detector achieves 68.51% as an F1 score without having fake samples. The detector attains a maximum 70.59% of F1 score when the training set has 500 fake samples.

A hybrid deep learning scheme [33] is used to detect the botnet in the IoT network. A long-short-term memory autoencoder (LAE) is implemented to reduce the dimensionality of network traffic features. Then, the long-term interrelated network traffic behavior is analyzed with the help of bidirectional long-short-term memory (BLSTM) to achieve generalization ability. In this work, binary multiclassification problems are addressed in the BoT-IoT dataset [28] for the classification of network traffic. In general, 6 features were derived from 37 features of the dataset [28] with the help of the LAE and BLSTM classifier that achieved 100% precision, 93.17% MCC (Matthews correlation coefficient).

Alauthman *et al.* [34] have proposed a traffic reduction mechanism that integrates the reinforcement learning technique in three datasets. The first dataset is information security and objects technology (ISOT) that contains Storm Bot, Waledac Bot, and normal traffic. The second data set comprises four legitimate P2P applications (Vuze, uTorrent, Frostwire and eMule) and three P2P botnets (Zeus, Storm and Waledac) [35], and the third is the ISCX data set [32], which contains benign traffic. The authors have used real-world network traffic to evaluate their proposed approach and achieved a detection rate of 98.3% and a false positive rate of 0.012%.

Singh *et al.* [36] have developed a quasi-real-time intrusion detection system using open-source tools such as Hadoop, Hive, and Mahout to provide scalability for the identification of Peer-to-Peer botnet attacks. For this, the authors have built the packet capture module to process high data bandwidth in a quasi-real-time (within 5-30 s delay) and developed a distributed dynamic feature extraction framework to illustrate network traffic statistics of packet captures. The parallel processing power of Mahout (that is, a machine learning library built on top of Hadoop) was used to build the Random Forest model that achieved a detection performance of 99% precision and recall.

B. FEATURE SELECTION

Feature selection aims to find the best subsets of features from input data to achieve better prediction results by eliminating unnecessary features [37]. The feature selection methods were classified mainly into three categories, such as filter, wrapper, and embedded [14]. Filter methods utilize statistical methods to rank features according to their discriminatory power. They are usually applied in an initial step before inducing the models. However, wrapper methods use a machine learning model to evaluate the merits of a given set of features in terms of model performance to identify the optimal set. Embedded methods blend the advantageous factors of both the filter and wrapper methods so that they perform feature selection and training of the ML algorithm in parallel. This feature selection method is an integral part of the classification or regression model.

Many feature selection approaches have been applied to evaluate the importance of features related to the context of botnet detection. Entropy, impurity, RelieF and principal component analysis (PCA) [38] were used with the neural network classification algorithm. 99.20% detection rate was achieved with the top 10 features based on the entropy of a total of 29 features in two botnet datasets, ISOT [39] and ISCX [32].

Velasco-Mata *et al.* [40] has tested the feature sets 5, 6, 7 with two filter methods, Information Gain and Gini Importance, over Decision Tree, Random Forest, k-NN for botnet detection for multiclass classification. Finally, the set of five features produced an 85% detection rate with a decision tree classifier induced for the QB-CTU13 [41] and EQB-CTU13 [41] datasets.

Guerra-Manzanares *et al.* [19] proposed a hybrid approach by combining filter and wrapper methods with random forest and k-NN classifiers. Eighteen features are selected by

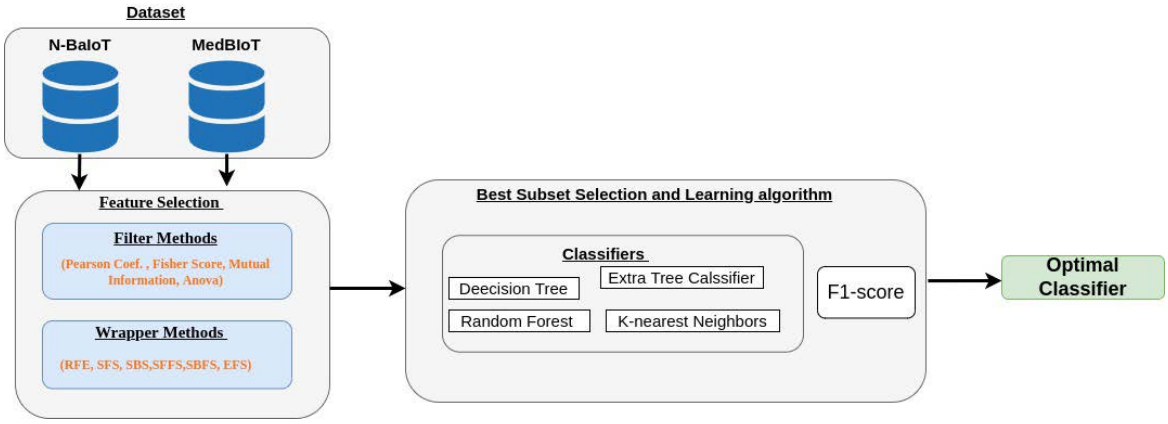


FIGURE 1. We used the filter and wrapper method feature selection approaches over the N-BaIoT [22] and MedBIoT [23] datasets to find the optimal feature subset.we evaluated all the feature subsets with four classifiers - DT, ET, RF and k-NN.

Pearson's correlation, and the top 20 features are selected with Fisher score. This study used the botnet dataset, N-BaIoT [22], which has 115 statistical features extracted from network traffic in an IoT network. The feature sets obtained from the filter methods are processed by wrapper methods, Sequential Forward Selection and Sequential Backward Elimination. Finally, a five-element set of features is used for the detection of IoT botnets formulated as a binary classification problem.

Correlation-based feature selection, consistency-based subset evaluation and principal component analysis [42] are used to select features that are then evaluated with decision trees, the Naive Bayes classifier, and the Bayesian Network classifier to detect botnet traffic based on peer-to-peer (P2P). With these selection methods, 5, 8, and 12 features were identified, respectively. 99% accuracy achieved with the decision tree based on the ISOT dataset [39].

Pektaş and Acarman [43] used linear models penalized with the L1 norm (also called Lasso), recursive feature elimination (RFE), tree-based feature selection methods (random forest feature importance) for the ISOT dataset [39]. Random forest feature selection produced 99% highest detection among all these feature selection methods.

The studies proposing feature selection do not create and compare the optimal sets that can be obtained for different multiclass problem formulations. In this paper, we address this gap by inducing various learning models for two datasets as explained in detail in Section II-C.

C. DATASET

In this study, we used two datasets, N-BaIoT [22] and MedBIoT [23]. Both datasets comprise legitimate IoT traffic as well as traffic with various types of attacks that originate from compromised IoT devices acting as bots.

N-BaIoT and MedBIoT have 115 and 100 features (mainly descriptive statistics measures), respectively, which are extracted from network traffic. These traffics are generated by bots deployed in a controlled testing environment. Both datasets have the same features, except that the MedBIoT dataset does not include network traffic coded as "H" in Table 1. More specifically, the features that are defined for each data point reflect the aggregated statistics of the raw streams of the network in five time windows (100 ms, 500 ms, 1.5 s, 10 s, and 1 min), which are coded L5, L3, L1, L0.1 and L0.01, respectively. There are five main feature categories, host-IP (traffic originated from a specific IP address, coded as H), host-MAC and IP (traffic originated from the same MAC and IP, coded MI), channel (traffic between specific hosts, coded HH), socket (traffic between specific hosts, including ports, coded HpHp), and network jitter (time interval between packets in channel communication, coded as HH_jit). For each major category, the packet count, mean and variance packet sizes are calculated. There have been extra statistical values like the correlation coefficient (PCC) of packet size, radius, covariance, magnitude, which are derived for Channel and Socket categories along with packet count, mean, variance. In this paper, we used a specific notation to name the features. The feature name is the concatenation of three keywords. The first one represents the category type (e.g., MI, HH), the second one shows the time window, and the third one indicates the statistical measurement function. For instance, "HH_L0.01_mean" means this feature is about the channel type that belongs to a 1-min interval with a mean function.

In this study, we have developed six different ML classification problems using these two datasets, as detailed in Table 2. The N-BIoT dataset is used for three classification problems, namely, binary, 3-class, and 9-class. Binary classification basically discriminates malicious traffic from benign traffic. 3-class provides greater scrutiny of malware type by classifying data points into categories, mirai, gafgyt, and benign. For the 9-class classification, the data points have been classified into different attack types: ack, benign,

TABLE 1. Summary of the features of the N-BaIoT and MedBIoT datasets features.

				N-BaIoT	MedBIoT
Feature Category	Category Code	Statistical Value Feature	Time Frame Window	No. of Features	No. of Features
Host Mac& IP	MI	Packet Count, Mean Variance	100 Micro Seconds	15	15
Host IP	H		500 Micro seconds	15	–
Network Jitter	HH_Jit		1.5 Seconds	15	15
Channel	HH	Packet Count, Mean Variance, Magnitude, Radius, Covariance, Correlation	10 Seconds	35	35
Socket	HpHp		1 Minute	35	35

TABLE 2. Classification problems addressed in this study.

Dataset	Classification Task	Class Name	Description of the Class name
N-BaIoT	Binary	Benign	Legitimate Network Traffic
		Attack	Malicious Network Traffic (Mirai, Gafgyt)
	3-class	Mirai	Mirai malware-infected network traffic
		Benign	Legitimate Network Traffic
		Gafgyt	Gafgyt malware-infected network traffic
	9-class	ACK	Gafgyt malware Sending Spam data
		Benign	Legitimate Network Traffic
		COMBO	Gafgyt malware Sending spam data and opening a connection to IP, port
		JUNK	Mirai Malware ACK-Flooding
		SCAN	Scans the network devices for vulnerabilities,(Mirai &Gafgyt)
		SYN	Mirai Malware SYN-Flooding
		TCP	Gafgyt malware TCP Flooding
		UDP	UDP flooding (Mirai & Gafgyt)
		UPDPLAIN	Mirai malwar UDP flooding with Less of an option for higher packet per second
MedBIoT	Binary	Benign	Legitimate Network Traffic
		Attack	Malicious Network Traffic (Mirai, Bashlite, Torii)
	3-class	Benign	Legitimate Network Traffic
		C&C	network traffic for C&C
		Spread	Spread Attack network traffic
	4-class	Bashlite	Bashlite malware-infected network traffic
		Benign	Legitimate Network Traffic
		Mirai	Mirai malware-infected network traffic
		Torii	Torii malware-infected network traffic

compact, junk, scan, syn, tcp, udp, and udpplain. These three-class and nine-class problem formulations address the attack phase of the botnet life cycle from two perspectives. The former identifies the types of malware that can be instrumental in detecting infected hosts in an organizational setting. The latter aims to discriminate against attacks carried out by bots, which better informs organizations that are targeted by such attacks.

MedBIoT is used for three classification formulations, binary, 3-class, and 4-class. As this dataset is collected at the C&C or formation phases, such formulations reveal which features are important in those phases. More specifically, 3-class addresses the identification of the phase (i.e., classes are benign, C&C and Spread), whereas 4-class aims to detect malware category (i.e., classes are benign, Bashlite, Mirai, and Torii).

In this work, we have experimented with 20,000 samples of each class label for the addressed classification type. For example, if the classification problem contains two classes, we randomly selected 40,000 samples from the source dataset.

III. FEATURE SELECTION METHODS

Within the framework of the present investigation, two types of feature selection methods are considered. The first is called the filter model, which evaluates a feature or a subset of features using a class-sensitive discriminating criterion [44]. These techniques do not depend on the particular classification algorithm. The second type of technique is the wrapper model. Techniques of this type use the characteristics of the specific classification algorithm to choose the feature set.

A. FILTER MODELS

In the domain of numeric feature sets, there are four main types of techniques. The first utilizes the linear correlations between the features. The second is based on the relationship between the inter-class and intra-class separation. The third uses entropy, and the fourth is based on the analysis of variance.

1) PEARSON's CORRELATION BASED TECHNIQUE

Based on Pearson's correlation coefficient (see (1)), the technique requires one to compute the collinearity matrix for the entire set of features to find the redundancy of the features. Pearson's correlation technique computes the linear correlation relationship between two variables. Pairwise correlations between features are analyzed to find the redundancy of features. $\mathcal{P}$-value of correlation coefficients bounds the ranges between -1 and 1. Two features contain a perfect positive correlation if the value is $\mathcal{P} = 1$. There is no correlation between the two features if the value $\mathcal{P} = 0$, and a perfect negative correlation is accepted if the value $\mathcal{P} = -1$. The formula for the Pearson correlation

$$\mathcal{P} = \frac{\sum_{i=1}^{n}[(x_i - \mu_x)(y_i - \mu_y)]}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\sum_{i=1}^{n}(y_i - \mu_y)^2}} \quad (1)$$

In (1), μ_x and μ_y denote the means of features x and y respectively. Greater absolute values of Pearson's correlation coefficient indicate stronger linear dependence between the features.

2) FISHER SCORE

Fisher score [44] is designed for the numeric features and measures the ratio of the average inter-class separation to the average intra-class separation. It is also referred to as Fisher's ratio [45]. Formally defined in (2) and denoted as F_s (not to be confused with F1 score), the numerator calculates the average inter-class separation and, the denominator calculates the average intra-class separation.

$$F_s = \frac{\sum_{j=1}^{K} p_j(\mu_j^i - \mu^i)^2}{\sum_{j=1}^{K} p_j(\sigma_j^i)^2} \tag{2}$$

where μ_j^i and σ_j^i are the mean and standard deviation of the j-th class and i-th feature, p_j is the proportion of data points of class belonging to the class j. Greater Fisher's score values indicate greater discriminating power of the feature.

3) MUTUAL INFORMATION

Among the different techniques implementing mutual information exclusion idea *normalized mutual information feature selection* [46] was chosen. For the case of continuous variables mutual information (MI) is defined by [46] as follows:

$$I(\mathcal{X}, Y) = \int \int p(x, y) \log \frac{p(x, y)}{p(x), p(y)} dxdy \tag{3}$$

Here, $p(x, y)$ is the joint probability density function (PDF) of the variables $\mathcal{X}$, Y and $p(x)$ and $p(y)$ are the marginal PDFs of the respected variables. For the case of discrete variables, [46] defines MI as follows:

$$I(X; Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}. \tag{4}$$

In (5) $p(x, y)$ denotes the joint probability mass, the function, the function, and $p(x)$ and $p(y)$ are the marginal probabilities. Mutual information values fall in the interval given below.

$$0 \leq I(X; Y) \leq \min\{H(X), H(Y)\} \tag{5}$$

To make this paper self-sufficient, the main steps of the MI -based feature selection algorithm proposed by [46] are presented below. Denote $I(C; S)$ the MI between the class variable C and the subset of selected features S. Also define measure G as

$$G = I(C; f_i) - \frac{1}{|S|} \sum_{f_s \in S} NI(f_i; f_s). \tag{6}$$

1) Initialize the initial feature set F that includes all available features and the empty set S of the selected features.
2) Calculate $I(f_i, C)$ or each feature $f_i \in F$.
3) To select the first feature, find $\hat{f}_i$ such that $\hat{f}_i = \max_{i=1,\ldots,N}\{I(f_i, C)\}$.
4) Update sets F and S as follows: $F = F \setminus \hat{f}_i$ and $S = \hat{f}_i$
5) Repeat until $|S| = k$.
   a) Compute $I(f_i; f_s)$ for all pairs of features such that $f_i \in F$ and $f_s \in S$.
   b) Select the feature $f_i \in F$ that maximizes the measure (6).
   c) Update sets F and S as follows: $F = F \setminus \hat{f}_i$ and $S = \hat{f}_i$
6) Return the set S

4) ANOVA F-TEST

ANOVA is one of the most well-known feature selection techniques, therefore, does not require an in-depth explanation. This method usually answers the question of whether the values of the given features are independent of the target classification label or not. It is performed in the form of statistical hypothesis testing, where the null hypothesis states that the values of the feature are independent of those of the target label, and the alternative hypothesis states the opposite. The application of this method requires the user to utilize only the features whose values are not independent of the target labels.

B. WRAPPER METHODS

Unlike the filter models, wrapper methods are classifier-agnostic and choose the most suitable feature set for the particular classifier. The wrapper method is used to calculate the weights of the features using the classification algorithm to measure the performance of the features. Wrapper methods employ the inductive algorithm as an evaluation or criterion function [47], [48]. This approach uses a classification algorithm to evaluate subsets of features based on their predictive accuracy (in test data) after cross-validation of the dataset. In the context of our research, we have evaluated subsets of features using the F1 score. Usually, the feature set is being constructed iteratively by adding (forward selection) or deleting (backward elimination) the features. Within each branch, particular methods differ by evaluating the significance of the features, the goodness criteria of the model, and the number of features added or removed. In the preliminary stage of the investigation, the authors have experimented with six different wrapper techniques. Among them, Recursive Feature Elimination (RFE) [49], Sequential Backward Selection (SBS), and Sequential Forward Selection (SFS) [50] have shown the best results and are included in the comparison.

1) RECURSIVE FEATURE ELIMINATION

Recursive feature elimination (RFE) is a greedy algorithm based on feature ranking techniques [49]. Based on a characteristic of the feature-ranking criterion, the RFE starts with a complete set of features and then removes the least relevant feature one by one to choose the most significant features. The RFE is used with the following classification algorithms, DT, ET, and RF. This method uses the following steps to evaluate the significance of the features.

1) Initialize the initial set of features F that includes all available features, set each element of the feature ranking list R to $1/n$.
2) Repeat the following steps until the feature set $F = \emptyset$
   - Train with the classification algorithm and calculate the importance of the feature in set F. Order the features corresponding to their importance and update the list R accordingly.
   - Eliminate the feature of the smallest importance.
3) Output: List of Feature Rankings R.

2) SEQUENTIAL FORWARD SELECTION

We have used two sequential algorithms [50] that work based on greedy search algorithms. SFS [50] is a stepwise search approach that can avoid excessive computational time consumption. It works in a bottom-to-top approach. The following steps are involved in the SFS Algorithm.

1) Start with an empty set $S = \emptyset, F = f_1, f_2, \ldots f_n$
2) **while** |F|>0
   # |F| is size of the feature set F
3) $f_i = argmin_{j \in F}[J(S + f_i)]$
   (Select the feature $f_i \in F$ with the maximum performance of the classification algorithm and join to the set S (the features selected subsequently combine with the initial selected feature)
4) $S = S + f_i$
5) $F = F - f_i$

Consider F to be a set of features. Then select the best feature among the F features using some evaluation criterion function J that maximizes the performance of the classification algorithm. The F1 score is considered an objective evaluation criterion function. At each iteration, a new feature subset is created with the help of one of the remaining available features and the previous feature subset. The new subset of features should provide the maximum classification performance compared to the addition of any other feature. This iteration continues until the total number of features is completed in the set F. SFS method is the best and most rapid method when a small subset of optimal features is available.

3) SEQUENTIAL BACKWARD SELECTION

In contrast to SFS, SBS (Sequential Backward Selection) operates in a top-to-bottom approach. The selection of features starts from a set F with n being the total number of features. Therefore, the evaluation function produces the maximum performance of the classification algorithm for all n numbers of features. Each feature is removed one at a time. For every iteration, the new subset is created by the $n-1$ features computed with the help of the evaluation function, and then the worst feature is discarded from the next subset of features. This procedure continues until the total number of features is left.

1) S = feature set, $F = f_1, f_2, \ldots f_n$
2) **while** |F|>1 do
   #|F| is size of the feature set F,
3) $f_i = argmin_{j \in F}[J(S - f_i)]$
4) $S = S - f_i$
5) $F = F - f_i$

TABLE 3. Tuning of learning algorithm hyperparameters.

Algorithm	Description	Range
Decision Tree	Maximumn Depth of the tree	5-50 in steps of 1
	Minimum number of samples required to split an internal node	2-30 in steps of 1
	the minimum number of samples required to be at a leaf node.	2-30 in steps of 1
	the impurity of a split	Gini, Entropy
Random Forest Extra Tree Classifier	Maximum number of levels in each decision tree	5, 500 in steps of 50
	Maximum number of features	Square root, Auto ,
	Maximumn Depth of the tree	5-50 in steps of 1
	the minimum number of samples required to be at a leaf node.	2-30 in steps of 1
	the minimum number of samples required to be at a leaf node.	2-30 in steps of 1
	the impurity of a split	Gini, Entropy
k-nearest neighbors	number of neighbors	1-25 in steps of 1
	Distance metric	minkowski,euclidean,manhattan

C. APPLICATION OF THE MACHINE LEARNING WORKFLOW

For the computational experiments, the classical machine learning workflow was used. The initial datasets are large enough to provide samples that can be balanced with respect to all characteristics of the dataset, malware type, attack type, and device type. In the preprocessing step, balanced samples were drawn from the dataset of interest. Then, the division into training and testing subsets was carried out proportionally 80/20. Initial experiments have demonstrated that among the k-nearest neighbors classifier (kNN), decision tree classifier (DT), random forest classifier (RF), extremely randomized trees classifier (ET), logistic regression, support vector machine, and Ada-boost classifier, the last three have demonstrated much lower performance and were excluded from further investigation. For each remaining classifier and feature selection technique, a ten-fold cross-validation was performed, while, to ensure better results and the best configuration for each classification algorithm, a randomized search was used to find the optimal hyperparameters for each classifier. The range of hyperparameters is described in Table 3.

We use the three steps to evaluate the distinct subsets of features in both datasets. First, the F1 score metric is used to evaluate the set of features. Second, computational time is the total time it takes a computer with a particular processor to complete a task. Third, Performance computed the ratio between the F1 score and the computational time. Intrusion detection systems must respond as quickly as possible without sacrificing accuracy. Response time is essential when thwarting the threat in the early stages would limit the degree of losses. For this motivation, time must be considered when evaluating any detection of the model along with the model metrics. The F1 score (see Eq. (7)) is defined as a harmonic mean of precision (P) and recall(R) [51]. In this research work, precision is the fraction of correctly identified botnet samples to all botnet samples identified as a botnet.

TABLE 4. Filter method feature sets for the N-BaIoT and MedBIoT dataset.

Dataset	Classificaiton type	Pearson Correlation	Fisher Score	Mutual Information	Anova
N-BaIoT	Binary	33	5	3	3
	3-class	33	6	3	5
	9-class	33	68	28	59
MedBIoT	Binary	34	51	36	85
	3-class	34	42	38	49
	4-class	34	46	41	52

On the other hand, recall is the fraction of correctly identified botnet samples for all botnet samples in the dataset [52]. The F1 score provides a more suitable measure of incorrectly classified cases than the accuracy measure. We have used the harmonic mean of the F1 score, as it penalizes the extreme values. F1 score as follows;

$$\text{F1 score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{7}$$

In our experiments, we used the computational time to calculate the computational cost of classifying a sample. We did not consider the training time of the ML algorithms. We have experimented with all tasks on the same CPU. Finally, to measure the performance of a set of features derived from filter and wrapper methods, we calculated the ratio between the F1 score and the computational time to allow measurement of the gain in detection ability relative to the computational expense of this detection [40].

The experiment carried out in this work was carried out on a Ubuntu 20.04.4 LTS machine with 60 GB of DDR4-2666 R ECC RAM and 2 x Intel Xeon Gold 6148 20C 2.40 GHz. We developed our scripts using Python 3, Scikit-learn [53] and mlextend libraries [54].

IV. RESULTS

This section gives experimental results of the learning models induced for six classification problems listed in Table 2. We analyze the importance of the features obtained by filter and wrapper feature selection methods in each problem and perform a comparison between the results. Tables 4 and 5 show the numbers of features selected by the filter and wrapper methods for each classification problem, respectively. We provided detailed analysis of the result of each classification problem in the following subsections.

A. N-BaIoT

1) BINARY CLASSIFICATION

In this part, we use filter and wrapper feature selection methods to find the optimal feature subsets for binary classification of the N-BaIoT dataset. Based on the ratio of the highest detection rate of the minimal feature set to its computational time, as given in Fig. 2, we selected the best model for the implementation of four classifiers with different feature selection methods. In this binary problem formulation, we identified 33 features with fewer correlations according to Pearson's correlation values. For each filter method,

TABLE 5. Wrapper methods feature sets for N-BaIoT and MedBIoT.

Dataset	Classification Type	Feature Selection Approach	DT	RF	ET	Knn
N-BaIoT	Binary	RFE	3	4	4	
		SFS	3	3	3	3
		SBS	3	3	3	3
	3-class	RFE	3	4	4	
		SFS	3	3	3	3
		SBS	3	3	3	3
	9-class	RFE	28	23	25	
		SFS	3	3	3	3
		SBS	3	3	3	3
MedBIoT	Binary	RFE	29	27	24	
		SFS	7	7	7	7
		SBS	7	7	7	7
	3-class	RFE	26	27	24	
		SFS	7	7	7	7
		SBS	7	7	7	7
	4-class	RFE	29	24	22	
		SFS	7	7	7	7
		SBS	7	7	7	7

we select the best features based on their scores. Furthermore, we induce models with feature sets that have increasing numbers to understand how many features are enough to pass the 99% F1 score. Finally, we select the best 3, 5, 3 features for the ANOVA, Fisher Score and mutual information methods, respectively (see Table 4). On the other hand, the wrapper methods usually select three features (for example, DT selects three features in each method), as presented in Table 5.

Almost all classifier and feature pairs produce a high detection rate above 99%, as shown in Table 6. Based on the minimal set and computational performance, we selected three pairs and reported more detailed performance results, accuracy, precision, recall, and F1 score values in Table 7. These pairs are: DT with mutual information (that is, three features), Fisher (that is, five features), and SBS (that is, three features). DT with SBS achieves the highest performance metric, as shown in Fig. 3. Among the wrapper methods, Anova provides better results than the others.

The mutual information method selected the features, {MI_dir_L0.1_weight, MI_dir_L0.01_weight, H_L0.01_ weight}, fisher score selected {MI_dir_L5_weight, HH_jit_L5_mean, MI_dir_L5_mean, MI_dir_L0.01_weight, MI_dir_L0.01_mean}, Anova identified the feature set, {MI_dir_L1_weight, MI_dir_L0.1_weight, H_L0.1_weight}. SBS selected {MI_dir_L5_weight, MI_dir_L3_weight, MI_dir_L1_ weight}. Almost all features belong to the host category; except one that is a network jitter-type feature.

It is important to note that we computed the computational time of the models (i.e. the testing-time performance) after selecting the features in all filter and wrapper methods. Thus, the time required for feature selection is not reported in this paper, as testing time is a more significant aspect compared to training, which is not done so frequently, and, when needed, high resources can be assigned for such task. In this sense, the calculated time can be affected by the number of features and characteristics of the corresponding learning model. However, in our experiments, as expected, we observed that

TABLE 6. F1 scores for binary classification models using feature subsets (represented in Table 4 and 5) of feature selection algorithms in the N-BaIoT dataset.

Binary N-BaIoT					
FS Method	**Approach**	**DT**	**ET**	**RF**	**KNN**
Filter	Pearson Correlation	0.9987	0.9983	0.9983	0.9957
	Fisher Score	0.9997	0.9997	1.0000	0.9847
	Mutual Information	0.9990	0.9990	0.9990	0.9977
	Anova	0.9973	0.9970	0.9970	0.9970
Wrapper	RFE	0.9983	0.9983	0.9987	
	SFS	0.9996	1.0000	0.9999	0.9994
	SBS	0.9998	0.9999	0.9997	0.9966

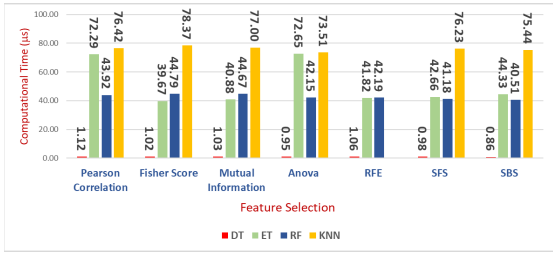


FIGURE 2. Computational time required to classify a sample by binary classification models on N-BaIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.

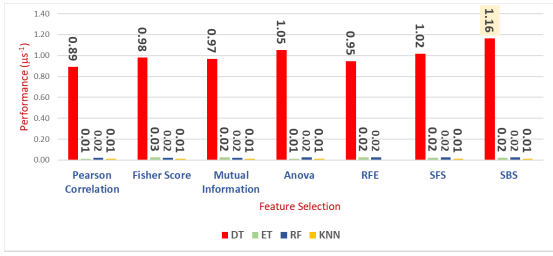


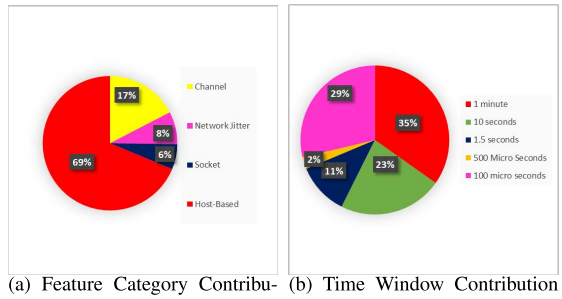
FIGURE 3. Performance achieved by binary classification models over the N-BaIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.

training the wrapper models is more computationally expensive compared to filter methods. Among the wrapper methods, sequential feature selection algorithms (SBS, SFS) are more expensive than recursive feature elimination.

After identifying the optimal feature subsets from the dataset for binary classification, we performed a frequency analysis to scrutinize which feature category and time windows are used primarily by the selection methods, as shown in Fig. 4. Host-based feature categories are observed to play an important role in discriminating malicious traffic from benign traffic. The features of network jitter and socket are less preferred. Although the features regarding the longest time window, 1 minute, have contributed greatly to the

TABLE 7. Accuracy, precision, recall, F1, Binary classification scores of the selected model with performance based on feature sets in the N-BaIoT dataset.

Feature Selection	Feature Subset	Model	Class Name	Accuracy	Precision	Recall	F1-Score
Mutual Information	3	DT	Attack	0.99	0.99	0.99	0.99
			Benign	0.99	0.99	0.99	0.99
Anova	3	DT	Attack	0.99	0.99	0.99	0.99
			Benign	0.99	0.99	0.99	0.99
SBS	3	DT	Attack	1	1	1	1
			Benign	1	1	1	1



(a) Feature Category Contribution to Overall Botnet detection (b) Time Window Contribution to overall botnet detection

FIGURE 4. Contribution of feature categories and time windows in selected feature sets for binary classification in the N-BaIoT dataset.

TABLE 8. F1 scores for 3-class classification models using feature subsets (shown in Table 4 and 5) of feature selection algorithms on the N-BaIoT dataset.

3-Class N-BaIoT					
FS method	**Approach**	**DT**	**ET**	**RF**	**KNN**
Filter	Pearson Correlation	0.9989	0.9989	0.9987	0.9338
	Fisher Score	0.9989	0.9989	0.9987	0.9338
	Mutual Information	0.9991	0.9989	0.9989	0.9730
	Anova	0.9965	0.9956	0.9921	0.9910
Wrapper	RFE	0.9991	0.9964	0.9904	
	SFS	0.9997	0.9997	0.9998	0.9970
	SBS	0.9996	0.9998	0.9994	0.9790

detection, there is no clear increasing or decreasing pattern regarding the time duration, as the shortest duration, 100 microseconds, also plays a significant role in the model performance.

2) 3-CLASS CLASSIFICATION

In the N-BaIoT dataset, Mirai and Gafgyt malware are used to infect IoT devices. In this part, we report the findings of the three-class classification models that discriminate network traffic as Mirai, Gafgyt, and legitimate. Similarly, we evaluated the feature selection method and the pairs of learning models according to the same performance metric we used for binary classification and presented the F1 scores in Table 8.

All pairs, except some KNN models, provide more than 99% F1 scores. Pearson correlation still found 33 features. We identified six, three, and five features by using filter methods, fisher score, mutual information, and ANOVA,

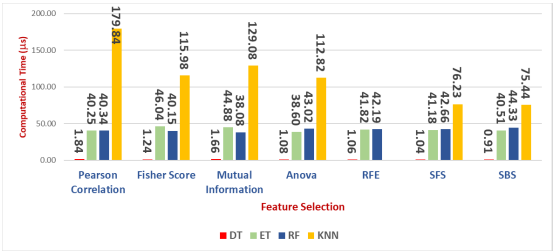


FIGURE 5. Computational time required to classify a sample using 3 class classification models in the N-BaIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.

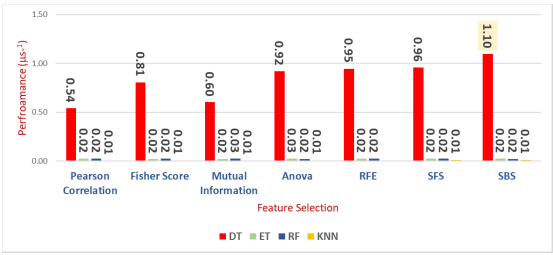


FIGURE 6. Performance achieved by 3-class classification models in the N-BaIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.

respectively, as shown in Table 4. The wrapping methods mostly selected three features (see Table 5).

Among all the feature selection methods, the DT and SBS pair again achieves the highest performance, as shown in Fig. 6. Anova is the best compared to other filter methods. Table 9 shows the detailed performance metrics for DT and three feature selection methods, Fisher Score, Mutual Information, and SBS. It is obvious that the detection performance is higher than 99% for all metrics.

The optimal feature set selected by the mutual information feature set is {MI_dir_L0.1_mean, MI_dir_L0.01_mean, H_L0.01_mean}, the set of Fisher Score is { MI_dir_L5_weight, MI_dir_L5_mean, MI_dir_L0.01_mean, MI_dir_L0.01_weight, H_L0.01_mean}.

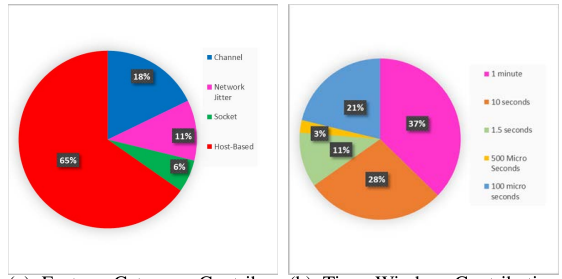
Compared to binary classification, we were unable to identify clear differences between the results. Learning models can easily identify the type of malware in this dataset. However, a small number of features, 3-5, achieve high detection rates regardless of the feature selection method. SBS and DT are the pair that performs best. The analysis of category distributions for the classification of 3 classes is given in Fig. 7. The results are very similar to those of binary classification. Host-based features have again played an essential role, and the time-window distribution does not show a distinct outcome.

3) 9-CLASS CLASSIFICATION

In the 9-class formulation, we consider eight different types of attack and benign as distinct categories, as presented in Table 2. The results of this classification are quite different from the results of the binary and 3-class classification with

TABLE 9. Accuracy, Precision, Recall, F1 of 3-class classification of the selected model with feature set-based performance over the N-BaIoT dataset.

Feature Selection	Feature Subset	Model	Class Name	Accuracy	Precision	Recall	F1-Score
Fisher Score	5	DT	mirai	0.99	0.99	0.99	0.99
			benign	0.99	0.99	0.99	0.99
			gafgyt	1	1	1	1
Mutual Information	3	DT	mirai	0.99	0.99	0.99	0.99
			benign	0.99	0.99	0.99	0.99
			gafgyt	1	1	1	1
SBS	3	DT	mirai	0.99	0.99	0.99	0.99
			benign	0.99	0.99	0.99	0.99
			gafgyt	0.99	0.99	1	0.99



(a) Feature Category Contribution to Overall Botnet detection (b) Time Window Contribution to overall botnet detection

FIGURE 7. Contribution of feature category and time window in the selected feature set for 3-class classification in the N-BaIoT dataset.

TABLE 10. F1 scores for 9-class classification models using feature subsets (see in Table 4&5) of feature selection algorithms in the N-BaIoT dataset.

		9-class N-BaIoT			
FS Method	Approach	DT	ET	RF	KNN
Filter	Pearson Correlation	0.9946	0.9949	0.9944	0.7051
	Fisher Score	0.9955	0.9954	0.9941	0.9611
	Mutual Information	0.9937	0.9948	0.9912	0.9453
	Anova	0.9956	0.9952	0.9925	0.9601
Wrapper	RFE	0.9943	0.9962	0.9967	
	SFS	0.9927	0.9941	0.9942	0.9928
	SBS	0.9944	0.9947	0.9940	0.8502

respect to filter methods, as the learning models with these selection methods require a very high number of features to achieve an F1 score greater than 99%. More specifically, 68, 28 and 59 features should be fed into the models when Fisher score, mutual information, and ANOVA methods are used, respectively. However, 33 features are identified as not highly correlated by the Pearson correlation method. Wrapper methods show very interesting results. Although RFE provides higher detection results using 20-28 features depending on the type of learning model, SFS and SBS achieved higher detection with only three features.

Table 10 shows the F1 scores achieved by the nine sets of classification features of the classes. Except for KNN, all

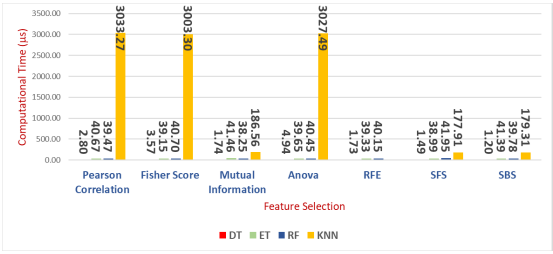


FIGURE 8. Computational Time required to classify a sample using 9-class classification models on the N-BaIoT dataset using feature sets (see Table 4 and 5) of feature selection methods.

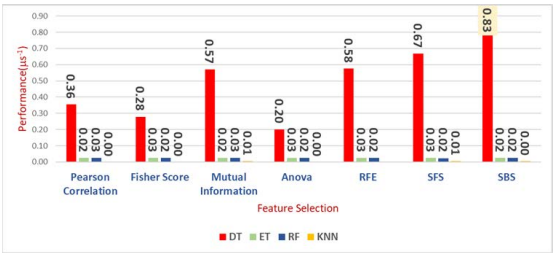


FIGURE 9. Performance achieved by 9-class classification models in the N-BaIoT dataset using feature sets (in Table 4 and 5) of feature selection methods.

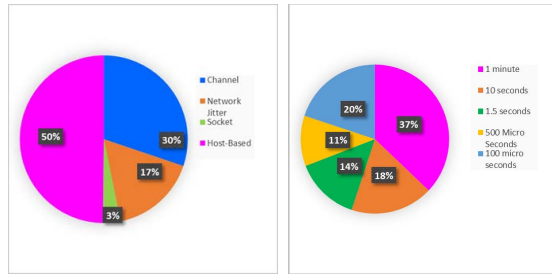
TABLE 11. Accuracy, Precision, Recall, F1 summary of classification of results mutual information and SBS features, DT with 28-feature set and 3-feature set respectively for 9-class classification over N-BaIoT dataset.

	Accuracy		Precision		Recall		F1-Score	
Class Name	MI	SBS	MI	SBS	MI	SBS	MI	SBS
Ack	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Benign	0.99	0.99	0.98	0.99	0.99	0.99	0.99	0.99
Combo	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Junk	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99
Scan	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Syn	0.99	1	1	0.99	0.99	0.99	0.99	0.99
TCP	1	1	0.98	0.98	0.99	0.99	0.99	0.99
UDP	0.98	0.98	0.99	0.99	0.98	0.98	0.98	0.98
UDPPLain	0.99	0.99	1	1	0.99	0.99	0.99	0.99

other models achieve more than 99% in all selection methods. The result of the overall performance metric indicates that SBS and DT are the best pair in the 9-class classification (see Fig. 9). Among the wrapper methods, DT and mutual information emerge as the leading performer.

3 features used by the SBS and DT pair are as follows: MI_dir_L0.01_mean, HH_L0.01_std, HH_jit_L0.01_mean.

Table 11 shows the detailed classification performance of the 9-class classification with mutual information based on the 28-feature set and the SBS with the 3-feature set (that is, DT is the learning model in both cases). Although the detection rates of some classes (e.g., junk accuracy, accuracy, recall and F1 UDP scores) decrease to 98%, the remaining metrics show figures equal to or greater than 99%.



(a) Feature Category (b) Time Window

FIGURE 10. Feature category and time window in each set of features for 9-class classification.

The frequency analysis of the feature categories shows that the host-based features are still the most important category for the 9-class classification (see Fig. 10). However, the selected features of the channel category are higher compared to the binary and 3-class formulations. The contribution of the network jitter category is also more important in this classification task. This means that learning models need to resort to other features, which provide statistics about network activities between hosts and time intervals between network packets to differentiate attack types. When many types of attack are considered, including various denial-of-service attacks, such features are instrumental in making a distinction between them. Time window analysis provides a similar distribution, except that lower time intervals (i.e. 1.5 seconds, 500 microseconds, and 100 microseconds) have closer distributions to each other.

4) THE STANDARD FEATURE SET FOR BINARY, 3-CLASS AND 9-CLASS CLASSIFICATIONS OVER N-BaIoT

In this part, our objective was to discover a feature set that provides high performance for all classification models induced with the N-BaIoT dataset. Here, we do not claim to obtain the feature set that has been proven to be the best for all formulations, but we show that a working set is possible. Intuitively, for this purpose, we have tested the best feature sets of each classification in the other classification tasks. The best feature set obtained from the 9-class classification provided high detection rates for the remaining binary and 3-classification tasks. However, we were unable to obtain such high results in the reverse situation where binary or 3-class classification features are applied to a 9-class formulation. More specifically, the feature set, {MI_dir _L0.01_mean, HH_L0.01_std, HH_jit_L0.01_mean} that is determined by the SBS and DT pair for the 9-class classification is utilized to induce models for all classification types, and we obtained the results given in Table 12. Except for the Junk and UDP classes in the 9-class formulation, all results are equal to or greater than 99%, demonstrating the effectiveness of this common set in all classification types.

TABLE 12. Classification results using the standard 3-Feature Set for all classification tasks in the N-BaIoT dataset.

Classification Type	Class Name	Accuracy				Precision				Recall				F1-Score			
		DT	RF	KNN	ET	DT	RF	KNN	ET	DT	RF	KNN	ET	DT	RF	KNN	ET
Binary	Benign	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	Attack	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
3-class	Mirai	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	Benign	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	Gafgyt	0.99	0.99	0.99	0.99	0.99	1	1	1	1	0.99	0.99	0.99	0.99	0.99	0.99	0.99
9-class	Ack	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	Benign	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	Combo	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
	Junk	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	1
	Scan	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1	0.99	1	0.99	0.99	0.99	1
	Syn	0.99	0.99	0.99	0.99	0.99	1	1	1	1	0.99	0.99	0.99	0.99	1	0.99	0.99
	TCP	0.99	0.99	0.99	0.99	0.99	1	1	1	0.99	0.99	1	0.99	0.99	0.99	1	0.99
	UDP	0.98	0.99	0.99	0.99	0.99	0.97	0.99	0.99	0.97	0.99	0.99	0.99	0.99	0.98	0.99	0.98
	UDPPLain	0.99	0.99	0.99	0.99	0.99	0.99	1	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

TABLE 13. Classification performance of sequential back- ward selection, DT With 7-feature set for binary classification over MedBIoT dataset.

Feature Selection Method	Feature Set	Model	Class Name	Accuracy	Precision	Recall	F1-Score
SBS	7	DT	Attack	0.99	0.99	0.99	0.99
			Benign	0.99	0.99	0.99	0.99

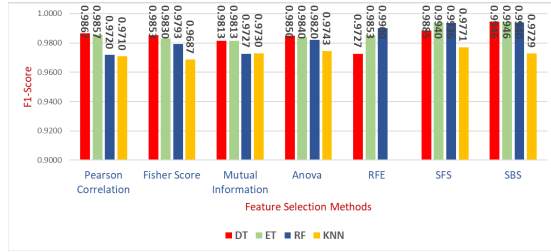


FIGURE 11. F1-scores for binary classification models in the MedBIoT dataset using feature subsets (see in Table 4&5) of feature selection algorithms.

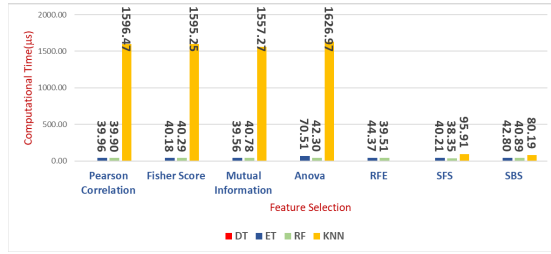


FIGURE 12. Computational time required to classify a sample by binary classification models over MedBIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.

B. MedBIoT

MedBIoT dataset has malicious network traffic from Mirai, BashLite, and Torii botnet malware, which were deployed on 83 real or emulated IoT devices. In this subsection, we report the experimental results of the binary, 3- and 4-class classification models induced with this data set (see Table 2 for the details of classification formulations).

1) BINARY CLASSIFICATION

We identified that 34 features are not highly correlated according to Pearson's correlation scores in the MedBIoT data set. A high number of features are required for filter methods to achieve a reasonable detection threshold rate equal to or above 98%. More specifically, ANOVA, Fisher Score, and Mutual Information can achieve that threshold rate with 85, 51, and 36, respectively, as shown in Table 4. On the other hand, RFE reaches the threshold value of 24-27 features depending on the type of learning model, while 7 features are enough for SBS and SFS (see Table 5). We present the F1 scores for all model and feature selection pairs in Fig. 11. Although the pairs do not exceed 98%, at least one learning model achieved this threshold for each feature selection method. In this data set and in the formulation of the problem, SBS still provides the best performance metric, as shown in Fig. 13. The results presented in Table 13 indicate that SBS achieves a score greater than 99% in all performance metrics.

7 features selected by the SBS and DT pair are as follows: {HH_L1_pcc, HH_L0.01_magnitude, HH_jit_L1_std, HH_jit_0.01_weight, HpHp_L1_pcc, HpHp_L0.01_weight, HpHp_L0.01_magnitude}. The distributions of the features according to the category of features and the duration of the time window are given in Fig. 14. When this feature set is compared to the selected feature sets in N-BaIoT, it is observed that the channel category is the dominant category instead of the host-based one. As MedBIoT covers malicious activities regarding the the C&C and formation phases of the botnet life cycle, the features that characterize host-to-host communications become more important. In contrast, N-BaIoT, which covers the attack phase, can discriminate malicious activities based on host-based features.

Similar to N-BaIoT, MedBIoT does not show any specific pattern on time periods, indicating whether longer or shorter periods are preferred. Although the longest period, 1 minute, provides more discriminative features among the others, still,

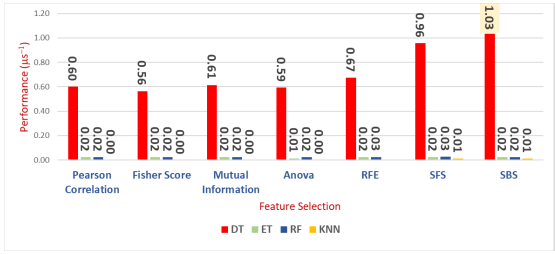
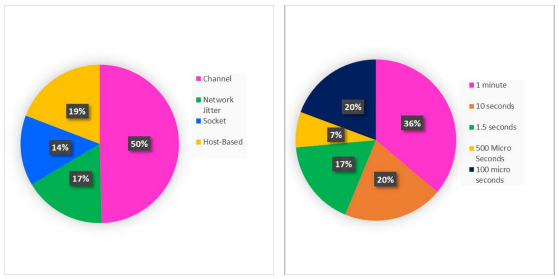


FIGURE 13. Performance achieved by binary classification models over the MedBIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.



(a) Feature Category (b) Time Window

FIGURE 14. Feature category and time window contribution in each feature set for binary classification over the MedBIoT datase.

the second-best category is 100 microseconds, which is the smallest one.

2) 3-CLASS CLASSIFICATION

The 3-class classification of the MedBIoT dataset aims to identify whether the instance that represents a portion of network traffic belongs to the spreading or C&C phases of a botnet life cycle. The third class in this formulation is benign traffic. Similarly to binary classification, filter methods require a greater number of features to achieve high detection rates. More specifically, features 42, 38 and 49 should be included by Fisher score, Mutual information, and Anova, respectively, to achieve 98% detection rate (see Fig. 15. Wrapper methods, SFS and SBS, identified a set with 7 features. On the other hand, RFE requires 24-27 features.

SBS and DT are still the best pair of models and feature selection methods, as shown in Fig. 15. This highest performance is obtained from the following feature set: {HH_L3_magnitude, HH_L0.01_weight, HH_L0.01_radius, HH_jit_L1_weight, HH_jit_L0.1_std, HpHp_L5_pcc, HpHp_L0.1_magnitude}. The detection results given in Fig. 15 indicate that it is possible to find learning models for each feature selection method that gives a performance greater than 99%.

Fig. 18 shows that channel-based features are more useful than other network categories to achieve the highest performance. Compared to binary classification, the ratios of

TABLE 14. Summary of the classification results with the selected model and feature sets based on the performance of the 3-class classification in the Med BIoT dataset.

Feature Selection	Class Name	Class Name	Accuracy	Precision	Recall	F1-Score
SBS	DT	Benign	0.99	0.98	0.99	0.99
		CC	0.99	0.99	0.99	0.99
		Spread	0.99	0.98	0.99	0.99

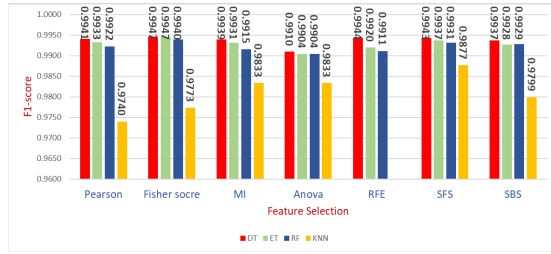


FIGURE 15. F1-scores for 3-class classification models in the MedBIoT dataset using feature subsets (see in Table 4&5) of feature selection algorithms.

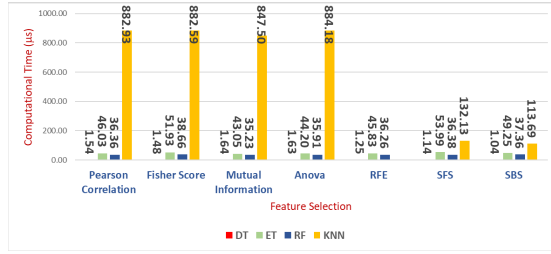


FIGURE 16. Computational Time required to classify a sample using 3-class classification models on MedBIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.

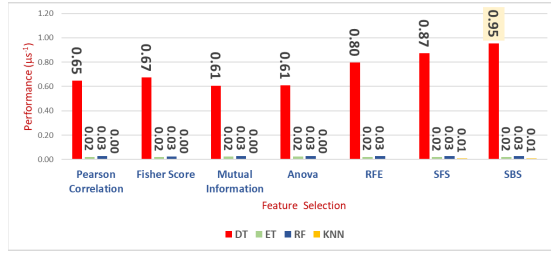
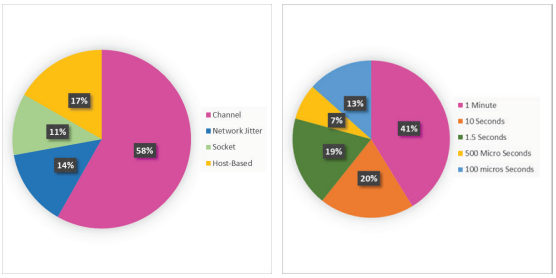


FIGURE 17. Performance achieved by 3-class classification models in the MedBIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.

channel features are more frequent. The time window results are similar to the binary classification outcome.

3) 4-CLASS CLASSIFICATION

In the 4-class classification, we consider the identification of the source malware that generates malicious traffic. Thus, the labels in this formulation are Mirai, Bash-Lite, Torii, and Benign. Fisher score, mutual information, and Anova require 46, 41 and 52 features, respectively



(a) Feature Category (b) Time Window

FIGURE 18. Feature category and time window contribution in each feature set for 3-class classification in MedBIoT dataset.

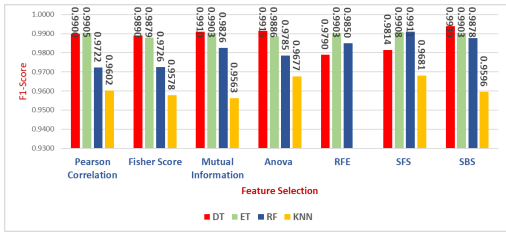


FIGURE 19. F1-scores for 4-class classification models in the MedBIoT dataset using feature subsets (see in Table 4&5) of feature selection algorithms.

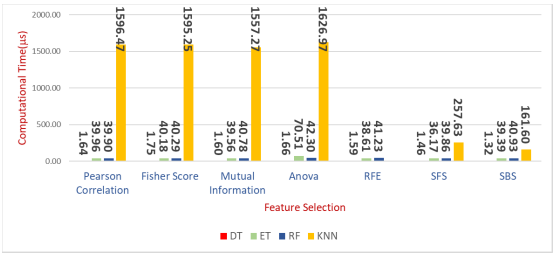


FIGURE 20. Computational Time required to classify a sample by 4-class classification models in the MedBIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.

(see Table 4). SBS and SFS methods with any learning model achieve a higher detection with 7 features, whereas the feature numbers within the range of 22-29 are sufficient in RFE. F1 score of 99% can be achieved by a learning model in each feature selection method, as shown in Fig. 19. SBS and DT are the best pair of performers and use the following feature set:{MI_dir_L0.1_weight, HH_L1_pcc, HH_L0.01_magnitude, HH_jit_L0.01_weight, HH_jit_L0.01_std, HpHp_L0.01_weight, HpHp_L0.01_std}. Fig. 22 shows that the channel category is the most important category.

4) STANDARD FEATURE SET FOR BINARY, 3-CLASS AND 4-CLASS CLASSIFICATION TASKS OVER MedBIoT DATASET

To find a standard feature set that works for binary, 3-class and 4-class classification problems in the MedBIoT data set, similar to the case of N-BaIoT, we tested the performance

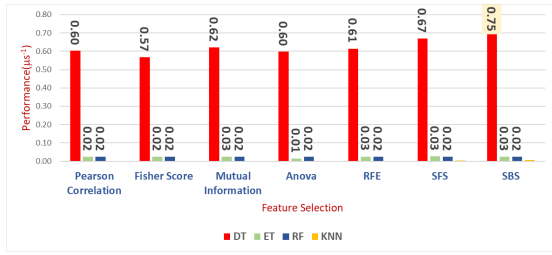
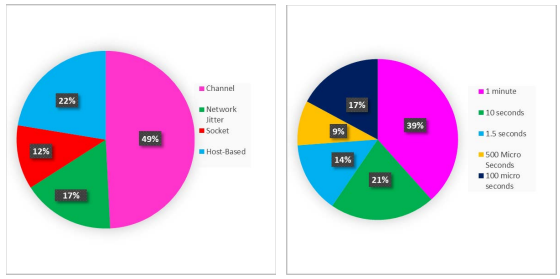


FIGURE 21. Performance achieved by 4-class classification models on the MedBIoT dataset using feature sets (see in Table 4 & 5) of feature selection methods.



(a) Feature Category (b) Time Window

FIGURE 22. Feature category and time window contribution in each feature set for 4-class classification in the MedBIoT dataset.

of the selected feature set of one classification on the other classification problem. We identified that the feature set of 4-class classification also works better in all other classifications, as shown in Table 15.

V. DISCUSSION

In this study, it is shown that all the machine learning problem formulations realized for the detection of IoT botnet attacks in two datasets, N-BaIoT and MedBIoT, achieved high detection performance in more than 99% with a limited number of features (i.e. 3 and 7 features).

In our experiments, we used various filter and wrapper methods for feature selection, in addition to four main machine learning methods to induce the models. In the case where we use filter methods, the results of feature selection are fed into the models. In wrapper methods, models are used directly for the assessment of feature subset alternatives. Performance evaluation was carried out based on the relationship between the F1 score and the computational time required to classify a sample. The wrapper method, SBS, with the DT model has achieved the most satisfactory trade-off between detection capacity and computational cost, exceeding the other alternative feature selection and learning model pairs.

Using feature selection approaches, tree-based models (DT, ET, and RF) achieved the best results in all classification types for both datasets, especially in multiclass classification types. k-NN classifier was not suitable for multiclass

TABLE 15. Summary of classification results using the standard 7-Feature Set for binary, 3-class and 4-class classification tasks in the MedBIoT dataset.

Classification Type	Class Name	Accuracy				Precision				Recall				F1-Score			
		DT	RF	KNN	ET	DT	RF	KNN	ET	DT	RF	KNN	ET	DT	RF	KNN	ET
Binary	Benign	0.99	0.99	0.96	0.99	0.99	0.99	0.95	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.96	0.99
	Attack	0.99	0.99	0.98	0.99	0.99	0.99	0.97	0.99	0.99	0.99	0.94	0.99	0.99	0.99	0.96	0.99
3-class	Benign	0.99	0.99	0.97	0.99	0.99	0.98	0.96	0.98	0.98	0.99	0.98	0.99	0.99	0.99	0.97	0.99
	CC	0.99	0.99	0.97	0.99	0.98	0.99	0.97	0.99	0.99	0.99	0.97	0.99	0.99	0.99	0.97	0.99
	Spread	0.99	0.98	0.97	0.99	0.99	0.99	0.98	0.99	0.99	0.98	0.96	0.99	0.99	0.98	0.97	0.99
4-class	Bashlite	0.99	0.99	0.94	0.99	1	1	0.97	0.99	0.99	0.99	0.94	0.98	0.99	0.99	0.96	0.99
	Benign	0.99	0.99	0.94	0.99	0.99	0.98	0.91	0.98	0.98	0.98	0.92	0.99	0.98	0.98	0.91	0.98
	Mirai	0.99	0.99	0.89	0.99	0.98	0.98	0.88	0.98	0.99	0.99	0.89	0.98	0.98	0.98	0.88	0.98
	Torii	0.99	0.99	0.98	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98	0.99	0.99	0.99	0.98	0.99

classification and also took the longest computational time to classify the sample compared to tree-based models.

However, there are some differences between the results of the MedBIoT and N-BaIoT data sets. The former requires seven features, whereas three features in the latter data set are enough for high detection rates. Compared to N-BIoT, which addresses the attack stage of the botnet lifecycle, MedBIoT differentiates post-attack and C&C phases. It can be argued that the detection at the attack stage would be relatively easier, as this stage is usually accomplished by sending an enormous number of packets (i.e., spam, packet flooding). Therefore, more features are needed for other attack stages.

On the other hand, we observed a remarkable difference between filter and wrapper methods in some classification formulations. High accuracy rates are achieved with more than 28 features with filter methods for 9-class classification with N-BaIoT and all classifications with MedBIoT. On the other hand, the wrapper methods, SFS and SBS, identify an optimal set with 3 and 7 features for the respective formulations. One interesting observation is that the wrapper method, RFE, demonstrates quite different results for these formulations when compared to the other wrapper methods, so that, similarly to filter methods, it demands a high number of features. RFE applies a greedy approach by evaluating each feature one by one. Despite the differences in the statistical approach, filter methods also evaluate features in a similar fashion, one by one; thus, more composite feature set evaluation of SFS and SBS provides remarkable results in our case.

Another significant finding is obtained by comparing the feature categories that are prioritized by the feature selection methods. We identified that host-based features are more influential for the N-BaIoT dataset, whereas channel-based features show a more discriminatory property for the MedBIoT dataset. As the latter data set focuses on the spreading and C&C activity of the IoT malware within the target network, statistical features that are derived by tracking which network node communicates with which other node help more discriminate the malicious activity from the benign one or determine the type of malicious activity.

We conducted additional experiments to demonstrate the influence of feature categories. For this purpose, we induced models with only the features of the corresponding categories and reported the F1 scores for the ET, RF, DT and kNN

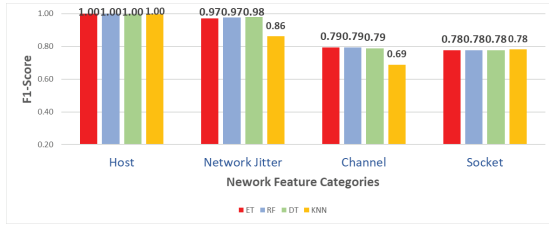


FIGURE 23. Comparison according to the feature categories - N-BaIoT dataset.

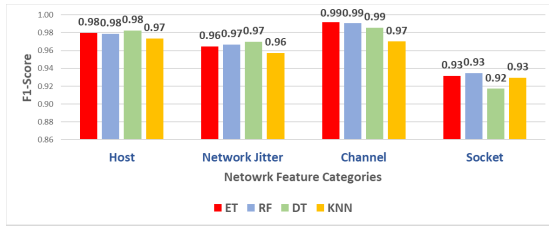


FIGURE 24. Comparison according to the feature categories - MedBIoT data set.

models. As shown in Figure 23, the use of all host features achieves a perfect model with a 1.00 F score, while network jitter would be helpful for higher rates for the N-BaIoT data set. However, the features of the channel category achieve 99% rates, and the host and network jitter categories would also be helpful for MedBIoT, as demonstrated in Figure 24.

Our results send a significant message to experts who design intrusion detection systems. The attacks originating from the bots (i.e., as simulated in the N-BaIoT dataset) can be easily detected by the sensors that track the incoming and outgoing packet statistics without considering the destination of the traffic. However, post-attack and C&C stages require the sensors to follow the sources and targets of traffic flows. Although some feature selection methods utilize the features of the socket category, the overall picture shows that the identification of receiving parties would be enough without using the source and destination ports.

Our comparison regarding the feature categories from the time interval perspective shows that the longest interval value,

TABLE 16. Comparison of selected feature counts and classification results with previous work.

DataSet	Classification Type	Feature Selection method	Number of features	Model	Accuracy	Precision	Recall	F1-score
N-BaIoT	9-class	LOGISTIC REGRESSION METHOD [21]	19	ANN	96.4	93.9	95.1	99.13
		Random forest [55]	40	XGB	99.96	NA	NA	99.94
		correlation based feature selection (CFS) [56]	75	LSTM	97.84	97.81	95	96.25
		In this paper	**3**	**DT**	**99.57**	**99.56**	**99.55**	**99.55**
MedBIoT	Binary	Chi-Squared [57]	20	DT	95.3	95	98	96
				RF	95.3	95	98	96
		Original MedBIoT [23]	NA	DT	93.15	94.48	93.15	92.93
		In this paper	**7**	**DT**	**99.34**	**99.33**	**99.32**	**99.34**
	4-class	Original MedBIoT [23]	NA	DT	95.16	95.84	95.16	94.99
		In this paper	**7**	**DT**	**99.41**	**99.36**	**99.38**	**99.46**

*NA -Not Applicable.

1 min, contributes more to the set with a higher discrimination property.

We also compared our proposal with the latest methods from recent models, and the results are summarized in Table 16. For the N-BaIoT dataset, the 9-class classification achieved better results with lower subsets of features than others. Abbasi *et al.* provided the 19 most important features for various attacks using LR(logistic regression) as feature selection. With these features, the ANN model performed well with 96.4% accuracy, 93.9% precision, 93.9% recall, 99.13% F1 score [21]. Parra *et al.* has created an LSTM model with the help of correlation-based feature selection to classify attacks and confirmed its model with 75 features achieved 97.84% precision, 97.81% precision, 95% Recall, 96.25% F1 score [56]. Faysal *et al.* proposed an XGBoost model that used 40 related features and stated that the model classified attacks with 99.96% accuracy and 99.94% F1 score [55].

Compared to existing models, the proposed framework achieved for botnet attack type classification in the N-baIoT dataset achieved good detection performance on the SBS-DT model (decision tree) with three features: precision of 99.57%, precision of 99.56%, recall of 99.55% and F1 score of 99.55%. The proposed methodology effectively distinguishes IoT botnet attacks from network traffic with high detection rates.

However, feature selection is applied less on the MedBIoT dataset. Gandhi and Li has proposed the decision tree, random forest models for binary classification, and selection of chi-square characteristics utilized. Twenty features used to detect malware type for DT and RF models with 99.3% precision, 95% precision, 98% recall, 96% recall. We also compared our detection rates with an original MedBIoT dataset.

Our proposed methodology achieved good detection performance for binary and 4-class classification in the MedBIoT dataset compared to the SBS-DT models. For binary classification, 99.34% precision, 99.33% precision, 99.32% recall, 99.34% f1 score. For 4-class classification, 99.41% precision, 99.36% precision, 99.38% recall, 99.46% f1 score. To maximize the classification performance of the learning model, a random search is used to determine the best set of hyperparameters for each classification formulation and is summarized in Table 17.

TABLE 17. SBS-DT optimal parameters for each classification in the dataset.

Dataset	Classification Type	Description	Best Parameter
N-BaIoT	Binary	Maximum Depth of the tree	38
		Minimum number of samples required to split an internal node	15
		the minimum number of samples required to be at a leaf node.	1
		the impurity of a split	Entropy
	3-class	Maximum Depth of the tree	19
		Minimum number of samples required to split an internal node	3
		the minimum number of samples required to be at a leaf node.	3
		the impurity of a split	Gini
	9-class	Maximum Depth of the tree	24
		Minimum number of samples required to split an internal node	3
		the minimum number of samples required to be at a leaf node.	2
		the impurity of a split	Entropy
MedBIoT	Binary	Maximumn Depth of the tree	47
		Minimum number of samples required to split an internal node	26
		the minimum number of samples required to be at a leaf node.	8
		the impurity of a split	Entropy
	3-class	Maximum Depth of the tree	24
		Minimum number of samples required to split an internal node	15
		the minimum number of samples required to be at a leaf node.	3
		the impurity of a split	Entropy
	4-class	Maximumn Depth of the tree	39
		Minimum number of samples required to split an internal node	5
		the minimum number of samples required to be at a leaf node.	1
		the impurity of a split	Gini

VI. CONCLUSION

Botnet attacks change the shape and volume to deplete the target resources on the entire IoT network system. Therefore, to mitigate the critical impact, a machine learning-based intrusion detection system is developed to accurately classify botnet attacks.

In this work, we propose a reduced set of features to detect and classify malicious activities of popular IoT botnet malware. We identified six different binary or multiclass classification problems using datasets, N-BaIoT and MedBIoT. We applied various filter and wrapper methods with four machine learning methods to these datasets. Finally, we derive an optimal set of features for each classification problem. To our knowledge, no detailed comparison between the optimal feature sets required for different classification problems of IoT botnet detection, which can vary depending on the stage of the botnet life cycle, has been done before.

We obtained very high detection rates for each classification problem with fewer features. The decision tree-based SBS takes less time to classify the samples with the highest detection rate. Wrapper methods, SFS and SBS, were effective in finding the optimal feature sets in each classification.

Filter methods provide suboptimal results in terms of feature numbers for 9-class classification with N-BaIoT and all classifications with MedBIoT. Host-based features are more instrumental in the detection rates for N-BaIoT, whereas channel features play a more important role for MedBIoT.

REFERENCES

[1] P. Srinivasulu, M. S. Babu, R. Venkat, and K. Rajesh, "Cloud service oriented architecture (CSoA) for agriculture through Internet of Things (IoT) and big data," in *Proc. IEEE Int. Conf. Electr., Instrum. Commun. Eng. (ICEICE)*, Apr. 2017, pp. 1–6.
[2] R. Shah and A. Chircu, "IoT and AI in healthcare: A systematic literature review," *Issues Inf. Syst.*, vol. 19, no. 3, pp. 1–9, 2018.
[3] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 3rd Quart., 2019.
[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
[5] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Botnet in DDoS attacks: Trends and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2242–2270, 4th Quart., 2015.
[6] G. K. Kishore and K. Rajesh, "Avoiding attacks using node position verification in mobile ad hoc networks," in *Next-Generation Networks*. Singapore: Springer, 2018, pp. 111–118.
[7] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, "Your botnet is my botnet: Analysis of a botnet takeover," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, 2009, pp. 635–647.
[8] R. C. Perkins, C. J. Howell, C. E. Dodge, G. W. Burruss, and D. Maimon, "Malicious spam distribution: A routine activities approach," *Deviant Behav.*, vol. 43, no. 2, pp. 196–212, Feb. 2022.
[9] J. Leonard, S. Xu, and R. Sandhu, "A framework for understanding botnets," in *Proc. Int. Conf. Availability, Rel. Secur.*, 2009, pp. 917–922.
[10] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.
[11] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. LISA*, vol. 99, Jun. 1999, pp. 229–238.
[12] E. Albin and N. C. Rowe, "A realistic experimental comparison of the Suricata and Snort intrusion-detection systems," in *Proc. 26th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Mar. 2012, pp. 122–127.
[13] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.
[14] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Comput. Surveys*, vol. 50, no. 6, pp. 1–45, Nov. 2018, doi: 10.1145/3136625.
[15] J. M. H. Jimenez and K. Goseva-Popstojanova, "The effect on network flows-based features and training set size on malware detection," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–9.
[16] H. Bahsi, S. Nomm, and F. B. La Torre, "Dimensionality reduction for machine learning based IoT botnet detection," in *Proc. 15th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2018, pp. 1857–1862.
[17] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Oct. 2014, pp. 247–255.
[18] F. V. Alejandre, N. C. Cortes, and E. A. Anaya, "Feature selection to detect botnets using machine learning algorithms," in *Proc. Int. Conf. Electron., Commun. Comput. (CONIELECOMP)*, 2017, pp. 1–7.
[19] A. Guerra-Manzanares, H. Bahsi, and S. Nomm, "Hybrid feature selection models for machine learning based botnet detection in IoT networks," in *Proc. Int. Conf. Cyberworlds (CW)*, Oct. 2019, pp. 324–327.
[20] F. Beer and U. Buhler, "Feature selection for flow-based intrusion detection using rough set theory," in *Proc. IEEE 14th Int. Conf. Netw., Sens. Control (ICNSC)*, May 2017, pp. 617–624.
[21] F. Abbasi, M. Naderan, and S. E. Alavi, "Anomaly detection in Internet of Things using feature selection and classification based on logistic regression and artificial neural network on N-BaIoT dataset," in *Proc. 5th Int. Conf. Internet Things Appl. (IoT)*, May 2021, pp. 1–7.
[22] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul./Sep. 2018.
[23] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network," in *Proc. 6th Int. Conf. Inf. Syst. Secur. Privacy*, 2020, pp. 207–218.
[24] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in *Proc. 16th Annu. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, Feb. 2008, pp. 235–252.
[25] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering analysis of network traffic for protocol-and structure-independent botnet detection," in *Proc. 17th Conf. Secur. Symp.*, CA, USA, 2008, pp. 139–154.
[26] M. S. Gadelrab, M. ElSheikh, M. A. Ghoneim, and M. Rashwan, "BotCap: Machine learning approach for botnet detection based on statistical features," *Int. J. Commun. Netw. Inf. Secur.*, vol. 10, no. 3, p. 563, Apr. 2022.
[27] M. Injadat, A. Moubayed, and A. Shami, "Detecting botnet attacks in IoT environments: An optimized machine learning approach," in *Proc. 32nd Int. Conf. Microelectron. (ICM)*, Dec. 2020, pp. 1–4.
[28] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.
[29] H.-T. Nguyen, Q.-D. Ngo, and V.-H. Le, "IoT botnet detection approach based on PSI graph and DGCNN classifier," in *Proc. IEEE Int. Conf. Inf. Commun. Signal Process. (ICICSP)*, Sep. 2018, pp. 118–122.
[30] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: A novel honeypot for revealing current IoT threats," *J. Inf. Process.*, vol. 24, no. 3, pp. 522–533, 2016.
[31] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *Proc. Int. Conf. Artif. Intell. Big Data (ICAIBD)*, May 2018, pp. 228–234.
[32] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.
[33] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, and H. Gacanin, "Hybrid deep learning for botnet attack detection in the Internet-of-Things networks," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4944–4956, Mar. 2021.
[34] M. Alauthman, N. Aslam, M. Al-kasassbeh, S. Khan, A. Al-Qerem, and K.-K. R. Choo, "An efficient reinforcement learning-based botnet detection approach," *J. Netw. Comput. Appl.*, vol. 150, Jan. 2020, Art. no. 102479.
[35] B. Rahbarinia, R. Perdisci, A. Lanzi, and K. Li, "PeerRush: Mining for unwanted P2P traffic," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Berlin, Germany: Springer, 2013, pp. 62–82.
[36] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Inf. Sci.*, vol. 278, pp. 488–497, Sep. 2014.
[37] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Jan. 2003.
[38] M. Alauthaman, N. Aslam, L. Zhang, R. Alasem, and M. A. Hossain, "A P2P botnet detection scheme based on decision tree and adaptive multilayer neural networks," *Neural Comput. Appl.*, vol. 29, no. 11, pp. 991–1004, 2018.
[39] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting P2P botnets through network behavior analysis and machine learning," in *Proc. 9th Annu. Int. Conf. Privacy, Secur. Trust*, Jul. 2011, pp. 174–180.
[40] J. Velasco-Mata, V. Gonzalez-Castro, E. F. Fernandez, and E. Alegre, "Efficient detection of botnet traffic by features selection and decision trees," *IEEE Access*, vol. 9, pp. 120567–120579, 2021.
[41] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Comput. Secur.*, vol. 45, pp. 100–123, Sep. 2014.
[42] P. Narang, J. M. Reddy, and C. Hota, "Feature selection for detection of peer-to-peer botnet traffic," in *Proc. 6th ACM India Comput. Conv.*, 2013, pp. 1–9.
[43] A. Pektaş and T. Acarman, "Effective feature selection for botnet detection based on network flow analysis," in *Proc. Int. Conf. Automatics Informat.*, 2017, pp. 1–4.
[44] C. Aggarwal, *Data Mining*. Cham, Switzerland: Springer, 2015.
[45] Q. Gu, Z. Li, and J. Han, "Generalized Fisher score for feature selection," 2012, *arXiv:1202.3725*.

[46] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 189–201, Feb. 2009.
[47] S. Beniwal and J. Arora, "Classification and feature selection techniques in data mining," *Int. J. Eng. Res. Technol.*, vol. 1, no. 6, pp. 1–6, 2012.
[48] M. ShakilPervez and D. Md. Farid, "Literature review of feature selection for mining tasks," *Int. J. Comput. Appl.*, vol. 116, no. 21, pp. 30–33, Apr. 2015.
[49] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 389–422, 2002.
[50] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern classifiers," *Pattern Recognit.*, vol. 33, no. 1, pp. 25–41, 2000.
[51] Y. Sasaki, "The truth of the F-measure," Univ. Manchester, Manchester, U.K., Lect. Notes, Oct. 2007. [Online]. Available: https://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf
[52] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *Proc. AAAI Workshop Learn. Text Categorization*, 1998, vol. 752, no. 1, pp. 41–48.
[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and M. Blondel, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Jan. 2011.
[54] S. Raschka, "MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack," *J. Open Source Softw.*, vol. 3, no. 24, p. 638, Apr. 2018.
[55] J. A. Faysal, S. T. Mostafa, J. S. Tamanna, K. M. Mumenin, M. M. Arifin, M. A. Awal, A. Shome, and S. S. Mostafa, "XGB-RF: A hybrid machine learning approach for IoT intrusion detection," *Telecom*, vol. 3, no. 1, pp. 52–69, Jan. 2022.
[56] G. De La Torre Parra, P. Rad, K.-K.-R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102662.
[57] R. Gandhi and Y. Li, "Comparing machine learning and deep learning for IoT botnet detection," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Aug. 2021, pp. 234–239.



RAJESH KALAKOTI (Graduate Student Member, IEEE) received the master's degree in information technology from JNTUK, Kakinada. He is currently pursuing the Ph.D. degree with the Tallinn University of Technology. He worked as a Lecturer of information technology and a Software Engineer in India. He is also an Early Stage Researcher of XAI in cyber security domain with the Tallinn University of Technology. He has (co)authored three publications in conference proceedings. His research interests include the IoT security, network traffic, malware, HTTP-based botnet, machine learning, and XAI. He is a member of the IEEE Computer Society.



SVEN NÕMM received the Ph.D. degree jointly from the Tallinn University of Technology (TalTech), Estonia, and the École Centrale de Nantes et L'Université de Nantes, France, in 2004. He is currently a Senior Research Scientist at the Department of Software Science, TalTech. He has published more than 100 articles in scientific. His research interests include human–machine interaction, analysis of human motions, and applications of AI to the problems of cybersecurity and geoscience.



HAYRETDIN BAHSI received the M.Sc. degree in computer engineering from Bilkent University and the Ph.D. degree in computer engineering from Sabanci University. He is currently a Research Professor at the Center for Digital Forensics and Cyber Security, Tallinn University of Technology, Tallinn, Estonia. His research interests include cyberphysical system security and the application of machine learning methods to cybersecurity problems.

Appendix 2

II

M. U. Rehman, R. Kalakoti, and H. Bahşi. Comprehensive feature selection for machine learning-based intrusion detection in healthcare iomt networks. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 2: ICISSP*, pages 248–259. INSTICC, SciTePress, 2025

Comprehensive Feature Selection for Machine Learning-Based Intrusion Detection in Healthcare IoMT Networks

Muaan Ur Rehman[1][a], Rajesh Kalakoti[1][b] and Hayretdin Bahşi[1,2][c]
[1]*Department of Software Science, Tallinn University of Technology, Tallinn, Estonia*
[2]*School of Informatics, Computing, and Cyber Systems, Northern Arizona University, U.S.A.*
{*muaan.ur, rajesh.kalakoti, hayretdin.bahsi*}*@taltech.ee*

Keywords: Feature Selection, Intrusion Detection, Machine Learning, Internet of Medical Things.

Abstract: The rapid growth of the Internet of Medical Things (IoMT) has increased the vulnerability of healthcare networks to cyberattacks. While Machine learning (ML) techniques can effectively detect these threats, their success depends on the quality and quantity of features used for training to improve detection efficiency in IoMT environments, which are typically resource-constrained. In this paper, we aim to identify the best-performing feature sets for IoMT networks, as measured by classification performance metrics such as F1-score and accuracy, while considering the trade-offs between resource requirements and detection effectiveness. We applied an ML workflow that benchmarks various filter-based feature selection methods for ML-based intrusion detection. To test and train our binary and multi-class models, we used two well-developed IoMT datasets (CICIoMT2024 and IoMT-TrafficData). We applied filter-based feature reduction techniques (Fisher Score, Mutual Information, and Information Gain) for different machine learning models, i.e., Extreme Gradient Boosting (XGBoost), K-Nearest Neighbors (KNN), Decision Tree (DT), and Random Forest (RF). Our study demonstrates that 3-4 features can achieve optimal F1-score and accuracy in binary classification, whereas 7-8 features give reasonable performance in most of the multi-class classification tasks across both datasets. The combination of Information Gain and XGBoost with 15 features provides excellent results in binary and multi-class classification settings. Key features—protocol types, traffic metrics, temporal patterns, and statistical measures—are essential for accurate IoMT attack classification.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The Internet of Medical Things (IoMT) is an interconnected network of sensors, wearable and medical devices, and clinical systems, enabling applications like remote monitoring, fitness tracking, chronic disease management, and elderly care while enhancing treatment quality, lowering costs, and facilitating prompt responses (Islam et al., 2015),(Dimitrov, 2016).

The security of IoMT is very crucial due to its role in healthcare, where sensitive patient data and critical medical systems are increasingly interconnected. IoMT devices are often targets of cyberattacks, posing risks to patient safety and data privacy (Kondeti and Bahsi, 2024). Intrusion detection systems (IDS) are essential to monitor and detect malicious activities, ensuring the reliability and security of these networks. Machine learning (ML) is vital for IDS in IoMT as it can identify complex attack patterns and adapt to evolving threats. However, IoMT devices have limited computational resources, making it essential to reduce data dimensions and select the most relevant features to ensure that ML-based IDS operates efficiently and effectively without overburdening the network. We applied filter-based feature reduction techniques (Fisher Score, Mutual Information, and Information Gain) for different machine learning models, i.e., XGBoost, KNN, Decision Tree, and Random Forest. present an analysis by utilising two benchmarking IoMT datasets CICIoMT2024 (Dadkhah et al., 2024) and IoMT-TrafficData (Areia et al., 2024) for training and testing our models. We applied filter-based feature reduction techniques (Fisher Score, Mutual Information, and Information Gain) for different machine learning models, i.e., XGBoost, KNN, Decision Tree, and Random Forest.

We evaluate the proposed model in terms of F1 score by focusing on both binary classification and multiclassification. Binary classification aims to dis-

[a] https://orcid.org/0009-0000-2656-0127
[b] https://orcid.org/0000-0001-7390-8034
[c] https://orcid.org/0000-0001-8882-4095

Rehman, M. U., Kalakoti, R. and Bahşi, H.
Comprehensive Feature Selection for Machine Learning-Based Intrusion Detection in Healthcare IoMT Networks.
DOI: 10.5220/0013313600003899
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 2*, pages 248-259
ISBN: 978-989-758-735-1; ISSN: 2184-4356
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

tinguish between benign and malicious traffic, providing a high-level detection mechanism, while multi-class classification goes further by categorizing traffic into specific attack types, enabling a granular understanding of threats. The CICIoMT2024 dataset includes traffic data for 18 types of cyberattacks (19 classes including benign traffic) grouped into five main categories (6 classes): DDoS, DoS, Reconnaissance, MQTT, and Spoofing. Similarly, the IoMT-TrafficData dataset comprises eight distinct cyberattack types, including Denial of Service, ARP Spoofing, and Network Scanning, alongside benign traffic, resulting in a 9-class classification problem.

Additionally, we evaluated the classification performance (accuracy, precision, recall, and F1) of the best-performing model, XGBoost, on both datasets, utilizing the top 15 features identified through the Information Gain (IG) feature selection. Furthermore, to address these security challenges, this study examines key network features within both datasets that are essential for identifying and classifying cyber-attacks in IoMT. Both datasets use network flow features extracted from benign and malicious traffic. Specifically, we focus on features, such as protocol type, traffic volume metrics, temporal patterns, and statistical attributes, in network flows to understand their role in distinguishing normal and attack traffic patterns.

There exists a line of research on feature selection for ML-based intrusion detection in IoT devices (Kalakoti et al., 2022; Bahşi et al., 2018). However, these studies present benchmarking results for IoT networks that include consumer IoT devices. It is necessary to understand the impact of feature selection and the best-performing features in IoMT networks, as benign traffic profiles and system components in these networks have distinct properties when compared to other IoT devices.

By highlighting critical features across IoMT datasets, this study contributes to more robust, feature-driven methods for accurate anomaly and attack detection in IoMT environments, ultimately aiming to strengthen the security and reliability of these healthcare networks. The uniqueness of our work is that we have conducted a cross-analysis between two well-developed datasets, which were released recently, to obtain more generalized findings regarding the best-performing features in IoMT networks. Our work puts a particular emphasis on feature selection in multi-class classification settings, which has not been elaborated well in the literature.

This paper is structured as follows. Section 2 reviews the related research. Section 3 presents the methodology used in our feature selection process. In Section 4, we show and discuss our results. Finally, Section 5 concludes the paper and discusses future directions.

2 RELATED WORK

In the literature, various papers employ different feature selection techniques for machine learning-based attack classification. Some studies have adopted a filter approach to identify the best feature subsets, while others have applied wrapper or embedded methods. A few works combined both filter and wrapper techniques to determine the optimal feature set. This section provides a comprehensive review of the state-of-the-art methods for feature selection in machine learning-based intrusion detection systems, as reported in the literature. In (Khammassi and Krichen, 2017), a Genetic Algorithm (GA) combined with a Logistic Regression (LR) wrapper was applied to the UNSW-NB15 and KDDCup99 datasets. Using 20 features from UNSW-NB15, the GA-LR method with a Decision Tree (DT) classifier achieved 81.42% accuracy and a false alarm rate (FAR) of 6.39%. For KDDCup99, it achieved 99.90% accuracy with 18 features. In (Osanaiye et al., 2016), a filter-based approach using Information Gain, Chi-Square, and Relief was applied for Distributed Denial of Service (DDoS) detection on the NSL-KDD dataset. Using 13 features, the DT classifier reached 99.67% accuracy and a FAR of 0.42%. The work in (Ambusaidi et al., 2016) introduced a filter-inspired reduction approach with Flexible Mutual Information (FMI) and Least Square SVM (LS-SVM), achieving 99.94% accuracy on NSL-KDD with 18 features. In (Ingre and Yadav, 2015), a filter-based feature reduction method for IDS using correlation and DT was applied to NSL-KDD, reducing the feature set to 14 attributes and achieving 83.66% accuracy for multiclass classification. In (Alazzam et al., 2020), the Pigeon Inspired Optimizer (PIO) was used for feature reduction on multiple datasets. The Sigmoid and Cosine PIO methods selected features with accuracy rates between 86.9% and 96.0%.

Janarthanan and Zargari (Janarthanan and Zargari, 2017) implemented various feature selection algorithms on UNSW-NB15, selecting optimal subsets of 5 and 8 features. Using Random Forest (RF), they achieved up to 81.62% accuracy. Vikash and Ditipriya (Kumar et al., 2020) applied Information Gain for feature reduction on UNSW-NB15, selecting 22 attributes, and their IDS achieved 57.01% Attack Accuracy (AAc) and 90% F-Measure. In (Almomani, 2020), PSO, Firefly, Grey Wolf Optimization (GO), and GA were used on UNSW-NB15, with

a 30-feature subset yielding 90.48% accuracy with the J48 classifier. Maajid and Nalina (Khan et al., 2020) used Random Forest (RF) to rank features on UNSW-NB15, selecting 11 attributes, with RF achieving 75.56% accuracy. In (Tama et al., 2019), a two-stage model combining PSO, GA, and Ant Colony Optimization (ACO) on UNSW-NB15 selected 19 features, achieving 91.27% accuracy. Some studies have also used feature selection methods prior to applying explainable techniques in IoT botnet detection problems(Kalakoti et al., 2024a; Kalakoti et al., 2024c; Kalakoti et al., 2024b; Kalakoti et al., 2023).

Zong et al. (Zong et al., 2018) proposed a two-stage model using Information Gain (IG) for feature selection on UNSW-NB15, achieving 85.78% accuracy. In (Kasongo and Sun, 2020), the authors applied a filter-based feature selection technique by utilizing the XGboost algorithm on the UNSW-NB15 intrusion detection dataset. The results illustrate that feature selection method based on XGBoost enables models like DT to improve test accuracy from 88.13% to 90.85% in the binary classification.

The domain of intrusion detection systems (IDS) within the Internet of Medical Things (IoMT) has attracted considerable attention in recent years due to the growing adoption of IoMT devices in healthcare systems. To protect the security and privacy of sensitive medical data, developing effective IDS is essential. While many researcheres have focused on IDS for traditional networks, there is a notable lack of studies dedicated to IDS for the IoMT (Alalhareth and Hong, 2023a).

Feature selection techniques are crucial for enhancing the performance of IDS in the Internet of Medical Things (IoMT) (Rbah et al., 2022),(Khalil et al., 2022). These techniques reduce the dimensionality of input features while retaining essential information (Wagan et al., 2023). Filter-based methods, like chi-square and Information Gain, evaluate features individually based on their contribution to the target variable (Awotunde et al., 2021). Wrapper-based methods, such as recursive feature elimination (RFE), use ML algorithms to iteratively select and remove features, assessing their impact on model performance.

Information theory-based feature selection methods, such as MIFS and MRMR, are commonly used in fields like intrusion detection for the Internet of Medical Things (IoMT) (Gökdemir and Calhan, 2022). However, these methods require large datasets to accurately estimate Mutual Information between features and the target variable, and limited data can lead to suboptimal results (Chaganti et al., 2022). Solutions to this issue include data augmentation techniques, like oversampling or synthetic data generation (Parimala and Kayalvizhi, 2021), and transfer learning, which applies knowledge from data-rich domains to improve performance in data-limited contexts (Awotunde et al., 2021). However, these approaches come with challenges, such as introducing bias or noise and increasing computational costs (Al-Sarem et al., 2021).

In (Alalhareth and Hong, 2023b) authors proposed an improved Mutual Information feature selection technique for IDS for the IoMT. This paper proposes a Logistic Redundancy Coefficient Gradual Upweighting MIFS (LRGU-MIFS) to enhance feature selection for IDS in the IoMT. LRGU-MIFS improves detection accuracy by addressing overfitting and non-linear feature redundancy, outperforming existing methods in identifying key features.

State-of-the-art IDS systems for IoMT, such as deep learning models, offer high accuracy but are computationally intensive and less adaptable to resource-constrained environments. In contrast, our integration of feature selection techniques with IDS significantly reduces computational overhead, enhancing suitability for IoMT applications. The studies on feature selection do not create or compare the optimal sets achievable for different multiclass problem formulations. They only focus on one dataset and derive conclusions. This paper addresses this gap by inducing various learning models, including various multi-class classification models, for two well-developed and comprehensive IoMT datasets (Dadkhah et al., 2024; Areia et al., 2024) released recently. These datasets contain a huge number of attack types, making them convenient for multi-class classification. This study also conducts a cross-analysis between two datasets to identify the commonalities.

3 METHODOLOGY

We applied an ML workflow that includes the stages, data preprocessing, feature selection, and model training/testing, as demonstrated in Figure 1. In the data pre-processing stage, we eliminated the correlated features using Pearson Correlation. We applied filter-based feature selection methods (i.e., Fisher Score, Information Gain, Mutual Information) to prioritize the features. In the last stage, we benchmarked various ML algorithms (i.e., k-NN, Decision Tree, XGBoost, Random Forest) with varying numbers of selected best features determined by filter-based selection methods.

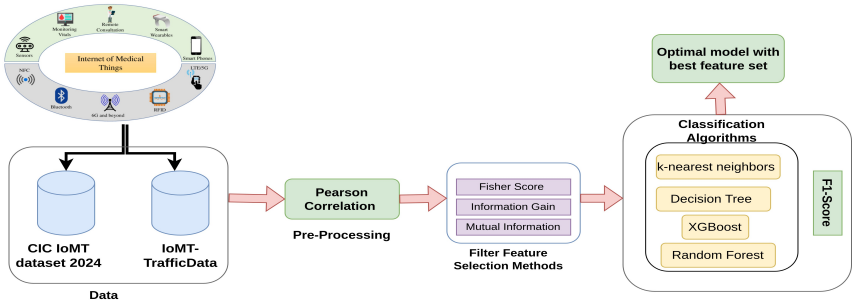


Figure 1: We employed filter methods for feature selection on the CICIoMT2024 Dataset (Dadkhah et al., 2024) and IoMT-TrafficData (Areia et al., 2024) to identify optimal features in IoMT networks. Four classifiers were used for evaluation: Decision Trees (DT), Random Forest (RF), k-Nearest Neighbors (k-NN), and XGBoost.

3.1 Datasets

We apply feature selection to the CICIoMTDataset 2024 dataset (Dadkhah et al., 2024) and IoMT-TrafficData (Areia et al., 2024), which focus on Internet of Medical Things devices in the healthcare sector. These datasets are designed to assess and improve the cybersecurity of IoMT devices through intrusion detection systems.

The CICIoMT2024 dataset (Dadkhah et al., 2024) includes traffic generated from 40 devices (25 real, 15 simulated) across multiple protocols like Wi-Fi, MQTT, and Bluetooth. The authors simulated 18 cyberattacks, categorized into five main categories i.e. DDoS, DoS, Recon, MQTT, and Spoofing.The features extracted from the attacks in the CICIoMT2024 dataset include Header Length, Duration, Rate, Srate, fin flag number, syn flag number, rst flag number, psh flag number, ack flag number, ece flag number, cwr flag number, syn count, ack count, fin count, rst count, IGMP, HTTPS, HTTP, Telnet, DNS, SMTP, SSH, IRC, TCP, UDP, DHCP, ARP, ICMP, IPv, LLC, Tot sum, Min, Max, AVG, Std, Tot size, IAT, Number, Radius, Magnitude, Variance, Covariance, Weight, and Protocol Type.

The IoMT-TrafficData dataset (Areia et al., 2024) is a comprehensive collection of network traffic data. It includes both benign and malicious traffic generated from eight different types of cyberattacks i.e. Denial of Service (DoS), Distributed Denial of Service (DDoS), ARP Spoofing, CAM Table Overflow, MQTT Malaria, Network Scanning, Bluetooth Reconnaissance, and Bluetooth Injection. The identified key features in the IP-based flows in the IoMT-TrafficData dataset cover various aspects of network communication. Protocol features include proto and service, which identify the transport and application protocols in use. Payload and packet metrics such as orig_bytes, resp_bytes, orig_pkts, and resp_pkts detail the volume and direction of data exchanged. Flow characteristics, including flow_duration and history, capture the overall session duration and connection state transitions. Packet directionality is covered by fwd_pkts_tot and bwd_pkts_tot for packet counts, and by fwd_pkts_payload and bwd_pkts_payload for payload bytes in each direction. Rate metrics (fwd_pkts_per_sec, bwd_pkts_per_sec, and flow_pkts_per_sec) provide packet transmission rates, while inter-arrival time features (fwd_iat, bwd_iat, and flow_iat) and active duration (active) reflect timing characteristics within the flow.

In this work, we used person correlation as the preprocessing step. The Pearson correlation coefficient, given by the equation (1) ,is used to compute the linear correlation between two variables. This technique involves calculating the collinearity matrix for all features to identify redundancy. The Pearson correlation coefficient P ranges from -1 to 1, where $P = 1$ indicates perfect positive correlation, $P = 0$ indicates no correlation, and $P = -1$ indicates perfect negative correlation. The formula for Pearson's correlation is:

$$P = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \mu_y)^2}} \quad (1)$$

Here, μ_x and μ_y represent the means of features x and y, respectively. Greater absolute values of P indicate a stronger linear relationship between the features.

3.2 Feature Selection Methods

Irrelevant features for classification problems are reduced to decrease the running time and improve the classification accuracy of machine learning algorithms. Feature selection methods are divided into three categories: wrapper, filter, and embedded techniques (Jović et al., 2015). Wrapper methods iteratively evaluate subsets of features using a machine

learning algorithm, but they can be computationally intensive for high-dimensional data. In contrast, filter methods rank features independently of the learning algorithm, which may result in suboptimal selections due to the lack of guidance. To reduce computational complexity, we opted for filter-based methods, which are highly efficient and well-suited for resource-constrained IoMT environments. The following three primary filter-based feature methods are commonly employed for numeric-based feature interclass and intra-class separation analysis and entropy-based methods as described below.

3.2.1 Fisher Score

The Fisher Score, also known as Fisher's ratio, measures the ratio of inter-class separation to intra-class separation for numeric features (Gu et al., 2012). The Fisher Score F_s is formally defined in equation 2 as:

$$F_s = \frac{\sum_{j=1}^{K} p_j(\mu_{ij} - \mu_i)^2}{\sum_{j=1}^{K} p_j \sigma_{ij}^2} \quad (2)$$

Where μ_{ij} and σ_{ij} represent the mean and standard deviation of the j-th class and i-th feature, while p_j denotes the proportion of data points in class j. A higher Fisher Score indicates greater discriminative power of a feature.

3.2.2 Mutual Information

Mutual Information (MI) quantifies the dependency between variables (Estévez et al., 2009). For continuous variables, MI is defined as:

$$I(X,Y) = \int \int p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \quad (3)$$

For discrete variables, MI is given by:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \quad (4)$$

Here, $p(x,y)$ is the joint probability, and $p(x)$, $p(y)$ are the marginal probabilities. MI values range as follows:

$$0 \leq I(X;Y) \leq \min\{H(X), H(Y)\}$$

To enhance the Mutual Information feature selection, the following goal function is used:

$$G = I(C;f_i) - \frac{1}{|S|} \sum_{f_s \in S} NI(f_i;f_s) \quad (5)$$

Where $I(C;f_i)$ is the Mutual Information between class C and feature f_i, and S is the set of selected features. The algorithm selects features by maximizing this measure. Function $NI(f_i;f_s)$ is the Normalized Mutual Information between features f_i and f_s.

3.2.3 Information Gain

Information Gain helps quantify how much information a feature contributes to classification by utilizing the concept of entropy. It measures the reduction in dataset entropy after knowing the values of a particular feature (Velasco-Mata et al., 2021). The initial entropy of the dataset, $H(X)$, is given by the following equation, which is based on the probability $p(x)$ of a sample belonging to class x. The conditional entropy, $H(X|Y)$, after knowing the values of feature Y, is defined based on the probability $p(y)$ of a sample having feature value $y \in Y$, and the probability $p(x|y)$ of a class x sample having feature value $y \in Y$.

$$H(X) = -\sum_{x=1}^{X} p(x) \log(p(x)) \quad (6)$$

$$H(X|Y) = -\sum_{y} p(y) \sum_{x} p(x|y) \log(p(x|y)) \quad (7)$$

3.3 Machine Learning Work Flow

In our study, we employed four machine learning algorithms for classifying cyberattacks in IoMT network flow data: Decision Tree (DT), Random Forest (RF), XGBoost (XGB) and K-Nearest Neighbors (K-NN). Decision Tree (DT) is a non-parametric supervised method for classification and regression. DTs classify data by evaluating attributes at each node until reaching a decision. Random Forest (RF) is an ensemble method of decision trees, chosen for its robustness, ability to manage complex datasets, and compatibility with diverse features. XGBoost, a gradient-boosting algorithm, optimizes using second-order gradients and applies L1/L2 regularization to reduce overfitting and enhance performance. Its efficiency, interpretability, and scalability make it ideal for large datasets. Lastly, K-Nearest Neighbors (KNN) is a distance-based algorithm for classification and regression.

Our classification models were evaluated for IoMT attack detection using confusion matrices for both binary and multi-class classification. For binary classification, True positives(TP) (correctly classified attacks), True negatives (TN) (correctly classified benign traffic), False negatives (FN) (misclassified attacks), and False Positives (FP) (misclassified benign traffic) were recorded. In this study, we have utilized the F1 score metric to evaluate distinct subsets of features. The F1 score is defined as the harmonic mean of precision (P) and recall (R). It provides a more appropriate measure of incorrectly classified cases compared to accuracy. We have employed the harmonic

mean of the F1 score, as it penalizes extreme values.

$$\text{F1 score} = \frac{2 \times \text{P} \times \text{R}}{\text{P} + \text{R}} \tag{8}$$

To train the models in binary classification, we have taken 5,000 samples of each class label. This results in a total of 10,000 samples from two labels. On the other hand, for the multi-class classification involving different distinct classes, we ensured an equal number of samples from each label, even for the classes with fewer instances, to maintain a balanced representation across all attack types. In the preprocessing step, after applying the Pearson correlation, balanced samples were drawn from the dataset of interest. Then, the datasets were divided into training and testing subsets in an 80/20 ratio. For evaluating each feature set with models, Random Search hyperparameter tuning was used for training the classification algorithms.

4 RESULTS AND DISCUSSIONS

This study analyzed the discriminatory power of network traffic flow features using filter-based feature selection techniques, including Fisher Score, Mutual Information, and Information Gain, for a machine learning-based intrusion detection function in IoMT healthcare networks. The analysis was conducted for binary and multiclass classification tasks on the CICIoMT2024 and IoMT-TrafficData datasets.

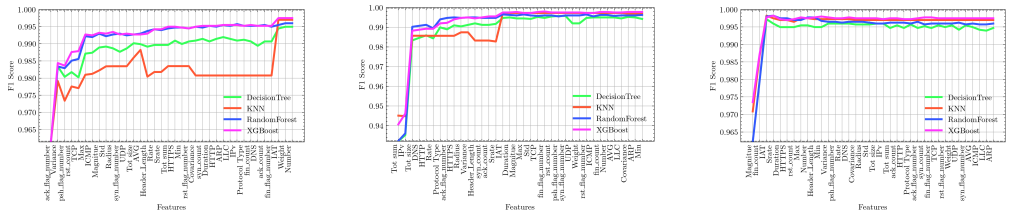
First, we applied Pearson's linear correlation coefficient (r) as a data preprocessing step to remove redundant and irrelevant data features. Any feature highly correlated with another feature ($|r| > 0.80$) was removed, keeping only one. As a result, out of the initial set of 44 features used to describe each sample in the dataset, 36 features remained in the final feature set. After removing the Pearson co-related features, in IoMT-Traffic dataset, we get 21 features, however, we also removed is_attack feature as it represent binary label. The final feature list contains 20 features.

After applying Pearson correlation and excluding unnecessary features, we applied three filter-based feature selection methods, i.e., Fisher Score, Mutual Information, and Information Gain. These methods were used to rank the importance of the remaining reduced features. An iterative, stepwise approach was used to train the ML models for each filter-based feature selection method (Fisher Score, Mutual Information, and Information Gain). Starting with the highest-ranked feature, we added one feature at a time, trained the model, and evaluated its performance progressively. For example, if the features were ranked as $f = \{f_1, f_2..f_n\}$, the model was first trained using only the top-ranked feature subset $\{f_1\}$, followed by training with $\{f_1$ and $f_2\}$, then with $\{f_1, f_2, \ldots f_n\}$ This process was repeated for all (n) ranked features in each method for both datasets. At each step, we added the next highest-ranked feature, as determined by the feature selection method, to the feature set incrementally to assess its impact on the model performance. The performance classifiers—Decision Tree (DT), Random Forest (RF), K-Nearest Neighbor (KNN), and XGBoost (XGB)—were evaluated based on the F1 score for both binary and multiclass classification tasks.

From the CICIoMT-2024 dataset, Binary classification was used to differentiate between benign and attack traffic. Two types of studies were performed for multi-class classification: category-based and attack-based classification. In the category-based classification, we identified six categories of network traffic: benign, MQTT attacks, DDoS, DoS, Reconnaissance, and ARP spoofing attacks, referred to as the 6-class classification. In the attack-based classification, there were 19 classes, which included various attack types such as ARP Spoofing, Ping Sweep Scan, Reconnaissance VulScan, OS Scan, Port Scan, Malformed Data Packets, Connect Flood (DoS), Publish Flood (DDoS), Publish Flood (DoS), Connect Flood (DDoS), TCP (DoS), ICMP (DoS), SYN (DoS), UDP (DoS), SYN (DDoS) , TCP (DDoS) , ICMP (DDoS) , and UDP (DDoS). Attack-based detection is referred to as a 19-class classification.

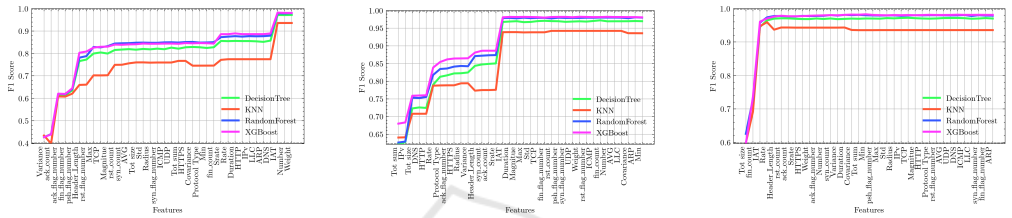
Fig. 2 shows the algorithm's performance comparison using different feature selection methods on the CICIoMT2024 dataset for binary classification. Across all three feature selection methods, the classifiers' performance rapidly improves by adding the first few features. However, the performance plateau shows only marginal improvements as more features are added. When all 36 features were included, a small subset of highly informative features had already achieved high performance across models. Most classifiers achieved high F1 scores (above 0.99) with only 5-10 features. Notably, XGB and RF consistently reached near-optimal performance with fewer than five features, while DT and KNN demonstrated more gradual improvements as features were added, achieving their best results after more features were incorporated into the model.

Attacks categories (Figure 3) and 19-classes based classification (Figure 4) show almost the same performance in comparison with binary classification as XGBoost and Random Forest again perform best, particularly when fewer features are used. However, Mutual Information demonstrates overall higher model



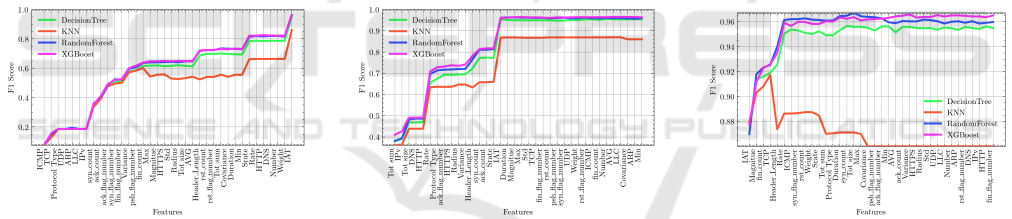
(a) Fisher Score (b) Mutual Information (c) Information Gain

Figure 2: Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for Binary Classification.



(a) Fisher Score (b) Mutual Information (c) Information Gain

Figure 3: Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for 6-class Classification.



(a) Fisher Score (b) Mutual Information (c) Information Gain

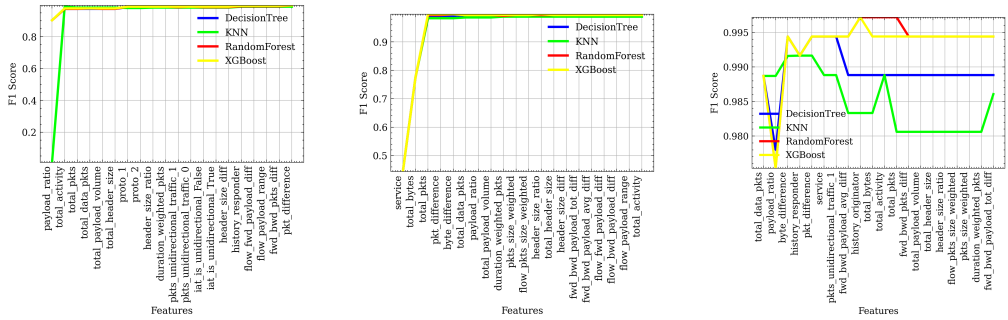
Figure 4: Comparison of algorithms performance using Feature selection methods over CICIoMT2024 data set for 19-class Classification.

performance early on, i.e., for the first three features in binary classification compared to multi-class classifications. Figure 4 shows that KNN performance drops dramatically after the first 4 features in the case of Information Gain. This shows that KNN does not work well for multi-class classification in the CICIoMT2024 dataset.

From the IoMT-TrafficData dataset, binary classification was used to differentiate between benign and attack traffic and multi-class classification was used to classify the attack traffic further. In the attack-based classification, there were 9 classes, which included 8 different types of cyberattacks i.e. DoS (Appachekiller, Slowread, Rudeadyet, Slowloris), Distributed Denial of Service (DDoS), ARP Spoofing, Buffer Overflow (Camoverflow), MQTT Malaria, and Network Scanning (Netscan).

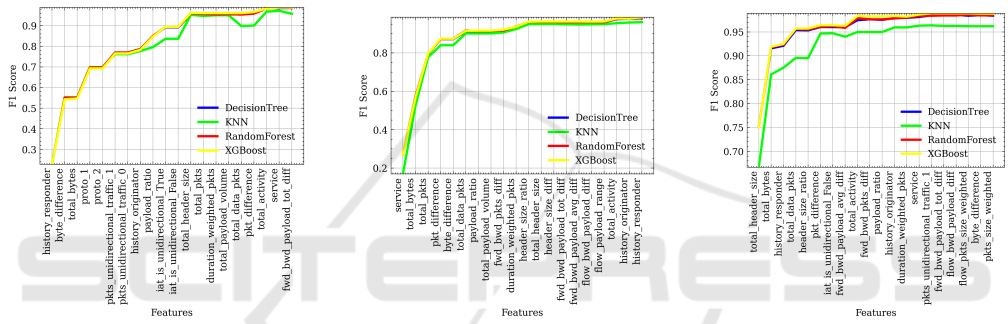
Table 1 presents the classification performance report of the XGBoost model on both CICIoMT2024 and IoMT-TrafficData, using selected top-15 features from Information Gain (IG) feature selection. Both datasets show excellent binary classification performance, with accuracy, precision, recall, and F1-score around 0.997 for both classes (attack and benign), indicating strong classification ability. On the CICIoMT2024 dataset, the model performs well across 6 and 19-class classifications, with high accuracy (0.977) and consistent metrics, though performance slightly drops for complex classes like Recon and ARP Spoofing.

On the IoMT-TrafficData dataset, accuracy remains high at 0.987, with perfect precision and recall



(a) Fisher Score Features (b) Mutual Information (c) Information Gain

Figure 5: Comparison of algorithms performance using Feature selection methods over IoMT-TrafficData dataset for Binary Classification.



(a) Fisher Score (b) Mutual Information (c) Information Gain

Figure 6: Comparison of algorithms performance using Feature selection methods over IoMT-TrafficData dataset for Multi-class classification (9 classes).

Table 1: Classification report of selected top-15 features from Information Gain (IG) feature selection for the CICIoMT2024 dataset & IoMT-TrafficData , using the XGBoost model for all three classification types.

Dataset	Classification type	Binary			6-Class classification report					
	Metric|class	Attack	Benign	Metric|class	ARP Spoofing	Benign	DDoS	DoS	MQTT	Recon
	Accuracy	0.997	0.997	Accuracy	0.977	0.977	0.977	0.977	0.977	0.977
	Precision	0.999	0.994	Precision	0.918	0.959	0.999	1.000	0.997	0.994
	Recall	0.995	0.999	Recall	0.967	0.945	1.000	0.998	0.993	0.962
CICIoMT2024 dataset	F1-Score	0.997	0.997	F1-Score	0.942	0.952	1.000	0.999	0.995	0.978

	19-Class classification																		
	Metric/Class	ARP Spoofing	Benign	MQTT-DDoS -Connect Flood	MQTT-DDoS -Publish Flood	MQTT-DoS -Connect Flood	MQTT-DoS -Publish Flood	MQTT-Malformed Data	Recon-OS Scan	Recon-Ping Sweep	Recon-Port Scan	Recon-VulScan	TCP IP-DDoS-ICMP	TCP IP-DDoS-SYN	TCP IP-DDoS-TCP	TCP IP-DDoS-UDP	TCP IP-DoS-ICMP	TCP IP-DoS-SYN	TCP IP-DoS-TCP	TCP IP-DoS-UDP
	Accuracy	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967	0.967
	Precision	0.902	0.931	1.000	1.000	1.000	1.000	0.899	0.870	0.947	0.902	0.915	1.000	0.997	1.000	0.997	1.000	1.000	1.000	1.000
	Recall	0.877	0.922	1.000	0.987	0.997	0.997	0.958	0.831	0.967	0.902	0.927	1.000	1.000	1.000	1.000	0.997	0.997	1.000	0.997
	F1-Score	0.889	0.926	1.000	0.994	0.998	0.998	0.927	0.850	0.957	0.902	0.921	1.000	0.998	1.000	0.998	0.998	0.998	1.000	0.998

	Classification type	Binary			9-Class classification report								
	Metric|class	Attack	Benign	Metric|class	Apachekiller	Arpospoofing	Camoverflow	Mqttmalaria	Netscan	Normal	Rudeadyet	Slowloris	Slowread
	Accuracy	0.997	0.997	Accuracy	0.987	0.987	0.987	0.987	0.987	0.987	0.987	0.987	0.987
	Precision	0.997	0.997	Precision	0.993	1.0	1.0	0.996	1.0	0.974	0.982	0.977	0.965
	Recall	0.997	0.996	Recall	0.981	0.987	1.0	0.989	1.0	0.993	0.966	0.981	0.997
IoMT-Traffic dataset	F1-Score	0.997	0.997	F1-Score	0.987	0.993	1.0	0.993	1.0	0.983	0.974	0.979	0.977

for many attack types (e.g., Camoverflow, Netscan), but slightly lower performance for some attack classes like Slowread and Rudeadyet.

Fig. 5 shows the algorithm's performance comparison using different feature selection methods on the IoMT-TrafficData dataset for binary classification. Across Mutual Information, the classifiers' performance rapidly improves with the addition of the first few features. Fisher Score follows the same pattern; however, it gives a slightly lower Model performance. Information Gain effectively ranks features by their usefulness, as the performance improves significantly with the first few features. XGBoost emerges as the best-performing classifier, while KNN performance tends to decrease after the first 5 features. The results suggest that focusing on the top-ranked features can optimize classifier performance while reducing computational costs. Figure 6 compares algorithm per-

Table 2: Binary classification top 15 best features.

Type of features	CICIoMT2024 dataset	IoMT-TrafficDat dataset
Protocol	TCP UDP ICMP IPv DNS HTTP Protocol Type HTTPS	proto_2 service
Traffic Volume	Tot sum Tot size	total_bytes payload_ratio total_activity
Temporal metrics	Duration IAT	iat_is_unidirectional_True duration_weighted_pkts
Packets rate	Rate srate	
Flags	ack flag number psh flag number syn flag number rst count syn count ack count fin count	history_responder hisoty_originator
Other statistical features	Std variance Max Magnitude Min Radius AVG	byte_difference fwd_bwd_pkts_diff fwd_bwd_payload_avg_diff fwd_bwd_payload_tot_diff flow_bwd_payload_diff flow_payload_range pkt_difference
Other features	Header Length	pkts_unidirectional_traffic_1

formance for multi-class classification (9 classes) on the IoMT-TrafficData dataset. Performance gradually improves with the increase in features in the case of the Fisher Score. However, models obtain higher performance earlier (after four features) in the case of Mutual Information and Information Gain. All models show comparable performance across the feature selection methods, except KNN, which lags behind when using the Information Gain method.

By examining the important features in both datasets, it is possible to identify the important network characteristics for attack detection in IoMT traffic. Table 2 illustrates the union of the top 15 features selected by different feature selection methods. The CICIoMT2024 dataset includes transport-layer protocol features, TCP, and UDP, while the IoMT-TrafficData dataset uses proto_2, which also represents transport-layer protocols. Therefore, we conclude that TCP and UDP are important features. Both datasets also emphasize application-layer protocols, such as HTTP, DNS, and SMTP (in CICIoMT2024) and service (in IoMT-TrafficData dataset), which identify application-layer protocols as well. Tot sum (CICIoMT2024) provides a key metric to understand traffic volume when considered alongside total_pkts and total_bytes in the IoMT-TrafficData dataset. Flags in CICIoMT2024 directly capture counts of specific TCP flags, while IoMT-TrafficData's history_responder encapsulates the sequence of connection states, reflecting the flags' transitions. Variability measures in packet lengths in a flow, such as Std and Variance (Ratio of the variances

Table 3: 19-class classification top 15 best features in CICIoMT2024 dataset.

Type of features	Fisher Score	Mutual information	Information Gain
Protocol	ICMP TCP UDP ARP LLC Ipv Protocol Type	IPv DNS HTTP HTTPS Protocol type	ICMP TCP Protocol type
Traffic Volume		tot sum tot size	tot sum
Temporal metrics			IAT Duration
Packets rate		Rate srate	Rate srate
Flags	ack flag number psh flag number syn flag number fin flag number	ack flag number	syn flag number
Header Attributes		Header_Length	Header_Length
Other statistical features	Variance	Variance	Variance Mangnitude Weight
Other	syn_count ack_count fin_count	syn_count ack_count	syn_count rst_count fin_count

Table 4: 9-class classification top 15 best features in IoMT-TrafficData dataset.

Type of features	Fisher Score	Mutual information	Information Gain
Protocol	proto_2	service	service
Traffic Volume	total_bytes payload_ratio total_pkts total_header_size total_payload_volume	total_bytes payload_ratio total_data_pkts total_header_size total_payload_volume	total_bytes payload_ratio total_data_pkts
Temporal metrics	duration_weighted_pkts iat_is_unidirectional_True iat_is_unidirectional_False	duration_weighted_pkts	duration_weighted_pkts total_activity iat_is_unidirectional_False
Flags	history_responder history_originator		history_responder history_originator
Header Attributes	header_size_diff	header_size_ratio	header_size_ratio
Other statistical features	byte_difference pkts_unidirectional_traffic_1 pkts_unidirectional_traffic_0	byte_difference pkt_difference fwd_bwd_pkts_diff fwd_bwd_payload_avg_diff fwd_bwd_payload_tot_diff	pkt_difference pkts_unidirectional_traffic_1 fwd_bwd_pkts_diff fwd_bwd_payload_avg_diff

of incoming to outgoing packet lengths in the flow) in CICIoMT2024, along with pkt_diff, byte_difference and fwd_bwd_payload_tot_diff, which capture the fluctuations and differences in packet lengths in IoMT-TrafficData, are essential metrics in identifying anomalies in IoMT networks. Temporal features such as Duration and IAT (Inter-arrival time) in the CICIoMT2024 dataset can be compared with duration_weighted_pkts and iat_is_unidirectional_False in the IoMT TrafficData dataset, which provides additional directional features, i.e., unidirectional/bidirectional that enhance understanding of packet arrival patterns.

Table 3 identifies several common features across Fisher Score, Mutual Information, and Information Gain methods (selected by at least two methods) for 19-class classification in CICIoMT2024 dataset. Features that relate to protocols (ICMP, TCP, Protocol type), traffic volume (tot sum), packet transmission rate (Rate, Srate) , flags (ack flag number, syn flag number, syn_count), and statistical properties (variance), are pivotal for distinguishing patterns and detecting multiple attacks in IoMT traffic, highlighting their relevance in network security analysis.

The common important features underlined in Table 4 reveal crucial insights into detecting multiple attacks in IoMT-TrafficData dataset for 9-class classification. In traffic volume, features like total_bytes, payload_ratio, total_header_size, and total_payload_volume appear frequently, emphasizing the significance of overall data transferred and packet structure. Temporal metrics are also prominent, with duration_weighted_pkts, which capture the rate or proportion of packets over time within a flow, and iat_is_unidirectional_False, capturing consistency of IAT with bidirectional traffic flowing traffic. Among flags, history_responder and history_originator recur, reflecting connection state transitions. It should also be noted that Mutual Information could not grasp any flag information. The header attribute feature, header_size_ratio, refers to the proportion of the header size relative to the total size of a packet also highlights the significance of packet header size. Lastly, statistical features like byte_difference (difference in payload bytes between the originator and responder), fwd_bwd_pkts_diff (difference in the number of packets sent forward and backward in the connection), fwd_bwd_payload_avg_diff (difference in average payload size per packet between forward and backward traffic), and pkts_unidirectional_traffic_1 (indication of unidirectional traffic) show significance in multi-class attack classification in IoMT traffic.

The analysis of Figures 2, 3, 4, 5, and 6 reveals that filter-based methods exhibit excellent performance for binary classification across both datasets. Notably, these models achieve higher performance levels early on, often after selecting just 3 to 4 features using information gain for feature selection. This indicates that these methods are effective in differentiating between benign and malicious traffic with a minimal set of features. As the number of selected features increases, the models' performance steadily improves. Significant accuracy is attained with 7 to 8 features, particularly for multi-class classifications (6-class, 9-class, and 19-class) using information gain. Among the evaluated models, the XGBoost model achieved the highest performance with fewer features selected through information gain feature selection.

5 CONCLUSION AND FUTURE WORK

In this work, we performed filter-based feature selection methods (Fisher Score, Mutual Information, Information Gain) to identify the best features in two IoMT datasets (CICIoMT2024 and IoMT-TrafficData,). We compared the performance of four machine learning algorithms (Decision Tree, Random Forest, K-Nearest Neighbors, and XGBoost) in both datasets. We checked the performance for binary and multi-class classifications in both datasets.

Fisher Score works well for both datasets, especially for classifiers like Decision Tree and KNN, which show gradual improvements as more features are added. Mutual Information is highly effective across both datasets, particularly for Random Forest and XGBoost, which reach optimal performance with fewer features. For the CICIoMT2024 dataset in binary classification, XGBoost and Random Forest perform best with Fisher Score or Mutual Information, requiring fewer features for optimal results, while Multi-class (6-class & 19-class) observed a similar trend with XGBoost and Random Forest consistently outperforming other models when using with the mentioned methods.

Information Gain works better for CICIoMT2024 datasets but shows a different pattern for binary classification in the IoMT-TrafficData dataset, where performance does not improve as rapidly compared to the other methods. Furthermore, the binary classification of IoMT-TrafficData with XGBoost and Random Forest shows superior performance with Mutual Information and Fisher Score, achieving near-optimal results with only a few features. Fisher Score and Mutual Information are again the most effective in Multi-class classification, especially for Random Forest and XGBoost in IoMT-TrafficData.

Our paper highlights key features for IoMT attack detection across both datasets, including essential transport-layer protocols (TCP, UDP), application-layer identifiers (e.g., HTTP, DNS), and traffic volume metrics (e.g., total_bytes, payload_ratio). Temporal and directional metrics, like Duration, IAT, and connection-state flags (history_responder), enhance understanding of packet flows, while variability and statistical measures (variance, byte_difference) are crucial for identifying attack patterns, underscoring their importance in multi-class attack classification in IoMT traffic. Furthermore, the XGBoost model demonstrates excellent performance in both binary and multi-class classification across the CICIoMT2024 and IoMT-TrafficData datasets, with minor variations in handling certain attack types. Our study shows that filter-based methods perform well in binary classification with 3-4 features, while multi-class classification achieves significant accuracy with 7-8 features across both datasets. Furthermore, this study also illustrates that using the top-15 features of the selection of information gains (IG) features for the XGboost model, achieving excellent binary classification results (0.997 accuracy, precision, recall, and F1

score) and very good performance in multiclass classifications, with slight drops for few complex attacks, thereby opening doors for further research.

In future work, exploring hybrid feature selection methods, such as combining Mutual Information with optimization techniques like Genetic Algorithms, could improve feature relevance. Implementing non-stationary models to dynamically adapt to new features and unseen attacks would also enhance the robustness of intrusion detection systems in healthcare IoMT networks. Furthermore, extending the work to include other types of datasets, such as telemetry, software, hardware threats, or monitored data from implantable devices, could broaden the applicability of the results.

ACKNOWLEDGMENT

This study was co-funded by the European Union and Estonian Research Council via project TEM-TA5.

REFERENCES

Al-Sarem, M., Saeed, F., Alkhammash, E. H., and Alghamdi, N. S. (2021). An aggregated mutual information based feature selection with machine learning methods for enhancing iot botnet attack detection. *Sensors*, 22(1):185.

Alalhareth, M. and Hong, S.-C. (2023a). An improved mutual information feature selection technique for intrusion detection systems in the internet of medical things. *Sensors*, 23(10).

Alalhareth, M. and Hong, S.-C. (2023b). An improved mutual information feature selection technique for intrusion detection systems in the internet of medical things. *Sensors*, 23(10):4971.

Alazzam, H., Sharieh, A., and Sabri, K. E. (2020). A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer. *Expert systems with applications*, 148:113249.

Almomani, O. (2020). A feature selection model for network intrusion detection system based on pso, gwo, ffa and ga algorithms. *Symmetry*, 12(6):1046.

Ambusaidi, M. A., He, X., Nanda, P., and Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE transactions on computers*, 65(10):2986–2998.

Areia, J., Bispo, I., Santos, L., and Costa, R. L. d. C. (2024). Iomt-trafficdata: Dataset and tools for benchmarking intrusion detection in internet of medical things. *IEEE Access*.

Awotunde, J. B., Abiodun, K. M., Adeniyi, E. A., Folorunso, S. O., and Jimoh, R. G. (2021). A deep learning-based intrusion detection technique for a secured iomt system. In *International Conference on Informatics and Intelligent Applications*, pages 50–62. Springer.

Bahşi, H., Nõmm, S., and La Torre, F. B. (2018). Dimensionality reduction for machine learning based iot botnet detection. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1857–1862. IEEE.

Chaganti, R., Mourade, A., Ravi, V., Vemprala, N., Dua, A., and Bhushan, B. (2022). A particle swarm optimization and deep learning approach for intrusion detection system in internet of medical things. *Sustainability*, 14(19):12828.

Dadkhah, S., Neto, E. C. P., Ferreira, R., Molokwu, R. C., Sadeghi, S., and Ghorbani, A. A. (2024). Ciciomt2024: A benchmark dataset for multi-protocol security assessment in iomt. *Internet of Things*, 28:101351.

Dimitrov, D. V. (2016). Medical internet of things and big data in healthcare. *Healthcare informatics research*, 22(3):156–163.

Estévez, P. A., Tesmer, M., Perez, C. A., and Zurada, J. M. (2009). Normalized mutual information feature selection. *IEEE Transactions on neural networks*, 20(2):189–201.

Gökdemir, A. and Calhan, A. (2022). Deep learning and machine learning based anomaly detection in internet of things environments. *Journal of the Faculty of Engineering and Architecture of Gazi University*, 37(4):1945–1956.

Gu, Q., Li, Z., and Han, J. (2012). Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*.

Ingre, B. and Yadav, A. (2015). Performance analysis of nsl-kdd dataset using ann. In *2015 international conference on signal processing and communication engineering systems*, pages 92–96. IEEE.

Islam, S. R., Kwak, D., Kabir, M. H., Hossain, M., and Kwak, K.-S. (2015). The internet of things for health care: a comprehensive survey. *IEEE access*, 3:678–708.

Janarthanan, T. and Zargari, S. (2017). Feature selection in unsw-nb15 and kddcup'99 datasets. In *2017 IEEE 26th international symposium on industrial electronics (ISIE)*, pages 1881–1886. IEEE.

Jović, A., Brkić, K., and Bogunović, N. (2015). A review of feature selection methods with applications. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1200–1205.

Kalakoti, R., Bahsi, H., and Nõmm, S. (2024a). Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection. *IEEE Internet of Things Journal*.

Kalakoti, R., Bahsi, H., and Nõmm, S. (2024b). Explainable federated learning for botnet detection in iot networks. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 01–08.

Kalakoti, R., Nõmm, S., and Bahsi, H. (2022). In-depth feature selection for the statistical machine learning-based botnet detection in iot networks. *IEEE Access*, 10:94518–94535.

Kalakoti, R., Nõmm, S., and Bahsi, H. (2023). Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai). In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 595–601. IEEE.

Kalakoti, R., Nõmm, S., and Bahsi, H. (2024c). Enhancing iot botnet attack detection in socs with an explainable active learning framework. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 265–272. IEEE.

Kasongo, S. M. and Sun, Y. (2020). Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset. *Journal of Big Data*, 7(1):105.

Khalil, A. A., E Ibrahim, F., Abbass, M. Y., Haggag, N., Mahrous, Y., Sedik, A., Elsherbeeny, Z., Khalaf, A. A., Rihan, M., El-Shafai, W., et al. (2022). Efficient anomaly detection from medical signals and images with convolutional neural networks for internet of medical things (iomt) systems. *International Journal for Numerical Methods in Biomedical Engineering*, 38(1):e3530.

Khammassi, C. and Krichen, S. (2017). A ga-lr wrapper approach for feature selection in network intrusion detection. *computers & security*, 70:255–277.

Khan, N. M., Madhav C, N., Negi, A., and Thaseen, I. S. (2020). Analysis on improving the performance of machine learning models using feature selection technique. In *Intelligent Systems Design and Applications: 18th International Conference on Intelligent Systems Design and Applications (ISDA 2018) held in Vellore, India, December 6-8, 2018, Volume 2*, pages 69–77. Springer.

Kondeti, V. and Bahsi, H. (2024). Mapping cyber attacks on the internet of medical things: A taxonomic review. In *2024 19th Annual System of Systems Engineering Conference (SoSE)*, pages 84–91. IEEE.

Kumar, V., Sinha, D., Das, A. K., Pandey, S. C., and Goswami, R. T. (2020). An integrated rule based intrusion detection system: analysis on unsw-nb15 data set and the real time online dataset. *Cluster Computing*, 23:1397–1418.

Osanaiye, O., Cai, H., Choo, K.-K. R., Dehghantanha, A., Xu, Z., and Dlodlo, M. (2016). Ensemble-based multi-filter feature selection method for ddos detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2016:1–10.

Parimala, G. and Kayalvizhi, R. (2021). An effective intrusion detection system for securing iot using feature selection and deep learning. In *2021 international conference on computer communication and informatics (ICCCI)*, pages 1–4. IEEE.

Rbah, Y., Mahfoudi, M., Balboul, Y., Fattah, M., Mazer, S., Elbekkali, M., and Bernoussi, B. (2022). Machine learning and deep learning methods for intrusion detection systems in iomt: A survey. In *2022 2nd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, pages 1–9. IEEE.

Tama, B. A., Comuzzi, M., and Rhee, K.-H. (2019). Tseids: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE access*, 7:94497–94507.

Velasco-Mata, J., González-Castro, V., Fernández, E. F., and Alegre, E. (2021). Efficient detection of botnet traffic by features selection and decision trees. *IEEE Access*, 9:120567–120579.

Wagan, S. A., Koo, J., Siddiqui, I. F., Qureshi, N. M. F., Attique, M., and Shin, D. R. (2023). A fuzzy-based duo-secure multi-modal framework for iomt anomaly detection. *Journal of King Saud University-Computer and Information Sciences*, 35(1):131–144.

Zong, W., Chow, Y.-W., and Susilo, W. (2018). A two-stage classifier approach for network intrusion detection. In *Information Security Practice and Experience: 14th International Conference, ISPEC 2018, Tokyo, Japan, September 25-27, 2018, Proceedings 14*, pages 329–340. Springer.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Appendix 3

III

R. Kalakoti, H. Bahsi, and S. Nõmm. Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection. *IEEE Internet of Things Journal*, 11(10):18237–18254, 2024

Improving IoT Security With Explainable AI: Quantitative Evaluation of Explainability for IoT Botnet Detection

Rajesh Kalakoti, *Graduate Student Member, IEEE*, Hayretdin Bahsi, and Sven Nõmm

***Abstract*—Detecting botnets is an essential task to ensure the security of Internet of Things (IoT) systems. Machine learning (ML)-based approaches have been widely used for this purpose, but the lack of interpretability and transparency of the models often limits their effectiveness. In this research paper, our aim is to improve the transparency and interpretability of high-performance ML models for IoT botnet detection by selecting higher quality explanations using explainable artificial intelligence (XAI) techniques. We used three data sets to induce binary and multiclass classification models for IoT botnet detection, with sequential backward selection (SBS) employed as the feature selection technique. We then use two post hoc XAI techniques such as local interpretable model-agnostic explanations (LIME) and Shapley additive explanation (SHAP), to explain the behavior of the models. To evaluate the quality of explanations generated by XAI methods, we employed faithfulness, monotonicity, complexity, and sensitivity metrics. ML models employed in this work achieve very high detection rates with a limited number of features. Our findings demonstrate the effectiveness of XAI methods in improving the interpretability and transparency of ML-based IoT botnet detection models. Specifically, explanations generated by applying LIME and SHAP to the extreme gradient boosting model yield high faithfulness, high consistency, low complexity, and low sensitivity. Furthermore, SHAP outperforms LIME by achieving better results in these metrics.**

***Index Terms*—Botnet, complexity, consistency, explainable artificial intelligence (XAI), faithfulness, feature importance, Internet of Things (IoT), local interpretable model-agnostic explanations (LIME), posthoc XAI, robustness, Shapley additive explanation (SHAP).**

I. Introduction

THE RISE of Internet of Things (IoT) botnets [1] has become a significant security challenge for devices connected to the Internet, including homes, businesses, and critical infrastructure. A botnet is a group of Internet-connected devices controlled by a single entity, known as the botmaster, who can use the botnet for various malicious purposes, such as Distributed Denial-of-Service (DDoS) attacks, spamming, and cryptocurrency mining. Due to the large number of devices connected to the Internet, the sizes of the botnets consisting of IoT devices can be much larger than traditional botnets, posing a more significant threat to Internet security [2]. Detecting IoT botnets is a complex task due to their distributed nature and the various devices that they infect [1]. Traditional detection methods, such as signature-based and anomaly-based methods, are insufficient for detecting IoT botnets, as attackers can rapidly modify their behavior to evade detection. Machine learning (ML) has emerged as a promising technique for detecting IoT botnets, as it can analyze large amounts of data and identify patterns that humans may find difficult to recognize [3], [4], [5].

Despite the impressive results achieved by many ML techniques in the cyber security domain, significant concern arises due to the inherent lack of interpretability in ML models, which implies that security experts may encounter difficulties trusting the outputs of ML models because they do not fully understand how a model reaches a particular decision or classification. Trust is a considerable challenge in intrusion detection systems (IDSs) and many other critical systems. Due to this challenge, new approaches have been developed in the last few years with the goal of enhancing the explainability of ML models so that their output is more interpretable. This notion is known in the literature as explainable artificial intelligence (XAI) [6]. The integration of XAI techniques has caused a new trend in cybersecurity research that emphasizes the inclusion of additional layers of explainability for humans in the loop [7], [8]. Several works have used XAI techniques for IoT botnet detection systems [9], [10]. Furthermore, some survey articles have extensively highlighted open questions and future research directions in this domain [11], [12], [13], [14].

Many ML models, such as convolutional neural networks (CNNs), are often regarded as "black boxes," lacking transparency in decision-making. The goal of XAI is to improve the understanding of how ML models operate. For example, when identifying potential IoT botnets, an XAI method can reveal which network features are most critical in detecting the presence of malicious activity on IoT devices. This additional information is known as an "explanation." Explanations help users understand the models and trust the decisions they make. Model designers and ML experts can use explanations to enhance the performance of the model. XAI explanations can be categorized into global and local scopes. The local

Manuscript received 27 November 2023; revised 12 January 2024; accepted 29 January 2024. Date of publication 31 January 2024; date of current version 9 May 2024. *(Corresponding author: Rajesh Kalakoti.)*

Rajesh Kalakoti and Sven Nõmm are with the Department of Software Science, Tallinn University of Technology, 12616 Tallinn, Estonia (e-mail: rajesh.kalakoti@taltech.ee; sven.nomm@taltech.ee).

Hayretdin Bahsi is with the Department of Software Science, Tallinn University of Technology, 12616 Tallinn, Estonia, and also with the School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, AZ 86011 USA (e-mail: hayretdin.bahsi@taltech.ee).

Digital Object Identifier 10.1109/JIOT.2024.3360626

2327-4662 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

explanation in the model focuses on explaining individual predictions, whereas the global explanation unveils the overall behavior of the model. In XAI, models can achieve explanations through an intrinsic (model-specific) approach, in which explainability is integrated into the model during training and is not transferable, or a post hoc (model-agnostic) approach. The post-hoc XAI method is independent of any architecture and can be applied to any trained ML model.

Despite using an enormous number of tools and systems, experts are still highly involved in cyber security operations (e.g., incident handling, security monitoring, and vulnerability management). Identifying the incident, justifying the relevant findings, determining the possible course of action, and selecting and applying the most feasible countermeasure are complex tasks in which complete automatization is challenging to achieve. Specifically, in intrusion detection (ID) tasks, experts want to be sure of the findings, especially in high-stake situations, as such findings are essential input for many follow-up operations. Thus, explainability is as crucial as high detection capability in IDS using learning models that are of a "black-box" nature [15].

Post hoc explainability methods that are agnostic to any learning model have gained attraction in research circles due to their greater applicability [16]. Local interpretable model-agnostic explanations (LIME) [17] and Shapley additive explanation (SHAP) [18] have emerged as prominent local explainers that provide detailed explanations of the given instance and its model output. More specifically, they assign importance scores to features that play a key role in model decisions. Thus, experts can gain insight into the model output by checking whether such features make sense within the context of the corresponding cyber-incident. However, the quality of these explainability methods has emerged as a concern, as these methods use additional instruments (e.g., linear models at the point of interest, reflect game theory in identifying the importance score), which may introduce errors into the explanations [19]. Specifically, numerous current XAI techniques may result in unreliable explanations in real-world scenarios due to limited qualitative evaluation [20]. Addressing such issues is vital for IDSs in IoT networks, especially those capable of revealing sensitive information, such as temporal patterns of device activity. XAI approaches for ID are still in their early stages, with limited available literature [11], [12], [13], [14]. As XAI-based solutions for IDSs continue to increase, it becomes crucial to thoroughly evaluate their explainability components. This evaluation is essential to ensure that the explainability techniques employed are truly able to be deployed in real-world scenarios and contribute to understanding model outputs. Therefore, the design of an XAI system that includes robust qualitative and quantitative evaluation procedures for XAI methods used in the real world is critically essential for effective adoption in IDSs.

To overcome the above-mentioned challenges, our research emphasizes the significance of considering the quality of explainability using quantitative evaluation alongside the detection performance of models. We advocate for including this evaluation as a significant criterion for building models that security experts can trust in XAI-based IDS. In this research paper, first, we employed sequential backward selection (SBS) to perform feature selection on three IoT botnet data sets and then created learning models. Furthermore, we used LIME and SHAP as posthoc local explanations to provide insight into the models' behavior. We evaluated the quality and usefulness of the local explanations generated by these two posthoc explainers. For the evaluation purpose, we used four criteria: 1) faithfulness (how well the explanations match the actual behavior of the models); 2) consistency (how similar the explanations are for similar inputs); 3) complexity (how understandable the explanations are); and 4) robustness (how well the explanations hold up under perturbations of the input). Our evaluation of the local explanations using these criteria allowed us to determine their reliability and informativeness in explaining the outcomes of the models for XAI-based IDS. Our investigation highlights a notable gap in the existing literature-based on our knowledge, no empirical study has been performed that evaluates the quality of post hoc local explainability methods for network ID tasks, in general, and IoT botnet detection tasks, in particular. By providing insights into the significance of XAI within the cybersecurity domain, this article aims to contribute to the ongoing efforts to make AI more transparent, reliable, and beneficial for society. The contribution of this work can be summarized as follows.

1) A framework employing SBS feature selection to enhance explainability by selecting a subset of discriminating features, thus reducing the complexity of the model and facilitating a more transparent understanding of the output.
2) Evaluation of the quality of local explanations provided by two widely used post hoc XAI methods, LIME and SHAP, for explaining model decisions in the context of IoT botnet detection. The evaluation includes quantitative metrics, such as faithfulness, monotonicity, complexity, and sensitivity.
3) Evaluation and comparison of local explanations applied to various black box nature ML models for IoT botnet detection in binary and multiclass classification problems within a detailed benchmarking setting.

The findings of the present research suggest that a carefully designed benchmarking study can identify high-performance detection models, which can also induce high-quality explanations. Thus, it is possible that security experts do not need to sacrifice detection performance over explainability for the ML task addressed.

The structure of the research work is described as follows. Section II provides the background and review of the study literature. Section III focuses on IoT botnet detection and XAI techniques. The results of the study are presented in Section IV. Section V provides a discussion of the main findings of this research work. Finally, conclusions are drawn in Section VI.

II. Background and Related Work

The idea of explaining ML models [21], [22] has become increasingly important and has been explored by researchers in various domains [23]. Researchers have investigated

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

the explainability in healthcare care [24], [25], [26], social science [27], and in the field of human–computer interaction (HCI) [28], [29], [30].

An explanation aims primarily at improving the human ability to interpret an event. It can be considered an answer to the following questions: "What?", "Why?", "How?", "What if?", "What else?", "Why should?", and "Why not?" [27], [31], [32]. Each question reveals various aspects of an event or the process that guided it. On the other hand, the primary function of explantation is to demonstrate the reasons for an event ("Why?"), which could also be related to the particular context [27]. Clearly, for crucial context-dependent domains such as ID, The questions "what else?" refer to additional information or context surrounding an event or alert that can help stakeholders better understand the situation and make more informed decisions. For example, if the system detects an attempted intrusion, this type of question might include the following. What other activities were occurring on the network at the time? Were there any changes to the system made recently that could have created a vulnerability?

While the early AI frameworks were transparent and easy to comprehend, on the other hand, opaque and nontransparent techniques, such as deep neural network (DNN) models, have gained experimentally outstanding achievements in the field of AI due to the assortment of efficient learning algorithms and their extensive parametric field. DNN models are high-dimensional, nonconvex, resource- and data-hungry but lack interpretability. Therefore, DNNs are considered sophisticated black-box models because they contain thousands of layers and millions of parameters (weights), so the behavior of these models cannot be understood [33]. The need for XAI among AI stakeholders is increasing as black-box models are used to produce effective predictions in critical settings [34]. The jeopardy lies in making and carrying out decisions that are not rational, legitimate, or allow for far-reaching explanations of their activities [35]. Explanations that delineate the output of the model are crucial. For example, in the medical domain, experts (doctors) must disclose the reasons for the prediction identified in the model, which requires faithful explanations of AI decisions on diagnosis [36].

ID is a critical component of the cybersecurity domain. It allows us to detect malicious activities that can affect the confidentiality, integrity, and availability (CIA) properties of information assets [37]. The widespread utilization of IoT devices has created various attack opportunities for malicious actors. Such devices in high-stakes applications can be directly targeted by attackers or constitute part of the attack infrastructure as bots. Therefore, attacks launched against IoT devices should be detected by IDSs to initiate the relevant countermeasure actions. IDSs use different detection models, such as signature-based, anomaly-based, and hybrid [38]. Signature-based systems rely on the detection rules created by experts, whereas anomaly-based systems profile benign system utilization and detect deviations from such profiles. Hybrid models benefit from both approaches to achieve higher detection performance. ML models can eliminate the need for signatures by learning from attacks and benign system activities. More importantly, they are expected to identify unknown attacks that signatures cannot detect. ML models can also automatically create profiles of benign activities. Besides, some studies have employed feature selection methods to obtain optimal features for enhanced performance [39], [40], [41].

However, due to the black-box nature of most ML methods, the decisions made by such systems are difficult to comprehend by security experts. The lack of transparency is a significant issue in security operations, including ID, in which XAI can ensure that the predictions of the models are easily understood [8]. In some cases where experts suspect a misclassification of the IDS decision, explanations are necessary to diagnose the case and improve the system to prevent future attacks and take the appropriate steps [7].

It is important to note that, despite the problems regarding the detection of unknown and newly evolved attacks in conventional signature-based systems, experts can easily understand the detection decisions of such systems, as signature formats reveal the technical details. ML models equipped with XAI methods constitute a complete detection solution that can automatically adapt and improve over time without requiring manual effort while providing high detection performance.

A. Related Work

In the literature, various studies adapt explainability methods to network ID problems. Szczepański et al. [42] introduced the hybrid Oracle Explainer IDS, which combines artificial neural networks (ANNs) and Decision Trees, leveraging microaggregation techniques for improved performance [42]. The system aims to achieve high accuracy while providing human-understandable explanations for its decisions. The authors developed an Oracle-based Explainer module that measures the distance between clusters formed from the CICIDS2017 data set [43] and the test instances. The closest cluster is then used to generate an explanation for the decision.

In the study by [44], the authors propose an explainable hybrid IDS that integrates a rule-based approach with human experience and ML, creating a complete hybrid system. The decision tree, a white box model known for its intrinsic explainability, is used to provide rule-based explanations for the ID model, making it more comprehensible to experts.

Wang et al. [8] presented a framework that uses SHAP to harmonize local and global explanations, thus improving the interpretability of ID decisions [8]. Local explanations clarify the rationale behind predictions for specific instances, while global explanations showcase the prominent features derived from the model, illustrating the relationships between feature importance and attack types. Furthermore, the study compares the explanations generated for two classifier configurations, namely, the one-versus-all classifier and the multiclass models, using the NSL-KDD data set [45], as done in [8].

Le et al. [46] used SHAP in conjunction with Decision Tree and random forest (RF) models to facilitate a complete IDS. The implementation of a heatmap allowed for the visualization of the impact of individual features on the overall model, while a Decision Plot was used to explain specific instances of the data set. The ToN-IoT [47] and BoT IoT [48] data sets were used for this study.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

Suryotrisongko et al. [9] have proposed a novel model for the detection of domain generation algorithm (DGA) of botnets on Alexa's Top 1M domain names and 803 333 domain names of ten DGA botnet families employed in Conficker, Cryptolocker, Goz, Matsnu, New_Goz, Pushdo, Ramdo, Rovnix, Tinba, and Zeus [49]. The authors have used five classification algorithms: 1) Logistic Regression; 2) RF; 3) Naive Bayes; 4) Extra Tree; and 5) Ensemble. Of these five algorithms, RF has achieved 95.7% accuracy using a set of five features. Open-source intelligence (OSINT) and post hoc XAI methods, including SHAP and LIME, were incorporated into their work to provide a solution to incredulity toward the model's output and enhance the trust of the DGA detection model.

Araki et al. [50] proposed a two-step subspace clustering method to cluster botnets and classified their functionalities using the DBSCAN algorithm [51]. The proposed method divides features into multiple subspaces to individually target hosts and generate sublabels for each subspace to illustrate partisan aspects, such as low-size flows or a high TCP-SYN rate. The effectiveness of the proposed method was evaluated using two network traffic data sets: the MAWI [52] and ISP data sets. The method combines subspace clustering and frequent pattern mining to represent and explain essential features. The results demonstrate that the method identified and classified 60 bot groups that comprise 61 167 IP addresses in the MAWI data set and 295 bot groups that comprise 408 118 IP addresses in the ISP data set. Self-explained clustering dendrograms were used to explain the types of bots present in the network.

Mazza et al. [53] have proposed a bot detection technique named RTbust (Retweet Buster) that emanates from the previous investigation of retweeting behaviors. The proposed model [53] takes advantage of unsupervised feature extraction and clustering. They have used long short-term memory (LSTM) and variational autoencoders (VAEs) approaches, which transform the retweet time series into latent feature vectors. These vectors are then clustered with a hierarchical density-based algorithm. Twitter accounts belonging to large clusters are represented by malicious retweeting patterns, labeled bots. The RTT scatter plot visualization tool (ReTweet-Tweet) is used to come out of the black nature of a model after RTbust has been used to understand the characteristics of those Twitter accounts classified as bots.

Zolanvari et al. [54] have proposed a model-agnostic XAI framework named transparency based on statistical theory (TRUST) for numerical applications. The framework employs factor analysis to convert input features into latent features, uses mutual information to rank features, and then uses a multimodal Gaussian distribution to find a new sample that belongs to each class label. TRUST XAI has been evaluated through a case study on three data sets, namely, NSL-KDD [45], UNSW [55], and a data set collected from their testbed experiment, called "WUSTL-IIoT," on intrusion network traffic data. Finally, compared to LIME, the TRUST XAI model has achieved a success rate 98% in explaining random test samples. The paper [8] addressed the challenge of explaining IDS in computer networks. They have used deep learning (DL) to build an IDS. Then, they developed an XAI framework to optimize the transparency of the model. The authors operated on the NSL-KDD [56] data set to design the DL-based IDS. They used four XAI methods, such as SHAP, LIME, ProtoDash, and contrastive explanations to explain the DL ID model. A study analyzed the impact of feature selection on the reliability of the explanations induced by the post hoc local interpretability method, LIME [57]. In this study, the varying number of features selected by the filter method are used to create a model and then utilized by the explainer. An entropy-based metric is proposed to assess the quality of explanations based on feature selection.

With the advancement of DL in cyber security, various pilot studies have been performed to understand the behavior of botnet network traffic. However, cyber-security stakeholders need help building their faith in the results of current DL models due to bad decisions made by complex neural network models. To deal with such problem, Kundu et al. [58] performed extensive experiments by employing a combination of synthetic and real network traffic created by the IXAI breaking point system [59]. 1DCNN model is tested over three kinds of data sets. The synthetic data set is generated by the IXAI appliance, and the others are the Stratosphere IPS Project data set [60] and Kitsune data set [61] in botnet traffic. The authors have shown that the proposed DCNN botnet detection models perform better than the previous ML models, with an improvement of up to 15% for all classification results metrics. At the same time, SHAP was deployed to provide an understandable explanation of model decision-making and achieve the faithfulness of cyber security stakeholders.

Even with the significant improvements in recent times, there are significant gaps in the existing literature concerning XAI for IoT-based IDSs, which necessitate more in-depth investigation. A critical gap that stands out is the absence of standardized evaluation metrics to evaluate the effectiveness and utility of XAI techniques within the context of the ID domain. Current research in cyber security relies primarily on qualitative assessments of XAI approaches. Developing a comprehensive suite of benchmarks to measure the domain-specific facets of explainability would enable more robust comparisons across diverse XAI methodologies and streamline identifying optimal practices. While the works mentioned above can be viewed as initial steps toward introducing explainability into IoT botnet detection systems, as of our current understanding, there still needs to be a more quantitative evaluation of XAI techniques in the literature pertaining to cybersecurity. This highlights a need for more rigorously evaluating the quality of generated explanations, a vital prerequisite to establishing confidence in the explanation outputs of AI systems built upon XAI principles [8], [42], [44], [46], [53], [54], [62], [63], [64], [65].

III. IoT Botnet Detection and XAI Techniques

This section presents the workflow of a framework designed for IoT Botnet detection using XAI, as shown in Fig. 1. In the first step, we applied feature selection to the original data set and selected the optimal features for the classification. The

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

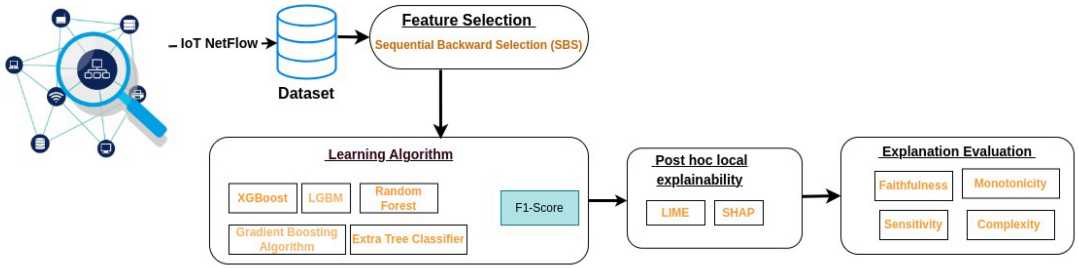


Fig. 1. Framework used in this study: we have applied SBS feature selection and models evaluated by F1-Score. Explained models using post hoc local explainability (LIME&SHAP). Evaluated the quality of explanations by faithfulness, monotonicity, sensitivity, and complexity.

TABLE I
SUMMARY OF THE FEATURES OF THE N-BAIOT AND MEDBIOT DATA SETS FEATURES

				N-BaIoT	MedBIoT
Feature Category	Category Code	Statistical Value Feature	Time Frame Window	No. of Features	No. of Features
Host Mac& IP	MI	Packet Count, Mean Variance	100 Micro Seconds	15	15
Host IP	H		500 Micro seconds	15	–
Network Jitter	HH_Jit		1.5 Seconds	15	15
Channel	HH	Packet Count, Mean Variance, Magnitude, Radius, Covariance, Correlation	10 Seconds	35	35
Socket	HpHp		1 Minute	35	35

second step, creation of the learning models for the classification tasks, is followed by a post-hoc step that applies XAI. The workflow ends with the evaluation of the explanations generated in the previous step.

The primary focus of this study is on two types of classification problem: Binary classification and multiclass classification. In the binary classification task, we used network traffic data from all three data sets to distinguish between malware and benign traffic. In this multiclass classification task, we focus on the detection of two distinct types: 1) botnet malware type detection, involving the classification of deployed botnets and benign traffic, using N-BaIoT (containing Mirai, Gafgyt, and Benign) and MedBIoT (Bashlite, Mirai, Torii, and Benign). On the other hand, 2) botnet attack type detection, including classes, such as ACK, benign, compact, junk, scan, syn, TCP, UDP, and UDP plain. Steps of the framework given in Fig. 1 are explained in detail in the following sections.

A. Data Set Description

1) N-BaIoT and MedBIoT Data Sets: The N-BaIoT [3] and MedBIoT [66] data sets comprise features extracted from bot-produced network traffic in a controlled testing environment. With 115 and 100 features extracted from network traffic, respectively, the data sets contain aggregated statistics of the raw network streams in five-time windows: 100 ms, 500 ms, 1.5 s, 10 s, and 1 min. These time windows are coded L5, L3, L1, L0.1, and L0.01, respectively. The features are classified into five main categories, which are host-IP(H), host-MAC&IP (MI), channel (HH), socket (HpHp), and network jitter (HH_Jit). Statistical values such as packet count, mean, and variance packet sizes are calculated for each significant category. Additional values such as the correlation coefficient (PCC) of packet size, radius, covariance, and magnitude are derived for the Channel and Socket categories (see Table I). The N-BaIoT data set consists of network traffic derived from Mirai and Gafgyt botnets and includes network traffic from nine different infected devices. The MedBIoT data set contains network traffic from Bashlite, Mirai, and Torri botnets, and it is collected at the command and control (C&C) or formation phases. The characteristics defined for each data point reflect the importance of specific statistical measures during the C&C or formation phases. In the N-BaIoT and MedBIoT data sets, attacks are performed using C&C servers, which are typically used by botmasters to control a network of infected IoT devices (bots). The C&C server acts as the central command point for the botmaster to issue commands and receive data from infected devices.

In the case of the N-BaIoT data set, Mirai and Gafgyt botnets use C&C servers to propagate malware and coordinate DDoS attacks. These botnets target various IoT devices, including routers, cameras, and other smart devices, by exploiting various vulnerabilities. Similarly, in the case of the MedBIoT data set, the Bashlite, Mirai, and Torii botnets also use C&C C servers to control infected devices and coordinate attacks. These botnets also target a wide range of IoT devices, including Locks, Switch, Fan, and Light by exploiting various vulnerabilities in the IoT botnet attack test bed environment.

2) BoT-IoT: The Bot IoT data [48] set was generated using the Ostinato tool, a network traffic generator that can simulate realistic network traffic in a controlled environment. In the case of BoT-IoT data set generation, the tool was used to generate network traffic data for a cloud server consisting of virtual machines and Kali Linux machines, on which various services, such as DNS, SSH, FTP, and HTTP, were deployed. Kali machines were used to simulate various IoT devices, and Node-RED [67] was used as a programming tool to create realistic behavior of the IoT devices. The data set contains five different IoT scenarios, each representing a different

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

type of IoT device: a weather station, a smart refrigerator, motion-activated lights, a remotely activated garbage door, and a smart thermostat. The captured network traffic data included various types of attacks, such as UDP, TCP, OS fingerprint, service scan, HTTP, keylogging, and data exfiltration. Based on the source paper of the data set, we have taken the top ten features for work in our workflow.

1) Numerical representation of feature state (state_number).
2) Argus sequence number (seq).
3) Average duration of aggregated records (Mean).
4) Standard deviation of aggregated records (StdDev).
5) Minimum duration of aggregated records (Min).
6) Maximum duration of aggregated records (Max).
7) Source-to-destination packets per second (Srate).
8) Destination-to-source packets per second (Drate).
9) Number of inbound connections per source IP (N_IN_Conn_P_SrcIP).
10) Number of inbound connections per destination IP (N_IN_Conn_P_DstIP).

B. Feature Selection

Based on our previous research [41] on the selection of in-depth IoT features on these N-BaIoT and MedBIoT, we selected SBS of features based on the greedy search algorithm. SBS is a feature selection method that iteratively removes the least important features of the data set. It starts with the entire set F of n features and evaluates the performance of the classification algorithm using all features. Then, it removes one feature at a time and creates a new subset of features with $n-1$ features. The evaluation function is used again to calculate the algorithm's performance with the reduced set of features. The process continues until the best optimal number of features is derived or until the algorithm's performance stops improving. SBS works in a top-to-bottom method; the worst feature is removed at each iteration. Here are the following steps.

1) S = feature set, $F = f_1, f_2, \ldots, f_n$.
2) *while* |F|>1 do
   #|F| is size of the feature set F.
3) $f_i = \text{argmin}_{j\in F}[J(S - f_i)]$.
4) $S{=}S - f_i$.
5) $F = F - f_i$.

C. Classification

The methodology used to perform ML experiments on a large data set that contains network traffic samples. The initial data set was large enough to provide samples that could be balanced based on different characteristics of the data set, such as malware type, attack type, and device type. The data was preprocessed to ensure balanced samples were drawn and then proportionally divided into training and testing sets (80/20). Several classification algorithms were tested, including extreme gradient boosting (XGBoost), light gradient boost machine (LGBM), gradient boosting classifier (GBC), RF, extremely randomized trees (ET), logistic regression, support vector machine (SVM), and Ada-boost. The last three algorithms had lower performance and were excluded from further investigation. A tenfold cross-validation was performed to find the optimal hyperparameters for each classifier and feature selection technique. A randomized search was used to ensure the best configuration for each classification algorithm.

This study uses the F_1 score metric to assess the feature set derived from the wrapper method. Defined as the harmonic mean of precision (P) and recall (R) [68], the F_1 score [see (1)] offers a more appropriate evaluation of misclassified instances compared to accuracy. Precision represents the proportion of accurately identified botnet samples among all samples classified as botnets, while recall corresponds to the fraction of correctly identified botnet samples with the total botnet samples in the data set. The choice of the harmonic mean of the F_1 score is driven by its ability to penalize extreme values, ensuring a more balanced assessment

$$F_1\text{-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \tag{1}$$

D. XAI Methods

In our research, we are studying two types of posthoc xai methods for understanding IoT botnet predictions. These approaches, known as SHAP and LIME, are widely acknowledged and extensively employed in the field of XAI. Our main goal is to use these methods to explain how predictions are made, which is crucial for generating alerts. Additionally, we aim to evaluate these XAI methods when applied to IoT botnet detection. Below, we provide a succinct summary of LIME and SHAP methods and the evaluation metrics for these methods.

For data set $\mathcal{D}$, input $x \in \mathbb{R}^d$, where d is the dimensionality of the feature set, and the black box model $\mathcal{M}$ maps the input to an output $\mathcal{M}(x) \in \mathcal{Y}$. Denote $\mathcal{D} = \{(x^i, y^i)\}$ as the collection of all input-output pairs in the data set. A post hoc explanation, denoted as $\mathbf{g}$ as an explanation mapping that for predictor $\mathcal{M}$ and point of interest x returns an importance score $\mathbf{g}(\mathcal{M}, x) = \varphi_x \in \mathbb{R}^d$ for all features. Denote $D : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$ as a metric in the space of explanations and $S\mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$ a metric in the space of inputs. The evaluation criterion μ is the mapping that takes predictor $\mathcal{M}$, explainer $\mathbf{g}$ and the point of interest x as arguments and returns a scalar value for $\mathbf{g}$.

1) LIME: This XAI method aims to provide local, interpretable explanations for black-box models by approximating the model's behavior in the local region around a specific instance [17]. Consider an original instance $x \in R^d$ and let $g \in \mathcal{G}$ represent an explanation of the model $\mathcal{M}$, where $\mathcal{G}$ denotes a class of interpretable models suitable for visual representations to users (e.g., linear models). The explanation $\varphi(x)$ provided by Lime can be obtained through the following equation:

$$\varphi(x) = \underset{g\in\mathcal{G}}{\text{argmin}}\{\mathcal{L}(\mathcal{M}, g, \omega_x) + \Omega(g)\}. \tag{2}$$

In the case of a classification model $\mathcal{M}$, ω_x represents a proximity measure or weight between the original instance and the new one. A higher value of ω_x signifies a stronger similarity between the new and the original instances. $\mathcal{L}$ is a loss function used to measure the proximity between the predictions made by the explanation model and the original model, and $\Omega(g)$ quantifies the complexity of model g.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

Hence, LIME aims to train a locally interpretable model by minimizing the function $\mathcal{L}(\mathcal{M},g,\omega_x)+\Omega(g)$. Subsequently, predictions for an instance are made using the locally computed explanation model $\omega(x)$.

2) SHAPs: This method is also another popular method to interpret the output of ML models [18]. It is based on the concept of Shapley values from cooperative game theory [69] and explains each feature's contribution to the model prediction [70]. SHAP is typically applied to tabular data and exhibits the following properties: local accuracy, handling of missing data, and consistency [71]. Local accuracy ensures that the explanation model matches the original model. Missingness ensures that missing features in the original input do not have an impact. Consistency, where increasing the impact of a feature in the model should also result in a higher SHAP value for that feature, regardless of other features.

SHAP creates simplified inputs z by mapping x to z through $x = h_x(z)$. The original model $\mathcal{M}(x)$ can be approximated using binary variables with a linear function

$$\mathcal{M}(x) = g(z) = \varphi_0 + \sum_{i=1}^{d} \varphi_i Z_i \tag{3}$$

where $z = \{0, 1\}^d$, d represents the number of input features, $\varphi_0 = \mathcal{M}(h_x(0))$, and φ_i denotes the feature attribution value

$$\varphi_i = \sum_{S \in F \setminus \{i\}} \frac{|S|!(d - |S|! - 1)!}{|d|!} \big[\mathcal{M}_x(S \cup i) - \mathcal{M}(S)\big] \tag{4}$$

$$\mathcal{M}_x(S) = \mathcal{M}\Big(h_x^{-1}(z)\Big) = E\big[[\mathcal{M}(x)]|x_s\big] \tag{5}$$

where F represents the nonzero input set in z, S is the subset of F excluding the ith feature from F, and φ_i is the SHAP value, a unified measure of additive feature attributions.

Since computing $E[[\mathcal{M}(x)]|x_s]$ is particularly difficult, many approximation methods have been created, including Kernel SHAP, Deep SHAP, and Tree SHAP. In our study, we employed Tree SHAP due to its efficient computation of $E[[\mathcal{M}(x)]|x_s]$ values by leveraging decision tree structures, particularly well suited for ensemble learning models. With the number of trees T and the maximum number of leaves in any tree L, the initial complexity of computing $E[[\mathcal{M}(x)]|x_s]$ is $O(TL2^d)$. However, utilizing Tree SHAP with a maximum tree depth of h reduces the complexity to $O(TL2^2)$, significantly reducing the computational complexity from a high-order exponential level to a quadratic level.

So, Tree SHAP takes as input a trained tree-based model that was trained using input data D (an $m \times d$ matrix with m instances and d features). This input leads to the generation of an $m \times d$ matrix of SHAP values, where each value quantifies the contribution of a feature to the prediction of the respective instance.

E. Evaluation of XAI Methods

In the past, assessing attribution-based explainability's quality relied on qualitative and subjective evaluation. For instance, Eriksson and Grov [72] have used SHAP and LIME. To understand how XAI can be applied and customized to explain ML-generated alerts within IDSs by interviewing multiple security operations center (SOC) analysts and cybersecurity students [72]. This involved judging levels of satisfaction subjectively based on the explanation's utility. These evaluations were carried out by stakeholders of AI system. Nonetheless, due to the growing demand for more robust and objective evaluation methods, the field has recently shifted its attention toward crafting quantitative metrics. These metrics aim to measure the extent of quality and reliability in XAI techniques. Various quantitative metrics have been introduced in the literature to assess the results of explainability methods [20]. XAI evaluation is categorized into three distinct groups [20]: user-focused evaluation, application-focused evaluation, and functionality-focused evaluation. The initial two categories are viewed as components of human-centred evaluation and are further classified into subjective and objective measures. Four primary metrics, including high faithfulness, monotonicity, low complexity, and max sensitivity, were employed as suitable criteria for local explanations of LIME and SHAP in this work.

1) Faithfulness: The faithfulness metric $\mu_F(\mathcal{M}, g; x)$ measures how well the feature importance scores generated by the explanation function **g** reflect the actual importance of the features in the black-box model $\mathcal{M}$ for input x. This property is best computed using Pearson's correlation coefficient between the sum of the attributions of the features set to the best line value and the corresponding difference in the output values. Let $\mathcal{B}$ be the subset of indices whose features are set to a baseline value. Then the faithfulness metric is computed as follows:

$$\mu_F(\mathcal{M}, g; x) = \underset{\mathcal{B} \in \binom{|d|}{|\mathcal{B}|}}{\rho} \left(\sum_{i \in \mathcal{B}} g(\mathcal{M}, x)_i, \mathcal{M}(x) - \mathcal{M}(x_{\mathcal{B}}) \right) \tag{6}$$

where $x_{\mathcal{B}} = x_i | i \in \mathcal{B}\}$.

2) Monotonicity: Let $x, x' \in \mathcal{R}^d$ be two input points such that $x_i \leq x_i'$ for all $i \in 1, 2, \ldots, d$. $\mathcal{M}$ and g are said to be monotonic if the following condition holds: for any subset $S \subseteq 1, 2, \ldots, d$, the sum of the attributions of the features in S should be nonnegative when moving from x to x', that is

$$\sum_{i \in S} g(\mathcal{M}, x)i \leq \sum_{i \in S} g\big(\mathcal{M}, x'\big)_i$$

imply

$$\mathcal{M}(x) - \mathcal{M}\big(x_{[x_s = \bar{x}_s]}\big) \leq \mathcal{M}\big(x'\big) - \mathcal{M}\big(x'\big[x_s' = \bar{x}_s\big]\big).$$

3) Low Complexity: A complex explanation uses all d features to explain which features of x are important to $\mathcal{M}$. However, this explanation may be less interpretable, especially if d is large. To address this, a fractional contribution distribution is defined, where $|.|$ denotes the absolute value

$$P_g(i) = \frac{|g(\mathcal{M}, x)i|}{\sum j \in |d||g(\mathcal{M}, x)_j|}; P_g = P_g(1), \ldots, P_g(d). \tag{7}$$

Note that P_g is a valid probability distribution. Let $P_g(i)$ denote the fractional contribution of the characteristic x_i to the total magnitude of the attribution. If every feature had the same attribution, the explanation would be complex (even if it is faithful). The simplest explanation would be concentrated on one feature. complexity was defined as the entropy of P_g.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

TABLE II
SBS FEATURE SELECTION FEATURES FOR N-BAIOT, MEDBIOT, AND BOT-IOT DATA SETS FOR BINARY CLASSIFIERS

Dataset	Model	Feature Set	F1-score
N-BaIoT	XGB	[MI_dir_L5_weight , MI_dir_L1_weight , MI_dir_L0.01_weight]	0.9987
	RF	[MI_dir_L0.01_weight , HH_jit_L0.01_mean , MI_dir_L0.01_weight]	0.999
	ET	[MI_dir_L1_weight , MI_dir_L0.1_mean , HpHp_L0.01_weight]	0.999
	GBC	[HH_L0.01_radius , HH_L0.01_weight , HH_L1_mean]	0.9942
	LGBM	[HH_jit_L0.1_mean , HH_jit_L5_mean , HH_jit_L0.1_weight]	0.9992
Med-BIoT	XGB	[HH_L1_pcc, HH_L0.1_radius, HH_L0.01_pcc, HH_jit_L0.01_mean, HpHp_L5_pcc, HpHp_L3_magnitude, HpHp_L0.01_magnitude]	0.9988
	RF	[HH_L3_pcc, HH_L0.1_magnitude, HH_L0.01_weight, HH_jit_L0.01_mean, HH_jit_L0.01_std, HpHp_L3_pcc, HpHp_L0.01_magnitude]	0.9988
	ET	[HH_L1_pcc, HH_L0.01_weight, HH_L0.01_magnitude, HH_jit_L1_std, HH_jit_L0.01_mean, HpHp_L5_magnitude, HpHp_L0.01_magnitude]	0.9996
	GBC	[HH_L1_magnitude, HH_jit_L0.01_mean, HpHp_L5_pcc, HpHp_L3_magnitude, HpHp_L1_pcc, HpHp_L0.1_pcc, HpHp_L0.01_magnitude]	0.9988
	LGBM	[HH_L1_pcc, HH_L0.1_magnitude, HH_L0.01_pcc, HH_jit_L0.01_weight, HH_jit_L0.01_std, HpHp_L3_pcc, HpHp_L0.01_weight]	0.9996
BoT IoT	XGB	[drate , N_IN_Conn_P_SrcIP , srate , N_IN_Conn_P_DstIP]	0.9936
	RF	[N_IN_Conn_P_SrcIP , state_number , N_IN_Conn_P_DstIP , max]	0.9880
	ET	[seq , max , N_IN_Conn_P_SrcIP , N_IN_Conn_P_DstIP]	0.9674
	GBC	[srate , stddev , N_IN_Conn_P_SrcIP , N_IN_Conn_P_DstIP]	0.9851
	LGBM	[drate , N_IN_Conn_P_SrcIP , mean , N_IN_Conn_P_DstIP]	0.9917

Given a prediction $\mathcal{M}(x)$, an explanation function g, and a point x, the complexity of g at x is

$$\mu_C(\mathcal{M}, g; x) = -\sum_{i=1}^{d} P_g(i) \log P_g(i). \tag{8}$$

4) Max Sensitivity: To ensure that nearby inputs with similar model output have similar explanations, it is desirable for the explanation function g to have a low sensitivity in the region surrounding the point of interest x, assuming the differentiability of the predictor function $\mathcal{M}$.

The maximum sensitivity of an explanation function g at a point of interest x in its neighborhood is defined as follows.

Consider a neighborhood N_r of points within a radius r of x, denoted by $N_r = z \in D_x | p(x, z) \leq r$, $\mathcal{M}(x) = \mathcal{M}(x)(z)$, where D is the distance metric, and p is the proximity function. Given a predictor $\mathcal{M}(x)$, a distance metric D, a proximity function p, a radius r, and a point x, we define the maximum sensitivity of g at x as follows:

$$\mu_M(\mathcal{M}(x), g, r; x) = \max_{z \in N_r} D(g(\mathcal{M}(x), x), g(\mathcal{M}(x), z)). \tag{9}$$

In summary, Faithfulness measures the level of correlation between the importance scores for the characteristics identified by the explainers and the model output. The monotonicity metric evaluates whether the incremental changes in the input instance are reflected in the explanations in a consistent way. The complexity metric quantifies the number of features to explain the model decisions. The robustness of the explainers is tested against the sensitivity metric, which expects the closer instances in the feature space to have similar explanations.

The experiments were carried out on a computer running Pop!_OS 22.04 LTS x86_64 operating system with the following hardware configuration: 32 GB of DDR4-2666R ECC RAM, AMD Ryzen 5 5600G with Radeon Graphics (12) @ 3.900-GHz processor. The scripts were developed using the Python 3 programming language, and Scikit-learn and mlxtend were used.

IV. RESULTS

This section presents the results of our study, which are divided into two main parts. Section IV-A explores the binary classification of the three data sets, with an emphasis on the evaluation of XAI techniques. Sections IV-B and IV-C discuss multiclassification results and delve into the application of XAI techniques for N-BaioT, MedBIoT, and BoT IoT data sets.

A. Explaining Binary Classifiers for Botnet Detection

We employed SBS to determine the top k-best features appropriate for binary classification across three distinct data sets. For the N-BaIoT data set, the k-best feature set consisted of three features; for Med-BIoT data set, it was seven; for the BoT-IoT data set, it was four. Table II shows SBS feature selection results across various models for the N-BaIoT, MedBIoT, and BoT-IoT data sets used in binary classification.

In N-BaIoT data set, all models achieved F1-scores exceeding 99% with the top three optimal k-feature subsets. SBS-XGBoost (XGB) emphasized Host-based (MI) features, SBS-LightGBM (LGBM) derived Network Jitter (HH_jit) features, and SBS-GBC obtained Socket features. SBS-Extra Trees (ET) and SBS-RF effectively combined features from diverse network categories.

The network traffic category features of the MedBIoT data set are the same as those in the N-BaIoT data set. However, Torii botnets were also deployed to infect IoT devices, in addition to the Gafgyt and Mirai botnets for Med-BIoT data set. In contrast to N-BaIoT, the optimal feature sets using SBS for the Med-BIoT data set are predominantly derived from Socket (HpHp), Network Jitter (HH_jit), and Channel-based (HH) features. XGB exhibited the highest performance,

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

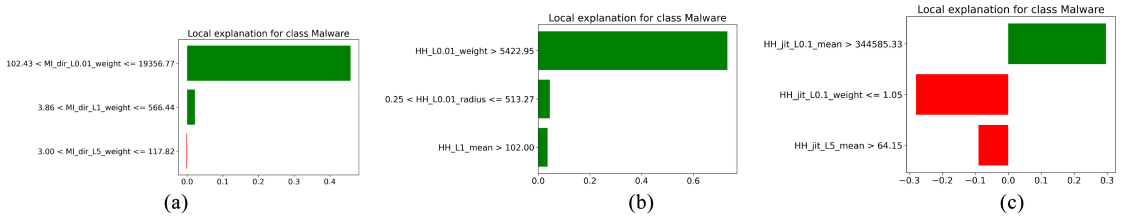


Fig. 2. LIME local explanations of malware instance of N-BaIoT data set for binary classifiers. (a) LIME explanations for XGB model over host-based category features. (b) LIME explanations for GBC model over socket features. (c) LIME explanations for LGBM over network-jitter category features.

TABLE III
ACTUAL DATA POINTS OF N-BAIOT DATA SET ACROSS THREE NETWORK CATEGORIES

Network Category	Model	Features	Data points
Host based	XGB	MI_dir_L5_weight	108.087471
		MI_dir_L1_weight	266.816511
		MI_dir_L0.01_weight	5843.19767
Net work jitter	LGBM	HH_jit_L5_mean	1.50591E+09
		HH_jit_L0.1_weight	1.00000E+00
		HH_jit_L0.1_mean	1.50591E+09
Socket Based	GBC	HH_L0.01_radius	4.00513E+01
		HH_L0.01_weight	3.73916E+04
		HH_L1_mean	5.53999E+02

achieving F1 score of 0.9936. Following closely, RF and LGBM also demonstrated strong performance with F1 scores of 0.9880 and 0.9917, respectively. ET performed less, attaining an F1 score of 0.9674. The remaining classifiers displayed notable performance, with F1 scores ranging from 0.9674 to 0.9851. Overall, these results emphasize the effectiveness of the selected 7-feature set using SBS for Binary Classifiers.

Within the BoT-IoT data set, it is evident that certain features, such as the number of inbound connections per source IP (N_IN_Conn_P_SrcIP) and the number of inbound connections per destination IP (N_IN_Conn_P_DstIP), are essential for attack detection across all models. Notably, XGB and LGBM achieved the highest F1-scores by incorporating additional features like destination-to-source packets per second (drate), source-to-destination packets per second (state), the average duration of aggregated records (mean), and the maximum duration of aggregated records. Additionally, it is worth noting that XGB and LGBM achieved the highest F1 scores.

To explain the outcomes of our black-box models, we employed two model-agnostic feature-importance XAI techniques: 1) LIME and 2) SHAP. In this section, we present local explanations generated by the SHAP and LIME techniques and evaluate the quality of these explanations.

The LIME method explains the rationale behind assigning probabilities to each class by comparing the probability values with the actual class of the data point. As a use case for illustrating local explanations provided by explainers, we selected a single actual malware data point (index 13 from the test data, see Table III) from each of the three network categories: Host-based, network jitter, and socket-based. The explanations generated by LIME can be observed in Fig. 2 for binary classifiers of Botnet detection. In these visualizations, green bars indicate features contributing to predicting a data point as malware, while red bars signify features contributing to predicting a data point as benign. Fig. 2 displays LIME local explanations for the N-BaIoT data set. For example, in Fig. 2(a), LIME explanations for the features in the Host network category demonstrate that the XGB Model predicted as malware class with 100% accuracy for the actual class label datapoint (mentioned in Table III). These explanations include the following rules.

1) 102.43 < MI_dir_L0.01_weight <= 19356.77: If the packet count of the host-based(MI) feature captured in a 1-min time window (L0.01) falls within the range of 102.43–19356.77, the XGB model is more inclined to classify the data point as malware, indicating a potential compromise of the host by malware.
2) 3.86 < MI_dir_L1_weight <= 566.44: If the packet count (weight) of the host-based (MI) feature captured in a 1.5-s time window (L1) falls within the range of 3.86–566.44, the XGB model implies a potential presence of IoT botnet malware. While this range indicates a lower level of network activity compared to the significant activity represented by the 1-min time window, it still signifies notable communication occurring within a shorter time frame. Data points within this range raise suspicion and might imply the existence of IoT botnet malware on the host system.
3) 3.00 < MI_dir_L5_weight <= 117.82: When the packet count of the host-based feature within 100 ms exceeds 117.28, the XGB model classifies the data point as benign, indicating that any malware does not compromise the host. Pocket data points within this range are considered normal and nonmalicious, indicating regular network traffic within extremely short time intervals.

Similar LIME explanations (for a datapoint in Table III) are shown in Fig. 2(b) and (c) for other network categories, with the GBC model predicting as malware class for the socket-based feature set and the LGBM model also predicting as malware for network jitter features over N-BaIoT data set.

The features of the MedBIoT data set are similar to those of the N-BaIoT data set. Thus, explanations generated by LIME and SHAP for the MedBIoT data set are similar. For illustrative purposes, we chose a single data point from the Med-BIoT data set. The explanations generated by LIME for

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

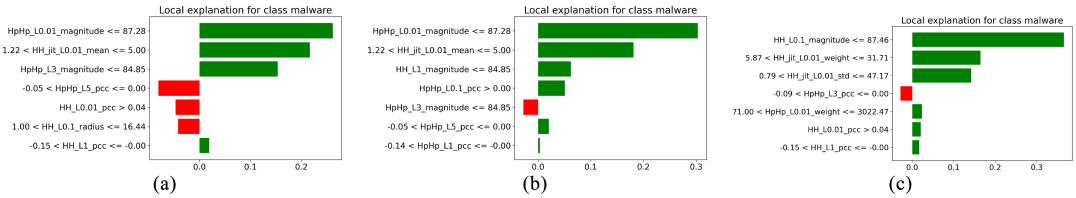


Fig. 3. LIME local explanations of malware instance of Med-BaIoT data set for binary classifiers. (a) LIME explanations for XGB model. (b) LIME explanations for GBC. (c) LIME explanations for LGBM.

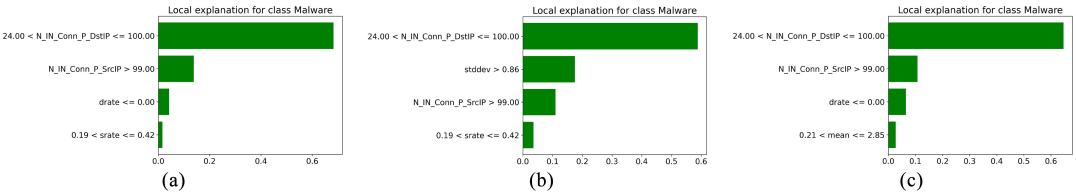


Fig. 4. LIME local explanations of malware instance of BoTIoT data set for binary classifiers. (a) LIME explanations for XGB model. (b) LIME explanations for GBC model. (c) LIME explanations for LGBM.



Fig. 5. SHAP local explanations of malware instance of N-BaIoT data set for binary classifiers. (a) SHAP explanations for XGB model over data point of host-based category features. (b) SHAP explanations for GBC model over data point of socket category features. (c) SHAP explanations for LGBM model over data point of network-jitter.

models XGB, GBC, and LGBM can be seen in Fig. 3(a)–(c), respectively. All these models predicted the actual data point as malware with 100%. Green bars show that features help to predict the malware, and the Red bar shows the features which help to predict the instance as Benign.

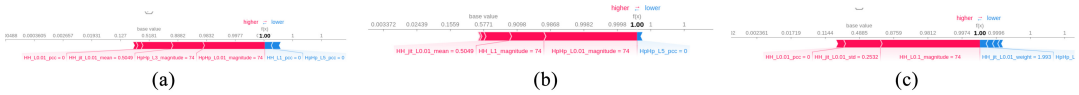
In the case of the BoT IoT data set, the models correctly predicted the actual data point as malware. The corresponding LIME explanations can be observed in Fig. 4(a)–(c). corresponding to models XGB, GBC, and LGBM.

Additionally, SHAP is widely employed for model explanations and furnishes local and global explanations. In local explanations, a specific data point is chosen, and the model prediction is explained to showcase the contribution of each feature. SHAP calculates Shapley values, demonstrating the contribution of features on model predictions. SHAP values were calculated over the Test data set. Fig. 5 illustrates a local explanations for a malware data point (see Table III) of N-BaIoT data set using a shap's force plot for XGB, GBC, and LGBM models, displaying the contribution of each feature to the prediction. The plot shows the base value, and the features containing a positive influence on the prediction are in red, and the features showing a negative influence on the predictions are in blue. The base value in the plots is the average of all prediction values. Each strip in the plot illustrates how the features influence the predicted value, either drawing it closer or pushing it farther away from the base value. Red strip features push the value to higher values, whereas blue strip features push the value to lower values. The contribution of features holding broader strips is more.

For example, Fig. 5(c) demonstrates the SHAP local explanations of LGBM model over network jitter category features for a data point (see Table III) over N-BaIoT data set. The base value is 0.566. Features HH_jit_L0.1_mean and HH_jit_L5_mean positively contribute to the prediction value, while feature HH_jit_L0.1_weight has a negative impact. HH_jit_L0.1_mean is the most crucial feature, as the contribution has a broader range. The total positive contribution is greater than the negative contribution, and the final predicted value is greater than the base value. As a result, the class is predicted as malware. Similarly, In Fig. 5(a), XGB model with Host-Based features reveals a base value of 0.4932 and a predicted value of 1 (for malware data point in Table III). MI_dir_L0.01_weight has a broader range and is the most crucial feature. In Fig. 5(b), the GBC model with socket features determines a base value of 0.5979. Here, HH_L0.01_weight was the most crucial feature, contributing to a predicted value of 1 for the malware class over N-BaIoT data set.

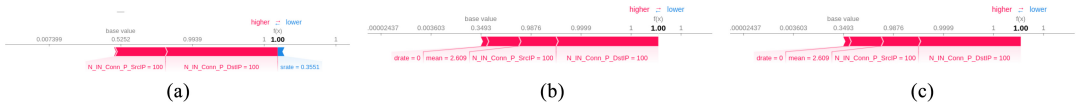
Similarly, when explaining a selected record of the Med-BaIoT data set using SHAP, it was observed from Fig. 6 that the base values were 0.5181 for XGB, 0.5771 for GBC, and 0.4885 for LGBM models. All three models predicted the data point as malware. Notably, the feature HpHp_L0.01_magnitude had a substantial impact and a wider range in Fig. 6(a) and (b) for XGB and GBC models. Additionally, in Fig. 6(c), the HH_L0.1_magnitude feature exhibited a broader range and is the most important feature for LGBM.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.



(a) (b) (c)

Fig. 6. SHAP local explanations of malware instance of Med-BaIoT data set for binary classifiers. (a) SHAP explanations for XGB. (b) SHAP explanations for GBC. (c) SHAP explanations for LGBM model.



(a) (b) (c)

Fig. 7. SHAP local explanations of malware instance of BoTIoT data set for binary classifiers. (a) SHAP explanations for XGB model. (b) SHAP explanations for GBC model. (c) SHAP explanations for LGBM model.

For BoT-IoT data set, base values from SHAP force plots [see Fig. 7(a)–(c)] were 0.5252, 0.3493, and 0.3493 for models XGB, GBC, and LGBM, respectively. Notably, N_IN_Conn_P_SrcIP had a wide range and was identified as the most important feature, followed closely by N_IN_Conn_P_DstIP. Both LIME and SHAP explainers for BoT IoT models highlighted these two features as the most significant contributors to model predictions.

In our research, we quantitatively evaluated the quality of XAI methods, specifically LIME and SHAP, using four metrics: Faithfulness, Monotonicity, Complexity, and Sensitivity. Faithfulness evaluates the correlation between the importance assigned by the XAI method to features and their impact on the prediction probabilities of the model. A high faithfulness of the XAI method indicates that the assigned feature importance closely aligns with their impact on the model's prediction probabilities, ensuring accurate and trustworthy explanations. Monotonicity measures the influence of individual features on model prediction probabilities by evaluating how the prediction probability changes when each attribute is incrementally added in order of increasing importance. As each feature is added, the model's probability consistently increases, resulting in monotonically increasing model prediction probabilities. A high monotonicity score suggests that the explanations by XAI method are consistent with the model's predictions for the given input. Besides, we calculate the low complexity metric by computing the entropy of feature attribution obtained by the XAI method. Likewise, the sensitivity metric evaluates the robustness of the explanations, ensuring that nearby inputs in the feature space have similar explanations output when the sensitivity value is low. In the sensitivity metric, for acquiring the nearest neighbor points associated with the prediction label of the explanations score, we utilized the Euclidean distance with a radius value of 0.1, which helps find data points in the feature space most proximate to the instance and have similar explanations for the predicted label.

We evaluated LIME and SHAP explanations across 2000 test points. Table IV displays the XAI metric results, including mean and standard deviations, for LIME and SHAP explanations across 2000 test points for binary classifiers (ET, LGBM, GBC, RF, and XGB) on N-BaIoT, Med-BaIoT, and BoT IoT data sets.

For the N-BaIoT data set, the evaluation of faithfulness for both LIME and SHAP shows varying performance across models. However, SHAP consistently outperforms LIME in capturing the models' behavior with higher faithfulness correlation mean values, especially for the LGBM and XGB models. Notably, the XGB model, when explained using SHAP, achieves an exceptional faithfulness score ($\mu_f = 0.99 \pm 0.13$), indicating the high fidelity of the explanations provided by SHAP for this model. Additionally, by evaluating the monotonicity of the explainer, a measure of how the explanations change monotonically with respect to the input features, both LIME and SHAP reveal high monotonicity values, implying their ability to provide consistent explanations across different models. Specifically, SHAP explanations for LGBM and XGB models, achieving monotonicity scores above 99%. Complexity measures the conciseness of explanations provided by the explainer. Lower complexity values indicate simpler and more interpretable explanations. SHAP explainer for XGB produced the most concise explanations among all models and explainers, with the lowest complexity ($\mu_c = 0.50 \pm 0.25$). Sensitivity measures the stability of explanations for nearby data points. Lower sensitivity values indicate more stable explanations. So, XGB model explained by SHAP showed the lowest sensitivity ($\mu_s = 0.001 \pm 0.001$), suggesting highly stable and reliable explanations for nearby data points within the same feature space used by XGB model.

Similarly, when evaluating the explainers for binary classifiers of both the Med-BIoT and BoT IoT data sets (see Table IV), SHAP explanations consistently outperform LIME for all the models. Notably, when applied to XGB, SHAP's explanations were higher fidelity, greater consistency, lower complexity, and more robust than LIME.

B. Explaining Multiclass Classifiers for Botnet Malware type

Similar to Section IV-A, we evaluated the quality of explanations obtained after applying LIME and SHAP to multiclass models, explicitly focusing on the detection of botnet malware type. In this study phase, we employed N-BaIoT and MedBIoT data sets.

N-BaIoT data set includes network traffic data from nine distinct IoT devices infected with Mirai and Gafgyt malware alongside legitimate traffic. Conversely, the MedBIoT data set

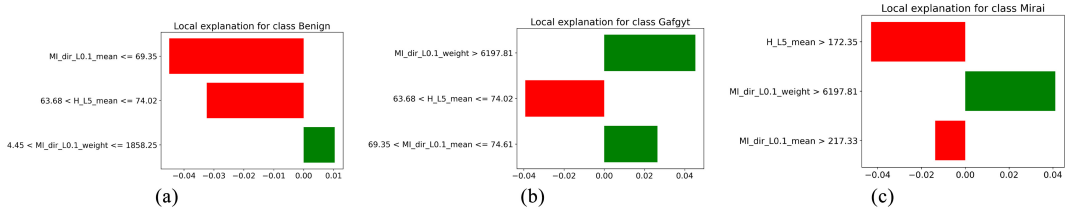
Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
RESULTS OF EVALUATING THE QUALITY OF LIME&SHAP USING FAITHFULNESS (μ_f), MONOTONICITY (μ_m), COMPLEXITY (μ_c), AND SENSITIVITY(μ_s) FOR IOT MALWARE DETECTION BINARY CLASSIFICATION MODELS OVER THREE DATA SETS: N-BAIOT, MEDBIOT, AND BOT-IOT

Dataset	Model	ET		LGBM		GBC		RF		XGB	
	XAI Metric\XAI method	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP
N-BaIoT	μ_f	0.28 ± 0.75	0.34±0.73	0.65 ± 0.65	0.84 ± 0.21	0.46 ± 0.67	0.48 ± 0.67	0.36 ± 0.65	0.91 ± 0.23	0.77 ± 0.57	**0.99 ± 0.13**
	μ_m	94.80%	95.38%	91.55%	99.25%	57.33%	73.08%	75.23%	79.33%	97.70%	**98.15%**
	μ_c	0.97 ± 0.13	0.80 ± 0.21	0.90 ± 0.24	0.99 ± 0.16	1.14 ± 0.14	0.85 ± 0.23	0.97 ± 0.11	0.91 ± 0.14	0.72 ± 0.24	**0.50 ± 0.25**
	μ_s	1.71 ± 1.54	0.02 ± 0.01	2.42 ± 2.22	0.12 ± 0.07	7.88 ± 10.35	0.09 ± 0.06	0.04 ± 0.02	0.01 ± 0.02	0.05 ± 0.02	**0.001 ± 0.001**
MedBIoT	μ_f	0.11 ± 0.84	0.14 ± 0.93	0.80 ± 0.39	0.87 ± 0.40	0.08 ± 0.85	0.68 ± 0.48	0.71 ± 0.57	0.87 ± 0.31	0.81 ± 0.40	**0.95 ± 0.12**
	μ_m	92.28%	94.34%	72.16%	84.76%	84.16%	87.70%	75.18%	78.28%	94.10%	**94.34%**
	μ_c	0.92 ± 0.09	0.87 ± 0.13	0.93 ± 0.09	0.73 ± 0.13	0.82 ± 0.17	0.79 ± 0.08	1.09 ± 0.01	0.76 ± 0.05	0.75 ± 0.09	**0.60 ± 0.19**
	μ_s	0.28 ± 0.71	0.03 ± 0.05	1.06 ± 3.33	0.03 ± 0.02	3.53 ± 9.03	0.10 ± 0.18	0.03 ± 0.05	0.01 ± 0.04	0.02 ± 0.01	**0.01 ± 0.01**
BoT-IoT	μ_f	0.43 ± 0.49	0.46 ± 0.53	0.06 ± 0.76	0.53 ± 0.37	-0.13 ± 0.48	0.20 ± 0.48	0.35 ± 0.37	0.59 ± 0.26	0.64 ± 0.38	**0.73 ± 0.24**
	μ_m	62.95%	63.98%	26.60%	37.41%	27.03%	30.30%	52.02%	60.40%	86.10%	**89.01%**
	μ_c	1.54 ± 0.21	1.34 ± 0.26	1.60 ± 0.12	1.46 ± 0.17	1.52 ± 0.09	1.50 ± 0.18	1.65 ± 0.09	1.55 ± 0.12	1.18 ± 0.12	**0.91 ± 0.28**
	μ_s	0.55±0.99	0.44±0.03	0.10±0.11	0.50±0.71	2.07±2.41	0.12±0.15	0.03±0.05	0.01±0.02	0.01±0.05	**0.01±0.01**

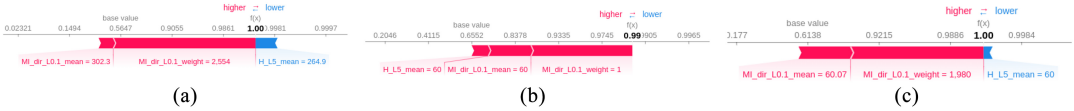
TABLE V
SBS FEATURE SELECTION FEATURES FOR MULTICLASS (BOTNET DETECTION) FOR N-BAIOT AND MEDBIOT DATA SETS

Dataset	Model	Feature set	F1-Score
N-BaIoT	GBC	[MI_dir_L0.1_weight, MI_dir_L0.01_variance, HH_L5_magnitude]	0.9995
	LGBM	[MI_dir_L0.01_weight, HH_L0.1_weight, MI_dir_L5_weight]	0.9919
	XGB	[MI_dir_L0.1_weight, MI_dir_L0.1_mean, H_L5_mean]	0.9995
	RF	[MI_dir_L5_weight, MI_dir_L0.1_mean, MI_dir_L0.01_weight]	0.99917
	ET	[MI_dir_L1_weight, MI_dir_L0.1_weight, MI_dir_L0.1_mean]	0.99933
Med-BaIoT	GBC	[HH_L3_pcc, HH_L0.1_pcc, HH_L0.01_magnitude, HH_jit_L5_weight, HH_jit_L0.01_mean, HpHp_L0.01_weight, HpHp_L0.01_radius]	0.9948
	XGB	[MI_dir_L0.01_weight, HH_L3_pcc, HH_L1_pcc, HH_L0.01_radius, HH_jit_L0.01_mean, HH_jit_L0.01_std, HpHp_L0.01_magnitude]	0.9954
	LGBM	[HH_L0.01_magnitude, HH_L0.01_pcc, HH_L0.1_magnitude, HH_L1_pcc, HH_jit_L0.01_mean, HH_jit_L0.01_weight, HpHp_L5_pcc]	0.9924
	RF	[HH_L3_pcc, HH_L0.1_radius, HH_L0.1_pcc, HH_L0.01_radius, HH_jit_L0.01_weight, HH_jit_L0.01_std, HpHp_L0.01_magnitude]	0.9942
	ET	[HH_L1_magnitude, HH_L0.1_pcc, HH_L0.01_radius, HH_L0.01_pcc, HH_jit_L0.01_std, HpHp_L0.01_magnitude, HpHp_L0.01_pcc]	0.995



(a) (b) (c)

Fig. 8. LIME explanations for XGB model in multiclass (botnet malware type) instance of the N-BaIoT data set. (a) LIME explanations for XGB over benign instance. (b) LIME explanations for XGB over Gafgyt instance. (c) LIME explanations for XGB over Mirai instance.



(a) (b) (c)

Fig. 9. SHAP explanations for XGB model in multiclass (botnet malware type) instance of the N-BaIoT data set. (a) SHAP explanations for XGB over benign instance. (b) SHAP explanations for XGB over Gafgyt instance. (c) SHAP explanations for XGB over Mirai instance.

contains malicious network traffic induced by Mirai, BashLite, and Torii botnet malware deployed on 83 real or emulated IoT devices. Therefore, the detection of botnet malware type involves three classes (Mirai, Gafgyt, and Benign) for the N-BaIoT data set and four classes (Mirai, BashLite, Torii, and Benign) for the Med-BaIoT. Table V shows the results of SBS feature selection for multiclass malware detection using various models on N-BaIoT and MedBIoT data sets. In the context of multiclass classification, the k-best feature set for the N-BaIoT data set is 3, while for the Med-BIoT

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

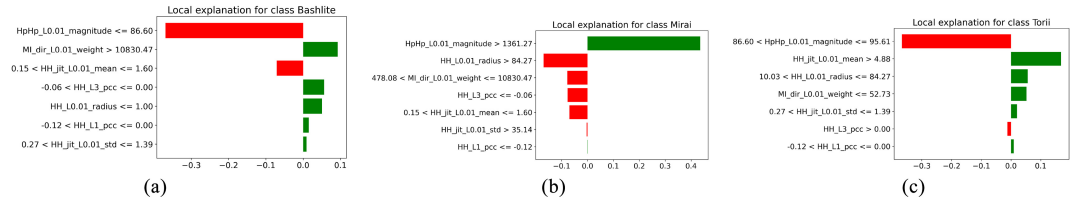


Fig. 10. LIME explanations for XGB model in multiclass (botnet malware type) instance of the Med-BaIoT data set. (a) LIME explanations for XGB over Bashlite instance. (b) LIME explanations for XGB over Mirai instance. (c) LIME explanations for XGB over Torii instance.

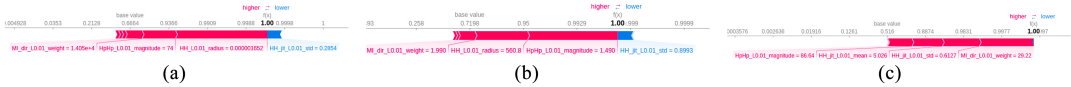


Fig. 11. SHAP explanations for XGB model in multiclass (botnet malware type) instance of Med-BaIoT data set. (a) SHAP explanations for XGB over Bashlite instance. (b) SHAP explanations for XGB over Mirai instance. (c) SHAP explanations for XGB over Torii instance.

TABLE VI
RESULTS OF EVALUATING THE QUALITY OF LIME&SHAP USING FAITHFULNESS (μ_f), MONOTONICITY (μ_m), COMPLEXITY (μ_c), AND SENSITIVITY(μ_s) FOR BOTNET MALWARE TYPE (MULTICLASS) ON TWO DATA SETS: N-BAIOT AND MEDBIOT

Dataset	Model	ET		LGBM		GBC		RF		XGB	
	XAI Metric	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP
N-BaIoT	Faithfulness	-0.026±0.757	-0.060±0.852	0.395±0.641	0.239±0.573	-0.124±0.502	0.268±0.033	0.130±0.615	0.152±0.608	0.898±0.129	**0.907±0.382**
	Monotonocity	74.70%	73.30%	56.90%	66.10%	57.70%	48.70%	58.60%	64.70%	89.00%	**100.00%**
	Complexity	1.017±0.086	0.994±0.079	0.772±0.195	0.753±0.109	0.926±0.542	0.887±0.215	0.990±0.092	0.885±0.155	0.710±0.236	**0.686±0.236**
	Sensitivity	0.021±0.012	0.054±0.137	0.028±0.023	0.034±0.015	0.789±1.995	0.878±0.324	0.025±0.012	0.082±0.204	0.042±0.043	**0.002±0.001**
Med-BaIoT	Faithfulness	0.11±0.38	0.46±0.43	0.05±0.32	0.29±0.45	0.12±0.59	0.20±0.49	0.31±0.38	0.57±0.25	0.62±0.27	**0.81±0.23**
	Monotonocity	36.00%	45.00%	35.00%	41.00%	28.00%	49.00%	27.00%	58.00%	78.00%	**84.00%**
	Complexity	1.45±0.23	1.41±0.30	1.59±0.15	1.57±0.14	1.68±0.15	1.92±0.22	1.64±0.11	1.48±0.20	1.09±0.31	**0.91±0.19**
	Sensitivity	0.02±0.01	0.00±0.01	0.82±1.46	0.05±0.02	0.74±0.23	0.04±0.02	0.03±0.03	0.01±0.03	0.02±0.09	**0.01±0.01**

data set, it is 7. Notably, all feature sets produced a strong performance in terms of F1 score, with models achieving scores exceeding 0.99. The features "MI_dir_L0.1_weight" and "MI_dir_L0.01_weight" appeared in multiple models, including XGB, LGBM, and ET, indicating they are vital in detecting botnet malware.

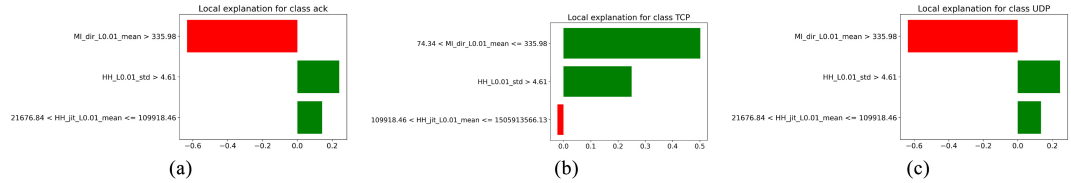
To guarantee the readability of our results, we have reported only results pertaining to explaining XGB model using LIME and SHAP. So the feature set from SBS-XGB is ["MI_dir_L1_weight," "MI_dir_L0.1_weight," and "MI_dir_L0.1_mean"]. As an illustrative use case demonstrating local explanations provided by explainers for multiclass classifiers in the context of botnet malware type detection, we have selected an instance from each class, shown in Figs. 8 and 9 for the N-BaIoT data set. In the figures, green bars indicate features contributing to predicting a data point for the particular selected class label, while the red color shows contributions for the remaining other class labels. In Fig. 8(a)–(c), LIME explanations were presented for instances corresponding to the labels Benign, Mirai, and Gafgyt, respectively. XGB model correctly predicted all these instances with high detection rates for each selected instance label. Additionally, Fig. 9 also showcases SHAP explanations for botnet malware type detection using the XGB model. It can be observed that "MI_dir_L0.1_weight" had a broader strip in all three figures [Fig. 9(a)–(c)] for class labels (Benign, Mirai, and Gafgyt), indicating its significance as the most essential feature for Botnet malware type detection in N-BaIoT data set and the predicted value is 1 for all three class labels.

In the context of botnet malware type detection using the XGB model, we similarly selected a data point in Med-BIoT data set. Fig. 10 presents LIME explanations for instances belonging to class labels Bashlite [Fig. 10(a)], Mirai [Fig. 10(b)], and Torii [Fig. 10(c)]. The XGB model accurately predicted class labels corresponding to their respective actual labels. On the other hand, Fig. 11 Shows the SHAP explanations for the same class labels. In Fig. 11(a), "HH_L0.01_radius" has a broader strip, indicating its significance as the most essential feature. In Fig. 11(b), "HpHp_L0.01_magnitude" is identified as the most influential feature for the Mirai class label, while "MI_dir_L0.01_weight" is highlighted [in Fig. 11(c)] as an essential feature for the Torii class label. The predicted value for all three class labels is 1 by the XGB model.

For botnet malware type detection, Table VI provides results of the quality of explanations using both LIME and SHAP for various models. For both data sets, Evaluating explanation quality has varied performance among models using explainers.

In the N-BaIoT data set, Faithfulness calculates the correlation between feature importance and model predictions, indicating that the XGB model consistently outperforms others, showing high correlation values for both LIME

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.



(a) (b) (c)

Fig. 12. LIME explanations for XGB model in multiclass (botnet attack type) instance of N-BaIoT data set. (a) LIME explanations for XGB over ACK attack instance. (b) LIME explanations for XGB over TCP attack instance. (c) LIME explanations for XGB over UDP attack instance.



(a) (b) (c)

Fig. 13. SHAP explanations for XGB model in multiclass (botnet attack type) instance of N-BaIoT data set. (a) SHAP explanations for XGB over ACK attack instance. (b) SHAP Explanations for XGB over TCP attack instance. (c) SHAP explanations for XGB over UDP attack instance.

TABLE VII
SBS FEATURE SELECTION FEATURES FOR XGB IN BOTNET ATTACK TYPE OVER N-BAIOT DATA SET

Dataset	Model	Feature set	F1-Score
N-BaIoT	XGB	[MI_dir_L0.01_mean, HH_L0.01_std, HH_jit_L0.01_mean]	0.99591

TABLE VIII
RESULTS OF EVALUATING THE QUALITY OF LIME&SHAP FOR XGB MODEL FOR IOT BOTNET ATTACK TYPE DETECTION

Datast	Detection type	Model	XGB	
N-BaIoT	Attack type	Metric	LIME	SHAP
		Faithfulness	0.12±0.27	0.66±0.21
		Monotonocity	47.00%	59.00%
		Complexity	1.48±0.13	1.28±0.07
		Sensitivity	0.12±0.7	0.03±0.11

(0.898±0.129) and SHAP (0.907±0.382). Monotonicity, highlighting the consistency of explanations, demonstrates that the XGB model consistently maintains high monotonicity scores, mainly with SHAP (100.00%). Regarding complexity, which evaluates explanation conciseness, XGB with SHAP (0.686±0.236) has the most concise explanations among all models. Sensitivity computes explanation stability for nearby data points and underscores the XGB model's stability, particularly with SHAP, showing the lowest sensitivity (0.002±0.001). Overall, results suggest that SHAP consistently provides more reliable and faithful explanations than LIME across different models.

For Med-BaIoT data set, the XGB model explained by SHAP achieved the highest Faithfulness (0.81±0.23). SHAP consistently outperforms LIME across all models, with XGB achieving an exceptional monotonicity score of 84.00%. Regarding complexity, which evaluates explanation conciseness, SHAP consistently produces more concise explanations than LIME for all models. The stability of explanations of SHAP, notably for XGB model, is highlighted with the lowest sensitivity (0.01±0.01). Overall, results in Table VI show that SHAP consistently outperforms LIME.

C. Explaining Multiclass Classifiers for Botnet Attack type

This section provides the results of XGB model explanations using LIME and SHAP for IoT botnet attack types. N-BaIoT data set was chosen for this scenario. To explain the attack type classifiers, data points are categorized into distinct types: ACK, benign, compact, junk, scan, syn, TCP, UDP, and UDP plain. The feature set for the attack type is ["MI_dir_L0.01_mean," "HH_L0.01_std," and "HH_jit_L0.01_mean"] (see Table VII). The XGB model achieved more than 99% of F1-score.

Figs. 12 and 13 illustrate LIME and SHAP explanations for the XGB model regarding attack type detection. Similar to the multiclassifier for botnet malware types, in the case of attack types, LIME explanations use green bars to indicate features contributing to predicting a data point for the selected class label. In contrast, red bars represent contributions to other labels. For illustrative purposes, we selected instances of the ACK, UDP, and TCP attacks. Fig. 12(a)–(c) displays LIME explanations, showcasing XGB accurate prediction of actual labels as ACK, UDP, and TCP. On the other hand, Fig. 13 presents SHAP explanations for the XGB model. HH_jit_L0.01_mean (Network Jitter) feature notably shows more strips for labels. Predicted values for ACK and TCP attacks [Fig. 13(a) and (b)] are both 1, indicating correct predictions. However, for the UDP attack Fig. 13(c), the predicted value is 0.98.

Results of the quality of explainers for the XGB model in attack type detection in the N-BaioT data set, as shown in Table VIII, demonstrate that SHAP consistently outperforms LIME across multiple evaluation metrics. Specifically, SHAP achieves substantially higher Faithfulness (0.66±0.21) than LIME (0.12±0.27), indicating a stronger correlation between feature importance and model predictions. Furthermore, SHAP demonstrates a higher monotonicity score (59%) than LIME (47%), highlighting more consistent explanations. Regarding complexity, SHAP provides more concise explanations with a lower complexity score (1.28±0.07) than LIME (1.48±0.13). Regarding sensitivity, SHAP displays more excellent stability with the lowest sensitivity score (0.03±0.11), while

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

LIME shows a higher sensitivity (0.12±0.7). Across all metrics, SHAP emerges as the superior explainer for the XGB model in the Med-BaIoT data set, offering more faithful, consistent, concise, and stable explanations than LIME.

V. Discussion

In this study, we examined the performance of various ML classifiers with their respective feature sets in predicting malware infections in IoT devices based on three data sets: 1) N-BaIoT; 2) MedBIoT; and 3) BoT-IoT. We used LIME and SHAP explainers to deduce interpretable explanations of the model predictions. The following discussion is based on the results obtained from the analysis of each data set.

In this study, SBS was used for each data set to identify a subset of features that significantly enhanced the predictive performance of the classifiers. Although the selected features varied between data sets, network traffic category features such as host, channel, network jitter, and socket consistently emerged as crucial factors in predicting IoT botnet malware (binary) and botnet type (multiclass) across various data sets. Host-based features were pivotal in the N-BaIoT data set, while channel- and socket-based features were more prominent in the Med-BIoT data set. Furthermore, the number of inbound connections per source IP and inbound connections per destination IP were identified as influential features of the BoT-IoT data set. Across all three data sets, XGB model consistently achieved the highest performance in terms of the F1 score, which indicates that the XGB model is an effective and robust choice for malware detection in IoT devices. RF and LGBM also performed well in most cases, while ET generally had the lowest performance.

Our results show that both the LIME and SHAP explanations exhibit high consistency and interpretability for the XGB model with the SBS-XGBoost feature set, as evidenced by the high monotonicity scores. This indicates that the explanations provided by both LIME and SHAP capture the underlying relationships between the feature set and the values of the prediction (malware and benign) of network traffic in malware classification consistently and interpretably. Moreover, both the LIME and SHAP explanations have high faithfulness correlation mean values, suggesting that security analysts can have a high degree of trust in the explanations provided for XGB model. The complexity scores are lower, in addition to high trustworthiness, which can help analysts better understand the model predictions and make informed decisions based on the model outputs. The explanations for XGB are also more robust due to lower sensitivity values, indicating that nearby inputs with similar model outputs have similar explanations.

Figs. 14–16 show the evaluation metrics for the LIME and SHAP distributions, providing comparative analysis results of the respective models for Multiclass from the N-baIoT and med-BaIoT data sets, as well as binary class from the BoT IoT data set. The evaluation of LIME and SHAP, using four metrics, faithfulness correlation, monotonicity, complexity, and sensitivity, on XGB model with its respective feature set (SBS-XGB), reveals promising distribution results. For

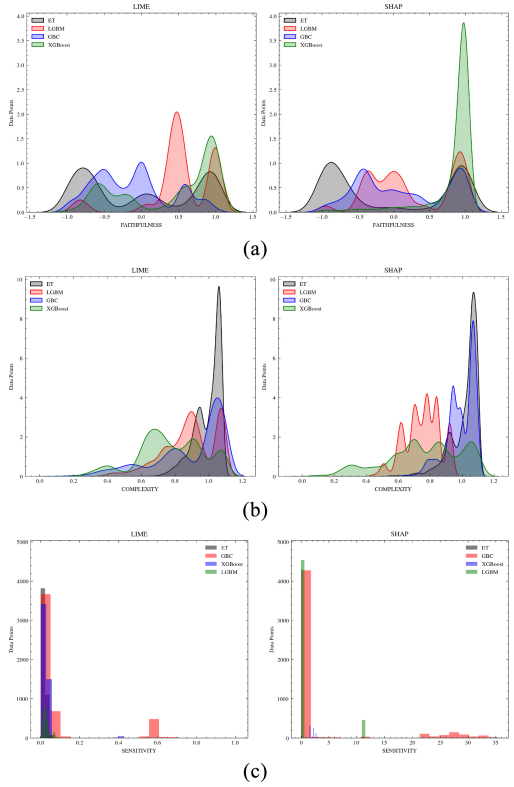


Fig. 14. Distribution of LIME and SHAP XAI metrics over N-BaIoT data set for models. (a) High faithfulness. (b) Low complexity. (c) Low sensitivity.

instance, the faithfulness correlation values are skewed toward higher levels, demonstrating a high degree of consistency through monotonicity when compared to alternative models. Moreover, the entropy values of feature importance scores for explainers tend to be distributed more toward the lower end in comparison to those of other models. The feature importance scores of explainers for the XGB model remain closely clustered compared to other models, particularly when assessed using sensitivity as an evaluation metric. For the GBC model with its respective feature set (SBS-GBC), the faithfulness correlation values for LIME and SHAP are very low, while the complexity values are high. Conversely, the sensitivity values are distributed more toward high values.

Other models, such as RF and LGBM, demonstrated varying levels of faithfulness and consistency, depending on the data set and the explainer used.

In conclusion, our study demonstrates the effectiveness of the XGB classifier for malware detection in IoT devices across different data sets. LIME and SHAP explainers provide valuable insights for security analysts, allowing them to prioritize alerts and focus their efforts on the most critical devices. Although LIME generally offers lower complexity and higher consistency across different models, still, SHAP explainer is

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

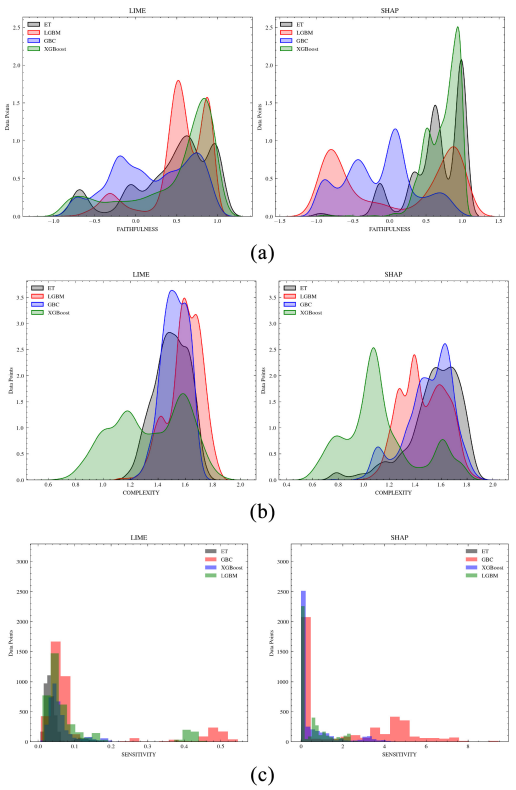


Fig. 15. Distribution of LIME and SHAP XAI metrics over MedBIoT data set for models. (a) High faithfulness. (b) Low complexity. (c) Low sensitivity.

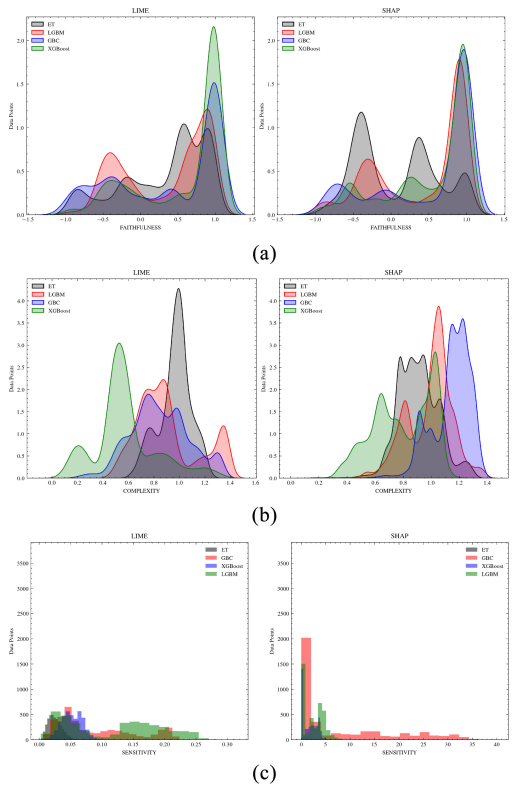


Fig. 16. Distribution of LIME and SHAP XAI metrics over BoT-IoT data set for models. (a) High faithfulness. (b) Low complexity. (c) Low sensitivity.

better than LIME. Overall, this research provides a strong foundation for developing interpretable and effective malware detection systems in the IoT network domain.

VI. Conclusion and Future Work

This research paper highlights the importance of using XAI methods to enhance the interpretability and transparency of ML models for detecting IoT botnets. The study uses three data sets for binary and multiclass classification of IoT botnets, with SBS employed as the feature selection technique. Local explanations using LIME and SHAP provide insights into the models' behavior. The quality of XAI methods is evaluated based on high faithfulness, monotonicity, complexity, and sensitivity metrics.

The results demonstrate that XAI techniques effectively improve the interpretability and transparency of ML-based IoT botnet detection models. Specifically, XGB with both explainers yields the most optimal explanations for all metrics. In general, model–explainer pairs provided varying results, indicating no consistent superiority of one explainer. However, the benchmarking results suggest that it is possible to find a set of features and a model that provides higher quality explanations and superior detection results for IoT botnet detection.

The findings of this research paper have important implications for security practitioners. By improving the transparency and interpretability of ML models for detecting IoT botnets, XAI techniques can enhance the ability of these practitioners to understand the models' behavior and the data's characteristics, leading to improved decision making and more effective strategies for detecting and preventing IoT botnets. A proper ML workflow containing feature selection, model development, and post hoc explanation stages can meet the expectations of security practitioners.

This study focused on LIME and SHAP as explanations for model predictions. However, other explainability methods could be explored, such as Integrated Gradients, Anchors, and DeepLIFT. Evaluating the performance and utility of these additional methods in the context of IoT malware detection may lead to the identification of more effective and insightful explanations. Future research could involve conducting qualitative user studies to assess the effectiveness of various XAI techniques in supporting security analysts in identifying and responding to IoT malware threats. Such research would contribute to developing more user-friendly explainability

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

tools that can be used in practical settings. Furthermore, the increasing sophistication of adversarial attacks in the IoT botnet network environment underscores the importance of evaluating the robustness of XAI methods against such attacks. By investigating the resilience of explainers to adversarial manipulation, it could lead to the development of more robust and secure explainability methods, which are essential for ensuring the trustworthiness and transparency of AI systems in the context of IoT botnet networks.

REFERENCES

[1] A. Marzano et al., "The evolution of bashlite and mirai IoT botnets," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, 2018, pp. 813–818.

[2] K. Angrishi, "Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT botnets," 2017, *arXiv:1702.03681*.

[3] Y. Meidan et al., "N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul.–Sep. 2018.

[4] M. Shafiq, Z. Tian, A. K. Bashir, A. Jolfaei, and X. Yu, "Data mining and machine learning methods for sustainable smart cities traffic classification: A survey," *Sustain. Cities Soc.*, vol. 60, Sep. 2020, Art. no. 102177.

[5] M. Shafiq, Z. Tian, Y. Sun, X. Du, and M. Guizani, "Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for Internet of Things in smart city," *Future Gener. Comput. Syst.*, vol. 107, pp. 433–442, Jun. 2020.

[6] D. Gunning, "Explainable artificial intelligence (xai)," *Def. Adv. Res. Proj. Agency (DARPA), nd Web*, vol. 2, no. 2, p. 1, 2017.

[7] D. L. Marino, C. S. Wickramasinghe, and M. Manic, "An adversarial approach for explainable ai in intrusion detection systems," in *Proc. 44th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, 2018, pp. 3237–3243.

[8] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73127–73141, 2020.

[9] H. Suryotrisongko, Y. Musashi, A. Tsuneda, and K. Sugitani, "Robust Botnet DGA detection: Blending XAI and OSINT for cyber threat intelligence sharing," *IEEE Access*, vol. 10, pp. 34613–34624, 2022.

[10] Z. A. El Houda, B. Brik, and S.-M. Senouci, "A novel IoT-based explainable deep learning framework for intrusion detection systems," *IEEE Internet Things Mag.*, vol. 5, no. 2, pp. 20–23, Jun. 2022.

[11] N. Moustafa, N. Koroniotis, M. Keshk, A. Y. Zomaya, and Z. Tari, "Explainable intrusion detection for cyber defences in the Internet of Things: Opportunities and solutions," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 3, pp. 1775–1807, 3rd Quart., 2023.

[12] S. Neupane et al., "Explainable intrusion detection systems (X-IDS): A survey of current methods, challenges, and opportunities," *IEEE Access*, vol. 10, pp. 112392–112415, 2022.

[13] İ. Kök, F. Y. Okay, Ö. Muyanlı, and S. Özdemir, "Explainable artificial intelligence (XAI) for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14764–14779, Aug. 2023.

[14] N. Capuano, G. Fenza, V. Loia, and C. Stanzione, "Explainable artificial intelligence in cybersecurity: A survey," *IEEE Access*, vol. 10, pp. 93575–93600, 2022.

[15] J. R. Goodall et al., "Situ: Identifying and explaining suspicious behavior in networks," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 204–214, Jan. 2019.

[16] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek, "Explainable AI methods-a brief overview," in *Proc. Int. Workshop Ext. Explain. AI Beyond Deep Models Classif. xxAI-Beyond Explain. AI Revis. Ext. Papers*, 2022, pp. 13–38.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?" explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2016, pp. 1135–1144.

[18] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Adv. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.

[19] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, 2021.

[20] L. Coroama and A. Groza, "Evaluation metrics in explainable artificial intelligence (XAI)," in *Proc. 2nd Int. Conf., Adv. Res. Technol., Inf., Innov. Sustain. Revis. Select. Papers, Part I*, 2022, pp. 401–413.

[21] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[22] Z. C. Lipton, "The mythos of model interpretability," 2019, *arXiv:1606.03490*.

[23] M. R. Islam, M. U. Ahmed, S. Barua, and S. Begum, "A systematic review of explainable artificial intelligence in terms of different application domains and tasks," *Appl. Sci.*, vol. 12, no. 3, p. 1353, 2022.

[24] H. W. Loh, C. P. Ooi, S. Seoni, P. D. Barua, F. Molinari, and U. R. Acharya, "Application of explainable artificial intelligence for Healthcare: A systematic review of the last decade (2011–2022)," *Comput. Methods Programs Biomed.*, vol. 226, Nov. 2022, Art. no. 107161.

[25] U. Pawar, D. O'Shea, S. Rea, and R. O'Reilly, "Explainable ai in healthcare," in *Proc. Int. Conf. Cyber Situat. Aware., Data Analy. Assess. (CyberSA)*, 2020, pp. 1–2.

[26] U. Pawar, D. O'Shea, S. Rea, and R. O'Reilly, "Incorporating explainable artificial intelligence (XAI) to aid the understanding of machine learning in the healthcare domain," in *Proc. AICS*, 2020, pp. 169–180.

[27] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," 2018, *arXiv:1706.07269*.

[28] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf, "Principles of explanatory debugging to personalize interactive machine learning," in *Proc. 20th Int. Conf. Intell. User Interf.*, 2015, pp. 126–137.

[29] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W.-K. Wong, "Too much, too little, or just right? ways explanations impact end users' mental models," in *Proc. IEEE Symp. Vis. Lang. Human Centric Comput.*, 2013, pp. 3–10.

[30] M. Nazar, M. M. Alam, E. Yafi, and M. M. Su'ud, "A systematic review of human–computer interaction and explainable artificial intelligence in healthcare with artificial intelligence techniques," *IEEE Access*, vol. 9, pp. 153316–153348, 2021.

[31] B. Y. Lim and A. K. Dey, "Assessing demand for intelligibility in context-aware applications," in *Proc. 11th Int. Conf. Ubiquitous Comput.*, 2009, pp. 195–204.

[32] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *Proc. IEEE 5th Int. Conf. Data Sci. Adv. Anal. (DSAA)*, 2018, pp. 80–89.

[33] D. Castelvecchi, "Can we open the black box of AI?" *Nat. News*, vol. 538, no. 7623, p. 20, 2016.

[34] A. Preece, D. Harborne, D. Braines, R. Tomsett, and S. Chakraborty, "Stakeholders in explainable AI," 2018, *arXiv:1810.00184*.

[35] D. Gunning and D. Aha, "DARPA's explainable artificial intelligence (XAI) program," *AI Mag.*, vol. 40, no. 2, pp. 44–58, 2019.

[36] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (XAI): Toward medical XAI," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 11, pp. 4793–4813, Nov. 2021.

[37] P. Svenmarck, L. Luotsinen, M. Nilsson, and J. Schubert, "Possibilities and challenges for artificial intelligence in military applications," in *Proc. Big Data Artif. Intell. Mil. Decis. Making Spec. Meet. (NATO)*, 2018, pp. 1–16.

[38] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[39] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "CorrAUC: A malicious Bot-IoT traffic detection method in IoT network using machine-learning techniques," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3242–3254, Mar. 2021.

[40] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "IoT malicious traffic identification using wrapper-based feature selection mechanisms," *Comput. Secur.*, vol. 94, Jul. 2020, Art. no. 101863.

[41] R. Kalakoti, S. Nõmm, and H. Bahsi, "In-depth feature selection for the statistical machine learning-based botnet detection in IoT networks," *IEEE Access*, vol. 10, pp. 94518–94535, 2022.

[42] M. Szczepański, M. Choraś, M. Pawlicki, and R. Kozik, "Achieving explainability of intrusion detection system by hybrid oracle-explainer approach," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2020, pp. 1–8.

[43] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSp*, 2018, pp. 108–116.

[44] T. Dias, N. Oliveira, N. Sousa, I. Praça, and O. Sousa, "A hybrid approach for an interpretable and explainable intrusion detection system," in *Proc. Int. Conf. Intell. Syst. Design Appl.*, 2022, pp. 1035–1045.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

[45] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Def. Appl.*, 2009, pp. 1–6.
[46] T.-T.-H. Le, H. Kim, H. Kang, and H. Kim, "Classification and explanation for intrusion detection system based on ensemble trees and SHAP method," *Sensors*, vol. 22, no. 3, p. 1154, 2022.
[47] N. Moustafa, "New generations of Internet of Things datasets for cybersecurity applications based machine learning: TON_IoT datasets," in *Proc. eRes. Aust. Conf., Brisb., Aust.*, 2019, pp. 21–25.
[48] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-iot dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.
[49] C. Patsakis, F. Casino, and V. Katos, "Encrypted and covert DNS queries for botnets: Challenges and countermeasures," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101614.
[50] S. Araki, K. Takahashi, B. Hu, K. Kamiya, and M. Tanikawa, "Subspace clustering for interpretable botnet traffic analysis," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–6.
[51] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discov. Data Min. (KDD)*, 1996, pp. 226–231.
[52] "MAWI working group traffic archive." MAWI working group, 2012. [Online]. Available: https://mawi.wide.ad.jp/mawi/
[53] M. Mazza, S. Cresci, M. Avvenuti, W. Quattrociocchi, and M. Tesconi, "RTbust: Exploiting temporal patterns for botnet detection on twitter," in *Proc. 10th ACM Conf. Web Sci.*, 2019, pp. 183–192.
[54] M. Zolanvari, Z. Yang, K. Khan, R. Jain, and N. Meskin, "Trust XAI: Model-agnostic explanations for ai with a case study on IIoT security," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 2967–2978, Feb. 2023.
[55] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1–6.
[56] "NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB," UNB. Accessed: Dec. 4, 2022. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html
[57] A. Guerra-Manzanares, S. Nõmm, and H. Bahsi, "Towards the integration of a post-hoc interpretation step into the machine learning workflow for IoT botnet detection," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, 2019, pp. 1162–1169.
[58] P. P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, and S. G. Teo, "Detection and classification of botnet traffic using deep learning with model explanation," *IEEE Trans. Dependable Secure Comput.*, early access, Jun. 15, 2022, doi: 10.1109/TDSC.2022.3183361.
[59] "Ixia." Accessed: Dec. 21, 2022. [Online]. Available: https://support.ixiacom.com/sites/default/
[60] "Datasets overview—Stratosphere IPS," Stratosphereips, Accessed: Dec. 21, 2022. [Online]. Available: https://www.stratosphereips.org/datasets-overview
[61] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*.
[62] J. A. Recio-Garcia, M. G. Orozco-del Castillo, and J. A. Soladrero, "Case-based explanation of classification models for the detection of SQL injection attacks," in *Proc. XCBR'23 Workshop Case-Based Reasoning Explan. Intell. Syst. (ICCBR)*, 2023, pp. 1–12.
[63] M. M. Alani and E. Damiani, "XRecon: An explainbale IoT reconnaissance attack detection system based on ensemble learning," *Sensors*, vol. 23, no. 11, p. 5298, 2023.
[64] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya, "An explainable deep learning-enabled intrusion detection framework in IoT networks," *Inf. Sci.*, vol. 639, Aug. 2023, Art. no. 119000.
[65] S. R. Islam, W. Eberle, S. K. Ghafoor, A. Siraj, and M. Rogers, "Domain knowledge aided explainable artificial intelligence for intrusion detection and response," 2020, *arXiv:1911.09853*.
[66] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network," in *Proc. 6th ICISSP*, 2020, pp. 207–218.
[67] "Node-RED." Accessed: Nov. 21, 2023. [Online]. Available: https://nodered.org/
[68] Y. Sasaki, "The truth of the F-measure," *Teach Tutor Mater*, vol. 1, no. 5, pp. 1–5, 2007.
[69] L. S. Shapley, "A value for n-person games," in *Contributions to Theory of Games*, vol. 2. Princeton, NJ, USA: Princeton Univ. Press, 1953, pp. 307–317.
[70] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowl. Inf. Syst.*, vol. 41, pp. 647–665, Dec. 2014.
[71] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," 2019, *arXiv:1802.03888*.
[72] H. S. Eriksson and G. Grov, "Towards XAI in the SOC–a user centric study of explainable alerts with SHAP and LIME," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2022, pp. 2595–2600.



Rajesh Kalakoti (Graduate Student Member, IEEE) received the master's degree in information technology from Jawaharlal Nehru Technological University Kakinada, Kakinada, India, in 2015. He is currently pursuing the Ph.D. degree in XAI within the cybersecurity domain with Tallinn University of Technology, Tallinn, Estonia.

He is also a Early-Stage Researcher in XAI within the cybersecurity domain with Tallinn University of Technology. He has worked as a lecturer of information technology and a software engineer in India. His research interests focus on IoT security, network traffic, malware, HTTP-based botnet, machine learning, federated learning, deep learning, and XAI.

Mr. Kalakoti is a member of the IEEE Computer Society.



Hayretdin Bahsi received the M.Sc. degree in computer engineering from Bilkent University, Ankara, Türkiye, in 2002, and the Ph.D. degree in computer engineering from Sabanci University, Istanbul, Türkiye, in 2010.

He is a Research Professor with the Center for Digital Forensics and Cyber Security, Tallinn University of Technology, Tallinn, Estonia. His research interests include cyber–physical system security and the application of machine learning methods to cyber-security problems.



Sven Nõmm received the Ph.D. degree jointly from Tallinn University of Technology, Tallinn, Estonia, and the École Centrale de Nantes et L'Université de Nantes, Nantes, France, in 2004.

He is a Tenured Associate Professor with Tallinn University of Technology. His research interests include human–machine interaction, analysis of human motions, applications of artificial intelligence to cybersecurity problems, and geoscience.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:57 UTC from IEEE Xplore. Restrictions apply.

Appendix 4

IV

R. Kalakoti, S. Nõmm, and H. Bahsi. Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai). In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 595–601. IEEE, 2023

Improving Transparency and Explainability of Deep Learning based IoT Botnet Detection using Explainable Artificial Intelligence (XAI)

Rajesh Kalakoti
Department of Software Science
School of Information Technology,
Taltech, Tallinn, Estonia.
rajesh.kalakoti@taltech.ee

Sven Nõmm
Department of Software Science
School of Information Technology
Taltech, Tallinn, Estonia
sven.nomm@taltech.ee

Hayretdin Bahsi
Department of Software Science
School of Information Technology,
Taltech, Tallinn, Estonia
hayretdin.bahsi@taltech.ee

2023 International Conference on Machine Learning and Applications (ICMLA) | 979-8-3503-4534-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICMLA58977.2023.00088

***Abstract*—Ensuring the utmost security of IoT systems is imperative, and robust botnet detection plays a pivotal role in achieving this goal. Deep learning-based approaches have been widely employed for botnet detection. However, the lack of interpretability and transparency in these models can limit these models' effectiveness. In this research, we present a Deep Neural Network (DNN) model specifically designed for the detection of IoT botnet attack types. Our model performs exceptionally, demonstrating outstanding performance of classification metrics with 99% accuracy, F1 score, recall, and precision. To gain deeper insights into our DNN model's behaviour, we employ seven different post hoc explanation techniques to provide local explanations. We evaluate the quality of Explainable AI (XAI) methods using metrics such as high faithfulness, monotonicity, complexity, and sensitivity. Our findings highlight the effectiveness of XAI techniques in enhancing the interpretability and transparency of the DNN model for IoT botnet detection. Specifically, our results indicate that DeepLIFT yields high faithfulness, high consistency, low complexity, and low sensitivity among all the explainers.**

***Index Terms*—XAI, Deep learning, IoT Botnet, Post-hoc explanation, explainable artificial intelligence**

I. Introduction

Cybersecurity incorporates extensive practices and measures to protect networks, devices, and data against unauthorized access or illicit usage. It entails the critical task of striking a delicate balance to uphold the security properties of confidentiality, integrity, and availability (CIA) for information protection [1]. Meanwhile, cyber defensive mechanisms emerge at multiple levels, including application, network, host, and data, to fortify the overall security infrastructure [2]. Artificial Intelligence techniques, particularly machine learning (ML) and deep learning (DL) algorithms have demonstrated remarkable capabilities in various cybersecurity domains. These techniques have exhibited impressive performance levels when applied to benchmark datasets for tasks such as intrusion detection, spam email filtering, botnet detection, fraud detection, and identification of malicious applications [3]. It is essential to acknowledge that these techniques are not error-free. They can occasionally produce errors that may be more costly than traditional cyber defensive approaches. On the contrary, cyber security developers have sometimes prioritized achieving higher accuracy even at the expense of interpretability, creating complex and challenging-to-understand models [4]. The European Union's General Data Protection Regulation (GDPR) has addressed the issue of explainability, emphasizing the importance of understanding the underlying logic behind Artificial Intelligence algorithmic decisions that may adversely affect individuals [5]. Consequently, to instil trust in cyber security systems, Artificial Intelligence must be transparent and interpretable. Various strategies have been proposed to enhance the intelligibility of Artificial Intelligence decisions for human understanding to meet these requirements. These techniques, commonly called "XAI" (Explainable Artificial Intelligence), have already been implemented in numerous domains, including healthcare, Natural Language Processing, and financial services [6].

As the complexity and volume of cyber attacks, such as malware, intrusion, and spam, continue to rise, effectively managing them is becoming increasingly challenging [7]. In the cybersecurity domain, conventional approaches such as rule-based, statistics-based, and signature-based have traditionally been employed for intrusion detection [8]. However, the surge in data transmission over the Internet and the emergence of new networking paradigms like the Internet of Things (IoT), cloud computing, and fog/edge computing [9, 10] have rendered these traditional approaches inadequate in processing large volumes of data and have resulted in high computing costs [9]. Furthermore, Artificial Intelligence (AI) has emerged as a fundamental technology in Industry 4.0 [11]. Consequently, AI techniques, such as machine learning (ML) algorithms and deep learning (DL) algorithms, have become increasingly essential in providing intelligent cybersecurity services and management. For example, Ucci et al. [7] highlighted the successful utilization of ML methods for various aspects of malware analysis, including malware detection, similarity analysis, and category analysis. Additionally, Kwon et al. [12] employed DL-based and feature selection [13] approaches for network anomaly detection and network traffic analysis, demonstrating these techniques' efficacy in cyber security.

1946-0759/23/$31.00 ©2023 IEEE
DOI 10.1109/ICMLA58977.2023.00088

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:56 UTC from IEEE Xplore. Restrictions apply.

However, despite the potential benefits, AI-based approaches in cyber security also face significant challenges. Limitations such as the availability of cybersecurity-related data [14], the vulnerability of AI models to adversarial attacks [15], and concerns regarding ethics and privacy [16] are inherent issues encountered by AI-based cybersecurity systems. Among these challenges, the black-box nature of AI models is a particularly noteworthy limitation that requires careful consideration when integrating AI models into the cybersecurity domain [17]. The opaque nature of AI models leads to cybersecurity decisions being generated without clear rationale or justifiability, making it difficult for humans to comprehend how these decisions are reached. Consequently, this lack of explainability renders the cyber defensive mechanisms as black-box systems, rendering them highly susceptible to information breaches and AI-based cyber threats [18]. Explainable AI (XAI) has emerged as a response to the growing black box problem addressing the limitations of using AI. XAI enables users and experts to better understand the underlying logic and key data evidence behind the results generated by AI-based statistical models [19]. In conclusion, there are several motivations for applying Explainable AI (XAI) in cyber security, and they are like Building trust: Integrating XAI in cyber security aims to enhance trust by promoting transparency and understanding of decision models related to cybersecurity. Justice, social responsibility, and risk mitigation: Applying XAI in cyber security is driven by addressing severe social problems, potential risks to human lives, and the ethical considerations involved. It goes beyond simple cost-benefit calculations and ensures fairness and social responsibility.

This research comprehensively evaluates seven post hoc local explainability methods, including Integrated Gradients, Gradient * Input, DeepLIFT, Saliency, SHAP, feature ablation and LIME. The evaluation is based on quantitative metrics such as faithfulness, monotonicity, complexity, and sensitivity. Specifically, we assess and compare the effectiveness of these local explanation methods when applied to multiclass attack types for IoT botnet detection in a detailed benchmarking setting. The experiments are performed on a deep learning-based model over the N-BaIoT dataset, carefully selected from the problem domain to ensure the robustness and relevance of our findings. To the best of our knowledge, very few empirical studies have evaluated the quality of post hoc local explainability methods for network intrusion detection tasks, particularly in IoT botnet detection. Our research aims to bridge this gap by systematically assessing and comparing multiple explainability methods, providing valuable insights into the effectiveness of these methods for enhancing the interpretability and trustworthiness of deep learning-based models in IoT security applications.

The content of our paper is presented as follows: Section II provides related information about the addressed topic and summarizes the literature. Section III describes data utilized in this study. Section IV explains the methodology of our research. Section V presents the results and discussions obtained. Our work is concluded in Section VI.

II. Related Work

Several studies in the literature have applied explainability methods to network intrusion detection problems. For instance, Szczepanski et al. introduced a hybrid Oracle Explainer Intrusion Detection System that combines Artificial Neural Networks (ANNs) and Decision Trees, utilizing micro aggregation techniques to enhance performance [20]. In this paper [21], the authors employed the Explainable Boosting Machine (EBM) as a glass-box classifier for detecting network intrusions. The performance of EBM was compared with other AI classifiers, including decision tree, logistic regression, Support Vector Machine (SVM), and Deep Neural Network (DNN). The ASNM-NPBO dataset [22] (Advanced Security Network Metrics & Non-Payload-Based Obfuscations) consisted of ASNM features extracted from tcpdump capture of obscured malicious and legitimate TCP transmissions on specified vulnerable network services.

Zolanvari et al. proposed a model-agnostic XAI framework named TRUST (Transparency Based on Statistical Theory) designed for numerical applications. This framework utilizes factor analysis to transform input features into latent features, ranks features using mutual information and employs a multimodal Gaussian distribution to generate new samples for each class label. The TRUST XAI model underwent evaluation through a case study using three datasets: NSL-KDD [24], UNSW [25], and a custom dataset named "WUSTL-IIoT," derived from their testbed experiment involving intrusion network traffic data. The evaluation showed that the TRUST XAI model achieved a 98% success rate in explaining random test samples, outperforming LIME. Hariharan et al. employ a comprehensive set of interpretability algorithms, including Permutation Importance, SHapley Additive exPlanation, Local Interpretable Model-Agnostic Explanation, Contextual Importance, and Utility algorithms [26] over NSL-KDD [24] dataset. These algorithms are applied to Intrusion Detection Systems (IDSs) implemented on three machine learning models: Random Forest, eXtreme Gradient Boosting, and Light Gradient Boosting. The study thoroughly compares explanations based on accuracy, consistency, and stability to enhance the understanding of cyber-attack predictions in network traffic for cyber security personnel. Additionally, the paper presents a case study focusing on DoS attack variants, which offers valuable insights into the impact of features on prediction performance. A Bidirectional Long Short-Term Memory based Explainable Artificial Intelligence framework (BiLSTM-XAI) was developed, utilizing the Krill herd optimization (KHO) algorithm to select significant database features [27]. SHAP and LIME explainable AI algorithms were employed for analysis, and the proposed BiLSTM-XAI model achieved an accuracy of 98.2% when evaluated using the NSL-KDD dataset [24]. By implementing XAI models, the complexities of the BiLSTM framework are reduced, leading to improved detection accuracy with explanations provided for each learning prediction.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:56 UTC from IEEE Xplore. Restrictions apply.

III. Data set Description

In this work, N-BaIoT [28] is used. In this dataset, 115 network traffic features were extracted based mainly on descriptive statistics measures from the network traffic generated by bots deployed in a collected testing environment settings of 9 different IoT devices infected by Mirai and Gafgyt malware types. More significantly, the features that are specified for each instance reflect the aggregated descriptive statistics of the raw traffic of the network in five-time windows (100 ms, 500 ms, 306 1.5 s, 10 s, and 1 min), which are coded L5, L3, L1, L0.1 and L0.01, respectively as shown in Table I. There are five main feature categories: host-IP (traffic emanating from a specific IP address, coded as H), host-MAC and IP (traffic emanating from the same MAC and IP, coded MI), channel (traffic between specific hosts, coded HH), socket (traffic between specific hosts, including ports, coded HpHp), and network jitter (time interval between packets in channel communication, coded as HH_jit). The packet count, mean and variance packet sizes are calculated for each significant category. There have been extra statistical values like the correlation coefficient (PCC) of packet size, radius, covariance, and magnitude derived for network Channel and Socket along with packet count, mean, and variance. In this study, From the N-BaIoT dataset, attack type classification is developed. For this classification, the data points have been classified into eight attack types and legitimate network traffic: ACK, Benign, COMBO, JUNK, SCAN, SYN, TCP, UDP, and PLAIN. A description of these attacks can be found in Table II.

TABLE I: Details of Features of N-BaIoT dataset.

Feature Category	Category Code	No.Of features	Statistical Value Feature	Time Frame Windows
Hos Mac & IP	MI	15	Packet Count, Mean Variance	100 Micrso Sconds
Host IP	H	15		500 micro seconds
Network Jitter	HH_jit	15		1.5 Seconds
Channel	HH	35	Packet Count, Mean	10 Seconds
Socket	HpHp	35	Variance, Magnitude,Radius, CoVariance, Correlation	1 Minute

TABLE II: Botnet Attack types used in this Study for DNN model botnet attack detection

Class Name	Description
ACK	Gafgyt malware Sending Spam data
Beign	Legitimate Network Traffic
COMBO	Gafgyt malware Sending spam data and opening a connection
JUNK	Mirai Malware ACK-Flooding
SCAN	Scans The network devices for vulnerabilities,(Mirai & Gafgyt
SYN	Mirai Malware SYN-Flooding
TCP	Gafgyt malware TCP Flooding
UDP	UDP flooding (Mirai & Gafgyt)
UDPPLAIN	Mirai malware UDP flooding with Less of an option for higher packet per second

IV. Methodology

A. DNN for Botnet attack prediction

This research paper proposes a fully connected Deep Neural Network (DNN) model for detecting IoT botnet attacks in the N-BaIoT dataset through network traffic analysis. The dataset was split into training and testing sets using an 80:20 ratio. As a preprocessing step, we applied min-max normalization, scaling the feature values between 0 and 1. Model consists of 3 hidden layers, each comprising 9 hidden units, as specified in hyperparameters (see Tab. III). The Rectified Adam (RAdam) optimizer with a learning rate of 0.0129692 was employed to train the model. The SELU (Scaled Exponential Linear Unit) activation function is applied after each hidden layer to introduce non-linearity, allowing the DNN to learn complex patterns and relationships within the data effectively.

TABLE III: DNN Model Hyperparameters (Random Search)

Hyperparameter	Value
Hidden Layers	3
Hidden Units	9
Learning Rate	0.01296
Optimizer	RAdam
Activation	SELU
Batch Size	256
Epochs	21

The input layer receives a 115-dimensional feature vector, representing aggregated statistics from five different time windows. This input is passed through the first fully connected hidden layer with 9 units, applying the SELU activation function. The output from the first hidden layer is then fed to the second hidden layer, which consists of 9 units and employs the SELU activation function. Finally, the output layer is implemented with another fully connected layer, mapping the activations from the last hidden layer to the output size. This output size corresponds to the number of classes (ack, benign, combo, junk, scan, syn, tcp, udp, udpplain) used for botnet attack type classification. During the training phase, the model undergoes 21 epochs with a batch size 256. To compute the loss during training, we utilize the CrossEntropyLoss function, which is well-suited for multi-class classification tasks

To generate prediction labels, we rely on the softmax activation function. This function is crucial in providing prediction probabilities for each class, enabling a deeper understanding of the model's confidence in its predictions and the probability distribution across different classes(attacks and begin). Utilizing the softmax activation function enhances the interpretability and significance of our classifier's outputs, which is essential for evaluating explainers regarding Faithfulness, Monotonicity, and Sensitivity metrics mentioned in the section IV-C

B. Explaination Methods

Our research focused on enhancing the interpretability of deep learning-based intrusion detection systems in IoT networks by applying various Explainable AI (XAI) feature attribution methods. In this study, we employed seven distinct explanation methodologies, which were carefully selected based on their widespread usage in the literature. Below, we present a concise summary of each method. However, please note that this is not an exhaustive description of each method.

For data, the input $x \in \mathbb{R}^d$, where d is the dimensionality of the feature set, and the black box model $\mathcal{M}$ maps the input to an output $\mathcal{M}(x) \in \mathcal{Y}$. Denote $\mathcal{D} = (x^i, y^i)$ as the dataset consisting of all input-output pairs. $\mathbf{g}$ as an explanation mapping that for predictor $\mathcal{M}$ and point of interest x returns an

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:56 UTC from IEEE Xplore. Restrictions apply.

importance score $\mathbf{g}(\mathcal{M}, x) = \phi_x \in \mathbb{R}^d$ for all features. Denote $D : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$ as a metric in the space of explanations and $S\mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$ a metric in the space of inputs. The evaluation criterion μ is the mapping that takes predictor $\mathcal{M}$, explainer $\mathbf{g}$ and the point of interest x as arguments and computes a scalar.

1) Integrated Gradients: Integrated attributes [29] feature importance to inputs by integrating the model's gradients along a path from a baseline to the actual input, The authors define $\bar{x}$ as a baseline input representing the absence of a feature in the input x. Integrated Gradients is defined as $\mathbf{g}(\mathcal{M}, x) = \text{IG}(x) = (x - \bar{x}) \times \int_{\alpha=0}^{1} \frac{\partial \mathcal{M}(\bar{x}+\alpha\cdot(x-\bar{x}))}{\partial x}\, d\alpha$ where $\bar{x}$ is the baseline input.

*2) Gradient * Input:* Gradient * Input computes [30] the importance of each feature by multiplying the model's partial derivative with respect to that feature by the input feature value. The formula for Gradient * Input, denoted as $\mathbf{g}(\mathcal{M}, x)$, is given by: $\mathbf{g}(\mathcal{M}, x) = \frac{\partial \mathcal{M}(x)}{\partial x} \times x$

3) DeepLIFT: DeepLIFT (Deep Learning Important FeaTures) [31] attributes each input x is assigned value $C_{\Delta x_i \Delta y}$ that signifies a deviation from its original value towards a reference value. DeepLIFT utilizes a summation-to-delta property: $\sum_{i=1}^{n} C_{\Delta x_i \Delta o} = \Delta o$ where $o = \mathcal{M}(x)$ and Δo is the difference between the model output of the input and the reference value.

4) Saliency: The saliency-based [32] approach focuses on the gradient of the output of a neural network with respect to the input x. In our case, the output corresponds to the attack score $a(x)$. The basic idea behind the saliency is that small perturbations to important features will cause large changes in the model output; the saliency score measures the sensitivity of the model output against the input perturbation. Concretely, the saliency score s_ℓ of the ℓ-th dimension of input x is defined as $s_\ell(x) = \left|\frac{\partial a(x)}{\partial x_\ell}\right|$where x_ℓ is the ℓ-th dimension of x. For each malicious packet x, we calculate the saliency scores $s_\ell(x)$ for $\ell = 1, 2, \ldots, L$, and report the dimensions with high saliency scores as suspected parts.

5) Shap: SHAP [33] is a game-theoretic explanation method based on Shapley values. SHAP is defined as $\mathbf{g}(\mathcal{M}, x) = \phi_0 + \sum_{j=1}^{M} \phi_j$, where ϕ_j is the feature attribution of feature j.

6) Feature Ablation: Feature ablation assesses the importance of individual features in a model's predictions by systematically removing or ablating each feature and observing the change in the model's output. The process involves evaluating the model's performance on the modified input with the ablated feature(s) and comparing it to the original performance. The formula for feature ablation can be represented as follows:

$$\text{Ablation}(x, i) = \mathcal{M}(x) - \mathcal{M}(x_{\text{ablated}_i})$$

By comparing the model's output before and after ablation, feature ablation allows for understanding the influence of each individual feature on the model's decision-making process.

7) LIME: LIME [34] is an explanation method that approximates the model's behavior locally around a specific data instance x within the neighborhood $N(x)$ using a simpler interpretable model the prediction of the interpretable model around x is denoted as The approximation is obtained by minimizing the object function defined as $\underset{g \in G}{\text{argmin}} L(\mathcal{M}, g, \pi_x) + \Omega(g)$

C. XAI evaluation

Various metrics assess XAI explanation quality [35], grouped into three categories: user-focused, application-focused, and functionality-focused evaluations. The primary metrics employed to assess local explanations in this work include *high faithfulness*, *monotonicity*, *low complexity*, and *max sensitivity* for the explainers utilized.

1) High Faithfulness: Faithfulness measures how well the explanation function $\mathbf{g}$ aligns feature importance scores with the black-box model $\mathcal{M}$

$$\mu_F(\mathcal{M}, g; x) = \underset{\mathcal{B} \in \binom{|d|}{|\mathcal{B}|}}{corr} \left(\sum_{i \in \mathcal{B}} g(\mathcal{M}, x)_i, \mathcal{M}(x) - \mathcal{M}(x_{\mathcal{B}}) \right) \tag{1}$$

where $x_{\mathcal{B}} = x_i | i \in \mathcal{B}\}$

The faithfulness metric iteratively replaces a random subset of given attributions with a baseline value. Then it measures the correlation between the sum of this attribution and the difference in the model's output.

2) Monotonocity: monotonicity checks if the predictor's output changes consistently with changes in the sum of feature attributions when moving from one input point to another. Monotonicity ensures that the explanation provided by the explainer is consistent with the behavior of the predictor.

3) Low Complexity: The complexity metric computes the entropy of the fractional contribution of each feature to the total magnitude of the attribution individually.

$$\mu_C(\mathcal{M}, g; x) = -\sum_{i=1}^{d} P_g(i) \log P_g(i) \tag{2}$$

where

$$P_g(i) = \frac{|g(\mathcal{M}, x)i|}{\sum j \in |d| |g(\mathcal{M}, x)_j|}; P_g = P_g(1), ...P_g(d) \tag{3}$$

4) Maximum Sensitivity: The maximum sensitivity metric measures the maximum sensitivity of an explanation using a Monte Carlo sampling-based approximation. It assesses how sensitive the explanation function is to small changes in the input data in the neighbourhood of the point of interest. By sampling nearby points and comparing their explanations, the metric aims to ensure that similar inputs with similar model outputs receive consistent explanations

$$\mu_M(\mathcal{M}(x), g, r; x) = \max_{z \in N_r} Distance(g(\mathcal{M}(x), x), g(\mathcal{M}(x), z)) \tag{4}$$

where $N_r = z \in Distance_x | p(x, z) \leq r, \mathcal{M}(x) = \mathcal{M}(x)(z)$

In the above equation, N_r represents the set of points within a certain radius r around the point of interest x.

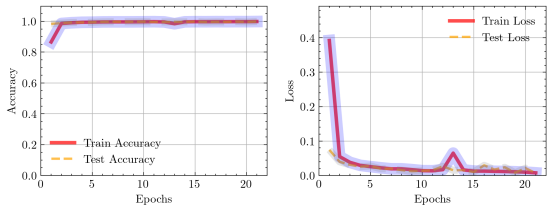
Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:56 UTC from IEEE Xplore. Restrictions apply.

The experiments were carried out on a computer running Pop!_OS 22.04 LTS x86_64 operating system with the following hardware configuration: 32 GB of DDR4-2666R ECC RAM, AMD Ryzen 5 5600G with Radeon Graphics (12) @ 3.900GHz processor. The scripts were developed using the Python 3 programming language and Pytorch library.

V. Results & Discussion

In this section, we will present the results of our research, which encompass an analysis of the model's performance, explanations provided by the explainers, and an evaluation of these explainers using four different metrics.

To evaluate the performance of our DNN model for botnet attack type detection, we utilized metrics such as classification accuracy, precision, recall, F1 score, and the confusion matrix (see results in Fig. 2). We present Fig. 1, which illustrates the graph of testing and training loss over epochs and the training and testing accuracies throughout the training process. In the classification report, the model achieved more than 99% accuracy for all attacks. However, the recall for TCP attacks is 97% , and the precision for UDP attacks is 98%, both of which are slightly lower than those for other attacks, respectively.



(a) Accuracy (b) Loss

Fig. 1: Training and Testing Accuracies and Loss Values of DNN Model for IoT Botnet Attack Detection

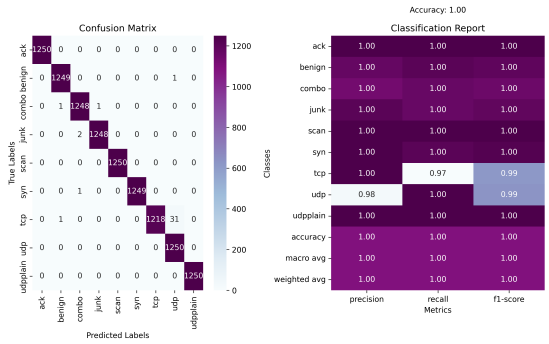
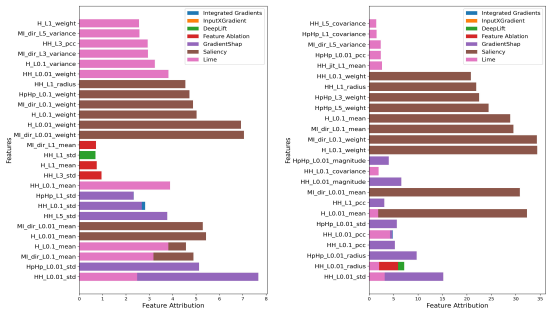


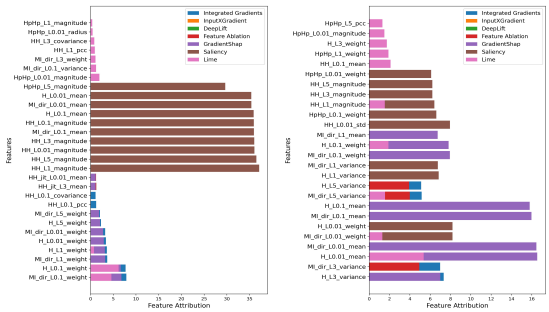
Fig. 2: Confusion matrix and classification report of DNN model for IoT botnet attack type detection

In our research, we employed seven explanation methodologies, namely Integrated Gradients, InputXGradient, DeepLIFT, Feature Ablation, SHAP, Saliency, and LIME, to explain the predictions of our DNN model on the test data. Each explanation method provided feature importance scores based on its specific approach (e.g., gradient-based or SHAP values). We aggregated the feature importance scores across all instances in the test data using absolute sum for each class of prediction label. Subsequently, we ranked the aggregated features based on their importance scores.



(a) TCP attack (b) UDP attack

Fig. 3: Example of Top 10 influential features from each explainer for ACK and UDP attack



(a) ACK attack (b) SYN Attack

Fig. 4: Example of Top 10 influential features from each explainer for ACK and SYN attacks

Fig. 3 shows the example of top 10 feature influences from every explainer in TCP and UDP attack predictions by the DNN model. Based on the explanation results provided by the different explainers, it is evident that the most influential features in the IoT botnet attack prediction by our DNN model are derived from the Host-based (MI & H) category. On the other hand, features belonging to the Channel (HH) and Socket-based (HpHp) network categories exhibit relatively less influence. Furthermore, the Network Jitter (HH_jit) features show minimal impact on the predictions of the model. Notably, we have observed similar trends in the influence of network

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:56 UTC from IEEE Xplore. Restrictions apply.

categories for other types of attack class predictions, for Example, for ACK& SYN attacks in Fig 4.

TABLE IV: comparison results of the evaluation of XAI metrics

Explainer/metric	High Faithfulness	Sensitivity	Complexity	Monotonocity
Integrated Gradients	0.44 ± 0.32	0.61 ± 1.44	3.32 ± 0.38	52.00%
Gradient * Input	0.42 ± 0.36	0.61 ± 1.44	3.75 ± 0.29	60.00%
DeepLIFT	**0.71 ± 0.12**	**0.48 ± 0.76**	**2.82 ± 0.12**	**69.19%**
SHAP	0.55 ± 0.18	8.17 ± 8.11	3.30 ± 0.40	44.00%
Feature Ablation	0.48 ± 0.31	0.38 ± 0.76	4.29 ± 0.41	39.00%
Saliency	0.27 ± 0.26	9.79 ± 21.36	4.33 ± 0.06	46.00%
LIME	0.39 ± 0.21	10.62 ± 5.68	4.30 ± 0.31	28.00%

In our research, we evaluated the faithfulness of explanation methods by analyzing the correlation between the importance assigned to attributes by these methods and their impact on the predictive model's probabilities. Additionally, we measure monotonicity to assess the effect of individual features on model probability by incrementally adding each attribute in order of increasing importance and evaluating its impact on the model's probability. The DNN model prediction probabilities are obtained using the Softmax activation function. A high faithfulness correlation value indicates that the explanations accurately capture the original model's behaviour and can be considered trustworthy. Similarly, a high monotonicity score indicates that the explanations are consistent with the model's predictions for the given input. Furthermore, we compute the low complexity metric by calculating the entropy of feature attribution derived from the explanations. Furthermore, the sensitivity metric assesses the robustness of the explainer output, ensuring that nearby inputs in the feature space have similar explanations when the sensitivity value is low. In this metric, for obtaining the nearest neighbour points related to the prediction label of the explanation, we utilized the Euclidean distance with a radius value of 0.1, which helps identify data points in the feature space closest to the instance and have similar explanations for the predicted label.

Table IV shows comparison results of the evaluation of XAI metrics. Among the explainers, DeepLIFT emerged as the most promising, achieving the highest mean faithfulness correlation of 0.71 $\pm$ 0.12. This indicates that the explanations provided by DeepLIFT are highly consistent with the DNN model's behaviour, making them more trustworthy and accurate. Additionally, DeepLIFT demonstrated a lower sensitivity value of 0.48 $\pm$ 0.76, implying robust and consistent explanations for nearby data points in the feature space. Moreover, DeepLIFT achieved an impressive Monotonicity score of 69.19%, indicating a strong consistency in how each feature influences the model's predictions. This consistent relationship between feature importance and model performance enhances the credibility of DeepLIFT's explanations. Furthermore, Gradient * Input exhibited the highest Monotonicity score of 60%, showing a strong trend of monotonically increasing model performance with adding features. SHAP achieved a respectable faithfulness correlation of 0.55 $\pm$ 0.18 but had a relatively higher sensitivity value, suggesting less stable explanations for neighbouring data points.

VI. Conclusion

This research paper highlights the importance of using XAI methods to enhance the interpretability and transparency of deep learning-based model for detecting IoT botnet attacks of multiclass classication. We used DNN model for classifying various IoT botnet attack types, achieving remarkable performance with 99% accuracy, F1 score, recall, and precision. However, the black-box nature of deep learning models posed challenges in understanding their decision-making process. We explored seven post hoc explanation techniques to address this issue to provide local explanations for our DNN model's behaviour. Through rigorous quantitative evaluation using high faithfulness, monotonicity, complexity, and sensitivity metrics, we thoroughly assessed the quality of these explanations. Notably, among all the explainers, DeepLIFT emerged as a standout performer, demonstrating high faithfulness, high consistency, low complexity, and low sensitivity. Its ability to furnish accurate and stable explanations signifies the reliability of our DNN model's decision-making process.

References

[1] "What is cybersecurity? | cisa," https://www.cisa.gov/news-events/news/what-cybersecurity, (Accessed on 07/17/2023).

[2] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, p. 122, 2019.

[3] S. M. Mathews, "Explainable artificial intelligence applications in nlp, biomedical, and malware classification: A literature review," in *Intelligent Computing: Proceedings of the 2019 Computing Conference, Volume 2*. Springer, 2019, pp. 1269–1292.

[4] M. Sahakyan, Z. Aung, and T. Rahwan, "Explainable artificial intelligence for tabular data: A survey," *IEEE access*, vol. 9, pp. 135 392–135 422, 2021.

[5] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"," *AI magazine*, vol. 38, no. 3, pp. 50–57, 2017.

[6] M. Nazar, M. M. Alam, E. Yafi, and M. M. Su'ud, "A systematic review of human–computer interaction and explainable artificial intelligence in healthcare with artificial intelligence techniques," *IEEE Access*, vol. 9, pp. 153 316–153 348, 2021.

[7] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, 2019.

[8] S. Han, M. Xie, H.-H. Chen, and Y. Ling, "Intrusion detection in cyber-physical systems: Techniques and challenges," *IEEE systems journal*, vol. 8, no. 4, pp. 1052–1062, 2014.

[9] D. Gümüşbaş, T. Yıldırım, A. Genovese, and F. Scotti, "A comprehensive survey of databases and deep learning

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:56 UTC from IEEE Xplore. Restrictions apply.

methods for cybersecurity and intrusion detection systems," *IEEE Systems Journal*, vol. 15, no. 2, pp. 1717–1731, 2020.

[10] R. Donida Labati, A. Genovese, V. Piuri, F. Scotti, and S. Vishwakarma, "Computational intelligence in cloud computing," *Recent Advances in Intelligent Engineering: Volume Dedicated to Imre J. Rudas' Seventieth Birthday*, pp. 111–127, 2020.

[11] I. H. Sarker, M. H. Furhad, and R. Nowrozy, "Ai-driven cybersecurity: an overview, security intelligence modeling and research directions," *SN Computer Science*, vol. 2, pp. 1–18, 2021.

[12] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, pp. 949–961, 2019.

[13] R. Kalakoti, S. Nõmm, and H. Bahsi, "In-depth feature selection for the statistical machine learning-based botnet detection in iot networks," *IEEE Access*, vol. 10, pp. 94 518–94 535, 2022.

[14] R. Al Nafea and M. A. Almaiah, "Cyber security threats in cloud: Literature review," in *2021 international conference on information technology (ICIT)*. IEEE, 2021, pp. 779–786.

[15] A. Kuppa and N.-A. Le-Khac, "Black box attacks on explainable artificial intelligence (xai) methods in cyber security," in *2020 International Joint Conference on neural networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[16] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," in *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*. IEEE, 2019, pp. 0452–0457.

[17] J. Gerlings, A. Shollo, and I. Constantiou, "Reviewing the need for explainable artificial intelligence (xai)," *arXiv preprint arXiv:2012.01007*, 2020.

[18] G. Jaswal, V. Kanhangad, and R. Ramachandra, *AI and deep learning in biometric security: trends, potential, and challenges*. CRC Press, 2021.

[19] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[20] M. Szczepański, M. Choraś, M. Pawlicki, and R. Kozik, "Achieving explainability of intrusion detection system by hybrid oracle-explainer approach," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[21] T. A. El-Mihoub, L. Nolle, and F. Stahl, "Explainable boosting machines for network intrusion detection with features reduction," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2022, pp. 280–294.

[22] "Asnm datasets," ASNM Datasets, (Accessed on 07/29/2023).

[23] M. Zolanvari, Z. Yang, K. Khan, R. Jain, and N. Meskin, "Trust xai: Model-agnostic explanations for ai with a case study on iiot security," *IEEE Internet of Things Journal*, 2021.

[24] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.

[25] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[26] S. Hariharan, R. Rejimol Robinson, R. R. Prasad, C. Thomas, and N. Balakrishnan, "Xai for intrusion detection system: comparing explanations based on global and local scope," *Journal of Computer Virology and Hacking Techniques*, vol. 19, no. 2, pp. 217–239, 2023.

[27] S. Sivamohan and S. Sridhar, "An optimized model for network intrusion detection systems in industry 4.0 using xai based bi-lstm framework," *Neural Computing and Applications*, vol. 35, no. 15, pp. 11 459–11 475, 2023.

[28] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot: Network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[29] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 3319–3328.

[30] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *arXiv preprint arXiv:1605.01713*, 2016.

[31] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International conference on machine learning*. PMLR, 2017, pp. 3145–3153.

[32] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[33] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[34] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[35] L. Coroama and A. Groza, "Evaluation metrics in explainable artificial intelligence (xai)," in *Advanced Research in Technologies, Information, Innovation and Sustainability: Second International Conference, ARTIIS 2022, Santiago de Compostela, Spain, September 12–15, 2022, Revised Selected Papers, Part I*. Springer, 2022, pp. 401–413.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:14:56 UTC from IEEE Xplore. Restrictions apply.

Appendix 5

V

R. Kalakoti, R. Vaarandi, H. Bahşi, and S. Nõmm. Evaluating explainable ai for deep learning-based network intrusion detection system alert classification. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, pages 47–58. INSTICC, SciTePress, 2025

Evaluating Explainable AI for Deep Learning-Based Network Intrusion Detection System Alert Classification

Rajesh Kalakoti[1][a], Risto Vaarandi[1][b], Hayretdin Bahşi[1,2][c] and Sven Nõmm[1][d]
[1]*Department of Software Science, Tallinn University of Technology, Tallinn, Estonia*
[2]*School of Informatics, Computing, and Cyber Systems, Northern Arizona University, U.S.A.*

Keywords: Network Intrusion Detection System, NIDS Alerts, SOC, Evaluation of Explainability.

Abstract: A Network Intrusion Detection System (NIDS) monitors networks for cyber attacks and other unwanted activities. However, NIDS solutions often generate an overwhelming number of alerts daily, making it challenging for analysts to prioritize high-priority threats. While deep learning models promise to automate the prioritization of NIDS alerts, the lack of transparency in these models can undermine trust in their decision-making. This study highlights the critical need for explainable artificial intelligence (XAI) in NIDS alert classification to improve trust and interpretability. We employed a real-world NIDS alert dataset from Security Operations Center (SOC) of TalTech (Tallinn University Of Technology) in Estonia, developing a Long Short-Term Memory (LSTM) model to prioritize alerts. To explain the LSTM model's alert prioritization decisions, we implemented and compared four XAI methods: Local Interpretable Model-Agnostic Explanations (LIME), SHapley Additive exPlanations (SHAP), Integrated Gradients, and DeepLIFT. The quality of these XAI methods was assessed using a comprehensive framework that evaluated faithfulness, complexity, robustness, and reliability. Our results demonstrate that DeepLIFT consistently outperformed the other XAI methods, providing explanations with high faithfulness, low complexity, robust performance, and strong reliability. In collaboration with SOC analysts, we identified key features essential for effective alert classification. The strong alignment between these analyst-identified features and those obtained by the XAI methods validates their effectiveness and enhances the practical applicability of our approach.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Many organizations use open-source (e.g., Suricata and Snort) or commercial (e.g., Cisco NGIPS) NIDS platforms to identify malicious network traffic (Day and Burns, 2011). Most widely used NIDS platforms use human-created signatures to identify malicious network traffic. However, this often results in many alerts, with only a tiny fraction deserving closer attention from security analysts (Jyothsna et al., 2011).

In a typical SOC operation, security analysts analyze the alerts based on their impact on the security of the organizational assets and categorize them as high or low priority. At this stage, analysts also identify the false positives that are benign system activities but are flagged as alerts by NIDS. Security analysts find it challenging to identify high-priority alerts (Jyothsna et al., 2011). Machine learning (ML) Deep Learning (DL) methods constitute a significant solution to automatize these prioritization tasks and, thus, reduce SOC workloads, especially in the lower-tier levels of security monitoring and incident handling processes in the related literature, with approaches divided into supervised, unsupervised, and semi-automated methods (Vaarandi, 2021; Vaarandi and Mäses, 2022; Kalakoti et al., 2022). However, the explainability or interpretability of ML models arises as a significant concern in alert prioritization despite their significant contribution.

Explainable Artificial Intelligence (XAI or Explainable AI) is necessary for experts to verify alert classifications and for industries to comply with regulations (Goodman and Flaxman, 2017). In cybersecurity, it's vital to explain flagged network activities as potential threats. XAI helps meet compliance standards and improve systems by clarifying NIDS alert classifications and identifying crucial features for data

[a] https://orcid.org/0000-0001-7390-8034
[b] https://orcid.org/0000-0001-7781-5863
[c] https://orcid.org/0000-0001-8882-4095
[d] https://orcid.org/0000-0001-5571-1692

Kalakoti, R., Vaarandi, R., Bahşi, H. and Nõmm, S.
Evaluating Explainable AI for Deep Learning-Based Network Intrusion Detection System Alert Classification.
DOI: 10.5220/0013180700003899
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 1*, pages 47-58
ISBN: 978-989-758-735-1; ISSN: 2184-4356
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

collection. In the event of a security breach, XAI offers valuable insights for forensic analysis, helping to understand why specific alerts were or were not triggered, which is crucial in reconstructing the timeline and nature of an attack (Alam and Altiparmak, 2024). NIDS usually struggles with high false positive rates. XAI can enable security analysts to understand why particular benign activities are mistakenly flagged as threats, enabling more transparent system tuning and reducing false alarms (Moustafa et al., 2023).

Explainable AI (XAI) methods address the model opacity problem through various global and local explanation methods (Rawal et al., 2021). Several studies have studied explainable AI methods in intrusion detection (Alam and Altiparmak, 2024; Szczepański et al., 2020; Senevirathna et al., 2024; Moustafa et al., 2023). However, it is crucial to note that these studies did not comprehensively evaluate Explainable AI methods under various intrusion datasets and miscellaneous sets of Black box nature of AI models. This lack of comprehensive evaluation significantly affects the generality of such methods, highlighting the urgent need for further research in this area. Although XAI-based IDS tools are expected to be an integral part of network security to help security analysts in SOCs to enhance the efficiency and precise in network defence and threat mitigation, a key challenge of deploying XAI-Based model into network intrusion detection is assessing such tools, testing their quality, and evaluating the relevant security metrics. These challenges undermine the trust in using the XAI-IDS model for real-world deployment in network IDS systems.

In this paper, we propose a Long Short-Term Memory (LSTM) model for NIDS alert prioritization to improve transparency and Reliability. This study evaluates various XAI methods to bridge the gap between the high accuracy of complex ML models and the need for transparent, explainable decision-making in the cybersecurity problem domain. Objectives of the study include creating an explainable LSTM model for NIDS alert classification, comparing four advanced XAI methods, evaluating their performance using comprehensive metrics, and validating XAI-generated explanations based on four criteria: Faithfulness, Complexity, Robustness, and Reliability.

Faithfulness estimates how accurately the explanation reflects the model's behaviour, assuring that the local explanation represents the model's decision-making process. Robustness evaluates the stability of explanations under small input perturbations, which is vital for building faith in local explanations. Complexity assesses the simplicity of the explanations, as more detailed explanations are generally more interpretable and valuable for human understanding. Reliability guarantees that the explanations are consistent with established knowledge, such as the features identified by SOC analysts in this case.

We propose that explainable AI methods can provide explanations for the decision-making processes of the LSTM model, prioritizing NIDS alerts and ultimately boosting the trust and usefulness of these systems. This research particularly examined a real-world dataset of NIDS alerts using LSTM, interpreting the output decisions made by these models and evaluating them through both quantitative and qualitative (expert) evaluations. This study emphasizes artificial intelligence (XAI) in high-risk threat detection settings. Our research offers a perspective to the existing literature as the aspect of interpretability has not been explored in relation to the significance of NIDS alerts. This research suggests that a well-designed benchmarking study can identify high-performance detection models that provide high-quality explanations. Therefore, security experts may not need to sacrifice detection performance over a model for explainability in the addressed ML studies.

Our paper is structured as follows: Section 2 reviews related work on NIDS and XAI in NIDS, Section 3 outlines our methodology, Section 4 presents our results and discussions, and Section 5 provides our conclusions.

2 RELATED WORK

ML and DL have advanced the analysis of NIDS alerts. This section reviews key contributions in NIDS alert processing, focusing on classification, clustering, and explainable AI methods. It delves into studies addressing challenges such as alert prioritization, false positive reduction, and interpretable models in cybersecurity.

(Kidmose et al., 2020) proposed a three-phase method for NIDS alert classification (Kidmose et al., 2020). They used an LSTM and latent semantic analysis to convert textual alerts into vectors, clustered the vectors using the DBSCAN algorithm, and classified incoming alerts based on their similarity to the core points of the clusters. (Van Ede et al., 2022) developed a semi-automated method for classifying NIDS alerts and other security events,which involved detecting and analyzing event sequences using deep learning models, clustering with the DBSCAN algorithm, and human analysts labeling the resulting clusters (Van Ede et al., 2022). Labeled database was then used for semi-automated classification of additional

event sequences, with human analysts manually reviewing unclustered events.

In a paper(Mane and Rao, 2021), the authors utilized SHAP, LIME, Contrastive Explanations Method (CEM), ProtoDash, and Boolean Decision Rules via Column Generation (BRCG) over the NSL-KDD dataset (Tavallaee et al., 2009) for intrusion detection system (IDS). They demonstrated the factors that influence the prediction of cyber-attacks.

(Ban et al., 2023) proposed a method using an IWSVM-based classifier to detect critical NIDS alerts. The classifier assigned higher weights to repeated data points and the minority class of critical alerts. A clustering algorithm grouped alerts representing the same incident based on attributes such as IP addresses, service ports, and alert occurrence time. (Shin et al., 2019) developed an organizational platform using machine learning to analyze NIDS alert data with support for binary SVM and one-class SVM methods (Shin et al., 2019). In a paper (Feng et al., 2017), authors described another organizational implementation for processing NIDS alerts and other security events to identify at-risk users. (Wang et al., 2019) used a graph-based method to eliminate false alerts and applied GBDT algorithms for alert classification. (Ban et al., 2021) used a large NIDS dataset to evaluate seven supervised machine learning methods (Ban et al., 2021). They found that Weighted SVM, SVM, and AB (Adaboost) produced the best results, while two isolation forest-based unsupervised algorithms provided lower precision than the evaluated supervised algorithms.

It is important to note that a large body of research is devoted to replacing NIDS with ML-based systems (Tsai et al., 2009). However, organizations use signature-based NIDSs due to the wide availability of this technology and complex SOC processes evolving around these systems. Thus, prioritizing NIDS alerts is a significant real-world challenge in SOCs. Various research studies have addressed the explainability of ML-based NIDS systems. However, to our knowledge, the explainability of the ML models developed for NIDS alert prioritization has not been studied in the literature.

(Szczepański et al., 2020) introduced the hybrid Oracle Explainer IDS, which combines artificial neural networks and decision trees to achieve high accuracy and provide human-understandable explanations for its decisions (Szczepański et al., 2020). In a paper (Senevirathna et al., 2024), authors have developed an Oracle-based Explainer module that uses the closest cluster to generate an explanation for the decision. A study explores how explanations in the context of 5G security can be targeted and weakened using scaffolding techniques. The authors suggest a framework for carrying out the scaffolding attack within a security setting, which involves selecting features and training models by combining explainable AI methods. (Zolanvari et al., 2021)(Zolanvari et al., 2021) introduced a model-agnostic XAI framework called TRUST for numerical applications. It uses factor analysis to transform input features, mutual information to rank features, and a multimodal Gaussian distribution to generate new samples for each class label.

Some other studies have explored explainable AI methods in intrusion detection (Alam and Altiparmak, 2024; Szczepański et al., 2020; Kumar and Thing, 2024; Kalakoti et al., 2024a; Kalakoti et al., 2024c; Kalakoti et al., 2024b; Kalakoti et al., 2023). In contrast to studies on machine learning-based Network Intrusion Detection Systems (NIDSs), our research emphasizes the significance of making NIDS alerts understandable through model transparency. Our approach incorporates eXplainable AI (XAI) techniques to evaluate their effectiveness in clarifying NIDS alert classifications. We worked with a real world NIDS dataset from an environment making our findings more relevant than those based on old data sets. Our evaluation criteria cover aspects such as the reliability, faithfulness, robustness and complexity of explanations assessing explainability within this domain. By engaging Security Operations Center (SOC) analysts in verifying our XAI findings we bridge the gap, between machine learning models and human knowledge. This progress enhances XAI in the field of cybersecurity, offering perspectives for developing transparent and reliable NIDS alert critical prioritization systems.

3 METHODOLOGY

3.1 Dataset

Our study makes use of a NIDS alert dataset taken from a Suricata NIDS system deployed at the Security Operations Center (SOC) of Tallinn University of Technology (Taltech). The dataset was gathered using the Customized Stream Clustering Algorithm for Suricata (CSCAS) to analyze alerts from Suricata NIDS at TalTechs SOC. Data was collected over a span of 60 days, from January to March 2022 during which Suricata generated alerts, for network activity involving 45,339 hosts and 4401 TalTech hosts. The categorized dataset can be accessed at the link; https://github.com/ristov/nids-alert-data.

Throughout the data collection phase CSCAS op-

erated with settings; SessionLength = 300 seconds (5 minutes) SessionTimeout = 60 seconds (1 minute) ClusterTimeout = 604,800 seconds (1 week) CandTimeout = 36,000 seconds (10 hours) MaxCandAge = 864,000 seconds (10 days) and $\alpha = 0.01$. These configurations have been employed for CSCAS in an environment since 2021 and were determined to be optimal as outlined in (Vaarandi, 2021). NIDS Alerts are classified as either "important" or "irrelevant." Data points of network traffic were generated by a customized version of SCAS, a stream clustering algorithm, and have labels indicating whether they are regarded as inliers or outliers by SCAS. Data points are labeled by humans to indicate if they represent important or irrelevant alert groups. Important alerts are prioritized in the SOC security monitoring processes. Irrelevant alerts include low-priority threats (e.g., frequent scanning for old vulnerabilities) or false positives (e.g., alerts related to attempts to resolve botnet C&C server DNS names not originating from infected computers but from specific security applications). The description of the dataset (Vaarandi and Guerra-Manzanares, 2024) is given below:

- Timestamp – alert group reporting time
- SignatureText – human readable alert text
- SignatureID – numerical signature ID
- SignatureMatchesPerDay – Average matches per day by the triggering signature (set to 0 if first match was less than 24 hours ago).
- AlertCount – the number of alerts in the current alert group
- Proto – numerical protocol ID (e.g., 6 denotes TCP and 17 UDP)
- ExtIP – anonymized IP address of the external host (extipN, where N is a number that identifies the given IP address)
- ExtPort – port at the external host, set to -1 if alerts involve multiple external ports
- IntIP – Anonymized IP address of the internal host (intipN), set to -1 if alerts involved multiple internal IP addresses.
- IntPort – port at the internal host, set to -1 if alerts involve multiple internal ports.
- Similarity – The overall similarity of this alert group to others in the same cluster or, if it's an outlier, to other outlier alert groups. The value ranges from 0 to 1, with higher values indicating a high degree of similarity.
- SCAS – The label assigned by the customized version of SCAS. Here, 0 denotes an inlier and 1 denotes an outlier.
- AttrSimilarity – similarity for the network IDS alert attribute Attr (there are 34 attributes in total). Set to -1 if the attribute Attr is not set for the given signature, otherwise ranges from 0 to 1. The field indicates how often the attribute value has been observed in other alert groups from the same cluster (or in other outlier alert groups if the current alert group is an outlier).

We collaborated with Security Operations Center (SOC) analysts from TalTech, Estonia to estimate the reliability of the post-hoc explanations generated for the decisions of the black-box model, which is the DL model induced for alert classification in this work. A detailed description of TalTech SOC can be found in (Vaarandi and Mäses, 2022). Leveraging their expertise in managing Network Intrusion Detection System (NIDS) alerts, the SOC team at TalTech identified the five features for determining alert significance as outlined in Table 1. These features act as benchmarking reference features in our research to evaluate how well our XAI algorithms perform.

Table 1: Key Features Identified by Taltech SOC Analyst for Determining NIDS Alert Significance.

SignatureMatchesPerDay
Similarity
SCAS
SignatureID
SignatureIDSimilarity

For our work, the dataset excluded 'SignatureText' and 'Timestamp' features as external IP addresses ("ExtIP" feature) and internal IP addresses ("IntIP" feature) prior, to model training.

3.2 Long Short-Term Memory for NIDS Alerts

In this study, we proposed long-term memory (LSTM) to classify whether a given NIDS alert group needs immediate attention (Important class label) or can be assessed as less critical (Irrelevant class label). LSTM is a neural network designed to address the long-term dependence problem in traditional recurrent neural networks. It introduces forget, input, and output gates to control the flow of information and maintain long-term memory. Figure 1 shows structure of the hidden layer of the LSTM network. The forget gate adapts to the context, discarding unnecessary information. It uses a sigmoid function to produce a value between 0 and 1, then multiplied by the previous cell state. A value of 0 means complete forgetting, while 1 means fully retained.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

The input gate enhances the necessary information for the new cell state, and its output is a sigmoid

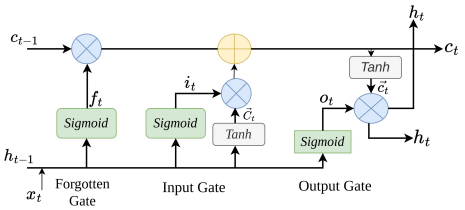


Figure 1: Hidden Layer Architecture of LSTM Network.

function with a range of 0 to 1, which is multiplied by the current cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{3}$$

Then the old and new state information can be combined to construct the final new cell state.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{4}$$

The output is determined by the output gate, which uses a sigmoid function to select information to be output along with the final cell state and the Tanh function.

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = O_t \times \tanh(C_t) \tag{6}$$

For training LSTM model, We selected 10,000 data points for each class label ('irrelevant' and 'important'), resulting in a total of 20,000 samples. The The dataset was divided into training and testing sets at an 80 20-split ratio. We applied the data normalization technique to the dataset to convert the values to a standard scale. We used Min-Max normalization, one of several available techniques, to transform and normalize the input features to scale them within a range of 0 to 1, as shown in Equation 7.

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{7}$$

where $x_{\min}$ is the smallest value of the feature, $x_{\max}$ is the largest value of the feature, and x is the actual value of the feature. The normalized feature, x', ranges between 0 and 1.

We used RandomSearch hyperparameter tuning with Ray Tune library[1] to train LSTM model. We evaluated the performance of LSTM model for NIDS alerts classification using a confusion matrix. In NIDS alerts classification, True Positives (TP) are the number of important alerts correctly classified as important, True Negatives (TN) are the number of irrelevant alerts correctly classified as irrelevant, False Positives (FP) are the number of irrelevant alerts incorrectly classified as important. False Negatives (FN) are the number of important alerts incorrectly classified as irrelevant. we used the following evaluation metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

We used softmax activation function at the output layer to predict class labels, which provides prediction probabilities for each class and enables us to understand the model's confidence and the probability distribution. It's also crucial to evaluate XAI techniques based on metrics like faithfulness, monotonicity and max sensitivity as discussed in section 3.4.

3.3 Explainable AI Methods

When explaining the model using Explainable AI, there are two approaches: model agnostic and model specific. Explainable AI methods are also categorized into two types explanations. Local explanations interpret individual predictions and global explanations that offer an overview of the model's behaviour. Our goal is to enhance the explainability of NIDS alerts detected by LSTM model. We have utilized four popular XAI feature attribution methods. Will provide a brief overview of each one. The following outlines the four methods (LIME, SHAP, Integrated Gradients (IG) and DeepLIFT) all designed to clarify instances and shed light on how the model makes decisions, for specific predictions. Let $x \in \mathbb{R}^d$ be the input, where d is the feature set dimensionality. The black box model $\mathcal{M}$ maps input to output $\mathcal{M}(x) \in \mathcal{Y}$. Dataset $\mathcal{D} = (x^i, y^i)$ contains all input-output pairs. The explanation mapping $\mathbf{g}$ for predictor $\mathcal{M}$ and point x returns importance scores $\mathbf{g}(\mathcal{M}, x) = \phi_x \in \mathbb{R}^d$ for all features. Let $D : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R} \geq 0$ be a metric in the explanation space and $S : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R} \geq 0$ a metric in the input space. The evaluation criterion μ maps predictor $\mathcal{M}$, explainer $\mathbf{g}$, and point x to a scalar.

[1] https://docs.ray.io/en/latest/tune/index.html

3.3.1 SHAP

SHAP (Lundberg and Lee, 2017) uses Shapley values from game theory to attribute the importance of each feature to a model's prediction, providing a unified measure of feature importance. SHAP based on Shapley values, is defined as: $\mathbf{g}(\mathcal{M},x) = \phi_0 + \sum_{j=1}^{M} \phi_j$ where ϕ_j is the feature attribution of feature j. SHAP's DeepExplainer was used in this study.

3.3.2 LIME

LIME (Ribeiro et al., 2016) (Local Interpretable Model-agnostic Explanations) constructs a locally interpretable model around a specific prediction. It works by perturbing the input and fitting a simple model, like a linear model, to explain the behaviour of the black box model in the vicinity of the prediction of interest. LIME approximates model behavior locally around (x) by minimizing: $\underset{g \in G}{\operatorname{argmin}} L(\mathcal{M},g,\pi_x) + \Omega(g)$ where g is an interpretable model in the neighborhood of (x).

3.3.3 Integrated Gradients

Integrated Gradients (IG) (Sundararajan et al., 2017) attributes the prediction of a deep network to its inputs by integrating the gradients along a straight-line path from a baseline input to the actual input. This method satisfies desirable axioms like completeness and sensitivity, providing a theoretically sound approach to feature attribution. IG attributes feature importance by integrating model gradients from a baseline $\mathbf{g}(\mathcal{M},x) = \text{IG}(x) = (x-\bar{x}) \times \int_{\alpha=0}^{1} \frac{\partial \mathcal{M}(\bar{x}+\alpha\cdot(x-\bar{x}))}{\partial x} d\alpha$ where $\bar{x}$ is the baseline input.

3.3.4 DeepLIFT

DeepLIFT (Shrikumar et al., 2017) assigns each input (x) a value $C_{\Delta x_i \Delta y}$ representing its deviation from a reference value, satisfying: $\sum_{i=1}^{n} C_{\Delta x_i \Delta o} = \Delta o$ where $o = \mathcal{M}(x)$ and Δo is the difference between model output and reference value.

3.4 Evaluation of Explainable AI Methods

The evaluation of Explainable AI methods is crucial to ensure that the explanations they provided are transparent, also accurate and reliable. We employ four key metrics to assess the quality of our explanations for LSTM Model based NIDS alerts: Reliability, Faithfulness, Robustness and Complexity. These metrics provide a comprehensive evaluation framework that addresses different aspects of explanation quality. XAI evaluation is categorized into three groups (Coroama and Groza, 2022): user-focused evaluation, application-focused evaluation, and functionality-focused evaluation. The first two categories are part of human-centered evaluation and are broken down into subjective and objective measures.

3.4.1 Reliability

An explanation should be centered around the region of interest, the ground truth **GT**. $\mathbf{g}(\mathcal{M},\mathbf{x}) = \mathbf{GT}$. 'Major' parts of an explanation should lie inside the ground truth mask GT(x) for both Relevance Mass Accuracy and Relevance Rank Accuracy metrics used in this work, and the Ground truth mask ([0,1]) was determined by the features SOC Analysts identified (see Table. 1).Truth-based measures relevance rank accuracy and relevance mask accuracy are derived from (Arras et al., 2022).

(a) Relevance Rank Accuracy (RRA) (Arras et al., 2022): Relevance rank accuracy measures how much of the high-intensity relevance lies within the ground truth. We sort the top K values of $\mathbf{g}(\mathcal{M},\mathbf{x})$ in decreasing order $\mathbf{X}_{\text{top}K} = \{x_1,...,x_K \mid \mathbf{g}(\mathcal{M},\mathbf{x})_{x_1} > ... > \mathbf{g}(\mathcal{M},\mathbf{x})_{x_K}\}$.

$$RRA = \frac{|\mathbf{X}_{\text{top}_k} \cap \mathbf{GT}(\mathbf{x})|}{|\mathbf{GT}(\mathbf{x})|}$$

Here top_k are features Identified by SOC Analyst.

(b) Relevance Mass Accuracy (RMA) (Arras et al., 2022): The relevance mass accuracy is calculated as the sum of the explanation values within the ground truth mask divided by the sum of all values.

$$\text{RMA} = \frac{\sum_i \mathbf{g}(\mathcal{M},\mathbf{x})_i \cdot \mathbf{GT}(x_i)}{\sum_i \mathbf{g}(\mathcal{M},\mathbf{x})_i}$$

3.4.2 Faithfulness

The explanation algorithm $\mathbf{g}$ should replicate the model's behavior. $\mathbf{g}(\mathcal{M},\mathbf{x}) \approx \mathcal{M}(\mathbf{x})$. Faithfulness quantifies the consistency between the prediction model $\mathcal{M}$ and explanation g. For evaluating the Faithfulness of explanations, the Faithfulness correlation (Bhatt et al., 2020) and Monotonocity (Luss et al., 2019) metrics were used.

(a) High Faithfulness Correlation: Faithfulness measures how well the explanation function $\mathbf{g}$ aligns feature importance scores with the black-

box model $\mathcal{M}$

$$\mu_F(\mathcal{M},g;x) = \underset{\mathcal{B}\in\binom{|d|}{|\mathcal{B}|}}{\text{corr}} \left(\sum_{i\in\mathcal{B}} g(\mathcal{M},x)_i, \mathcal{M}(x) - \mathcal{M}(x_{\mathcal{B}}) \right) \tag{12}$$

where $x_{\mathcal{B}} = x_i | i \in \mathcal{B}\}$ High Faithfulness correlation metric iteratively substitutes a random subset of given attributions with a baseline value $\mathcal{B}$. Then, it measures the correlation between the sum of these attributions and the difference in the model's output.

(b) Monotonicity: Let $x, x' \in \mathcal{R}^d$ be two input points such that $x_i \leq x'_i$ for all $i \in 1,2,\ldots,d$. $\mathcal{M}$ and g are said to be monotonic if the following condition holds: For any subset $S \subseteq 1,2,\ldots,d$, the sum of the attributions of the features in S should be nonnegative when moving from x to x', that is, $\sum_{i\in S} g(\mathcal{M},x)i \leq \sum_{i\in S} g(\mathcal{M},x')_i$ implies

$$\mathcal{M}(x) - \mathcal{M}(x_{[x_s=\bar{x}_s]}) \leq \mathcal{M}(x') - \mathcal{M}(x'[x'_s = \bar{x}_s])$$

3.4.3 Robustness

Robustness refers to similar inputs should result in similar explanations. $\mathbf{g}(\mathcal{M},\mathbf{x}) \approx \mathbf{g}(\mathcal{M},\mathbf{x} + \varepsilon)$ for small ε.

(a) Max Sensitivity: Max sensitivity (Bhatt et al., 2020): is used to ensure that nearby inputs with similar model output have similar explanations, it is desirable for the explanation function g to have a low sensitivity in the region surrounding the point of interest x, assuming the differentiability of the predictor function $\mathcal{M}$. Maximum sensitivity of an explanation function g at a point of interest x in its neighbourhood is defined as follows: Consider a neighbourhood N_r of points within a radius r of x, denoted by $N_r = z \in D_x | p(x,z) \leq r, \mathcal{M}(x) = \mathcal{M}(x)(z)$, where D is the distance metric, and p is the proximity function. Given a predictor $\mathcal{M}(x)$, a distance metric D, a proximity function p, a radius r, and a point x, we define the maximum sensitivity of g at x as follows:

$$\mu_M(\mathcal{M}(x),g,r;x) = \max_{z\in N_r} D(g(\mathcal{M}(x),x), g(\mathcal{M}(x),z)) \tag{13}$$

3.4.4 Complexity

Explanations using a smaller number of features are preferred. It is assumed that explanations using a large number of features are difficult for the user to understand.min $\|\mathbf{g}(\mathcal{M},\mathbf{x})\|_0$.

(a) Low Complexity: Low complexity (Bhatt et al., 2020) metric computes the entropy of each feature's fractional contribution to the total attribution magnitude individually.

$$\mu_C(\mathcal{M},g;x) = -\sum_{i=1}^{d} P_g(i) \log P_g(i) \tag{14}$$

where

$$P_g(i) = \frac{|g(\mathcal{M},x)i|}{\sum j \in |d||g(\mathcal{M},x)_j|}; P_g = P_g(1),....P_g(d) \tag{15}$$

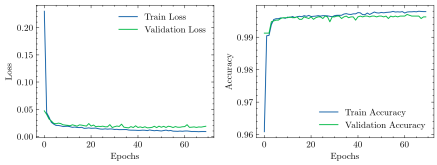
The experiments were carried out on a computer running Pop!_OS 22.04 LTS x86_64 operating system with the following hardware configuration: 32 GB of DDR4-2666R ECC RAM, AMD Ryzen 5 5600G with Radeon Graphics (12) @ 3.900GHz processor. The scripts were developed using the Python 3.9 programming language and Pytorch library. For the implementation of the Integrated Gradients and DeepLIFT explainers, Captum library was used.

4 RESULTS & DISCUSSIONS

In this section, we present the results of our research, including an analysis of the LSTM model's performance and explanations of LSTM model using Explainable AI methods and the quality of evaluation for these explanations based on four criteria: faithfulness, complexity, reliability, and robustness.

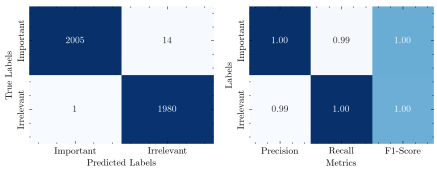
Figure. 3a shows the confusion matrix, indicating the model's strong classification performance for test data of 4000. It correctly classified 2005 important alerts and 1980 irrelevant alerts, with only 14 misclassifications of irrelevant alerts as necessary, demonstrating high accuracy and a low false positive rate. Figure. 2a shows the training and validation loss over 70 epochs obtained through random search parameter tuning. Initially, both decrease rapidly before stabilizing, indicating convergence without overfitting. The close alignment of the training and validation loss curves represents good generalization to unseen data. Figure. 2b shows the training and validation accuracy, which quickly stabilizes above 99.5%, indicating strong model performance. In Figure. 3b, from the classification report, the model achieves near-perfect precision, recall, and F1-score scores for both classes.

In this paper, we utilized 4 different explainable AI methods (LIME, SHAP, IG, and DeepLift) to explain the predictions of our LSTM model on the test data. LIME analyzes how the model assigns probabilities to categories by comparing these probabilities with the actual category of the data point. SHAP method provides single-data-point explanations for models, giving insights. In explanations, a particular



(a) Loss (b) Accuracy.

Figure 2: Loss and Accuracy from Best LSTM Performance Model.



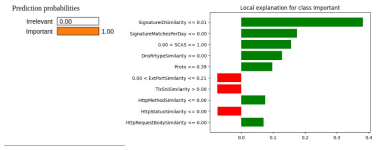
(a) Confusion Matrix (b) Classification Report

Figure 3: Confusion Matrix and Classification Report.

data point is selected to demonstrate how each feature influences the model's prediction.

Fig. 4a shows an local explanation from LIME method for a NIDS alert labeled as "Important.". Left side presents prediction probabilities with a 100% probability for the "Important" class. On the right side it illustrates the impact of features. For instance, when the feature 'SignatureIDSimilarity' is less than or equal to 0.01, it positively affects the "Important" classification of NIDS alert. Additionally, 'SignatureMatchesPerDay' and 'SCAS' being less than or equal to 1.00 also contribute positively. Conversely, 'ExtPortSimilarity' and 'TlsSniSimilarity' have impacts, suggesting that some NIDS alerts may not be relevant. SHAP employs Shapley values to showcase how features influence model predictions in Fig. 4b of force plot, red bar signifies the positive impact while blue bar indicates the negative impact on the model output. Each bar demonstrates whether the features bring the predicted value closer to or farther from the base value of 0.02463. The plot's base value is the average of all prediction values. Each strip in the plot displays the impact of the features on moving the predicted value closer to or farther from the base value. Final prediction is deemed an "important class label", with a value of 1.00 for this NIDS alert. Features, like 'IntPort' (Internal Port) 'SignatureIDSimilarity'. ExtPort' (External Port) along with 'SignatureID' play a role in indicating the importance of NIDS alert. However, the feature 'HttpStatusSimilarity' might suggest that this alert could be a less critical feature to its impact.

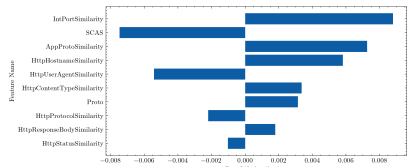
DeepLift is a technique used to attribute the output of LSTM model to its input features by comparing neuron activation to a reference activation and assigning contribution scores based on the variance. Fig. 4c illustrates the significance of features using the DeepLift explainer for the 10 features of a NIDS alert data point labeled as "important." The negative attribution of 'SCAS' suggests its influence on classifying as "Important" in NIDS alerts. Additionally 'HttpMethodSimilarity' and 'IntIP' show negative attributions while HttpContentTypeSimilarity has a slight positive impact countering the "Important" classification. IG attribute a LSTM model's prediction its input features by integrating gradients of the model's output with respect to the input along from a baseline to the input. This explanation technique works best for models that use linear activation functions. Fig. 4d showcases feature importance using IG explainer for a data point in the "Important" NIDS alert class label among the 10 features. Features such, as 'SignatureID' 'SCAS,' and 'HttpStatusSimilarity' display



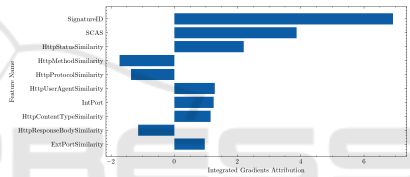
(a) LIME explanations for important NIDS alerts using an LSTM model



(b) SHAP explanations for an important NIDS alert data point using an LSTM model



(c) DeepLIFT feature importance for an important NIDS alert data point using an LSTM model

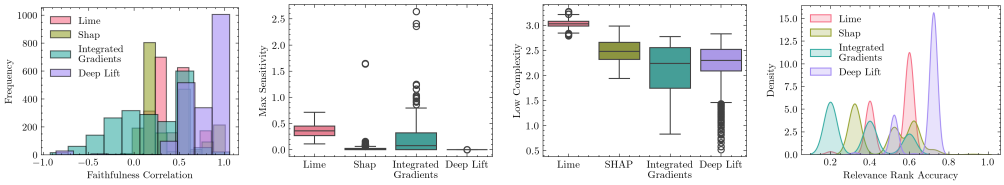


(d) Integrated Gradients feature importance for an important NIDS alert data point using an LSTM model

Figure 4: Explanations for an important NIDS alert data point using an LSTM model.

Table 2: Evaluation Results of Explainable AI Methods: Mean (μ) and Standard Deviation (σ) Values.

Explanation Criterion	Faithfulness		Robustness	Complexity	Reliability	
Explainer/Metric	High Faithfulnes $\mu \pm \sigma$	Monotonicity μ	Max Sensitvity $\mu \pm \sigma$	Low Complexity $\mu \pm \sigma$	Relevance Mass Accuracy $\mu \pm \sigma$	Relevancy Rank Accuracy $\mu \pm \sigma$
Lime	0.4209 ± 0.1835	59.55%	0.3617 ± 0.1152	3.0318 ± 0.0703	0.6234 ± 9.7008	0.5250 ± 0.1041
Shap	0.3959 ± 0.2928	64.45%	0.0245 ± 0.0862	2.4677 ± 0.2074	0.6527 ± 3.8334	0.4743 ± 0.1418
IG	0.1761 ± 0.3815	73.70%	0.1774 ± 0.2505	**2.1745 ± 0.4134**	0.5939 ± 0.6840	0.3410 ± 0.1545
Deep Lift	**0.7559 ± 0.2681**	**78.35%**	**0.0008 ± 0.0004**	2.2635 ± 0.3299	**0.7812 ± 25.2805**	**0.6754 ± 0.0897**



(a) High Faithfulness (b) Max Sensitivity (c) Low Complexity (d) Relevancy Rank Accuracy

Figure 5: Quality of Explainable AI evaluation metrics distribution.

attributions.

Our analysis comparing the features identified by the TalTech SOC analyst closely aligned with those derived by explainers used in our LSTM model to classify "important" NIDS alerts. The 5 features recognized by SOC experts in Table 1 proved significant across explainers, although their order of feature importance varied. For instance, 'SignatureIDSimilarity' and 'SignatureID', highlighted by SOC analysts, impacted the SHAP explainer for NIDS alerts. The presence of "SCAS" was notable in LIME, IG, and DeepLift, confirming its significance. The importance of 'SignatureMatchesPerDay' varied among explainers within LIME. Notably upon reviewing the 10 features highlighted by each explainer, we noticed an overlap with the features identified by SOC analysts particularly emphasizing 'SignatureID', 'SignatureIDSimilarity', 'SCAS' and 'SignatureMatchesPerDay'. We assessed the quality explanation of XAI methods, for LSTM model based alerts using metrics based on four criteria: faithfulness, robustness, complexity and reliability.

We evaluated the quality of explanations obtained by XAI methods for Long Short-Term Memory (LSTM) network-based NIDS alert classification across 2000 data points using metrics based on four criteria: Faithfulness, robustness, complexity, and reliability. Table 5 shows the results of the quality of explanation for XAI methods. LSTM model prediction probabilities were computed using the Softmax activation function. To evaluate the Faithfulness of explanations, we employed high faithfulness correlations and monotonicity. High Faithfulness of XAI methods was evaluated by studying the correlation between attribute importance assigned by the XAI method and their impact on the model's probabilities. A high faithfulness correlation value suggests that the explanations effectively capture the model's behaviour and can be regarded as faithful. Table. 2 shows the evaluation results of xai methods. Mean (μ) and standard deviation (σ) values were calculated for the test data of XAI computed metrics for 2000 test data points. Deep Lift achieved the highest Faithfulness mean and standard deviation correlation values of 0.7559 ± 0.2681 for test data points. We also analyzed the monotonicity of the explanation to understand how individual features affect model probability by adding each attribute to enhance its importance and observing its influence on the model's probability. By assessing the monotonicity of the explainer, we can measure how the explanations change monotonically with respect to the input features. Deep LIFT achieved high monotonicity with 78% (μ).

To measure complexity, we calculate the entropy of feature attribution in the explanations. Complexity measures the conciseness of explanations derived by the explainer. Among xai methods assessed by low complexity metric, Integrated Gradients (IG) achieved lower complexity (2.174 ± 0.413) closely followed by DeepLift (2.264 ± 0.330.)

The sensitivity metric assesses the consistency of the explainers' output, ensuring that similar inputs in the feature space of model outputs have similar explanations when sensitivity is low. For this metric, we used the Euclidean distance with a radius value of 0.1 to find the nearest neighbour points related to the prediction label of an explanation which helps to identify data points in the feature space with similar expla-

Table 3: Statistical Comparison of Explainers Across Multiple Metrics (p-values).

Metric	Explainer	Shap	IG	Deep Lift
Faithfulness	LIME	L (3.34e-41)	L (1.03e-134)	D (5.61e-185)
	SHAP	-	S (6.22e-91)	D (1.03e-169)
	IG		-	D (1.30e-230)
Max Sensitivity	LIME	S (0.00e+00)	I (1.64e-221)	D (0.00e+00)
	SHAP	-	S (1.38e-185)	D (3.54e-126)
	IG		-	D (3.29e-126)
Low Complexity	LIME	S (0.00e+00)	I (0.00e+00)	D (0.00e+00)
	SHAP	-	I (5.45e-146)	D (1.26e-88)
	IG		-	I (1.07e-42)
RMA	LIME	S (5.12e-25)	L (1.97e-80)	D (6.22e-83)
	SHAP	-	S (4.67e-155)	D (6.47e-91)
	IG		-	D (2.82e-54)
RRA	LIME	L (7.61e-39)	L (3.07e-210)	D (0.00e+00)
	SHAP	-	S (5.52e-155)	D (3.97e-253)
	IG		-	D (0.00e+00)

D (Deep Lift), L (LIME), S (SHAP), and I (Integrated Gradients)
indicate the better performing explainer in each pairwise comparison.
$p > 0.05$ — No significant evidence against H_0; H_0 is not rejected
$0.01 < p \leq 0.05$ — Significant evidence against H_0; H_1 is accepted at 95% confidence level
$0.001 < p \leq 0.01$ — Strong evidence against H_0; H_1 is accepted at 99% confidence level
$p \leq 0.001$ — Very strong evidence against H_0; H_1 is accepted at 99.9% confidence level.

nations for the predicted label. Deep LIFT achieved Lower sensitivity with max sensitivity metric (0.0008 ± 0.0004).

Two metrics, Relevance Mass Accuracy and Relevance Rank Accuracy, were used to evaluate the reliability of explanations. These metrics validated the explanations by comparing them to a ground truth mask based on features identified through collaboration with an SoC analyst. For both Relevance Mass Accuracy (0.781 ± 25.281) and Relevancy Rank Accuracy (0.6754 ± 0.089) metrics, Deep lift explanations were reliable. Figure. 5 illustrates the distribution of XAI metric results for 2000 data points, highlighting that DeepLIFT's explanations demonstrate high faithfulness, lower sensitivity, lower complexity, and more relevance rank accuracy. Faithfulness correlation values for DeepLIFT indicate a strong skew towards higher levels, showing a high degree of consistency through monotonicity. Moreover, the entropy values of feature importance scores for IG and DeepLIFT are more evenly spread towards the lower end than other explainers. The sensitivity values for the DeepLIFT explainer are also more evenly spread to lower values in maximum sensitivity metrics. Additionally, using Relevance Rank Accuracy, DeepLIFT consistently achieves a high relevance rank accuracy with less variation, centred around 0.8.

Following established practices in the statistical analysis of XAI methods evaluation (Jesus et al., 2021), we employed the Wilcoxon signed-ranks test (Woolson, 2005) to evaluate the statistical significance of differences (Demšar, 2006) in XAI metric scores between pairs of explainers (i.e., $explainer_A$, $explainer_B$) for NIDS alert classification. The null hypothesis (H_0) is that the explainable AI metric scores of the explainers are equivalent, i.e., there is no significant difference between the explainers (XAI Metric Score($explainer_A$) = XAI Metric Score($explainer_B$)). The alternative hypothesis (H_1) is that they are not equivalent (XAI Metric Score($explainer_A$) $\neq$ XAI Metric Score($explainer_B$)), indicating a significant difference in their explainer metric scores. XAI metrics used in this study are High Faithfulness, Max Sensitivity, Low Complexity, Relevance Mass Accuracy, and Relevancy Rank Accuracy. This test was conducted separately for each metric to assess the performance differences among the explainers comprehensively.

The statistical analysis in Table 3 shows significant differences among the explainers for all metrics, with p-values consistently below 0.05, demonstrating strong evidence against the null hypothesis. DeepLift explainer is better regarding faithfulness, max sensitivity, RMA, and RRA when compared pairwise ($p < 0.001$ for all comparisons) with other explainers. The relative performance of SHAP, LIME, and IG varies across metrics can be seen Table 3.

We have also provided a global explanation us-

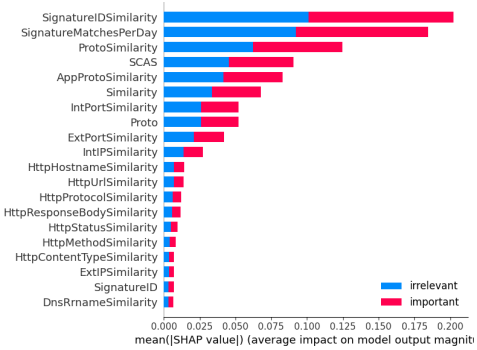


Figure 6: SHAP global explanation for LSTM model.

ing SHAP values for all the testing data of the LSTM model. A higher value positively impacts the prediction, while a lower value contributes negatively. Figure. 6 shows the global explanation of the LSTM model. The graph illustrates the average impact of each feature on the model's output magnitude for the class labels, "irrelevant" and "important" classifications. SignatureIDSimilarity, SignatureMatchesPerDay, ProtoSimilarity and SCAS are most impact ful features for important nids alerts. Notably, these top features align with those identified by human expert SOC analysts. Lower-ranked features such as HTTP-related similarities (e.g., HttpHostnameSimilarity, HttpUrlSimilarity) and IP-related features (e.g., ExtIPSimilarity) have comparatively less impact on the model's decisions.

5 CONCLUSIONS AND FUTURE WORK

This research presents explainable artificial intelligence (XAI) based Network Intrusion Detection Systems (NIDS) alert classification utilizing a Long Short-Term Memory (LSTM) model. We have showcased how enhancing the explainability and trustworthiness of AI-powered cybersecurity systems can be achieved by clarifying the output predictions of these LSTM models through four XAI techniques: LIME, SHAP, Integrated Gradients, and DeepLIFT. Our thorough assessment of the XAI framework, considering the aspects of faithfulness, complexity, robustness, and reliability, has evaluated how well these XAI methods explain NIDS alerts. The superior performance of DeepLIFT across these evaluation metrics underscores its potential as a preferred method for interpreting NIDS alert classifications. Notably, the substantial alignment between explanations generated by XAI techniques and features identified by SOC analysts validates their effectiveness in capturing domain expertise. This research makes a contribution by bridging the gap between the high accuracy of opaque machine learning models and the necessity for transparent decision-making in cybersecurity operations. By proposing a framework to explain black box model decisions and assess XAI in NIDS applications, we provided comprehensive benchmarking results, including evaluation metrics for developing transparent and interpretable AI systems in crucial security domains.

REFERENCES

Alam, S. and Altiparmak, Z. (2024). Xai-cf–examining the role of explainable artificial intelligence in cyber forensics. *arXiv preprint arXiv:2402.02452*.

Arras, L., Osman, A., and Samek, W. (2022). Clevr-xai: A benchmark dataset for the ground truth evaluation of neural network explanations. *Information Fusion*, 81:14–40.

Ban, T., Samuel, N., Takahashi, T., and Inoue, D. (2021). Combat security alert fatigue with ai-assisted techniques. In *Proceedings of the 14th Cyber Security Experimentation and Test Workshop*, pages 9–16.

Ban, T., Takahashi, T., Ndichu, S., and Inoue, D. (2023). Breaking alert fatigue: Ai-assisted siem framework for effective incident response. *Applied Sciences*, 13(11):6610.

Bhatt, U., Weller, A., and Moura, J. M. (2020). Evaluating and aggregating feature-based model explanations. *arXiv preprint arXiv:2005.00631*.

Coroama, L. and Groza, A. (2022). Evaluation metrics in explainable artificial intelligence (xai). In *International conference on advanced research in technologies, information, innovation and sustainability*, pages 401–413. Springer.

Day, D. and Burns, B. (2011). A performance analysis of snort and suricata network intrusion detection and prevention engines. In *Fifth international conference on digital society, Gosier, Guadeloupe*, pages 187–192.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30.

Feng, C., Wu, S., and Liu, N. (2017). A user-centric machine learning framework for cyber security operations center. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 173–175. IEEE.

Goodman, B. and Flaxman, S. (2017). European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57.

Jesus, S., Belém, C., Balayan, V., Bento, J., Saleiro, P., Bizarro, P., and Gama, J. (2021). How can i choose an explainer? an application-grounded evaluation of post-hoc explanations. In *Proceedings of the*

2021 ACM conference on fairness, accountability, and transparency, pages 805–815.

Jyothsna, V., Prasad, R., and Prasad, K. M. (2011). A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 28(7):26–35.

Kalakoti, R., Bahsi, H., and Nõmm, S. (2024a). Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection. *IEEE Internet of Things Journal*.

Kalakoti, R., Bahsi, H., and Nõmm, S. (2024b). Explainable federated learning for botnet detection in iot networks. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 01–08.

Kalakoti, R., Nõmm, S., and Bahsi, H. (2022). In-depth feature selection for the statistical machine learning-based botnet detection in iot networks. *IEEE Access*, 10:94518–94535.

Kalakoti, R., Nõmm, S., and Bahsi, H. (2023). Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai). In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 595–601. IEEE.

Kalakoti, R., Nõmm, S., and Bahsi, H. (2024c). Enhancing iot botnet attack detection in socs with an explainable active learning framework. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 265–272. IEEE.

Kidmose, E., Stevanovic, M., Brandbyge, S., and Pedersen, J. M. (2020). Featureless discovery of correlated and false intrusion alerts. *IEEE Access*, 8:108748–108765.

Kumar, A. and Thing, V. L. (2024). Evaluating the explainability of state-of-the-art machine learning-based iot network intrusion detection systems. *arXiv preprint arXiv:2408.14040*.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Luss, R., Chen, P.-Y., Dhurandhar, A., Sattigeri, P., Shanmugam, K., and Tu, C.-C. (2019). Generating contrastive explanations with monotonic attribute functions. *arXiv preprint arXiv:1905.12698*, 3.

Mane, S. and Rao, D. (2021). Explaining network intrusion detection system using explainable ai framework. *arXiv preprint arXiv:2103.07110*.

Moustafa, N., Koroniotis, N., Keshk, M., Zomaya, A. Y., and Tari, Z. (2023). Explainable intrusion detection for cyber defences in the internet of things: Opportunities and solutions. *IEEE Communications Surveys & Tutorials*, 25(3):1775–1807.

Rawal, A., McCoy, J., Rawat, D. B., Sadler, B. M., and Amant, R. S. (2021). Recent advances in trustworthy explainable artificial intelligence: Status, challenges, and perspectives. *IEEE Transactions on Artificial Intelligence*, 3(6):852–866.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Senevirathna, T., Siniarski, B., Liyanage, M., and Wang, S. (2024). Deceiving post-hoc explainable ai (xai) methods in network intrusion detection. In *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, pages 107–112. IEEE.

Shin, I., Choi, Y., Kwon, T., Lee, H., and Song, J. (2019). Platform design and implementation for flexible data processing and building ml models of ids alerts. In *2019 14th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 64–71. IEEE.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Szczepański, M., Choraś, M., Pawlicki, M., and Kozik, R. (2020). Achieving explainability of intrusion detection system by hybrid oracle-explainer approach. In *2020 International Joint Conference on neural networks (IJCNN)*, pages 1–8. IEEE.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee.

Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., and Lin, W.-Y. (2009). Intrusion detection by machine learning: A review. *expert systems with applications*, 36(10):11994–12000.

Vaarandi, R. (2021). A stream clustering algorithm for classifying network ids alerts. In *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 14–19. IEEE.

Vaarandi, R. and Guerra-Manzanares, A. (2024). Stream clustering guided supervised learning for classifying nids alerts. *Future Generation Computer Systems*, 155:231–244.

Vaarandi, R. and Mäses, S. (2022). How to build a soc on a budget. In *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 171–177. IEEE.

Van Ede, T., Aghakhani, H., Spahn, N., Bortolameotti, R., Cova, M., Continella, A., van Steen, M., Peter, A., Kruegel, C., and Vigna, G. (2022). Deepcase: Semi-supervised contextual analysis of security events. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 522–539. IEEE.

Wang, T., Zhang, C., Lu, Z., Du, D., and Han, Y. (2019). Identifying truly suspicious events and false alarms based on alert graph. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5929–5936. IEEE.

Woolson, R. F. (2005). Wilcoxon signed-rank test. *Encyclopedia of Biostatistics*, 8.

Zolanvari, M., Yang, Z., Khan, K., Jain, R., and Meskin, N. (2021). Trust xai: Model-agnostic explanations for ai with a case study on iiot security. *IEEE internet of things journal*, 10(4):2967–2978.

Appendix 6

VI

R. Kalakoti, S. Nõmm, and H. Bahsi. Explainable transformer-based intrusion detection in internet of medical things (iomt) networks. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 1164–1169. IEEE, 2024

Explainable Transformer-based Intrusion Detection in Internet of Medical Things (IoMT) Networks

Rajesh Kalakoti[1], Sven Nõmm[2] and Hayretdin Bahsi[3]
[1,2]School of Information Technology, TalTech, Tallinn, Estonia.
[3]School of Informatics, Computing and Cyber Systems, Northern Arizona University, United States
Email: [1]rajesh.kalakoti@taltech.ee, [2]sven.nomm@taltech.ee, [3]hayretdin.bahsi@nau.edu

***Abstract*—Internet of Medical Things (IoMT) systems have brought transformative benefits to patient monitoring and remote diagnosis in healthcare. However, these systems are prone to various cyber attacks that have a high impact on security and privacy. Detecting such attacks is crucial for implementing timely and effective countermeasures. Machine learning methods have been applied for intrusion detection tasks in various networks, but explaining the reasons for detection decisions remains an obstacle for security analysts. In this paper, we demonstrate that Transformer architecture, the core of the recent revolutionary large language models, constitutes a promising solution for intrusion detection in IoMT networks. We utilized a comprehensive dataset, CICIoMT2024, recently released specifically for these networks. We created a binary classification model for discriminating attacks from benign traffic and a multi-class model for the identification of specific attack types. We applied Explainable AI (XAi) methods such as LIME and SHAP to generate post-hoc explanations for the model decisions. We evaluated and compared the quality of explanations based on three metrics: faithfulness, sensitivity, and complexity. Our findings demonstrate that the applied XAI methods enhance transparency in the predictions of Transformer-based intrusion detection models for IoMT networks, proving that both transparency and high performance can be achieved simultaneously.**

***Index Terms*—Transformer, Evaluation of Explainable AI Intrusion detection, IoT, Health Care IoMT Intrusion detection**

I. Introduction

Internet of Medical Things (IoMT) has advanced patient monitoring but also introduced security vulnerabilities that attackers can exploit, posing risks to patient privacy and safety [1]. DDoS and DoS attacks on IoMT devices could disrupt critical medical equipment, endangering patient lives [2]. IoMT devices are prime targets for attackers due to the sensitive medical data they collect. Breaches could expose personal patient information [2], violating privacy and enabling further malicious activities. Attackers could exploit zero-day vulnerabilities in medical devices to gain unauthorized access and control, even manipulating device functionality to physically harm patients. Researchers have proposed security assessment frameworks for IoMT to evaluate healthcare system security [1]. Machine learning and deep learning intrusion detection systems are being explored to monitor network traffic and device behavior for anomalies [3]. Explainable Artificial Intelligence (XAI) is crucial for detecting attacks on healthcare IoMT networks, offering transparency for stakeholders to understand the rationale behind machine learning model predictions [4–7].

This paper proposes an explainable transformer-based model for detecting and categorizing network attacks in Internet of Medical Things (IoMT) networks. We use transformer models inspired by GPT architecture to enhance IDS within IoMT environments. Our study includes implementing a sliding window approach to adapt transformer models for analyzing network traffic as sequences, improving detection accuracy. We also aim to enhance model explainability by integrating XAI methods such as LIME and SHAP. We assessed the effectiveness of XAI methods in explaining predictions made by transformer-based IoMT attack detection models. Using the CICIoMT2024 dataset, we compared transformer models with traditional ML models and demonstrated their effectiveness in binary and multi-class classification tasks within the IoMT security domain. Our study is the first to provide explanations and conduct a detailed evaluation of Explainable ai for Intrusion detection of Transformer in healthcare.

II. Literature review

Several works have implemented the power of transformer architectures in various IDS contexts. Wu et al. proposed that RTIDS balances dimensionality reduction and feature retention in imbalanced datasets using positional embedding and a stacked encoder-decoder neural network [8]. It uses a self-attention mechanism for classifying network traffic types and achieved F1 scores of 99.17% and 98.48% on the CICIDS2017 and CIC-DDoS2019 datasets, respectively. Nguyen et al. developed a multi-class Intrusion Detection System (IDS) for in-vehicle CAN bus using a transformer-based attention network (TAN) with self-attention to classify and detect replay attacks [9]. It identifies intrusion messages without the need for message labeling and uses transfer learning to improve performance with small data from other car models. The paper [10] introduces the FlowTransformer framework, which offers a method for implementing NIDS using transformer models to capture long-term network patterns and allows for easy substitution of transformer components. The models were tested using different transformer architectures (GPT 2.0, BERT) on three NIDS datasets. It was found that the classification head type significantly affects model performance, and specific input encoding and classification head choices can reduce the model size by over 50%. In their paper [11], the authors introduce GTID, an ID model based on n-gram frequency and a time-aware transformer. GTID learns traffic

1946-0759/24/$31.00 ©2024 IEEE
DOI 10.1109/ICMLA61862.2024.00179

features hierarchically from packet-level and session-level data for XAI-Based IDS. Previous research has made strides in using transformer models for ID. In our study, we introduce a transformer model for IoMT networks to address security concerns in healthcare. We use the 2024 CIC IoMT dataset to keep up with emerging threats and integrate XAI methods to provide explanations for predictions and evaluate their quality.

III. Methodology

A. CICIoMT2024 Dataset Description

CICIoMT2024 dataset [12] was created at the Canadian Institute for Cybersecurity to simulate realistic IoMT environments and collect network traffic for cybersecurity research. It includes IoT devices connected via Wi-Fi, MQTT, and Bluetooth Low Energy (BLE). The setup included 25 real devices and 15 simulated devices, generating various network traffic, including data from healthcare devices and cameras connected via Wi-Fi. Attacks executed included ARP spoofing, DoS, DDoS, Port Scans, vulnerability scans, MQTT Connect Flood, MQTT Publish Floods, MQTT Malformed Data attack, and disruptions to BLE devices. In this work, we have excluded the network traffic from BLE-based 7 Bluetooth devices for our experiment since the creators of the dataset have not created feature extraction for the BLE protocol yet. 44 relevant features were extracted from network traffic PCAP files for each network flow. Description of features can be found in Table. I.

In this study, we employ a transformer model for two classification tasks in IoMT networks: 1. IoMT Attack Detection (Binary Classification): Distinguishing between benign and attack traffic. 2. IoMT Attack Category Detection (Multiclass Classification): Categorizing specific attack types. IoMT Attack category detection involves identifying six categories of network traffic: benign, MQTT attacks, DDoS, DoS, Reconnaissance, and ARP Spoofing attacks. For our work, we used 10,000 samples from each label. The dataset was split into an 80:20 ratio for training and testing purposes.

B. Transformer Model for IoMT attack Detection

Transformer is a sequence transduction model with an encoder that maps input sequence $x = (x_1, \ldots, x_n)$ to representations $z = (z_1, \ldots, z_n)$, and a decoder that maps z to output sequence $y = (y_1, \ldots, y_m)$. The attention function computes a weighted sum of values based on similarity scores between a query q and keys K: $\text{Attention}(q, K, V) = \sum_i \text{Score}(q, k_i) v_i$ [13]. Vaswani et al.[13] proposed "Scaled Dot-Product Attention" where queries and keys are d_k-dimensional and values are d_v-dimensional: $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$. Multiple attention heads are used, and self-attention captures dependencies between sequence elements without relying on recurrent architectures. According to Vaswani et al. [13], both the encoder and decoder consist of stacked layers. Each encoder layer uses self-attention with the input sequence as queries, keys, and values, followed by a feed-forward network. The decoder incorporates masked self-attention to ensure each position attends only to previous ones, along with encoder-decoder attention using outputs from the encoder and a feed-forward network.

Table I: Features Description of CICIoMT2024 dataset

#	Feature Description
1	Header Length - Length of the packet header
2	Duration - Lifetime of the packet in transit
3	Rate - Speed of packet transmission within a flow
4	Srate - Transmission speed of outgoing packets in a flow
5	Fin flag number - Value of the Fin flag in TCP/IP
6	Syn flag number - Value of the Syn flag in TCP/IP
7	Rst flag number - Value of the Rst flag in TCP/IP
8	Psh flag number - Value of the Psh flag in TCP/IP
9	Ack flag number - Value of the Ack flag in TCP/IP
10	Ece flag number - Value of the Ece flag in TCP/IP
11	Cwr flag number - Value of the Cwr flag in TCP/IP
12	Syn count - Tally of Syn flag occurrences in a flow
13	Ack count - Tally of Ack flag occurrences in a flow
14	Fin count - Tally of Fin flag occurrences in a flow
15	Rst count - Tally of Rst flag occurrences in a flow
16	IGMP - Denotes the use of IGMP in application layer protocols
17	HTTPS - Denotes the use of HTTPS in application layer protocols
18	HTTP - Denotes the use of HTTP in application layer protocols
19	Telnet - Denotes the use of Telnet in application layer protocols
20	DNS - Denotes the use of DNS in application layer protocols
21	SMTP - Denotes the use of SMTP in application layer protocols
22	SSH - Denotes the use of SSH in application layer protocols
23	IRC - Denotes the use of IRC in application layer protocols
24	TCP - Usage of TCP in the transport layer protocol
25	UDP - Usage of UDP in the transport layer protocol
26	DHCP - Presence of DHCP in the application layer protocol
27	ARP - Usage of ARP in the link layer protocol
28	ICMP - Usage of ICMP in the network layer protocol
29	IPv - Usage of IP in the network layer protocol
30	LLC - Usage of LLC in the link layer protocol
31	Tot sum - Total packet length within a flow
32	Min - Shortest packet length in a flow
33	Max - Longest packet length in a flow
34	AVG - Mean packet length in a flow
35	Std - Variability in packet length within a flow
36	Tot size - Length of the packet
37	IAT - Interval between the current and previous packet
38	Number - Total number of packets in the flow
39	Radius - Root mean square of the variances of incoming and outgoing packet lengths in the flow
40	Magnitude - Root mean square of the averages of incoming and outgoing packet lengths in the flow
41	Variance - Ratio of the variances of incoming to outgoing packet lengths in the flow
42	Covariance - Covariance between the lengths of incoming and outgoing packets
43	Weight - Product of the number of incoming and outgoing packets
44	Protocol Type - Type of protocol used (IP, UDP, TCP, etc.) expressed in integer values

We employed a sliding-window approach to network monitoring, followed by the work of Marino et al. [14]. In this approach, the extracted features from consecutive network flows are grouped into windows of a fixed length L_w. These windows are then used as input sequences to our transformer model. Each feature within the flows undergoes normalization before being processed by the model. Given a window $W = \{x_1, \ldots, x_{L_w}\}$ of grouped IoMT network flows as input to the encoder, a shifted version W_{shifted}, removing the last L_s flows, is fed to the decoder. The decoder reconstructs the

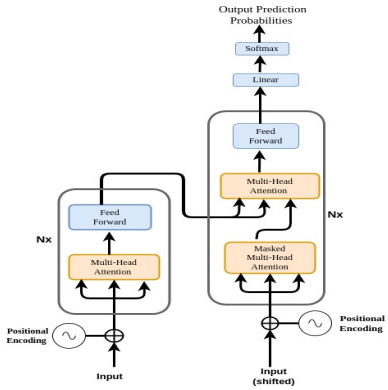


Figure 1: Transformer Architecture for IoMT attack detection

original window of length L_w from its shifted version of length $L_w - L_s$. We use Stochastic Gradient Descent (SGD) and cross-entropy as the loss function, defined as: $\text{Loss}(W) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$, where C is the number of classes, y_i is the true label (one-hot encoded), and $\hat{y}_i$ is the predicted probability. For binary classification ($C = 2$), the classes are Benign and Attack; for multi-class classification ($C = 6$), the categories are Benign, MQTT, DDoS, DoS, Recon, and ARP Spoofing. A score is assigned to each flow x_k based on the loss function computed over the window $W_x = (x_{k-L_w}, \ldots, x_k)$, with $\text{Score}(x) = \text{Loss}(W_x)$. Transformer model was fine-tuned for IoMT networks by adjusting hyperparameters such as the number of layers (N), window length (L_w), shift length (L_s), attention heads (H), epochs (E), and batch size (B). This flexibility optimizes the model for different configurations. In our experiments, we set L_s to one unit for next-flow prediction and tested L_w between 50 and 100 flows. A single attention head was used, as our feature space (44 attributes per flow) is smaller than typical NLP tasks. RELU activation was applied.

Transformer model was evaluated for IoMT attack detection using confusion matrices for both binary and multi-class classification. For binary classification, true positives (correctly classified attacks), true negatives (correctly classified benign traffic), false negatives (misclassified attacks), and false positives (misclassified benign traffic) were recorded. Evaluation metrics included Accuracy $(TP+TN)/(TP+TN+FP+FN)$, Precision $TP/(TP+FP)$, Recall $TP/(TP+FN)$, and F1-Score $2\times(P\times R)/(P+R)$. A weighted average was used for multi-class classification.

C. Explaining Transformer for IoMT attack detection

We can explain AI models using two approaches: model-agnostic and model-specific. There are also two types of explanations: local, which interpret individual predictions, and global, which offer an overview of the model's behavior. Our goal is to improve the explainability of Transformer-based IoMT attack detection. We used LIME and SHAP for local explanations of individual instances of the Transformer model, and SHAP for global explanations. Let $x \in \mathbb{R}^d$ be the input, where d is the feature set dimensionality. The black box model $\mathcal{M}$ maps input to output $\mathcal{M}(x) \in \mathcal{Y}$. Dataset $\mathcal{D} = (x^i, y^i)$ contains all input-output pairs. The explanation mapping $\mathbf{g}$ for predictor $\mathcal{M}$ and point x returns importance scores $\mathbf{g}(\mathcal{M}, x) = \phi_x \in \mathbb{R}^d$ for all features. Let $D : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}\geq 0$ be a metric in the explanation space and $S : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}\geq 0$ a metric in the input space. The evaluation criterion μ maps predictor $\mathcal{M}$, explainer $\mathbf{g}$, and point x to a scalar.

1) SHAP: SHAP [15] (SHapley Additive exPlanations) method utilizes Shapley values from game theory to determine the importance of each feature in a model's prediction, providing a comprehensive measure of feature significance. SHAP, based on Shapley values, is defined as follows: $\mathbf{g}(\mathcal{M}, x) = \phi_0 + \sum_{j=1}^{M} \phi_j$, where ϕ_j represents the feature attribution of feature j. In this study, SHAP's DeepExplainer was utilized.

2) LIME: LIME [16] (Local Interpretable Model-agnostic Explanations) creates locally interpretable models near specific predictions by perturbing the input and fitting a simple model (e.g., linear) to approximate the behaviour of the black-box model. LIME minimizes: $\underset{g\in G}{\text{argmin}} L(\mathcal{M}, g, \pi_x) + \Omega(g)$, where g is an interpretable model in the neighborhood of x.

D. Evaluating Explainable AI

Evaluating Explainable AI methods is necessary to ensure that their explanations are transparent. We employ three key metrics to evaluate the quality of our explanations for Transformer model: Faithfulness Correlation, Low complexity, Max sensitivity

1) High Faithfulness: : The explanation algorithm $\mathbf{g}$ should replicate the model's behavior, i.e., $\mathbf{g}(\mathcal{M}, \mathbf{x}) \approx \mathcal{M}(\mathbf{x})$. Faithfulness quantifies to what extent the explanation $\mathbf{g}$ follows the predictive behavior of the model $\mathcal{M}$. For evaluating the faithfulness of explanations, the faithfulness correlation [17] is used. Faithfulness measures how well the explanation function $\mathbf{g}$ aligns feature importance scores with the black-box model $\mathcal{M}$. The correlation metric is defined as $\mu_F(\mathcal{M}, g; x) = \underset{\mathcal{B}\in\binom{|d|}{|\mathcal{B}|}}{\text{corr}} \left(\sum_{i\in\mathcal{B}} g(\mathcal{M}, x)_i, \mathcal{M}(x) - \mathcal{M}(x_{\mathcal{B}})\right)$, where $x_{\mathcal{B}} = x_i \,|\, i \in \mathcal{B}$. The high faithfulness correlation metric iteratively substitutes a random subset of given attributions with a baseline value $\mathcal{B}$, then measures the correlation between the sum of these attributions and the difference in the model's output.

2) Max Sensitivity: Robustness of explanations represents similar inputs should result in similar explanations. $\mathbf{g}(\mathcal{M}, \mathbf{x}) \approx \mathbf{g}(\mathcal{M}, \mathbf{x} + \epsilon)$ for small ϵ. Max sensitivity was employed [17]. To ensure that nearby inputs, with model outputs, have the same explanations, it is important for the explanation function g to show low sensitivity in the vicinity of the point of interest x, assuming that the predictor function $\mathcal{M}$ is differentiable. The maximum sensitivity of an explanation function g at a point of interest x in its neighborhood can be defined as follows. Lets consider a neighborhood N_r

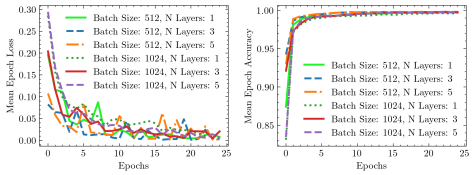
consisting of points within a radius r, from x denoted as $N_r = z \in D_x | p(x,z) \leq r, \mathcal{M}(x) = \mathcal{M}(z)$, where D represents the distance metric and p is the proximity function. With a predictor $\mathcal{M}(x)$, a distance D a proximity function p, a radius r and a point x we can define the sensitivity of function g at point x as follows: $\mu_M(\mathcal{M}(x), g, r; x) = \max_{z \in N_r} D(g(\mathcal{M}(x), x), g(\mathcal{M}(x), z))$

3) Low Complexity: Complexity explanations using a smaller number of features are preferred. A low complexity metric was employed in this work. Low complexity [17] metric computes the entropy of each feature's fractional contribution to the total attribution magnitude individually, $\mu_C(\mathcal{M}, g; x) = -\sum_{i=1}^{d} P_g(i) \log P_g(i)$, where $P_g(i) = \frac{|g(\mathcal{M},x)_i|}{\sum_{j \in d} |g(\mathcal{M},x)_j|}$ and $P_g = \{P_g(1), \ldots, P_g(d)\}$.

IV. Results& Discussions

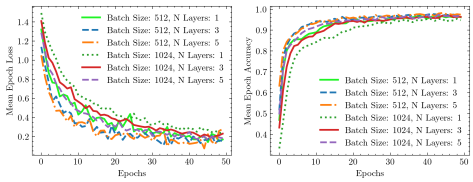
We propose an explainable transformer model for detecting and categorizing attacks in IoMT healthcare networks. Fig. 2 & Fig. 3 display the average training loss and accuracies of different transformer model configurations trained on CIC IoMT dataset 2024 for IoMT attack (binary) and IoMT attack-category (multi-class classification) detection. This evaluation involved testing various configurations of the transformer model to optimize its performance. Model configuration and training utilize a Window length ($L_w = 100$) for binary and multi-class classification types. Batch sizes ($B = [512, 1024]$). Number of encoder/decode (E/D) layers ($N = [1, 3, 5]$). Training epochs $E = 25$ for IoMT attack detection (binary classification) and epochs $E = 50$ for IoMT attack categorization (multi-class classification). Our experiments revealed that with a window length of $L_w = 100$, models required only one layer to fit the training data adequately. However, increasing the number of layers provided the model with improved fitting power, particularly for the more complex multi-class classification task. Models with 3 and 5 layers generally achieve lower loss values and higher accuracy for both attack-type and attack category-type. Model evaluation of testing data used $N = 3$ layers with $B = 512$ batch size. Fig. 2 shows the mean training loss and accuracy for IoMT attack detection (binary classification). All model configurations converge quickly in IoMT attack detection (binary), with significant loss reduction within 5 epochs. All models achieve >99% accuracy within 10 epochs, demonstrating the effectiveness of the transformer architecture for binary classification task. Classification report in Fig. 4a shows the model's exceptional performance, achieving over 99% accuracy. The model demonstrates perfect precision for benign traffic and almost perfect recall for attack detection, indicating a high ability to identify normal and malicious network activities.

In multi-class classification, Fig. 3 show that models with 3 and 5 layers consistently achieve lower loss values (see in Fig. 3a) and reach higher accuracy (see in Fig. 3b) more quickly. However, the single-layer model with a batch size of 1024 shows slower improvement, suggesting potential underfitting for this complex task. Fig. 4b shows classification report



(a) Loss (b) Accuracy.

Figure 2: Training loss and accuracy for Binary classification



(a) Loss (b) Accuracy.

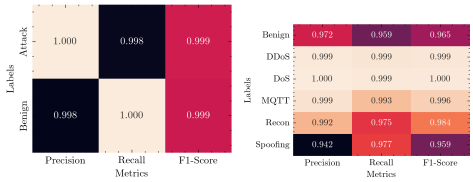
Figure 3: Training loss and accuracy for Multiclass

indicate strong performance across all attack categories, with an overall 97% accuracy for the test data. Model performed excellently in detecting DDoS and DoS attacks (F1-scores of 0.999 and 1.000, respectively), but it shows slightly lower performance for benign traffic and spoofing attacks (F1-scores of 0.965 and 0.959). While the model is highly effective overall, there's still a need to improve in distinguishing benign traffic and certain types of attacks, particularly spoofing. Table II compares our proposed transformer model with Decision Tree (DT) and k-nearest Neighbors (KNN) algorithms. Transformer model demonstrates superior performance with an accuracy better than DT and KNN for both IoMT attack and attack-category detection.

Table II: Transformer Model Comparison with KNN and DT

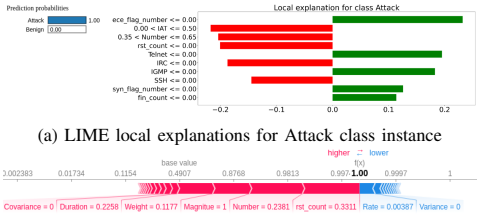
Accuracy of Transformer Comparision			
Classification type/model	DT	KNN	Transformer
IoMT attack detection (binary)	0.9955	0.99621	0.99847
IoMT Attack-category (Multi)	0.96167	0.88967	0.97426

In this work, we employed LIME and SHAP to explain predictions made by the Transformer model. We have selected a single instance of test data from each class to showcase the local explanations provided by these XAI methods for both binary and multi-classification types. The LIME method was used to explain selected instances in the testing data to get the probability for each class and explain why the probabilities were assigned to each class. Fig. 5a shows LIME explanations (first ten features) for "Attack" class of instance, left side of the figure shows the probability of each class, and the right side shows influential features for this class prediction. Model



(a) Binary Classification (b) Multiclass Classification

Figure 4: Classification Report for Binary class and Multiclass



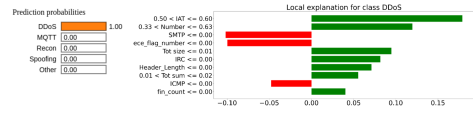
(a) LIME local explanations for Attack class instance

(b) SHAP local explanations for Attack class instance

Figure 5: Local explanations for binary classification

predicted an “Attack” class label with 100% accuracy. Right side of the bar chart shows the features which help to predict the instance as “Attack” class label, feature ‘Ece flag number’, ‘Telnet’, ‘IGMP’, ‘Syn flg number’, ‘fin count’ (shown with green bar). On the other hand, to predict an instance as not attack class (in red colour), the features ‘IAT’, ‘Number’, ‘Rst count’, ‘IRC’, and ‘SSH’ influence negatively. Similarly, we have provided local explanations for LIME (Fig. 6a) and SHAP (Fig. 6b) for the selected instances in IoMT attack-category detection (multi-class classification). The presented results are only for “DDoS” class label for LIME. In Fig. 6a, the model predicted the DDoS class label with 100% accuracy; features in the green bar show the most influential features in predicting the ’DDoS’ class. On the other hand, features in the green bar were the most influential in predicting not DDoS class.

SHAP is used to explain models and understand how the features are related to the predictions. SHAP provides the local and global explanation. In local explanation, we select a particular instance from test data points and explain the model prediction, showing each feature’s contribution to the prediction of the selected instance. SHAP calculates Shapley values, which shows the impact of features on the model predictions. Fig. 5b shows the local explanation’s force plot of the “Attack” class label instance, showing each feature’s positive contribution to predicting the “Attack” class label in red bar strip (‘Rst count’, ‘Number’, ‘Magnitude’, ‘Weight’, ‘Duration’, ‘Covariance’). ‘Rate’ and ‘Variance’ features in the blue strip contribute negatively to predicting the “Attack” class label. Plot’s base value, 0.4907, is the average of all prediction



(a) DDoS attack instance LIME local explanations



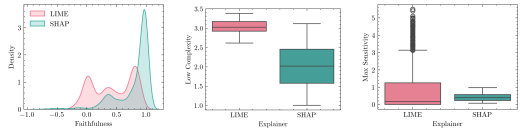
(b) SHAP local explanations for DDoS attack instance

Figure 6: local explanations for Multi-class classification

values. Each strip in the plot shows the impact of features in pushing the predicted value closer or farther from the base value. The total positive contribution is more significant than the negative contribution, and the final predicted value is greater than the base value, so the predicted class label is “Attack”. Similarly, Fig. 6b shows SHAP explanations for class label “DDoS” for Multi-class classification in Fig 6b shows the SHAP force plot that ‘Number’, ‘IAT’, ‘Magnitude’, and other features (red strip) positively push the model output from the based value of 0.35 to the final prediction (100%). ‘Header length’ feature (blue strip) negatively impacts the model prediction.

Table III: Quality Evaluation of local explanations results

Metrics	Explainer	Binary Classification $\mu \pm \sigma$	Multi-class $\mu \pm \sigma$
High Faithfulness	LIME	**0.78 ± 0.2**	0.44 ± 0.34
	SHAP	0.04 ± 0.74	**0.79 ± 0.3**
Max Sensitivity	LIME	**0.63 ± 0.18**	1.08 ± 1.6
	SHAP	0.88 ± 0.93	**0.41 ± 0.2**
Low Complexity	LIME	3.00 ± 0.11	3.04 ± 0.15
	SHAP	**2.51 ± 0.18**	**2.00 ± 0.5**

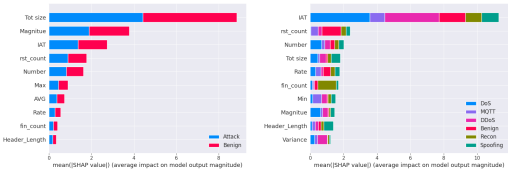


(a) High Faithfulness (b) Low Complexity (c) Max Sensitivity

Figure 7: Distribution of Quality Evaluation Metrics for LIME and SHAP in Multi classification

We evaluated the quality of explanations provided by LIME and SHAP XAI methods for the Transformer model across 2000 data points for both binary and multi-class classification using three metrics: High faithfulness, Max sensitivity, and Low complexity. The faithfulness of XAI methods was evaluated by analyzing the correlation between attribute importance assigned by the XAI method and its impact on Transformer model probabilities. Model probabilities were obtained by the Softmax activation function. A high faithfulness correlation

value suggests that the explanations effectively capture the model's behaviour and can be regarded as faithful. To evaluate the complexity of local explanations, we calculate the entropy of feature attribution in the explanations. The sensitivity metric evaluates the consistency of the explainer's output, ensuring that similar inputs in the feature space have explanations when sensitivity is low. For this metric, we used distance with a radius value of 0.1 to find the nearest neighbor points related to the prediction label of an explanation, which helps to identify data points to a given point in the feature space with similar explanations for the predicted label. Table III shows the results of quality explanation for local explanations of LIME and SHAP. Mean (μ) and standard deviation (σ) values were calculated for the test data of XAI computed metrics. For IomT attack detection, considering the high faithfulness metric, LIME performs better than SHAP, scoring 0.78 ± 0.2 compared to SHAP's result of 0.04 ± 0.74. This indicates that LIME explanations align closely with the model's behavior in predicting output. LIME demonstrates lower sensitivity (0.63 ± 0.18) than SHAP (0.88 ± 0.93), suggesting that LIME explanations are more reliable for inputs. Both methods have explanations, with SHAP (2.51 ± 0.18) being slightly less complex than LIME (3.00 ± 0.11). For IoMT Attack-category Detection (Multi-class), SHAP has achieved High faithfulness, Lower complexity, and Lower sensitivity. Fig. 7 shows the distribution of XAI metric evaluations for test points in IoMT attack category detection. SHAP explanations are generally reliable because the mean values of SHAP explanations of evaluation metric values tend to fall more on the positive side of correlation for high faithfulness and on the lower side for low complexity and max sensitivity metric.



(a) Binary Classification (b) Multiclass classification

Figure 8: SHAP Global explanations for Transformer Model

We provided a global explanation using a SHAP summary plot (Fig. 8) to highlight feature importance in the transformer model. This plot shows how each feature contributes to the model's output, averaged over the test data for the top 10 features. Fig. 8a illustrates the global feature importance for IoMT attack detection, while Fig. 8b presents the importance for attack category detection. Key features such as 'IAT', 'Number', 'Rst count', 'Magnitude', and 'Tot size' were highly influential in both cases.

V. Conclusion

This study proves the effectiveness of transformer-based models in detecting and categorizing attacks in Internet of Medical Things (IoMT) networks. Our transformer model achieved 99.85% accuracy in binary classification and 97.43% in multi-class classification on the CICIoMT2024 dataset, outperforming traditional ML approaches. This work uses LIME and SHAP as explainable AI methods to provide transparency for the transformer model's decisions. Our quantitative evaluation of local explanations of XAI methods revealed that LIME is suitable for explaining individual predictions in binary classification. At the same time, SHAP effectively explains IoMT attack types in multi-class classification. Global feature importance analysis highlighted key network traffic characteristics crucial in identifying various types of IoMT attacks in the healthcare domain.

References

[1] F. Alsubaei, A. Abuhussein, V. Shandilya, and S. Shiva, "Iomt-saf: Internet of medical things security assessment framework," *Internet of Things*, vol. 8, p. 100123, 2019.

[2] A. Chacko and T. Hayajneh, "Security and privacy issues with iot in healthcare," *EAI Endorsed Transactions on Pervasive Health and Technology*, vol. 4, no. 14, 2018.

[3] R. Kalakoti, S. Nõmm, and H. Bahsi, "In-depth feature selection for the statistical machine learning-based botnet detection in iot networks," *IEEE Access*, vol. 10, pp. 94 518–94 535, 2022.

[4] R. Kalakoti, H. Bahsi, and S. Nõmm, "Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection," *IEEE Internet of Things Journal*, 2024.

[5] R. Kalakoti, S. Nõmm, and H. Bahsi, "Enhancing iot botnet attack detection in socs with an explainable active learning framework," in *2024 IEEE World AI IoT Congress (AIIoT)*. IEEE, 2024, pp. 265–272.

[6] R. Kalakoti, H. Bahsi, and S. Nõmm, "Explainable federated learning for botnet detection in iot networks," in *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2024, pp. 01–08.

[7] R. Kalakoti, S. Nõmm, and H. Bahsi, "Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai)," in *2023 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2023.

[8] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "Rtids: A robust transformer-based approach for intrusion detection system," *IEEE Access*, 2022.

[9] T. P. Nguyen, H. Nam, and D. Kim, "Transformer-based attention network for in-vehicle intrusion detection," *IEEE Access*, 2023.

[10] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "Flowtransformer: A transformer framework for flow-based network intrusion detection systems," *Expert Systems with Applications*, vol. 241, p. 122564, 2024.

[11] X. Han, S. Cui, S. Liu, C. Zhang, B. Jiang, and Z. Lu, "Network intrusion detection based on n-gram frequency and time-aware transformer," *Computers & Security*, vol. 128, p. 103171, 2023.

[12] S. Dadkhah, E. Carlos Pinto Neto, R. Ferreira, R. Chukwuka Molokwu, S. Sadeghi, and A. Ghorbani, "Ciciomt2024: Attack vectors in healthcare devices-a multi-protocol dataset for assessing iomt device security," *Raphael and Chukwuka Molokwu, Reginald and Sadeghi, Somayeh and Ghorbani, Ali, CiCIoMT2024: Attack Vectors in Healthcare Devices-A Multi-Protocol Dataset for Assessing IoMT Device Security*, 2024.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[14] D. L. Marino, C. S. Wickramasinghe, C. Rieger, and M. Manic, "Self-supervised and interpretable anomaly detection using network transformers," *arXiv preprint arXiv:2202.12997*, 2022.

[15] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, 2017.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[17] U. Bhatt, A. Weller, and J. M. Moura, "Evaluating and aggregating feature-based model explanations," *arXiv:2005.00631*, 2020.

Appendix 7

VII

R. Kalakoti, S. Nõmm, and H. Bahsi. Enhancing iot botnet attack detection in socs with an explainable active learning framework. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 265–272. IEEE, 2024

Enhancing IoT Botnet Attack Detection in SOCs with an Explainable Active Learning Framework

Rajesh Kalakoti, Sven Nõmm
Department of Software Science
School of Information Technology,
TalTech, Tallinn, Estonia.
rajesh.kalakoti@taltech.ee, sven.nomm@taltech.ee

Hayretdin Bahsi
Department of Software Science
School of Information Technology, TalTech, Estonia &
School of Informatics, Computing and Cyber Systems
Northern Arizona University, United States
hayretdin.bahsi@nau.edu

2024 IEEE World AI IoT Congress (AIIoT) | 979-8-3503-8780-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/AIIoT61789.2024.10578957

***Abstract*—The widespread use of Internet of Things (IoT) devices has raised the threat of botnet attacks, presenting significant challenges for security operations centres (SOCs). While machine learning techniques have shown promising results in detecting these attacks, their effectiveness is often limited by the lack of labeled data and the need for greater transparency in the decision-making process of labeling. We propose an explainable active learning framework incorporating post-hoc explainability methods, such as LIME and SHAP, into the active learning process for detecting IoT botnet attacks in a multi-class classification setting. Our framework enables SOC analysts to provide informed annotations, while the explainability methods offer insights into the model's decision-making process. We employ uncertainty sampling and query-by-committee strategies to select the most informative instances for labeling, and we evaluate the quality of the explanations using various quantitative metrics. Experimental results demonstrate that our explainable active learning framework achieves high detection performance while enhancing the trust and transparency between the SOC analysts and the learning model.**

***Index Terms*—Active learning, SOC, IoT Botnet, Explainable AI, Post-Hoc explainability, LIME, SHAP**

I. Introduction

Increasing cyberattacks on IoT devices raise data protection concerns. DDoS attacks can disrupt IoT systems [1], and vulnerable devices can form botnets, posing security threats [2, 3]. Recent advances in machine learning (ML) and deeplearning (DL) have led to more accurate intrusion detection systems for IoT networks, outperforming traditional methods [4]. Training ML and DL models require high-quality labeled data, which is crucial for distinguishing normal behavior from threats in security operations. Active learning strategies, which select informative unlabeled data points for labeling, can significantly reduce manual annotation while improving model performance. These strategies perfectly fit cyber security tasks as annotation is a complex task due to the high costs of experts in this field. Integrating explainable AI (XAI) into active learning scenarios strengthens intrusion detection systems. XAI enhances the transparency of model predictions and helps experts understand the reasoning behind the predictions, which is crucial for trust in high-stakes environments. XAI in active learning bridges the model's decision process and the expert's knowledge, aiding in more informed decisions. Evaluating XAI methods ensures reliable and consistent explanations.

In this paper, we propose an active learning framework that integrates post-hoc explainability into its training cycles for IoT botnet attack detection in a multi-class classification setting. We employ local feature importance-based explanations, specifically LIME and SHAP, to enable analysts in the context of a security operation centre (SOC) to provide informed annotations and teach the learning model. Additionally, we incorporate a quantitative evaluation step for the explanations within this active learning loop using three metrics: (1) Faithfulness (how well the explanations match the actual behavior of the models), (2) Monotonicity (how similar the explanations are for similar inputs), and (3) Sensitivity (how well the explanations hold up under perturbations of the input). Our work aims to connect network forensic processes with the use of machine learning models in SOC environments. By giving SOC analysts access to XAI, our research enhances the explainability and trustworthiness of the ML models used in cybersecurity applications. This approach is well-suited for SOC environments, as network forensic specialists can assume the role of labelers in such settings. To the best of our knowledge, this is the first study that incorporates XAI into an active learning setting with a detailed evaluation of explanations in a network intrusion detection task.

Our paper is structured as follows: Section II covers active learning background, while Section II-A gives related work of XAI in intrusion detection systems. Section III outlines our methodology, followed by Section IV presenting results and discussions. Finally, conclusions are Section V.

II. Background work on Active learning

In the active learning method, a supervised classifier learns from a small set of labelled data called the seed dataset. The classifier selects points from an unlabeled data pool and asks a human expert to label them. After labelling, the expert adds these points to the training dataset to update it. The classifier is then retrained with the newly updated training data, and this cycle of selecting and annotating data points continues until the model reaches a threshold level of performance or stops improving. Additionally, to ensure the human expert's

979-8-3503-8780-3/24/$31.00 ©2024 IEEE

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:00 UTC from IEEE Xplore. Restrictions apply.

tasks remain manageable, the active learning process may be limited to expanding the training dataset to a specific size.

A widely used approach for choosing data points to label is known as uncertainty sampling, which selects data points one by one using a measure of how uncertain the model is [5]. Using this approach involves the classifier making label predictions for all the unlabeled data points and then selecting the one that it is most uncertain about, which means the point that the classifier has the lowest confidence in its prediction. There are various strategies to measure how uncertain a prediction is for a data point x with its predicted label y. The classification uncertainty score is one such strategy. This score is calculated as: $U(x) = 1 - P(\hat{y}|x)$ Subsequently, the data point with the highest score is selected based on this criterion. Classification margin is a score that calculates the difference between the probabilities of the most likely and the second most likely predictions. The smallest margin indicates the greatest uncertainty, so the sample with this smallest difference is selected. The classification margin score is defined as $M(x) = P(\hat{y_1}|x) - P(\hat{y}|x)$ Classification entropy (H) is a score that uses entropy to measure how uncertain a classification is. The data point selected has the highest entropy score, determined by the following equation: $H(x) = \sum_k p_k log(p_k)$, In this equation, p_k denotes the probability that a given unlabeled sample belongs to the k^{th} category, as determined by the current state of knowledge of the classifier.

Another employed approach for data point selection in AL is query-by-committee (QBC) [5]. In this approach, a set of different models, referred to as a committee, are all trained on the same labelled data but represent varying hypotheses within the hypothesis space. In this strategy, each model in the group votes on how to classify new examples. The most informative sample is the one with the most disagreement among the models about its class. Various methods can be used to measure the level of disagreement, including Vote Entropy (VE), Consensus Entropy (CE), and Maximum disagreement (ME). Vote entropy selects the query instance as the sample with the highest entropy in the vote distribution. It is calculated as: $VE = \text{argmax}_x - \sum_i \frac{V(y_i)}{C} \log\left(\frac{V(y_i)}{C}\right)$ Where y_i represents each possible class label and $V(y_i)$ is the count of votes that a label gets from the committee for a particular instance x, C represents the number of members in the committee. Consensus entropy works by first finding the average predicted probabilities of each class from all classifiers, known as consensus probability (P_{cs}). Next, it calculates the entropy of this consensus probability. The sample with the highest consensus entropy is then selected for labelling. The disagreement score is calculated as follows: $CE = -\sum_y P_{cs} \log(P_{cs})$. Where $P_{cs} = \frac{1}{C}\sum_{c=1}^{c} P(y_i)$. represents consensus probability. Finally, Maximum disagreement measures the level of disagreement by first creating a consensus probability, which is the average of the class probabilities predicted by each classifier. It then uses Kullback-Leibler divergence, rather than entropy, to measure how much each classifier's predictions differ from the consensus. The sample with the greatest divergence is selected as the query instance. The score of disagreement is computed as, $MD = \text{argmax}_x - \frac{1}{C}\sum_y D(P_{\theta^{(C)}}||P_{cs})$ where $D(P_{\theta^{(C)}}||P_{cs}) = \sum_y P(y_i|X;\theta^{(c)}) log \frac{P(y_i|x;\theta(C)}{P_{CS}}$ represents the corresponding Kullback–Leibler divergence. Here $\theta_{(C}$ denotes a specific classifier model within the committe and P_{CS} represents consensus probability.

A. Related Work on Explainable AI for Intrusion Detection

Several studies have investigated the application of explainable AI in the field of intrusion detection [6, 7]. For instance, Liu et al. proposed an explainable AI-based intrusion detection system that utilized a combination of rule-based reasoning and machine learning algorithms [8]. Muna et al. address the significant challenge of securing IoT infrastructures in smart cities against various cyber threats [9]. The study presents a novel approach that integrates machine learning, specifically Extreme Gradient Boosting (XGBoost), with XAI methods, including ELI5, LIME, and SHAP, to detect massive IoT attacks effectively and provide insights into the model's decision-making process. Performance evaluation using metrics confusion matrix, recall, precision, F1-score, and support on the IOTD20 [10] dataset exhibits that the model can efficiently classify several types of IoT attacks and contribute to reducing the cybersecurity threats in smart cities. ELI5 is employed for model debugging purposes, the SHAP summary plot is used for a global explanation purpose, and both the SHAP force plot and LIME are utilized to explain individual instances of IoT attack labels predicted by the model. The study [11] introduces an Improved Elman Neural Network (IENN) model, which is a refined version of ENN, to detect intrusions with high precision with the help of Glowworm Swarm Optimization (HEGSO) algorithm for feature selection purposes. IENN was used with the Enhanced Fruitfly Optimization (EFFO) algorithm for parameter optimization. LIME is used to explain the accurate classification of intrusions. The evaluation is based on the datasets CIC-IDS 2017 [12] and NSL KDD 2015 [13]. In the paper [14], a data engineering practice is employed wherein the Random Forest model is explained globally using SHAP feature importance. This methodology is applied to identify key features within the UNSW-NB15 dataset [15] for both binary and multiclass classification, aiming to effectively identify cyber threats with significant accuracy, precision, recall, and F1 score. SHAP summary plots revealed crucial features from the dataset, including Ct_state_ttl, Sttl, Dmean, and Dbytes. The review studies do not consider the explainability issues in active learning settings.

III. Methodology

In this work, we propose an explainable active learning (XAL) framework for IoT botnet detection see in Fig. 1. The framework combines AL with XAI methods to improve model performance while providing transparency and explainability. It begins with an ML model trained on a limited set of labeled samples (initial seed), which then selects informative instances

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:00 UTC from IEEE Xplore. Restrictions apply.

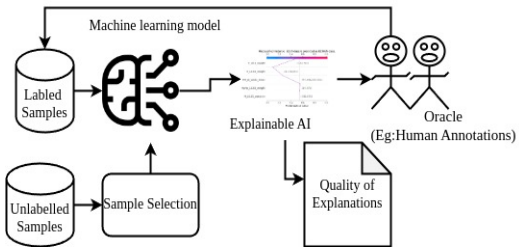


Fig. 1: Explainable Active Learning (XAL) Framework for IoT Botnet Attack Detection

from the unlabeled pool. The selected instances are passed to a post hoc XAI method to retrieve local explanations. After applying quantitative quality measurement metrics, the instances, their explanations and quality findings are presented to the human annotators who are security analysts in our context. Analysts review all the input and provide a final decision about the labels. These labeled instances are used to retrain and update the ML model. This iterative process enhances model performance in each cycle while improving human-machine collaboration by establishing trust.

A. Data Set

In this study, N-BaIoT dataset [16] was used, which contains 115 network traffic features extracted primarily through descriptive statistical values. A dataset is derived from network traffic generated by bots deployed in a testing environment with nine different IoT devices infected by Mirai and Gafgyt malware types. Notably, the features for each instance reflect aggregated descriptive statistics of raw network traffic across five-time windows ($100\mu s$, $500\mu s$, $1.5s$, $10s$, and 1_{min}), denoted as L5, L3, L1, L0.1, and L0.01, respectively (see Table-I). Dataset encompasses five main feature categories: host-IP (coded as H), host-MAC and IP (coded as MI), channel (coded as HH), socket (coded as HpHp), and network jitter (coded as HH_jit). For each category, metrics such as packet count, mean packet size, and variance in packet sizes are calculated. Additionally, statistical values, including Pearson correlation coefficient (PCC) of packet size, radius, covariance, and magnitude, are derived for network Channel and Socket, along with packet count, mean, and variance. This work mainly focuses on attack-type multiclass classification using the N-BaIoT dataset. Data points are classified into eight attack types and legitimate network traffic: ACK, Benign, COMBO, JUNK, SCAN, SYN, TCP, UDP, and PLAIN. A detailed description of these attack types can be found in Table II.

TABLE I: Details of Features of N-BaIoT dataset.

Feature Category	Category Code	No.Of features	Statistical Value Feature	Time Frame Windows
Hos Mac & IP	MI	15	Packet Count, Mean Variance	100 Micrso Sconds
Host IP	H	15		500 micro seconds
Network Jitter	HH_jit	15		1.5 Seconds
Channel	HH	35	Packet Count, Mean	10 Seconds
Socket	HpHp	35	Variance, Magnitude,Radius, CoVariance, Correlation	1 Minute

TABLE II: Botnet Attack types used in this Study for botnet attack detection

Class Name	Description
ACK	Gafgyt malware Sending Spam data
Beign	Legitimate Network Traffic
COMBO	Gafgyt malware Sending spam data and opening a connection
JUNK	Mirai Malware ACK-Flooding
SCAN	Scans The network devices for vulnerabilities,(Mirai & Gafgyt
SYN	Mirai Malware SYN-Flooding
TCP	Gafgyt malware TCP Flooding
UDP	UDP flooding (Mirai & Gafgyt)
UDPPLAIN	Mirai malware UDP flooding with Less of an option for higher packet per second

B. Feature Selection

In the feature selection phase, Pearson's correlation was employed as a technique (refer to Equation (1)). This method involves calculating a matrix for all the features to find which features are redundant. Pearson's correlation computes the linear correlation between pairs of variables. Pairwise correlations among features are analysed to identify feature redundancy. The $\mathcal{P}$-value of correlation coefficients is constrained within the range of -1 to 1. A perfect positive correlation is indicated when $\mathcal{P} = 1$ for two features. No correlation is observed if $\mathcal{P} = 0$, and a perfect negative correlation is acknowledged when $\mathcal{P} = -1$. The Pearson correlation formula is expressed as:

$$\mathcal{P} = \frac{\sum_{i=1}^{n}[(x_i - \mu_x)(y_i - \mu_y)]}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\sum_{i=1}^{n}(y_i - \mu_y)^2}} \quad (1)$$

Here, μ_x and μ_y represent the means of features x and y respectively. Larger absolute values of Pearson's correlation coefficient indicate a more linear relationship between the features. To rank the features, We have used Fisher score [17].

XGBoost (eXtreme Gradient Boosting) classifier was selected for all the benchmarking scenarios in AL process mentioned in Section II. Random search parameter tuning was used to find the best hyperparameters for the initial training, and these hyperparameters were maintained throughout the entire benchmarking process of AL.

C. XAI methods

Our research employed two post hoc XAI methods for understanding IoT botnet attack predictions. These methods, known as SHAP and LIME, are widely acknowledged and extensively employed in the field of XAI. These methods play a crucial role in the active learning paradigm. Specifically, they provide human experts, serving as oracles, with explanations to understand the underlying rationale behind model predictions. This explainability becomes particularly valuable when employing sophisticated sampling strategies like uncertainty sampling and querying by the committee in active learning scenarios. It enables the oracle to furnish more accurate and informed annotations, and the feedback obtained helps improve the learning model and trust in the predictions.

For a dataset $\mathcal{D}$, where each input $x \in \mathbb{R}^d$, with d representing the dimensionality of the feature set, and a black box model $\mathcal{M}$ maps the input to an output $\mathcal{M}(x) \in \mathcal{Y}$, where

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:00 UTC from IEEE Xplore. Restrictions apply.

$\mathcal{Y}$ denotes the output space. Let $\mathcal{D} = \{(x^i, y^i)\}$ represent the collection of all input-output pairs in the dataset. A post hoc explanation, denoted as $\mathbf{g}$, is an explanation mapping that, for predictor $\mathcal{M}$ and point of interest x, returns an importance score $\mathbf{g}(\mathcal{M}, x) = \varphi_x \in \mathbb{R}^d$ for all features. Denote $D : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$ as a metric in the space of explanations and $S : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$ as a metric in the space of inputs. The evaluation criterion μ is a mapping that takes predictor $\mathcal{M}$, explainer $\mathbf{g}$, and the point of interest x as arguments and returns a scalar value for $\mathbf{g}$.

1) Shap: SHAP [18] is a game-theoretic explanation method based on Shapley values. SHAP is defined as $\mathbf{g}(\mathcal{M}, x) = \phi_0 + \sum_{j=1}^{M} \phi_j$, where ϕ_j is the feature attribution of feature j.

2) LIME: LIME [19] is an explanation method that approximates the model's behavior locally around a specific data instance x within the neighborhood $N(x)$ using a simpler interpretable model the prediction of the interpretable model around x is denoted as The approximation is obtained by minimizing the object function defined as $\underset{g \in G}{\operatorname{argmin}} L(\mathcal{M}, g, \pi_x) + \Omega(g)$

D. XAI Metrics

1) Faithfulness: The faithfulness metric $\mu_F(\mathcal{M}, g; x)$ measures how well the feature importance scores generated by the explanation function $\mathbf{g}$ reflect the actual importance of the features in the black-box model $\mathcal{M}$ for input x. This property is best computed using Pearson's correlation coefficient between the sum of the attributions of the features set to the best line value and the corresponding difference in the output values. Let $\mathcal{B}$ be the subset of indices whose features are set to a baseline value. Then the faithfulness metric is computed as follows.

$$\mu_F(\mathcal{M}, g; x) = \underset{\mathcal{B} \in \binom{|d|}{|\mathcal{B}|}}{\rho} \left(\sum_{i \in \mathcal{B}} g(\mathcal{M}, x)_i, \mathcal{M}(x) - \mathcal{M}(x_{\mathcal{B}}) \right)$$

where $x_{\mathcal{B}} = x_i | i \in \mathcal{B}\}$

2) Monotonicity: Let $x, x' \in \mathcal{R}^d$ be two input points such that $x_i \leq x'_i$ for all $i \in 1, 2, \ldots, d$. $\mathcal{M}$ and g are said to be monotonic if the following condition holds: For any subset $S \subseteq 1, 2, \ldots, d$, the sum of the attributions of the features in S should be nonnegative when moving from x to x', that is, $\sum_{i \in S} g(\mathcal{M}, x) i \leq \sum_{i \in S} g(\mathcal{M}, x')_i$ implies

$$\mathcal{M}(x) - \mathcal{M}(x_{[x_s = \bar{x}_s]}) \leq \mathcal{M}(x') - \mathcal{M}(x'[x'_s = \bar{x}_s])$$

3) Max Sensitivity: To ensure that nearby inputs with similar model output have similar explanations, it is desirable for the explanation function g to have a low sensitivity in the region surrounding the point of interest x, assuming the differentiability of the predictor function $\mathcal{M}$. The maximum sensitivity of an explanation function g at a point of interest x in its neighbourhood is defined as follows: Consider a neighbourhood N_r of points within a radius r of x, denoted by $N_r = z \in D_x | p(x, z) \leq r, \mathcal{M}(x) = \mathcal{M}(x)(z)$, where D is the distance metric, and p is the proximity function. Given a predictor $\mathcal{M}(x)$, a distance metric D, a proximity function p, a radius r, and a point x, we define the maximum sensitivity of g at x as follows: $\mu_M(\mathcal{M}(x), g, r; x) = \max_{z \in N_r} D(g(\mathcal{M}(x), x), g(\mathcal{M}(x), z))$

E. Experimental Setup

The N-BaioT dataset was divided into 80% for training and 20% for testing to benchmark different active learning methods. Active learning requires many unlabeled data points, known as a pool. In our research, this pool always contained 100,000 data points taken from our larger training dataset. The testing dataset was exclusively used to evaluate the learning with the F1-score of the model and was not involved in the training process at any stage of the active learning cycles. Strategies were adopted for selecting queries, as outlined in Section 2. This study utilizes the F_1 score metric to evaluate the model. The F_1 score, defined as the harmonic mean of precision (P) and recall (R) [20], offers a more suitable evaluation of misclassified instances compared to accuracy

$$F_1\text{-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The research performed ten iterations for each model. In these iterations, beyond evaluating the models' performance, the study also applied explainable AI techniques like LIME and SHAP to the test data. These explanations for each test data point were then evaluated using XAI metrics, and the mean values of these metrics were calculated. Consequently, the reported outcomes for the models' performance and the XAI metrics are based on the average results obtained across the ten iterations.

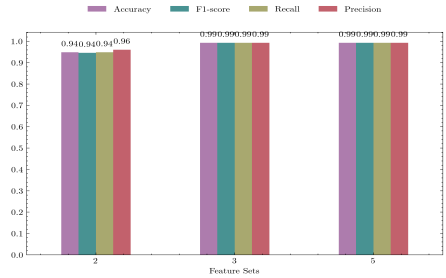


Fig. 2: Feature Sets Comparisons with metrics.

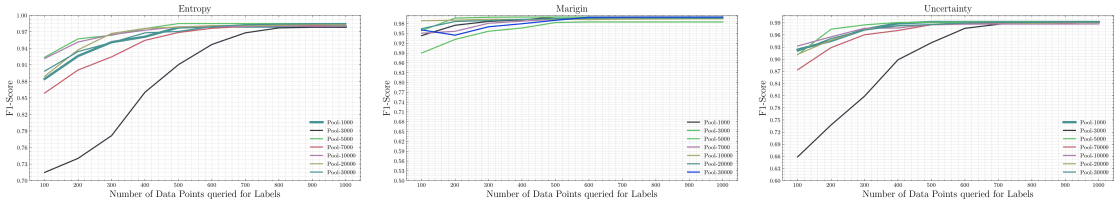
TABLE III: Top-5 Selected Features

Network Category	Feature
Host	H_L0.01_variance
	H_L0.1_weight
	H_L0.01_weight
Socket	HpHp_L0.01_weight
Networkjitter	HH_jit_L0.01_mean

IV. Results & Discussion

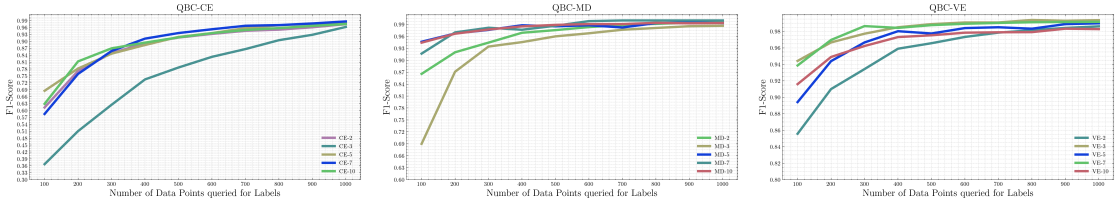
The N-BaIoT dataset contains a total of 115 features. We applied Pearson's linear correlation coefficient (r) to eliminate

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:00 UTC from IEEE Xplore. Restrictions apply.



(a) Classification Entropy (b) Classification Margin (c) Classification Uncertainty

Fig. 3: Uncertainty sampling: classification uncertainty results.



(a) Consensus entropy (b) Maximum disagreement (c) Vote entropy

Fig. 4: Query by committee results

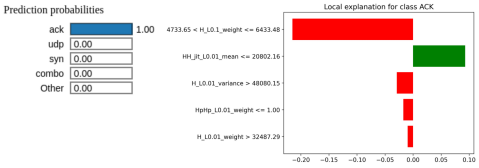
redundant and irrelevant data features. Features with a high correlation ($|r| > 0.80$) with any other feature were removed, leaving only one. Consequently, from the initial set of 115 features, only 33 essential features for describing each sample in the dataset remained. Fisher score was utilized to rank them. Various subsets were tested, including the top n features (i.e., 3, 5, 10, 15, 20). To test each feature set, we utilized an entire dataset sample of 100,000. For training and testing purposes, the dataset was split with an 80% to 20% ratio. Performance results for XGBoost (XGB model) are provided in Fig. 2. Top-3 and Top-5 features achieved the highest detection rates, including accuracy, F1-score, recall, and precision. Specifically, for this work, the top 5 features were selected and presented in Table III.

Fig. 3 shows the F1-score performance of active learning cycles using three uncertainty sampling strategies—classification entropy, classification margin, and classification uncertainty—for active learning in IoT botnet attack detection using XGB model. The experiment began with an initial set of 20 labelled data points, and various pool sizes (1000, 2000, 3000, 5000, 7000, 10000, 20000, and 30000) of unlabeled data points were utilized. The training sets were balanced in all cases. During the active learning-based training experiment, up to 1000 data points were chosen from the pool and presented to a human expert for labeling queries. Data point selection for querying was based on the classification uncertainty score from the three classification uncertainty strategies. Fig. 3a shows the F1-score performance of XGB model using Entropy sampling for various pool sizes (ranging from 1000 to 30000) across different numbers of data points queried for labels (ranging from 100 to 1000). The model achieves an F1-score of around 0.97 to 0.99 for most pool sizes when querying 1000 data points for labels, demonstrating high performance in detecting attack labels. For Margin Sampling (from Fig. 3b), similar to Entropy sampling, F1-score improves as more data points are queried for labels. The model achieves F1 scores ranging from 0.95 to 0.98 for most pool sizes when querying 1000 data points for labels. F1-score of Uncertainty in Fig. 3c is consistent with the previous two sampling techniques. F1 score improves as more data points are queried for labels, showcasing the effectiveness of active learning. The model achieves F1 scores ranging from 0.96 to 0.99 for most pool sizes when querying 1000 label data points. For all three strategies of classification uncertainty, as the number of data points queried for labels increases, F1 Score improves for all pool sizes, which indicates that Model performance increases with more labeled data. Larger pool sizes (e.g., 20000 and 30000) tend to achieve slightly higher F1 Scores compared to smaller pool sizes, especially when the number of queried data points is low (100 to 400).

The query-by-committee approach involves a committee of independent classifiers selecting data points from the unlabeled pool to label queries. Fig. 4 illustrates the performance of the QBC approach for active learning cycles using XGB model. The level of disagreement among the classifiers on the committee is quantified using different disagreement scores, i.e. consensus entropy score (QBC-CE) in Fig. 4a, vote entropy score (QBC-VE) in Fig. 4c, and maximum disagreement score (QBC-MD) in Fig. 4b. Experiments were performed with varying committee sizes (2, 3, 5, 7, and 10), and each committee member was an independent XGBoost classifier. A fixed pool of 10,000 data points was used, and the seed size was set to 100. High F1 scores of all three disagreement scores improve as the number of labeled data points increases. Across

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:00 UTC from IEEE Xplore. Restrictions apply.

different committee sizes, a committee size of 7 consistently outperforms others for all disagreement scores. As shown in Fig. 4b, QBC-MD with committee size 7 (QBC-MD-7) achieves the highest F1-scores among the three disagreement scores, showing the effectiveness of the QBC approach in actively selecting informative data points for labeling and showing improved model performance.



(a) LIME Explanations



(b) SHAP Local Explanations

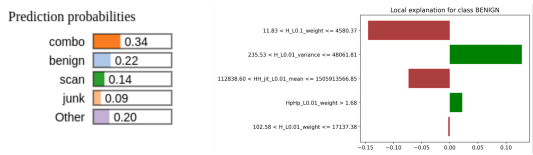
Fig. 5: LIME & SHAP Local Explanations for Correctly Classified Instance (ACK Attack)

In our work, we have employed and compared two post hoc explainability methods for understanding IoT botnet attack predictions in the active learning loop. The oracle, a security analyst, leverages these explanations to understand the model's decision-making process of predicting IoT botnet attacks by the XGB model. Integrating LIME and SHAP with active learning builds a foundation of trust and transparency between Oracle and the learning model. For a security analyst, understanding why a model made a specific prediction is essential, especially when deciding whether to trust the model's output and when providing precise labels or feedback for further model training. More specifically, posthoc methods, LIME and SHAP, explain the reasons for the model decisions by assigning an importance value to each feature.

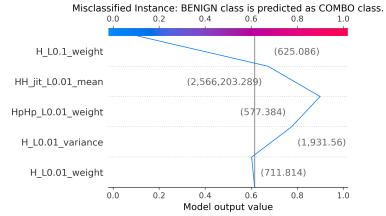
LIME and SHAP explanations for the model's features, particularly for a correctly classified sample as ACK attack type, are shown in Fig. 5. For Lime explanations (see Fig. 5a), red bars indicate features contributing to predicting a data point as ACK attack label correctly, while green bars signify features contributing to predicting a data point as other classes predicted by the model.

Prediction probabilities are assigned to each class on the left side of the figure in these visualizations. LIME explanations for the features demonstrate that the XGB Model predicted as ACK class with 100% accuracy for the selected actual class label. Specifically regarding certain rules generated by LIME:

- 4733.65 < H_L0.1_weight <= 6433.48: If, within a 10-second time window (L0.01), the packet count (weight) of the host (H) feature falls between 4733.65 and 6433.48, the XGB model is more inclined to classify the data point as an ACK attack, indicating a potential compromise of the host by ACK attack.



(a) LIME Explanations for Misclassified Instance



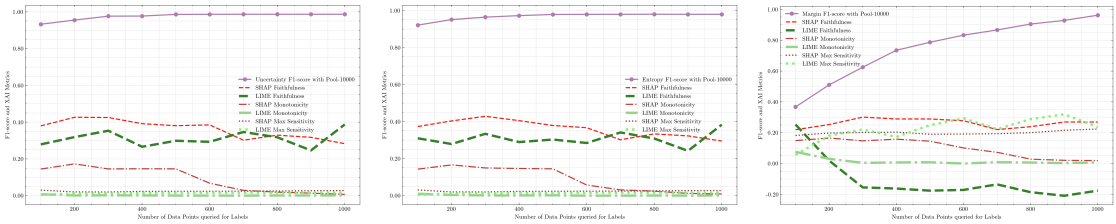
(b) SHAP Explanation for Misclassified Instance

Fig. 6: LIME & SHAP Local Explanations for Misclassified Instances

SHAP force plot is another explainable tool that visually represents the contribution of each feature to a specific prediction by demonstrating its relative importance and direction of impact. In local explanations of SHAP, a specific data point is selected, and the model prediction is explained to highlight the contribution of each feature, quantified by Shapley values. Fig. 5b illustrates a local explanation for a data point using a SHAP force plot. The plot illustrates the base value, with features that positively influence the prediction displayed in red, while those exhibiting a negative impact on the predictions are shown in blue. The base value represented in the plot is the average of all prediction values. Each strip in the plot indicates the influence of the features in pushing the predicted value closer or farther from the base value. Red strip features push the value to higher values, whereas blue strips push the value to lower values. Wider strips mean more contribution. Based on Fig. 5b, the base value for the ACK class attack is 0.5783, respectively. Feature "H_L0.01_variance" positively contributes to the prediction value, making it the most crucial feature due to its broader contribution range. The total positive contribution exceeds the negative contribution, resulting in a final predicted value higher than the base value. Consequently, the class is correctly classified as an ACK attack.

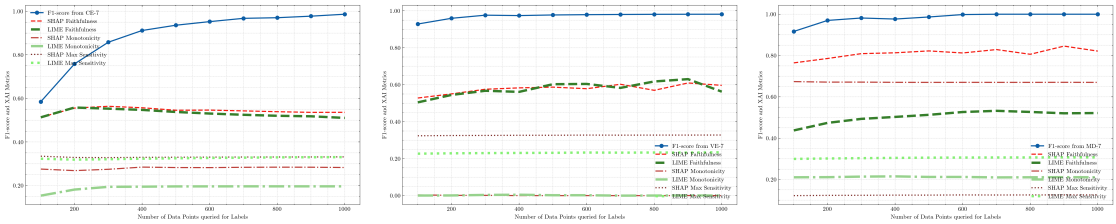
Fig. 6 demonstrates how LIME and SHAP can help explain misclassified instances, specifically when a benign traffic instance is incorrectly classified as a COMBO attack. In the LIME explanation of misclassified instance (Fig. 6a), prediction probabilities show that the COMBO attack is predicted with a probability of 0.34, while the benign probability is 0.22, and scan class prediction probability is 0.14. Feature contributions for this misclassified prediction are described on the left side of the figure, with red bars indicating features that contribute to COMBO attack prediction and green bars

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:00 UTC from IEEE Xplore. Restrictions apply.



(a) Classification Uncertainty (b) Classification Entropy (c) Classification Margin

Fig. 7: XAI metrics results for Uncertainty sampling strategies



(a) Vote entropy (b) Consensus entropy (c) Maximum disagreement

Fig. 8: XAI metrics results for QBC strategies

describing features that contribute to other class predictions. Likewise, a SHAP explanation (Fig. 6b) for the same misclassified instance (Benign class is predicted as a COMBO attack) is presented. SHAP contribution line shifts towards the blue line instead of the red line, indicating that the features are pushing the prediction towards the COMBO class, for the True label class is benign.

Through LIME explanations, the oracle can identify if a model relies on irrelevant features by rule-based explanations, thereby instructing the selection of the most informative instances for the next iteration of model training. SHAP explanations allow annotators to see the relative importance and direction of each feature's impact on an individual prediction. These explanations can help identify consistently important features across predictions, prioritizing which data points should be labelled next. The oracle uses these explanations to evaluate the model's predictions' validity and provide targeted feedback for model refinement. Quickly identify and prioritize which features or data points to focus on for providing feedback or further investigation. As such, they can guide the active learning process by selecting instances for querying that might correct these misclassified instances, improving the robustness and accuracy of the model over time. By providing locally understandable explanations for individual predictions, explainers empower the annotator to identify potential flaws and biases in the model's predictions. This feedback mechanism allows the expert to guide the model by actively selecting the most informative and representative samples for further labeling.

We evaluated the quality of explanations within the active learning loop by applying three quantitative metrics to determine whether the data points selected from the pool and presented as labeling queries to the human expert were effectively explained by the model. Evaluation of explanation quality was conducted using three metrics. Faithfulness computes the correlation between the feature importance scores provided by the explainers and the actual model predictions. The monotonicity metric evaluates whether the incremental changes in the input instance are reflected in the explanations in a consistent way. Max Sensitivity, used to assess the robustness of the explainers, is based on the premise that similar instances in feature space should give similar explanations. To ensure the clarity and readability of the paper, we have presented xai metrics results exclusively for the models that demonstrate the highest performance across sampling strategies. Specifically, the pool size of 10,000 provided the best results for uncertainty sampling. Furthermore, combining a pool size of 10,000 and a committee size of seven led to superior outcomes in the QBC approach. XAI metrics performance results presented in the graphs were the mean values of XAI metrics for explainers over the test data in active learning cycles.

Fig. 8 illustrates XAI metrics results for QBC strategies in the AL loop. Mean values of explainability metrics were computed for test data using explanations generated by LIME and SHAP explainers throughout the AL cycle. In QBC strategies, SHAP explanations consistently outperform LIME explanations regarding faithfulness, monotonicity, and maximum sensitivity. The only exception is that neither SHAP nor LIME have consistently better results for faithfulness results of consensus entropy. For QBC that uses maximum agreement

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:00 UTC from IEEE Xplore. Restrictions apply.

strategy with committee size 7 (QBC-MD-7) (see Fig. 8c), as the number of data points queried for labels increases, faithfulness of SHAP explanations remains high and stable, with correlation values above 0.8. In contrast, LIME's faithfulness is lower and more variable, with correlation values ranging from 0.4 to 0.6. The monotonicity of both explainers improves with more queried data points, indicating that the explanations become more consistent with the model's predictions as AL progresses. SHAP maintains a higher level of consistency, with monotonicity percentages far above 60%, while LIME's monotonicity consistency percentages range from 40% to below 60% in QBC-MD-7 results. Maximum sensitivity of both explainers decreases as more data points are queried, suggesting that the explanations become more robust. Lower sensitivity values indicate that similar instances in the feature space yield similar explanations. SHAP's maximum sensitivity values are consistently lower than LIME's, demonstrating that SHAP explanations are more robust. SHAP explanations for the QBC-MD-7, the highest performing model in F1 score, exhibit high faithfulness correlation values (above 0.8), consistent monotonicity percentages (far above 60%), and low maximum sensitivity values, indicating robust explanations.

QBC-CE-7 (Fig. 8b) and QBC-VE-7 (Fig. 8a) also demonstrate similar trends in XAI metrics, with SHAP explanations outperforming LIME regarding faithfulness, monotonicity, and maximum sensitivity. However, the QBC-MD-7 strategy consistently achieves the highest F1 scores and the most reliable explanations among the three query strategies with SHAP.

Within the uncertainty sampling strategies also, as depicted in Fig. 7, SHAP explanations were more reliable than LIME. However, when compared to the QBC strategy, the metrics of faithfulness and consistency of monotonocity exhibited fluctuations. Consequently, this suggests that the explanations about query strategies may not be consistently dependable. Therefore, Oracle's reliance should be directed toward the QBC sampling strategy, which has demonstrated robust performance with a pool size of 10,000 and a committee size of seven.

V. Conclusion

In our study, we proposed an active learning framework which is enhanced by XAI methods for detecting IoT botnet attacks in SOCs. The framework incorporates post-hoc explainability methods, LIME, and SHAP into the labeling procedures, thus improving the trust between security analysts and learning models. We also evaluated the quality of the explanations by using various quantitative metrics. Our research highlights the potential of explainable active learning to improve AI-based network intrusion detection tasks, exemplified by an IoT botnet case. As a future study, we plan to perform a comprehensive qualitative evaluation of the explanations to explore the practical implications of the proposed framework in SOC settings.

References

[1] P. Srinivasulu, M. S. Babu, R. Venkat, and K. Rajesh, "Cloud service oriented architecture (csoa) for agriculture through internet of things (iot) and big data," in *2017 IEEE international conference on electrical, instrumentation and communication engineering (ICEICE)*. IEEE, 2017, pp. 1–6.

[2] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] R. Kalakoti, S. Nõmm, and H. Bahsi, "In-depth feature selection for the statistical machine learning-based botnet detection in iot networks," *IEEE Access*, vol. 10, pp. 94 518–94 535, 2022.

[5] B. Settles, "Active learning literature survey," 2009.

[6] R. Kalakoti, S. Nõmm, and H. Bahsi, "Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai)," in *2023 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2023, pp. 595–601.

[7] R. Kalakoti, H. Bahsi, and S. Nõmm, "Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection," *IEEE Internet of Things Journal*, vol. 11, no. 10, pp. 18 237–18 254, 2024.

[8] T. Kim and W. Pak, "Early detection of network intrusions using a gan-based one-class classifier," *IEEE Access*, vol. 10, pp. 119 357–119 367, 2022.

[9] R. K. Muna, M. I. Hossain, M. G. R. Alam, M. M. Hassan, M. Ianni, and G. Fortino, "Demystifying machine learning models of massive iot attack detection with explainable ai for sustainable and secure future smart cities," *Internet of Things*, vol. 24, p. 100919, 2023.

[10] "Home," https://sites.google.com/view/iot-network-intrusion-dataset/home, (Accessed on 01/26/2024).

[11] L. Almuqren, M. S. Maashi, M. Alamgeer, H. Mohsen, M. A. Hamza, and A. A. Abdelmageed, "Explainable artificial intelligence enabled intrusion detection technique for secure cyber-physical systems," *Applied Sciences*, vol. 13, no. 5, p. 3081, 2023.

[12] "Ids 2017 | datasets | research | canadian institute for cybersecurity | unb," https://www.unb.ca/cic/datasets/ids-2017.html, (Accessed on 01/26/2024).

[13] "Nsl-kdd | datasets | research | canadian institute for cybersecurity | unb," https://www.unb.ca/cic/datasets/nsl.html, (Accessed on 01/26/2024).

[14] X. Larriva-Novo, C. Sánchez-Zas, V. A. Villagrá, A. Marín-Lopez, and J. Berrocal, "Leveraging explainable artificial intelligence in real-time cyberattack identification: Intrusion detection system approach," *Applied Sciences*, vol. 13, no. 15, p. 8587, 2023.

[15] "The unsw-nb15 dataset | unsw research," https://research.unsw.edu.au/projects/unsw-nb15-dataset, (Accessed on 01/26/2024).

[16] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot: Network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[17] C. C. Aggarwal *et al.*, *Data mining: the textbook*. Springer, 2015, vol. 1.

[18] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin, """ why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[20] Y. Sasaki, "The truth of the f-measure," *Teach Tutor mater*, 2007.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:00 UTC from IEEE Xplore. Restrictions apply.

Appendix 8

VIII

R. Kalakoti, H. Bahsi, and S. Nõmm. Explainable federated learning for botnet detection in iot networks. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 01–08. IEEE, 2024

2024 IEEE International Conference on Cyber Security and Resilience (CSR)

Explainable Federated Learning for Botnet Detection in IoT Networks

Rajesh Kalakoti
Department of Software Science
School of Information Technology,
Taltech, Tallinn, Estonia.
rajesh.kalakoti@taltech.ee

Hayretdin Bahsi
Department of Software Science
School of Information Technology
TalTech, Estonia &
School of Informatics, Computing, and Cyber Systems
Northern Arizona University, United States
hayretdin.bahsi@nau.edu

Sven Nõmm
Department of Software Science
School of Information Technology
Taltech, Tallinn, Estonia.
sven.nomm@taltech.ee

2024 IEEE International Conference on Cyber Security and Resilience (CSR) | 979-8-3503-7536-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/CSR61664.2024.10679348

***Abstract*—The widespread use of Internet of Things (IoT) devices has increased the vulnerability to botnet attacks, presenting significant challenges to network security. Federated learning (FL) is a promising approach for detecting IoT botnets while preserving data privacy. However, the black-box nature of FL models impedes their interpretability and transparency, which are crucial for trust and accountability in security applications. In this paper, we propose an approach to generate explanations for the server model induced for intrusion detection tasks in the FL setting without direct access to data of IoT device-based clients. This involves aggregating SHAP (SHapley Additive exPlanations) value explanations from individual IoT device-based client models to approximate the server model's explanations. We evaluated this approach by comparing the aggregated client-based explanations with the server-based explanations obtained when the server has access to the data of participating IoT device clients. We employed a deep neural network (DNN) model trained in a horizontal federated learning (HFL) setting with the federated averaging (FedAvg) algorithm. DNN model achieved high detection rates of Accuracy, Precision, Recall & F1-Score in Botnet Detection of multiclass classification on both IoT device-based client-side models and the server-side model. Additionally, we analyzed the importance of features contributing to IoT botnet detection using the generated SHAP explanations. The results demonstrate that the aggregation of client-based SHAP explanations closely approximates the server-based explanations, achieving comparable explainability without compromising data privacy.**

***Index Terms*—Federated Learning, Internet of Things, Botnet, Intrusion Detection, Explainable AI, Post-Hoc explainability, SHAP, Horizontal Federated Learning**

I. Introduction

Intrusion detection is crucial in IoT networks [1] to prevent unauthorized access to devices. There are two types of Intrusion Detection Systems (IDS): Signature-based and Anomaly-based. Signature-based IDS relies on pre-defined attack patterns, while Anomaly-based IDS detects malicious activity through deviations from normal behavior [2]. Though Signature-based detection performs better, it fails to detect new attacks. Adapting machine learning (ML) algorithms for Anomaly-based detection can enhance self-learning and develop more intelligent systems for detecting attacks in IoT environments [3]. Although ML techniques can enhance IDS approaches, most ML-enabled IDS deployments are centralized, leading to privacy issues. Federated Learning (FL) was proposed to address this issue [4]. FL is a collaborative learning approach where end devices share partial global model updates aggregated by a central entity, improving privacy. As ML is increasingly applied, users are more concerned about data privacy. To protect personal privacy, users may hesitate to share data. Although FL is commonly used to train deep learning models, its complexity makes it difficult to explainable, which is a challenge for high-stakes decision-making [5]. Meaningful and interpretable explanations of predictive models are critical to building trust between AI systems and human experts. These explanations help domain experts, such as security analyst and system administrators, make informed and accurate decisions, leading to better decision-making and accountability. More data privacy and security laws, such as General Data Protection Regulation (GDPR) [6] in the EU and China's Cybersecurity Law [7], were introduced to protect user privacy. Data cannot be shared or collected centrally from multiple parties due to strict regulations against leakage of private data. Using data from multiple parties to train ML models while complying with privacy regulations is an urgent issue.

This research explores the use of explainable AI (XAI) techniques in federated learning (FL) for botnet detection in IoT networks. This study aims to enhance the explainability and transparency of the decision-making process in FL-based intrusion detection systems while preserving data privacy. More specifically, this paper proposes an approach for explaining a prediction made by the global Deep neural Network model (server model) by aggregating explanations of clients who participated in the federation. This approach is based on the requirement that explaining the global model (server model) for the XAI method does not necessitate access to the training data of the participating clients. Using an FL-based IDS with XAI in IoT, we use this approach in the context of IoT botnet detection in a multiclass benchmarking setting. To assess the effectiveness of this approach, we use an analytical methodology to compare its approximation with a simulated scenario where a server with access to the training

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:01 UTC from IEEE Xplore. Restrictions apply.

data is involved. This study demonstrates the effectiveness of SHAP in generating explanations and identifying essential features for IoT botnet detection in FL Settings. To the best of our knowledge, this is the first work to investigate the integration of XAI techniques with federated deep learning for IoT intrusion detection. The insights gained from this research can benefit in developing more transparent and trustworthy FL-based security solutions. Furthermore, the findings contribute to the broader field of XAI by showcasing the applicability of xai methods in distributed learning environments.

Our paper is structured as follows: Section II covers background on Federated Learning, while Section II-A gives related work of federated learning in intrusion detection systems of IoT. Section III outlines our methodology, followed by Section IV presenting results and discussions. Finally, conclusions are Section V.

II. Background work on Federated Learning

Federated learning (FL), as defined by Yang et al., trains an ML model using data from multiple sources while ensuring data privacy [8]. Let N be the number of clients, denoted as $c_1, \ldots, c_N$. Each client c_i owns a set of samples denoted by U_{c_i}, feature space by X_{c_i}, label space by Y_{c_i}, and dataset by $D_{c_i} = \{u_{c_i}^{(j)}, x_{c_i}^{(j)}, y_{c_i}^{(j)}\}_{j=1}^{|D_{c_i}|}$. Here, data point $(u_{c_i}^{(j)}, x_{c_i}^{(j)}, y_{c_i}^{(j)})$ signifies that client c_i owns sample $u_{c_i}^{(j)}$ with features $x_{c_i}^{(j)}$ and label $y_{c_i}^{(j)}$. For instance, in IoT networks, U refers to the set of all IoT devices, X denotes features such as Network Traffic and Communication Protocols and Y indicates whether an IoT device is infected with Malware (Eg. Dos and DDos) or not. Based on data partition (X, Y, U) among clients, FL can be categorized into three types by Yang et al. [9]: (1). Horizontal Federated Learning (HFL): In HFL, clients share the same feature and label space ($X_i = X_j, Y_i = Y_j$) but have different sample spaces ($U_i \neq U_j$). (2). Vertical Federated Learning (VFL): In VFL, some common data samples are shared among clients ($U_i \cap U_j \neq \emptyset$), but feature and label spaces differ ($X_i \neq X_j$, $Y_i \neq Y_j$). (3). Federated Transfer Learning (FTL): FTL imposes no restrictions on sample, feature, and label space, allowing arbitrary differences. Furtherly, FL can be categorized as Cross-device FL (B2C) and Cross-silo FL (B2B). Cross-device (CDFL) involves many clients with lower resources, while Cross-silo (CSFL) involves companies or institutions with more robust computational and communication capabilities. Client-server architecture is the most popular HFL architecture, and Federated Averaging (FedAvg) algorithm [4] is based on it. Optimization objective of HFL as:

$$\min_{\theta} L(\theta) = \sum_{i=1}^{N} L_{c_i}(\theta) = \sum_{i=1}^{N} \frac{1}{|D_{c_i}|} \sum_{j=1}^{D_{c_i}} l(x_{c_i}^{(j)}, y_{c_i}^{(j)}; \theta) \quad (1)$$

Where θ represents model parameters, $L(\theta)$ is the global optimization objective. $L_{c_i}(\theta)$ is the optimization objective of client c_i based on its local data D_{c_i}, with $l(x, y; \theta)$ denoting the loss function, such as cross-entropy or mean squared loss. SGD (Stochastic gradient descent) is commonly used to optimize Equation. 1. The training process of HFL as follows:

1) Each client samples a batch of data B, and calculates gradients $g_{c_i} = \frac{1}{|B|} \nabla_\theta \sum_{x,y \in B} l(x, y, ; \theta)$ and sends g_{c_i} to the server.
2) The server Aggregates the gradients (eg. averaging $g = \frac{1}{N} \sum_{i=1}^{N} g_{c_i}$) and then distributes the aggregated gradient to all clients.
3) Each client updates the model by gradient descent based on the aggregated gradient g.

The process is executed iteratively until the loss function $L(\theta)$ converges. After training finishes, all clients share the same model parameters θ.

HFL has high communication overhead, especially in CDFL. FedAvg reduces communication costs by increasing local computation to converge the model faster in HFL. Instead of transmitting gradients for every batch, FedAvg transmits gradients every $\mathcal{E}$ epochs. Assuming current model parameters $\theta^{(t)}$, batch size b and and learning rate η, FedAvg algorithm follows these steps

1) Each client divides the local data D_{c_i} into batches of size b.
2) For each batch B, the clients perform parameter updates $\theta_{c_i}^{(t)} \leftarrow \theta_{c_i}^{(t)} - \frac{\eta}{b} \sum_{x,y \in B} \nabla_\theta l(x, y; \theta_{c_i}^{(t)})$ until the local dataset $\mathcal{D}_{c_i}$ is iterated $\mathcal{E}$ epochs.
3) Each client sends the updated parameters $\theta_{c_i}^{t}$ to aggregation server.
4) The aggregation server computes a weighted average of the model parameters $\theta^{t+1} = \sum_{c_i=1}^{N} w_{c_i} \theta_{c_i}^{(t)}$, where w_{c_i} is calculated from the size of each local dataset
5) the aggregation server distributes $\theta^{(t+1)}$ to the clients.

A. Related Work

ML-based IDS relies on ample data to build a model. However, centralizing all data may not be feasible in complex environments with multiple parties or clients, leading to privacy issues and high communication latency [10]. FL ensures data privacy without compromising model accuracy at low latency and communication costs. Bonawitz et al. designed a decentralized FL system, which differs from traditional ML architecture. FL techniques have been proposed for cybersecurity and intrusion detection in IoT networks [11]. Nguyen et al. proposed a self-learning anomaly detection system using FL for compromised IoT devices detection [12]. The system employs LSTM (Long short-term memory) and GRUs (Gated recurrent unit) for building the model. It constantly monitors the devices' network traffic and detects anomalous deviations from given communication profiles. The system has achieved an accuracy of 98.2% and can detect 95.6% of attacks in just 257ms with less false alarm rate. In a paper [13], authors presented an approach for intelligent intrusion detection using FL with LSTM recurrent neural network that achieved an accuracy and F1 score of 99.21% by comparing the efficiency of the proposed algorithm with conventional neural networks. Liu et al. propose a deep anomaly detection framework for Industrial IoT that uses FL approach on distributed edge devices [14]. It includes a CNN (Convolutional neural network)-LSTM model to extract fine-grained features of

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:01 UTC from IEEE Xplore. Restrictions apply.

historical observations and uses LSTM model for time-series predictions, preventing memory loss and gradient-dispersion problems. IoT devices are vulnerable to cyber attacks due to their architecture, making manual detection difficult. Attota et al. propose a FL based approach called MV-FLID, which trains on multiple views in a decentralized format. MV-FLID uses multi-view ensemble learning to detect, classify, and defend against attacks [15]. Evaluation results were shown that MV-FLID was more accurate than traditional non-FL centralized approaches. Other works [16, 17] have focused on Explainable AI (XAI) and it's quality of evaluation for centralized models of ID in IoT. Another study analyzed the impact of feature selection on the explanations obtained in the post-hoc interpretability step in IoT botnet detection [18]. However, these approaches did not address privacy concerns. This work is the first to focus on explaining the server model in an XAI-based IDS within a FL benchmarking setting.

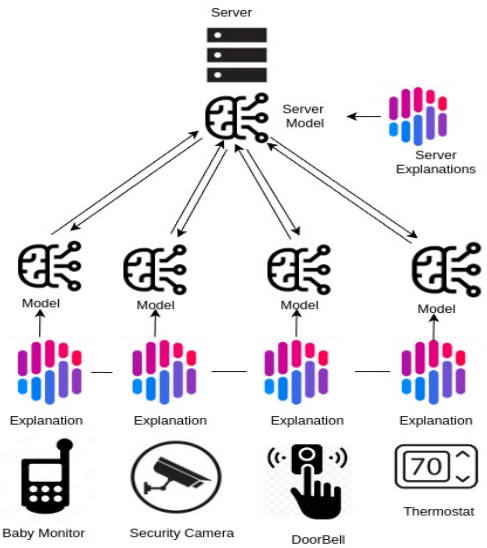


Figure 1: Explainable AI in Horizontal Federated Leaning for IoT botnet Detection

III. Methodology

Fig. 1 illustrates the architecture of the proposed HFL approach for IDS in IoT network, in which several devices are connected to the network and distributed across various sites. These devices are connected to gateways, and their network traffic data is monitored by profiling tools (e.g. Wireshark) for further investigation in the locally running host-based IDS on IoT devices. Each device has an ML model induced by its network data. Selected IoT devices share their parameters of ML models with a server-based aggregation platform, which aggregates these models to generate an enhanced intrusion detection model with optimized parameters. This sharing approach improves learning by enabling devices to detect intrusions based on comparable behaviour learned from different participating devices. After training the black box model in FL, XAI provides transparency into how the IDS makes its decisions. XAI identifies key features and patterns, making the detection process more interpretable and trustworthy. This ensures that security experts can understand and validate the actions of the IDS, thereby fostering user trust in the AI-based IDS. However, to explain the server model of FL, server needs to access the data. To address this, an approach for explaining the server model without sharing data was proposed in this paper, allowing for generating explanations while preserving privacy.

A. Dataset

N-BaIoT dataset [19] is appropriate for evaluating privacy-preserving collaborative training for IoT Botnet and B5G applications. It separates traffic data into files, allowing easy distribution into non-identically distributed parts. However, it only contains data from 9 IoT devices, which limits the number of clients for experiments. This dataset contains 115 network traffic features primarily extracted through descriptive statistical values. Dataset was derived from network traffic created by bots deployed in a controlled environment with 9 different IoT devices infected by Mirai and Gafgyt malware types.

Table I: Details of Features of N-BaIoT dataset.

Feature Category	Category Code	No.Of features	Statistical Value Feature	Time Frame Windows
Hos Mac & IP	MI	15	Packet Count, Mean	100 Micrso Sconds
Host IP	H	15	Variance	500 micro seconds
Network Jitter	HH_jit	15		1.5 Seconds
Channel	HH	35	Packet Count, Mean	10 Seconds
Socket	HpHp	35	Variance, Magnitude,Radius, CoVariance, Correlation	1 Minute

Table II displays the number of benign and Botnet deployed samples for each device. However, Doorbell2 and Webcam did not generate traffic while infected by Mirai Botnet. Notably, the features for each instance reflect aggregated descriptive statistics of the raw network traffic across five-time windows ($100\mu s$, $500\mu s$, $1.5s$, $10s$, and 1_{min}), denoted as L_5, L_3, L_1, $L_{0.1}$, and $L_{0.01}$, respectively (see Table I). Dataset encompasses five main feature categories: host-IP (coded as H), host-MAC and IP (coded as MI), channel (coded as HH), socket (coded as HpHp), and network jitter (coded as HH_jit). For each category, metrics such as packet count, mean packet size, and variance in packet sizes are calculated. Additionally, statistical values, including the Pearson correlation coefficient (PCC) of packet size, radius, covariance, and magnitude, are derived for network channels and sockets, along with packet count, mean, and variance.

B. Deep Neural Network for IoT Botnet Detection in FL

In this study, a Deep Neural Network (DNN) model was proposed to detect IoT botnets over N-BaIoT dataset using network traffic features. DNN model consists of an input layer, hidden layers, and an output layer containing neurons. Input

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:01 UTC from IEEE Xplore. Restrictions apply.

Table II: Number of Botnet malware & benign samples for each device over N-BaIoT dataset [19]

IoT Devcie Name	Short Name	Deployed BoTnet Samples		
		Mirai	Gafgyt	Benign
Danmini Doorbell	Doorbell1	652100	316650	49548
Philips B120N10 Baby_Monitor	BabyMonitor	610714	312723	175240
Provision PT 737E Security_Camera	SecurityCam1	436010	330096	62154
SimpleHome XCS7 1003 WHT Security Camera	SecurityCam2	514860	316438	19528
Ecobee Thermostat	Thermostat	512133	310630	13113
Provision PT 838 Security Camera	SecurityCam3	429337	309040	98514
SimpleHome XCS7 1002 WHT Security Camera	SecurityCam4	513248	303223	46585
Ennio Doorbell	Doorbell2	-	316400	39100
Samsung SNH 1011N Webcam	Webcam	-	323072	52150
Total		3668402 (57.9%)	2198800 (34.7%)	464682 (7.4/%)

layer had 115 neurons, which is the same as the number of network traffic features in dataset. The architecture of the model includes 7 hidden layers, each containing 81 hidden units was determined by experimentation using Random Search Hyperparameter Tuning with help of Ray Tune library[1] over data before distributing it into client-wise partitions. These optimized hyperparameters (see in Table III) were then passed into entire federated training setting to train DNN model.

Model was trained using the Rectified Adam (RAdam) optimizer with a learning rate of 0.000669451. The Scaled Exponential Linear Unit (SELU) activation function was applied after each hidden layer, introducing non-linearity into the model. The model's output layer is a fully connected layer that maps activations from the final hidden layer to the output size, indicating the number of classes for botnet malware type classification (Benign, Mira, Gafgyt). During the training phase, Each client model underwent 90 epochs with a batch size of 512 for each client's data in every communication round. Cross Entropy Loss function was utilized to compute the loss during training, making it well-suited for this multi-class classification task.

Table III: Parameters for DNN used in Federated Learning

Hyperparameter	Value
Hidden layers	7
Hidden units	81
Learning rate	0.000669451
Optimizer	RAdam
Activation	SELU
Batch size	512
epochs	90

C. Explainable AI in Federated Learning

This work used SHapley Additive exPlanations (SHAP) method [20]. SHAP is widely used and local post-hoc model-agnostic explanation method computes feature importance using Shapley values based on cooperative game theory [21]. A post hoc explanation, denoted as $\mathbf{g}$, is an explanation mapping that, for a black box model predictor $\mathcal{M}$ and point of interest x, returns an importance score $\mathbf{g}(\mathcal{M}, x) = \varphi_x \in \mathbb{R}^d$ for all features. SHAP is defined as $\mathbf{g}(\mathcal{M}, x) = \phi_0 + \sum_{j=1}^{M} \phi_j$, where ϕ_j is the feature attribution of feature j. SHAP values can be used for local explanation , which explain individual instances, and global explanation (overall model explanation), which can be computed using different explanation models. We used SHAP's *DeepExplainer*[2], which is model-agnostic.

SHAP Explainer was used to explain DNN model learned by FedAvg in HFL architecture mentioned in Section II. SHAP requires access to the training data or a reference set to create records for studying the impact of each feature value on the final prediction of the model. For server-based FL, the server needs access to its clients' complete set of training data or to compute the centroids of the dataset resulting from the union of the training sets of all the clients to explain the learned server model. In this work, we followed the approach outlined by Corbucci et al. [22],where an explanation for the server model was obtained by aggregating all client's explanations, thus avoiding data sharing to Server model.

After completing the training of FL, each client c_i receives its trained model $\mathcal{M}_{c_i}$, which is then sent to the server. Server then creates its own model $\mathcal{M}_s$ by averaging the weights of the received models using FedAvg algorithm. Each client c_i can then obtain SHAP explanation $\mathbf{g}_{c_i}(\mathcal{M}_{c_i}, x)$ of it's data points over their model $\mathcal{M}_{c_i}$, which strongly relies on the training data. To derive explanations from the server's perspective, additive nature of SHAP values were leveraged, which enables the generation of explanations for model $\mathcal{M}_s$ as an aggregation of explanations from the individual models $\mathcal{M}_c$ within U. Therefore, for a point of interest x, an explanation of the prediction performed by server model $\mathcal{M}_s$ is obtained by the following equation.

$$\mathbf{g}_s(\mathcal{M}_s, x) = E_{local} = \frac{1}{|N|} \sum_{c_i \in N} \mathbf{g}_{c_i}(\mathcal{M}_{c_i}, x). \qquad (2)$$

Hence, data sharing is not required for server to obtain explanations from server model $\mathcal{M}_s$. where, N is the number of client models who participated in FL.

To check if E_{local} explanations are sufficient for the server model, Server model $\mathcal{M}_s$ was provided data from all clients N and obtains explanations. These explanations are then

1. https://docs.ray.io/en/latest/tune/index.html

2. https://shap-lrjball.readthedocs.io/en/latest/generated/shap.DeepExplainer.html

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:01 UTC from IEEE Xplore. Restrictions apply.

aggregated for all clients by providing the combined training data to the SHAP explainer, resulting in explanation E_{global}. Finally, to check the correctness of the explanations, difference (d_{g-l}) between E_{global} and E_{local} was computed, i.e.,

$$d_{(g-l)} = E_{global} - E_{local}. \tag{3}$$

Difference between E_{global} and E_{local} represents the divergence between the server-based explanations (obtained by directly accessing the clients' data) and the average client's explanations (obtained by aggregating the individual client explanations without providing the data to server). This divergence calculation helps assess the quality and reliability of aggegagtion of client's explanations (E_{local}) compared to the server-based (E_{global}) ones. This approach enables explainability in FL while preserving data privacy.

In our case, SHAP values were computed on validation data. Each client's (IoT device) SHAP values explanations was an $m \times d$ matrix, where m is the number of data points from test data and d is the number of features. Aggregation of client-based explanations was computed by taking the element-wise average of individual client SHAP values explanations, resulting in E_{local}, which will also be an $m \times d$ matrix. Similarly, when server model was accessed test data of clients, server-based explanations (E_{global}) will also be an $m \times d$ matrix. Difference $d_{(g-l)}$ between E_{local} and E_{global} was computed by taking the element-wise difference between the two matrices, resulting in a matrix of size $m \times d$.

D. Experimental Setup

During federated learning, a deep neural network (DNN) was trained on data that is distributed from multiple IoT devices or clients. Each client's data is divided into training and validation sets using an 80:20 ratio. During each round of federated training, validation data from each client was tested on its respective client model. Once all clients have been processed, the combined validation data from all clients was tested on the server model. To prepare the data for training of DNN model, Min-max normalization was applied to scale feature values between 0 and 1 for each client's training and validation data.

In the evaluation of learning models in federated learning, a testing dataset was exclusively used on IoT devices. This study utilized Accuracy, Precision, Recall, and F1-score metrics to evaluate both the clients' and server models in federated learning over 200 communication rounds.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$\text{F1-Score} = \frac{2 \times TP}{(2 \times TP) + FP + FN} \tag{7}$$

From the N-BaIoT dataset, true positive (TP) is the number of network traffic statistical feature data points in the positive class that are correctly classified as positive; false positive (FP) is the number of network traffic statistical feature data points in the negative class that are misclassified as positive; true negative (TN) is the number of data points in the negative class that are correctly classified as negative; and false negative (FN) is the number of network traffic statistical feature data points in the positive class that are misclassified as negative.

Experiments were conducted on a server powered by an AMD Threadripper 3960X 24-Core/48-thread Processor, 128 GB RAM, and an NVidia 3090 GPU with 24 GB graphics memory, running Ubuntu 20.04 LTS. Experiments were implemented in Python.3.9, using Flower[3] [23] for FL with PyTorch[4].

IV. Results & Discussions

In this work, our proposal aims to use explainable AI to explain DNN model learned by FedAvg algorithm in the case of HFL architecture in a multiclass classification scenario for IoT botnet detection. This research used N-BaIoT dataset, which has 9 IoT devices attacked by botnets such as Gafgyt and Mirai. Mirai botnet did not infect 'Doorbell2' and 'Webcam' devices. Consequently, these two devices were excluded from the experiments to ensure consistency in the label space and input feature space during federated training using FedAvg algorithm because of the horizontal nature of FL approach.

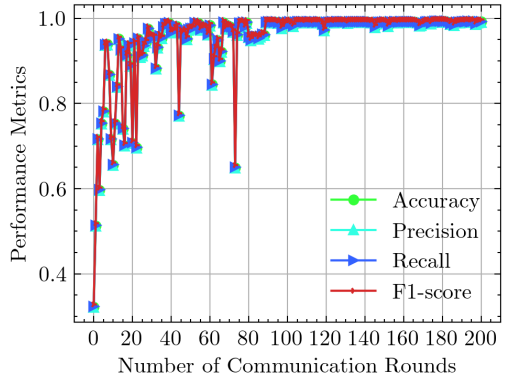
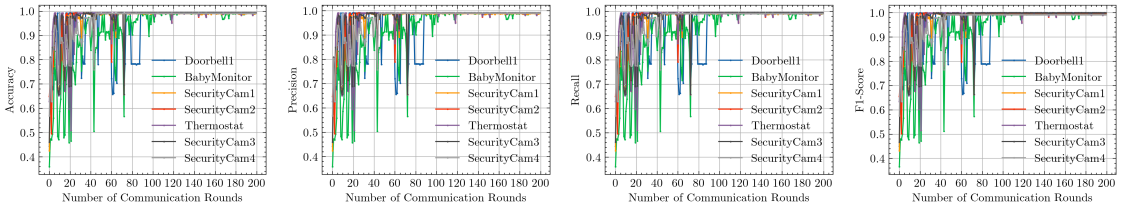


Figure 2: Evaluation of DNN Model Performance on Server side

Fig. 2 illustrates how well federated DNN model performed regarding Accuracy, Precision, recall and F1-score metrics. Server model was evaluated using combined test data from all participating clients (7 IoT devices). In federated learning, Evaluation was conducted over 200 communication rounds between the server and the clients, using the fedavg algorithm in a HFL setting. 50% of clients were randomly selected for

3. https://flower.ai/
4. https://pytorch.org/

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:01 UTC from IEEE Xplore. Restrictions apply.



(a) Accuracy (b) Precision (c) Recall (d) F1-score

Figure 3: Evaluation of DNN model performance on client's Side

each round of training and evaluation. This ensures diversity in client updates and provides a more robust measure of the model's generalization ability. All 7 clients were needed for the evaluation round to proceed, ensuring a reliable estimate of the model's performance. As the number of communication rounds increased, classification performance metrics notably improved. Initially, when server model parameters were initialized, metrics accuracy, precision, recall and F1-score began at approximately 0.4. However, after completing 200 communication rounds, classification performance metrics reached an impressive peak of around 0.99.

Fig. 3 presents classification performance metrics of client models when evaluated with its own parameters on each client's validation data. As communication rounds increased, each client's accuracy, precision, recall and F1-score improved, indicating successful learning. However, it was noted that there was variation in performance metrics among clients. By the end of the communication rounds, almost all clients achieved high classification performance, which indicates that model performed exceptionally well on the validation data.

As mentioned in Section III-C, SHAP was used to explain the DNN model learned by the FedAvg algorithm in the case of HFL architecture. In this work, we aim to investigate the differences in the explanations from the point of view of the feature's importance for the Server Model without providing data. For this purpose, this work considers the aggregation of explanations from client models as explanations for the server model. So, the Server does not need to access the data from clients to explain the server model. SHAP values were computed on each validation data of client to find how each feature contributes to a model's output. In this context, the model's output refers to the probability distribution over the class labels (Mirai, Gafgyt,and Benign) obtained using the softmax activation function. Therefore, the computation of SHAP values necessitates the calculation of the softmax function on the model's predictions. Following the completion of FL training, Client models and server models were taken at the 200th communication round for explanation generation.

SHAP values were computed for over 2000 test data points for each client. By aggregating the SHAP values of all clients, we obtained client-based explanations (E_{local}). We performed a comparative analysis to check the sufficiency of client-based explanations that are good for the server and validate their

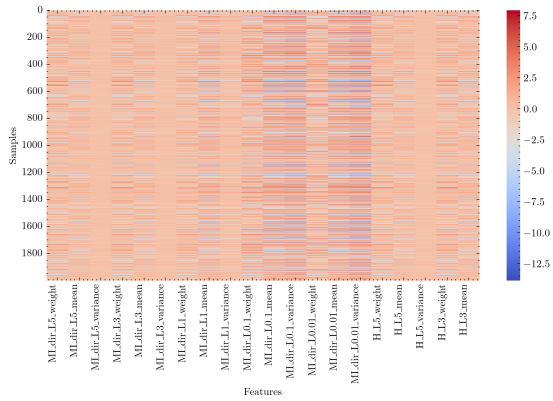
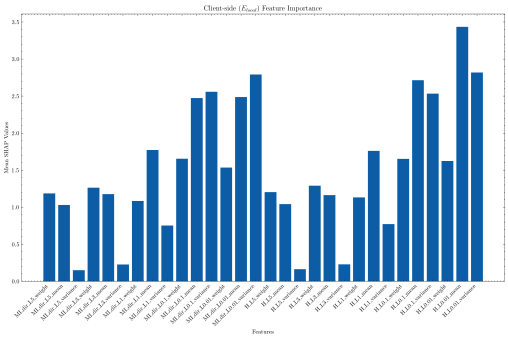


Figure 4: Heatmap for Difference Between aggregation of client's models Explanations and server model explanations

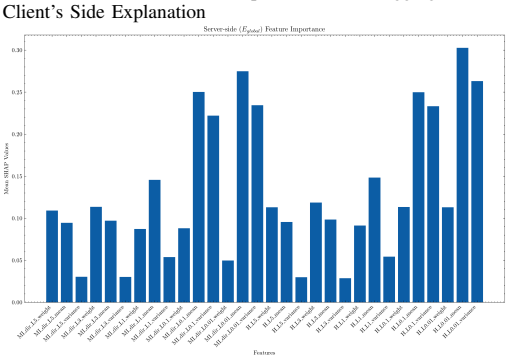
effectiveness in representing the server model's behaviour. So, To obtain the server-based explanations (E_{global}), the server model accessed each client's data, and SHAP values were computed and aggregated again, resulting in explanations of the same size as E_{local}. Finally, Difference (d_{g-l}) between the server-based explanations E_{global} and the client-based explanations (E_{local}) was then computed, representing the divergence between the two sides. Fig. 4 shows a heatmap that visualizes magnitude of the difference between server-based explanations and client-based explanations for each sample and feature. Heatmap shows discrepancy between client-based explanations and server-based explanations for 20 features. Only 20 features out of 115 were included for readability purposes. From heatmap, it can be obeserved that the overall divergence between the server-based and client-based explanations is relatively small. However, there are a few notable instances where the divergence is more significant. For example, some samples exhibit darker shades for certain features, indicating a larger discrepancy between the server-based and client-based explanations for those specific instances.

To provide a more detailed analysis of the features, We analyzed client-based and server-based explanations, and the

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:01 UTC from IEEE Xplore. Restrictions apply.



(a) SHAP values feature importance over Aggregation of Client's Side Explanation



(b) SHAP Value feature importance over Server model

Figure 5: SHAP value feature importance from both Client's side models and Server model

difference between them. We computed the average SHAP values for client-based, averaged server-based explanations (see in Fig. 5). First 30 features were selected for this analysis. It can be observed that the SHAP values of most features were comparable in both Fig. 5a & 5b, implying that aggregation of client-based explanations closely approximates server-based explanations. For Example, Feature 'H_L0.01_mean' is the top feature in both the client-side and server-side explanations, indicating its significant importance in the model's predictions for server model. This suggests that the feature 'H_L0.01_mean' (Host Based features with 1_{min} time window) plays a crucial role regardless of whether the explanations are obtained from the client-side without sending data or from the server model accessing the data directly. On the other hand, features such as 'MI_dir_L5_variance' (Host Mac & IP with $100_{\mu s}$), 'MI_dir_L3_variance' (Host Mac & IP with $500_{\mu s}$), 'H_L5_variance' (Host with $100_{\mu s}$), and 'H_L3_variance' (Host with $500_{\mu s}$) have relatively lower SHAP values in both the client-side and server-side explanations. This indicates that these features have a lesser impact on the model's predictions compared to other features. Fig 6 shows Difference between SHAP values obtained by the server-based explanations and those obtained by the client-based explanations. Fig. 6 illustrates that there is no discrepancy between the server-side and client-side explanations for certain features, as suggested by the SHAP values. This highlights the effectiveness of the FL approach in capturing feature importance while preserving data privacy. The similarity between the client-side and server-side explanations suggests that the aggregation of local explanations provides a good approximation of the global explanations, even without direct access to the clients' data.

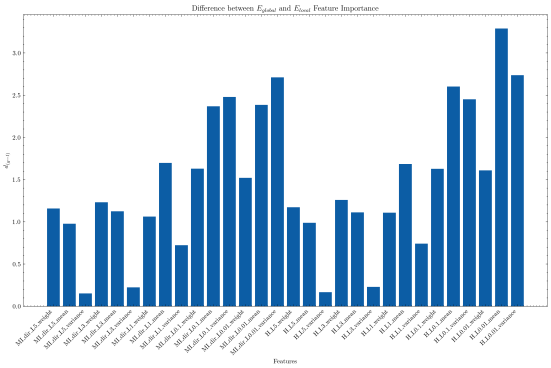


Figure 6: Average Feature importance of Difference client's-based Explanations and server model explanations

V. Conclusion & Future Work

In conclusion, our research highlights the potential of integrating XAI techniques with federated learning for enhanced interpretability and transparency in IoT botnet detection. The proposed approach of aggregating client-based SHAP explanations provides a promising solution for generating meaningful explanations while preserving data privacy. Our DNN Model has achieved more than 99% of detecting botnet in IoT networks. We conducted research on integrating explainable AI (XAI) techniques, specifically SHAP, with federated learning (FL) for IoT botnet detection. Our goal was to improve the interpretability and transparency of FL-based IDS while preserving data privacy. We proposed an approach that aggregates explanations from participating clients to generate global DNN model explanations without direct access to their training data. Our results showed that client-based SHAP explanations closely approximated server-based explanations, achieving comparable explainability without compromising data privacy. This approach provides valuable insights into the decision-making process of FL-based IoT botnet detection while maintaining client data confidentiality. Additionally, our study demonstrated the effectiveness of SHAP in identifying essential features for IoT botnet detection in FL settings. In future, we aim to enhance the security of explanations using secure multi-party computation and privacy preserving techniques to prevent attacks on ML models from black-box adversaries.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:01 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] P. Srinivasulu, M. S. Babu, R. Venkat, and K. Rajesh, "Cloud service oriented architecture (csoa) for agriculture through internet of things (iot) and big data," in *2017 IEEE international conference on electrical, instrumentation and communication engineering (ICEICE)*. IEEE, 2017, pp. 1–6.

[2] V. Jyothsna, R. Prasad, and K. M. Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26–35, 2011.

[3] R. Kalakoti, S. Nõmm, and H. Bahsi, "In-depth feature selection for the statistical machine learning-based botnet detection in iot networks," *IEEE Access*, vol. 10, pp. 94 518–94 535, 2022.

[4] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[5] F. Doshi-Velez and B. Kim, "Roadmap for a rigorous science of interpretability. arxiv," *arXiv preprint arXiv:1702.08608*, 2017.

[6] "General data protection regulation (gdpr) – official legal text," https://gdpr-info.eu/, (Accessed on 04/20/2024).

[7] F. Yang and J. Xu, "Privacy concerns in china's smart city campaign: The deficit of china's cybersecurity law," *Asia & the Pacific Policy Studies*, vol. 5, no. 3, pp. 533–543, 2018.

[8] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning: synthesis lectures on artificial intelligence and machine learning," *vol*, vol. 13, pp. 1–207, 2019.

[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[10] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Applied Sciences*, vol. 8, no. 12, p. 2663, 2018.

[11] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of machine learning and systems*, vol. 1, pp. 374–388, 2019.

[12] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[13] R. Zhao, Y. Yin, Y. Shi, and Z. Xue, "Intelligent intrusion detection based on federated learning aided long short-term memory," *Physical Communication*, vol. 42, p. 101157, 2020.

[14] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348–6358, 2020.

[15] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117 734–117 745, 2021.

[16] R. Kalakoti, H. Bahsi, and S. Nõmm, "Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection," *IEEE Internet of Things Journal*, 2024.

[17] R. Kalakoti, S. Nõmm, and H. Bahsi, "Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai)," in *2023 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2023, pp. 595–601.

[18] A. Guerra-Manzanares, S. Nõmm, and H. Bahsi, "Towards the integration of a post-hoc interpretation step into the machine learning workflow for iot botnet detection," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 1162–1169.

[19] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot: Network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[20] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[21] L. S. Shapley, "A value for n-person games," *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 307–317, 1953.

[22] L. Corbucci, R. Guidotti, and A. Monreale, "Explaining black-boxes in federated learning," in *World Conference on Explainable Artificial Intelligence*. Springer, 2023, pp. 151–163.

[23] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

Authorized licensed use limited to: Tallinn University of Technology. Downloaded on January 07,2025 at 12:15:01 UTC from IEEE Xplore. Restrictions apply.

Appendix 9

IX

R. Kalakoti, S. Nõmm, and H. Bahsi. Federated learning of explainable ai (fedxai) for deep learning-based intrusion detection in iot networks. *Computer Networks*, page 111479, 2025

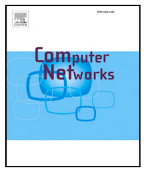
Computer Networks 270 (2025) 111479



Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Federated Learning of Explainable AI(FedXAI) for deep learning-based intrusion detection in IoT networks

Rajesh Kalakoti [a,*], Sven Nõmm [a], Hayretdin Bahsi [b,a]

[a] Department of Software Science, School of Information Technology, Tallinn University of Technology (TalTech), Akadeemia tee 15a, 12618, Tallinn, Estonia
[b] School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, AZ 86011, USA

ARTICLE INFO

Keywords:
Privacy-preserving
Explainable AI
Federated Learning
Deep learning
IoT botnet
Intrusion detection

ABSTRACT

The rapid growth of Internet of Things(IoT) devices has increased their vulnerability to botnet attacks, posing serious network security challenges. While deep learning models within federated learning (FL) can detect such threats while preserving privacy, their black-box nature limits interpretability, crucial for trust in security systems. Integrating explainable AI (XAI) into FL is significantly challenging, as many XAI methods require access to client data to interpret the behaviour of the global model on the server side. In this study, we propose a Federated Learning of Explainable AI (FedXAI) framework for binary and multiclass classification (botnet type and attack type) to perform intrusion detection in IoT devices. We incorporate one of the widely known XAI methods, SHAP (SHapley Additive exPlanations), into the detection framework. Specifically, we propose a privacy-preserving method in which the server securely aggregates SHAP value-based explanations from local models on the client side to approximate explanations for the global model on the server, without accessing any client data. Our evaluation demonstrates that the securely aggregated client-side explanations closely approximate the global model explanations generated when the server has access to client data. Our FL framework utilises a long-short-term memory (LSTM) network in a horizontal FL setup with the FedAvg (federated averaging) aggregation algorithm, achieving high detection performance for botnet detection in all binary and multiclass classification tasks. Additionally, we evaluated post-hoc explanations for local models client-side using LIME (Local Interpretable Model-Agnostic Explanations), Integrated Gradients(IG), and SHAP, with SHAP performing better based on metrics like Faithfulness, Complexity, Monotonicity, and Robustness. This study demonstrates that it is possible to achieve a high-performing FL model that addresses both explainability and privacy in the same framework for intrusion detection in IoT networks.

1. Introduction

Intrusion detection systems (IDS) protect ICT systems by identifying security attacks and threats through network traffic monitoring and analysis. IDS approaches are commonly categorised as signature-based and anomaly-based systems [1]. Signature-based systems rely on pre-established network patterns and challenges to detect new attacks. In contrast, anomaly-based systems analyse specific network traffic features to recognise deviations from normal behaviour, demonstrating a potential attack. In recent years, significant advances have been made in machine learning (ML) techniques, such as neural networks and clustering for IDS [2,3].

In the IoT context, recent efforts have been made to leverage Deep Learning (DL) techniques and various neural networks to detect attacks. Even though ML can enhance IDSs, most ML-based IDS systems are centralised. In such setups, network data from multiple IoT devices is sent to a central server to train the ML model. This server can access all network traffic and local device data during training, increasing privacy issues [4]. For example, transferring sensitive and private data via wearable devices or eHealth systems raises concerns about data breaches [4]. Due to the increased risk of data being exchanged, it is crucial to implement effective data management and protect privacy vulnerabilities [5]. Federated learning (FL) is an advanced approach to collaborative learning that Google [6] proposed in 2016 to address the privacy issues of traditional centralised ML approaches. In FL, end devices, clients, or parties do not share their data. Instead, they share only partial updates of a global model aggregated by a central server, regarded as an aggregator or a coordinator. FL ensures user privacy by not sharing participated client device data with third parties. FL scenarios involve numerous IoT-based client devices with varying amounts of distributed data. In real-life situations, the data is often

* Corresponding author.
E-mail addresses: rajesh.kalakoti@taltech.ee (R. Kalakoti), sven.nomm@taltech.ee (S. Nõmm), hayretdin.bahsi@taltech.ee (H. Bahsi).

https://doi.org/10.1016/j.comnet.2025.111479
Received 21 November 2024; Received in revised form 21 March 2025; Accepted 13 June 2025
Available online 25 June 2025
1389-1286/© 2025 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

non-independent and identically distributed (non-iid) [7]. For example, in an IDS deployed on a network, some devices may have attack-related traffic (e.g., Dos or DDoS), while others have normal operation traffic. Numerous studies have explored FL-based IDS in IoT. Some surveys have offered a comprehensive overview of the open challenges in FL with IDS, including issues related to data privacy, communication overhead, model accuracy, and handling heterogeneous IoT devices [8–10].

Despite the promising results of ML and DL for intrusion detection, challenges such as data labelling availability, cost, and achieving transparency through explainability persist. These issues need to be addressed to adopt ML models practically in the cybersecurity domain. ML community is increasingly focused on explainable Artificial Intelligence (XAI) to interpret models and make predictions, addressing the black-box nature of ML and DL models. There is significant concern due to the inherent lack of interpretability of ML models. In the cyber security domain, security experts (e.g., SOC analysts) often struggle to trust the ML model output because they do not fully understand how the model makes decisions. Some studies have also adopted XAI methods to increase trust in IDS among security analysts [11,12]. Post hoc explainability methods like LIME and SHAP are famous for providing detailed explanations of individual instances and model outputs. They assign importance scores to features critical in model decisions, allowing experts to verify the logic of these features within the corresponding cyber-incident context. However, the quality of these explainability methods has raised concerns. These methods use additional tools (such as linear models and game theory) to determine importance scores, which may lead to explanation errors. Due to limited evaluation, many current XAI methods may produce unreliable explanations. As XAI-based solutions for IDSs increase, thorough evaluation of their explainability components becomes crucial. This evaluation ensures that XAI-based IDS can be deployed in real-world scenarios, contributing to understanding the model output. Designing an XAI system with robust qualitative and quantitative evaluation procedures for use in the real world is essential for effective adoption in IDSs. Furthermore, some review articles extensively discuss open questions and future research directions in IDS [13–16].

In an IDS operating within a federated learning (FL) setting, security experts analysing intrusions on IoT devices using server-side models still need to understand how the models arrive at their decisions. Thus, in such settings, achieving both privacy and explainability requirements remains a significant challenge. The integration of XAI methods into FL is an area that has yet to receive much attention and presents additional challenges due to the complex distributed nature of FL, where the model is trained across multiple IoT devices. Post-hoc XAI methods, such as feature importance explainers (e.g., LIME or SHAP) and model interpretability, typically need to access the complete training dataset or trained model parameters, which can pose a privacy risk. In addition, integrating XAI into FL-based IDS for IoT is further complicated by the distributed complexity of IoT networks and the heterogeneity of IoT devices. IoT networks are characterised by limited resources, variable network connectivity, and diverse data generation patterns, making post hoc XAI challenging due to their underlying assumptions about the data and model architecture. More data privacy and security laws, such as the General Data Protection Regulation (GDPR) [17] in the EU and China's Cybersecurity Law [18], were introduced to protect user privacy. Data cannot be shared or collected centrally from multiple parties due to strict regulations against leakage of private data. Using data from multiple parties to train ML models while complying with privacy regulations is an urgent issue.

To address the challenges mentioned above, our research emphasises the importance of incorporating explainability into FL-based IDS. This work aims to fill this gap by proposing a method to detect IoT botnet network traffic and explain the black-box nature of a Deep learning model-based IDS model within an FL setting for IoT devices. In particular, we propose a FEDXAI-based IDS framework that integrates XAI into FL. This framework aims to make both the server model and the participating client models transparent and explainable within an IDS deployed on IoT network devices in FL settings. We operated a Long-Short-Term Memory (LSTM) model trained in a horizontal federated learning (HFL) setting with the federated averaging algorithm (FedAvg) for binary and multiclass classification settings. This paper proposes an approach to explain predictions made by the global LSTM (Fedavg aggregation-based server model) by aggregating explanations from participating clients while preserving data privacy by preventing the server model from directly accessing client IoT data. To this end, our proposal involves the secure aggregation of SHAP (SHapley Additive ExPlanations) value explanations from individual IoT device-based client models to approximate the server model's explanations. We evaluated this approach by comparing securely aggregated client-based Shapley value explanations with server-based Shapley explanations generated under the assumption that the server has access to the data from participating IoT device clients.

We also advocate for integrating XAI evaluation as a critical criterion to build trust in client models within FEDXAI-based IDS in FL. Additionally, we used LIME, integrated gradients (IG), and SHAP as post hoc local explanation methods to provide insights into the behaviour of these participant-client models in FL. We assessed the quality of the local post hoc explanations produced by these explainers. To evaluate the explanations, we used four criteria: Faithfulness (how accurately the explanations reflect the actual behaviour of the models), Consistency (the similarity of explanations for similar inputs), Complexity (the clarity and understandability of the explanations) and Robustness (the stability of the explanations when the nearby output has some explanations). Our evaluation of local explanations for client models in FL has assessed their reliability and informativeness in explaining outcomes for FEDXAI-based IDS.

The contribution of this work can be summarised as follows.

1. We proposed a FedXAI-based framework for intrusion detection in IoT devices, utilising an LSTM model within an HFL setting. The framework employs the Federated Averaging (FedAvg) algorithm to perform both binary and multiclass IoT botnet classification. The LSTM model was evaluated using accuracy, precision, recall, and F1 score on both the client-side and the server-side models.
2. Our methodology securely aggregates SHAP (SHapley Additive ExPlanations) values from individual client models of IoT devices to approximate the explanations of the server model.
3. We rigorously evaluated the reliability of the securely aggregated client-based explanations by comparing them with server-based explanations generated under conditions where the server had direct access to the data from participating IoT device clients.
4. We have provided post hoc local explanations using LIME, IG and SHAP to explain the decisions made by client-side models in FL, specifically for IoT botnet detection. The evaluation includes quantitative metrics such as faithfulness, monotonicity, complexity, and sensitivity.
5. Our work identified the most influential network category-type features for both the server model and participating client models in FL settings, addressing binary and multiclass classification tasks for IoT devices.

The existing FL settings guarantee privacy in model training. However, post hoc explainability methods require the collection and transfer of client data to the server, which leads to leakage of client data. Existing studies do not consider the preservation of privacy while extending the capabilities of FL solutions with explainability functions. The unique contribution of this paper is the secure aggregation of explanations obtained from clients to explain a server model in an FL setting to eliminate client data sharing. The proposed solution effectively integrates transparency and privacy in FL settings, achieving both

objectives within a unified framework. Although our contribution can be applied to other domains, in this study, we addressed the detection of IoT botnets as a problem domain.

This paper is structured as follows. Section 2 provides an overview of FL and reviews the literature related to FL in Intrusion Detection Systems (IDS), as well as Explainable AI (XAI) in the context of IDS and FL of explainable AI in IDS. Section 3.1 describes the data set used in this study. Section 3 Methodology focuses on the FL approach for implementing explainable AI for IDS in IoT network devices. The results of the study are presented in Section 4, and finally, conclusions are drawn in Section 7.

2. Background

2.1. Background on federated learning

FL trains ML models on smart devices without sharing local data. Devices share parameters and gradients with a global model hosted on a server. The global model aggregates local model updates by averaging individual model parameters or gradients, allowing each model to learn collaboratively from the global model. This ensures that sensitive raw data remains securely stored on client devices.

Let N be the number of clients, denoted as $c_1, \ldots, c_N$. Each client c_i has a set of samples denoted by U_{c_i}, the feature space by X_{c_i}, the label space by Y_{c_i}, and the data set by $D_{c_i} = \{u_{c_i}^{(j)}, x_{c_i}^{(j)}, y_{c_i}^{(j)}\}_{j=1}^{|D_{c_i}|}$. Here, the data point $(u_{c_i}^{(j)}, x_{c_i}^{(j)}, y_{c_i}^{(j)})$ indicates that the client c_i owns the sample $u_{c_i}^{(j)}$ with features $x_{c_i}^{(j)}$ and a label $y_{c_i}^{(j)}$. For example, in IoT networks, U refers to the set of all IoT devices, X denotes features such as Network Traffic and Communication Protocols, and Y indicates whether an IoT device is infected with Malware (for example Dos and DDos) or not. Based on data partition (X, Y, U) among clients, FL can be categorised into three types by Yang et al. [19]:

- Horizontal Federated Learning (HFL): In HFL, clients share the same feature and label space ($X_i = X_j, Y_i = Y_j$) but have different sample spaces ($U_i \neq U_j$).
- Vertical Federated Learning (VFL): In VFL, some common data samples are shared among clients ($U_i \cap U_j \neq \emptyset$), but the space between features and labels differs ($X_i \neq X_j, Y_i \neq Y_j$).
- Federated Transfer Learning (FTL): FTL does not impose restrictions on the sample, feature, and label space, allowing arbitrary differences.

In addition, Federated learning can be categorised as cross-device FL (business-to-consumer) and cross-silo FL (business-to-business). Cross-device FL involves many participants with low computational resources, while cross-silo FL consists of companies or institutions with stronger communication and computational capacities. The client–server architecture is the most popular HFL architecture and the Federated Averaging (FedAvg) algorithm [20] is based on it. The optimisation objective of HFL is as follows:

$$\min_{\theta} L(\theta) = \sum_{i=1}^{N} L_{c_i}(\theta) = \sum_{i=1}^{N} \frac{1}{|D_{c_i}|} \sum_{j=1}^{D_{c_i}} l(x_{c_i}^{(j)}, y_{c_i}^{(j)}; \theta) \tag{1}$$

where θ represents the model parameters, $L(\theta)$ is the global optimisation objective. $L_{c_i}(\theta) = \frac{1}{|D_{c_i}|} \sum_{j=1}^{D_{c_i}} l(x_{c_i}^{(j)}, y_{c_i}^{(j)}; \theta)$ is the client's optimisation objective c_i based on its local data D_{c_i}, with $l(x, y; \theta)$ denoting the loss function, such as cross-entropy or mean squared loss. SGD (stochastic gradient descent) is commonly used to optimise Eq. (1). The HFL training process is as follows:

1. Each client samples a batch of data B, and calculates gradients $g_{c_i} = 1/|B| \nabla_\theta \sum_{x,y \in B} l(x, y, ; \theta)$ and sends g_{c_i} to a server.
2. The server aggregates the gradients (e.g. averaging $g = 1/N \sum_{i=1}^{N} g_{c_i}$) and then distributes the aggregated gradient to all clients.
3. Each client updates the model by gradient descent based on the aggregated gradient g.

The process is executed iteratively until the loss function $L(\theta)$ converges. After training finishes, all clients share the same model parameters θ.

HFL has a high communication overhead, especially in CDFL. FedAvg reduces communication costs by increasing local computation to converge the model faster in HFL. Instead of transmitting gradients for every batch, FedAvg transmits gradients for every $\mathcal{E}$ epoch. Assuming current model parameters $\theta^{(t)}$, batch size b, and learning rate η, the FedAvg algorithm follows these steps.

1. Each client divides the local data D_{c_i} into batches of size b.
2. For each batch B, the clients perform parameter updates $\theta_{c_i}^{(t)} \leftarrow \theta_{c_i}^{(t)} - \eta/b \sum_{x,y \in B} \nabla_\theta l(x, y; \theta_{c_i}^{(t)})$ until the local dataset D_{c_i} is iterated $\mathcal{E}$ epochs.
3. Each client sends the updated parameters $\theta_{c_i}^{t}$ to aggregation server.
4. The aggregation server computes a weighted average of the model parameters $\theta^{t+1} = \sum_{c_i=1}^{N} w_{c_i} \theta_{c_i}^{(t)}$, where w_{c_i} is calculated from the size of each local dataset
5. the aggregation server distributes $\theta^{(t+1)}$ to the clients.

2.2. Related work

FL has recently gained significant attention due to its strengths in IoT applications. Recent studies have applied FL to improve IDS. A paper [21] performed a comprehensive evaluation of various ML models, such as Random Forest (RF), Decision Trees (DT), Support Vector Machine (SVM), and Multilayer Perceptron (MLP), in federated settings using blockchain for the aggregation process, incorporating local training with IoT device data and the KDDCup99 dataset [22]. Another Paper [23] utilised the NSL-KDD dataset [22] and MLP as an ML model for an FL-based IDS system. The approach is based on mimic learning, where a student model is trained with a public data set labelled by a master model trained with sensitive data [10] uses neural networks to propose an FL-based IDS considering three scenarios based on different data distributions regarding the types of attack on the NSL-KDD data set. The use of neural networks is also proposed by [24], which integrates a differential privacy approach into a scenario with non-iid data using the CSE-CIC-IDS2018 dataset [25]. We have observed that some studies use their own generated or simulated datasets. For example, in paper [26], the authors combined an FL approach with fog computing to enable fog nodes to work together to detect DDoS attacks [26]. The authors used Gated Recursive Units (GRUs) with FedAvg as the aggregation algorithm. Based on GRU, a paper [27] proposes the creation of communication profiles for IoT devices to detect potential attacks. The data set is generated from real devices and involves Mirai botnet traffic. Li et al. did not provide performance details, such as the number of participants and the training rounds. Khoa et al. [28] proposes using deep belief networks (DBN) in IoT gateways to detect potential attacks on a specific IoT subnet and then combines the models through FL. The proposed method uses various datasets, including the N-BaIoT [29] dataset, which contains traffic data from IoT devices. However, the authors do not provide details on the implementation being used or evaluation details, such as data distribution, number of clients, or training rounds. [30] uses the n-BaIoT dataset for binary classification with supervised and unsupervised learning. It incorporates aggregation techniques from [31] and evaluates them against adversarial attacks. The authors balanced the data set to ensure equal sample numbers and class proportions for all devices. Our work is similar to this approach in the model creation

stage for FL-based iDS prior to post hoc model interpretation. However, we develop our study to include multiclass classification scenarios using the N-BaioT data set to detect multiple attack types rather than focus on binary classification. We also emphasise explaining the black-box model through post hoc XAI methods in the FL-based IDS approach.

Most XAI techniques in cybersecurity focus on visualisation and model verification. Significant advances in DL have led to pilot studies that analyse network traffic in cybersecurity. The black-box nature of most DL models makes it difficult for security experts (stakeholders) to understand their decisions, posing a significant challenge in security operations like ID. XAI can help clarify the prediction of the model, which is crucial when experts suspect misclassifications in IDS. Although traditional signature-based systems struggle with unknown attacks, they provide clear technical details [32]. In contrast, XAI-enhanced ML models offer a powerful and adaptive detection solution that improves over time without manual intervention, maintaining high performance.

Wang et al. proposed a framework that uses SHAP to offer local and global explanations of IDS, helping security analysts interpret predictions. They analysed two supervised models trained on the NSL-KDD dataset, demonstrating that different attack types produce unique SHAP value patterns. However, They use it only for visualisation and do not analyse the values for further information.

Antwarg et al. employ SHAP to explain anomalies detected by an unsupervised auto-encoder model, offering insights for domain experts. They identify features with high reconstruction errors and use SHAP for clarification. Their method is tested on the KDD Cup 1999 dataset and other datasets, with visualisations furnished to help in understanding and triaging anomalies.

Liu et al. present FAIXID, a framework that enhances the explainability of IDS by improving data cleaning, clarifying the internals of the supervised model, providing local prediction explanations, and providing customised visualisations for security analysts.

Rao and Mane use an isolation forest on the NSL-KDD dataset to classify samples as normal or anomalous, employing SHAP and LIME for explanations and automatically labelling attacks based on significant features.

Barnard et al. introduce a two-stage network intrusion detection system (IDS). The first stage trains a supervised XGBoost model for binary classification of network flow data, using SHAP for interpretation. The second stage uses these SHAP explanations as input for an autoencoder to differentiate between known and unknown behaviours. Their hypothesis tests whether the first stage can identify normal versus anomalous flows, while the second can handle behaviour classification. The evaluation is based on the NSL-KDD dataset. However, the second stage's reliance on labelled data and the lack of analysis on different attack behaviour clusters are noted limitations.

It is important to note that the above studies [33–37] present their contributions on an old network data set that does not reflect the properties of the IoT networks.

Kundu et al. performed experiments using synthetic and real network traffic from the IXAI breaking point system. They tested the 1DCNN model on three datasets: a synthetic dataset from IXAI, the Stratosphere IPS Project dataset, and the Kitsune dataset [39] focussing on botnet traffic. Their DCNN botnet detection models outperformed previous machine learning models, with up to a 15% improvement in classification metrics. Furthermore, SHAP was used to explain the decision-making of the model, enhancing trust among cybersecurity stakeholders. In a paper [40], authors used SHAP for global and local explanations for RF, Adaboost, SVM, and SNN models for malware detection over a malware dataset.

Few studies focus on explainability in cybersecurity within FL settings. In a paper [41] using random forests (RF) as surrogates for the supervised FL model. Each client trains an RF on its local data. When the FL model misclassifies a sample, they use the RF trees to calculate feature importance values to detect attacks against the FL training process. These explanations are client-specific and require labelled data, making global interpretation challenging. Their main focus is on detecting potential attacks rather than explaining predictions. [42] propose a FL based anomaly detection architecture for industrial control systems. They use SHAP for model interpretation and verification, providing visualisations for domain experts. However, the SHAP explainer itself is not trained in a federated way and requires background data samples for the establishment of the baseline. The authors do not clarify how this baseline is obtained, which is crucial given the distributed nature of the data in FL settings. Many studies propose requiring labelled data at certain stages, which may not be feasible in practice [41]. Baseline refers to reference input data (e.g., training data) or values used by SHAP and similar explainability methods to compute feature attributions. This baseline is a reference point for evaluating the contributions of features to the predictions of the model. Some studies apply XAI methods in FL primarily to detect adversarial attacks [41] or to validate trained models [42]. However, explainer models are not developed in a federated manner, which may require violating FL assumptions or using different explainers for each client, complicating interpretation across the network. None of this works with SHAP in FL. Current research using SHAP in FL lacks discussion on how to extract a baseline, which is essential for generating explanations. This baseline selection is critical, as the explanations depend on it [43,44]. A paper [45] uses SHAP and LIME in federated settings but does not address the privacy risks of explaining the global model on the server side.

Despite recent improvements, there are still gaps in the literature on IoT for XAI-based IDS. A key issue is the absence of standardised evaluation metrics to evaluate the effectiveness of XAI techniques in this field. Current research is primarily based on qualitative assessments. Developing benchmarks to measure specific aspects of explainability would enable more rigorous comparisons of XAI methodologies and help identify best practices. Although some initial steps have been taken to incorporate explainability into IoT IDS systems, a quantitative evaluation of XAI techniques in the cybersecurity literature is still lacking. This emphasises the importance of thoroughly evaluating the quality of the generated explanations, which is crucial to building confidence in the results produced by AI systems that follow XAI principles [11,12,33–36,38,40]. In this work, we emphasise the importance of evaluating the quality of explainability through quantitative measures, alongside assessing the detection performance of models in FL settings for client-side model explanations. We advocate for including this evaluation as a critical criterion when developing models that security experts can trust in FedXAI-based IDS. To the best of our knowledge, this work is the first to use explanations generated through XAI methods within an FL architecture for FEDXAI-based IDS. This approach provides context for securely explaining server-side and client-side FL models without compromising privacy regarding feature importance. Previous research has not adequately addressed the privacy risks associated with providing explainability for the global (server) model. Since post hoc xai methods rely on data points for explanations, the lack of privacy measures raises significant concerns about potential data exposure. Our study ensures explainability by using client-based explanations, eliminating the need to rely on client data. Unlike prior works that assume centralised access to feature attributions, we introduce a privacy-preserving approach for interpreting the global model in a federated setting. Our method demonstrates that post hoc explanations can be provided effectively without compromising privacy, ensuring a secure and explainable FL-based IDS.

3. Proposed methodology

Fig. 1 illustrates the FedXAI (Federated learning of ai) based IDS framework for providing Client–Server Model Explanations in IoT network devices. It presents the architecture of the proposed HFL approach

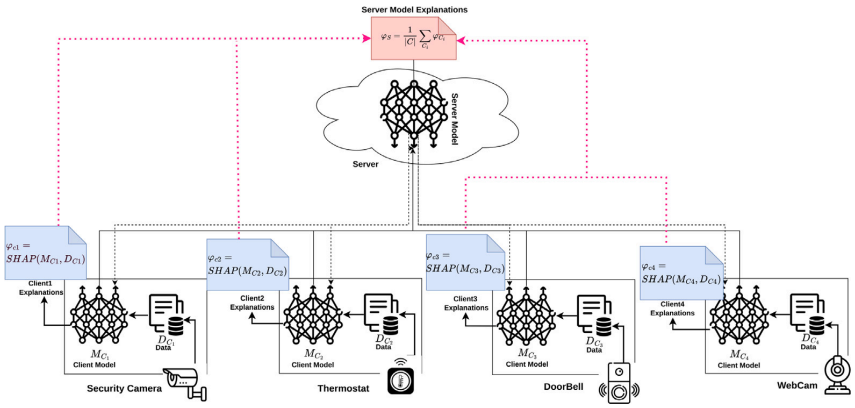


Fig. 1. FedXAI (Federated Learning of Explainable AI) framework for client–server intrusion detection model explanations in IoT network devices.

for IDS in IoT networks, where several devices are connected and distributed at various locations. Devices connect to gateways, and their network traffic is monitored with profiling tools (e.g., Wireshark) for analysis in the local host-based IDS on the IoT device. Each device has a ML model based on its network data. Selected IoT devices share their model parameters with a server, which aggregates them to create an enhanced intrusion detection model with optimised parameters. This collaborative approach improves learning by enabling devices to detect intrusions based on behaviour patterns learnt from other participating devices. After training the black-box model in FL, XAI provides transparency into how the IDS makes its decisions. XAI identifies key features, making the detection process more interpretable and trustworthy in FL. This allows security experts to validate the actions of the IDS and build user trust. To explain the FL server model, the server typically needs access to the data. This paper proposes a FedXAI-based IDS that explains the server model without sharing data, ensuring privacy while rendering explanations.

3.1. Dataset

N-BaIoT dataset [29] is suitable for evaluating privacy-preserving collaborative training in IoT botnet and B5G applications. Organise network traffic data into separate files, making distribution into non-identically distributed parts easier. However, it only includes data from nine IoT devices, which limits the number of available clients for experiments. This data set contains 115 network traffic features, extracted primarily through descriptive statistical analysis. Data were generated from network traffic produced by bots deployed in a controlled environment, using nine different IoT devices infected with Mirai and Gafgyt malware types. Mirai botnet was deployed on the Doorbell2 and Webcam IoT devices. Therefore, the data set contains no network traffic related to Doorbell2 and Mirai botnet Webcam. The features of each instance reflect the aggregated descriptive statistics of raw network traffic in five time windows (100 µs, 500 µs, 1.5 s, 10 s, and 1_{min}), denoted as L_5, L_3, L_1, $L_{0.1}$, and $L_{0.01}$, respectively (see Table 1). The data set encompasses five main feature categories: host-IP (coded as H), host-MAC and IP (coded as MI), channel (coded as HH), socket (coded as HpHp) and network jitter (coded as HH_jit). For each category, metrics such as packet count, mean packet size, and variance in packet sizes are calculated. In addition, statistical values, including the Pearson correlation coefficient (PCC) of packet size, radius, covariance, and magnitude, are derived for network channels and sockets, along with packet count, mean, and variance.

In this work, we have studied three types of classification: 1. Binary classification 2. Botnet-type detection 3. attack-type detection. Table 2 presents the class label distribution of the n-BaIoT data set for each IoT device and classification type. We developed three types of classification studies based on FL setup. For the binary classification type, the goal is to differentiate between malicious traffic and benign traffic. In botnet-type detection, we scrutinised the type of malware by classifying data points into three categories: Mirai, Gafgyt, and benign. Lastly, for attack-type detection, the data points were classified into different attack types, including ACK, benign, junk, scan, SYN, combo, TCP, UDP and plain UDP. Note that each client device has its own data D_{c_i} divided into training data $D_{c_i}^{\text{Train}}$ and testing data $D_{c_i}^{\text{Test}}$.

For data set D_{c_i} belonging to the client c_i, input $x_{c_i} \in \mathbb{R}^d$, where d is the dimensionality of the feature set, and the black box model $\mathcal{M}_{c_i}$ maps the input to an output $\mathcal{M}_{c_i}(x_{c_i}) \in \mathcal{Y}_{c_i}$. Denote $\mathcal{D}_{c_i} = \{(x_{c_i}^{(j)}, y_{c_i}^{(j)})\}$ as the collection of all input–output pairs for the client c_i. A post hoc explanation for the client c_i is $\mathbf{g_{c_i}}$, an explanation mapping that, for predictor $\mathcal{M}_{c_i}$ and point of interest x_{c_i}, returns an importance score $\mathbf{g}_{c_i}(\mathcal{M}_{c_i}, x_{c_i}) = \varphi_{x_{c_i}} \in \mathbb{R}^d$ for all features. Denote $\mathcal{D} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$ as a metric in the space of explanations and $S : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_{\geq 0}$ as a metric in the space of inputs. The evaluation criterion μ is a mapping that takes the predictor $\mathcal{M}_{c_i}$, explainer $\mathbf{g_{c_i}}$, and the point of interest $\mathbf{g_{c_i}}$ as arguments and returns a scalar value for $\mathbf{g_{c_i}}$.

3.2. Data prepossessing

Each client's data $\mathcal{D}_{c_i}$ was transformed and normalised using Min–Max feature scaling, standardised within the range $[0, 1]$ to enhance the accuracy and performance of the model. For each feature f where $f = 1, 2, \ldots, d$, the Min–Max normalisation is defined as:

$$x'^{(f)}_{c_i} = \frac{x^{(f)}_{c_i} - \min\left(x^{(f)}_{c_i}\right)}{\max\left(x^{(f)}_{c_i}\right) - \min\left(x^{(f)}_{c_i}\right)} \in \mathbb{R}^d$$

where $x^{(f)}_{c_i}$ is the actual value of the feature f for the client c_i, $\max\left(x^{(f)}_{c_i}\right)$ and $\min\left(x^{(f)}_{c_i}\right)$ are the maximum and minimum values of the feature f in the client data $\mathcal{D}_{c_i}$, respectively. The value $x'^{(f)}_{c_i}$ is the new normalised value for the feature f in the client data $\mathcal{D}_{c_i}$, which is in the range of 0 to 1.

3.3. Long short-term memory

To accurately classify network traffic in IoT edge devices, we propose a LSTM [46] model via Federated Averaging (FedAvg) in the HFL architecture for binary and multiclass classifications. LSTM is a neural network designed to overcome long-term dependency issues in traditional recurrent neural networks (RNN). It uses three types of gates

Table 1
Details of features of N-BaIoT dataset.

Feature category	Category code	No. Of features	Statistical value feature	Time frame windows
Hos Mac & IP	MI	15	Packet count, Mean Variance	100 μs
Host IP	H	15		500 μs
Network Jitter	HH_jit	15		1.5 s
Channel	HH	35	Packet count, Mean Variance, Magnitude, Radius, CoVariance, Correlation	10 s
Socket	HpHp	35		1 min

Table 2
Label distribution across different IoT devices for the classification type used in this study.

Device name	Classification type	Binary		Botnet-type detection			Attack type detection								
	Shortname	Malware	Benign	Mirai	Gafgyt	Benign	Udp	Scan	Syn	Ack	Tcp	Udpplain	Combo	Benign	Junk
Danmini Doorbell	Doorbell1	95.13%	4.87%	64.04%	31.10%	4.87%	33.74%	13.51%	12.04%	10.04%	9.05%	8.05%	5.86%	4.87%	2.85%
Philips B120N10 Baby_Monitor	BabyMonitor	84.05%	15.95%	55.59%	28.46%	15.95%	29.38%	11.97%	10.75%	8.29%	8.43%	7.36%	5.29%	15.95%	2.58%
Provision PT 737E Security Camera	SecurityCam1	92.50%	7.50%	52.64%	39.85%	7.50%	31.42%	15.22%	7.94%	7.31%	12.62%	6.84%	7.41%	7.50%	3.73%
SimpleHome XCS7 1003 WHT SecurityCamera	SecurityCam2	97.70%	2.30%	60.51%	37.19%	2.30%	30.57%	8.49%	14.40%	12.60%	11.53%	9.92%	6.98%	2.30%	3.22%
Ecobee Thermostat	Thermostat	98.43%	1.57%	61.27%	37.16%	1.57%	30.66%	8.46%	13.97%	13.55%	11.37%	10.45%	6.34%	1.57%	3.63%
Provision PT 838_Security_Camera	SecurityCam3	88.23%	11.77%	51.30%	36.93%	11.77%	31.46%	15.00%	7.39%	6.93%	10.68%	6.43%	6.87%	11.77%	3.47%
SimpleHome XCS7 1002 WHT Security Camera	SecurityCam4	94.60%	5.40%	59.47%	35.13%	5.40%	29.62%	8.55%	14.57%	12.92%	10.29%	9.07%	6.29%	5.40%	3.31%
Ennio Doorbell	Doorbell2	89.00%	11.00%		89.00%	11.00%	29.24%	7.91%			28.56%		14.91%	11.00%	8.38%
Samsung SNH 1011 N Webcam	Webcam	86.10%	13.90%		86.10%	13.90%	29.48%	7.38%			26.06%		15.64%	13.90%	7.54%

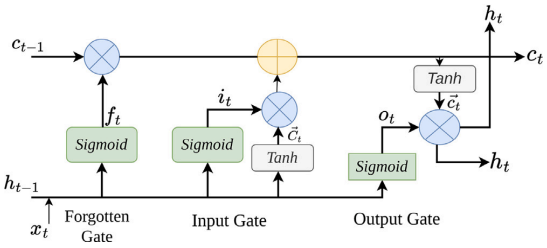


Fig. 2. Hidden layer architecture of LSTM network.

– forget, input, and output – to manage information flow and maintain long-term memory. The hidden layer structure of an LSTM network is shown in Fig. 2. The forget gate adapts to the context, discarding unnecessary information. It uses a sigmoid function to produce a value between 0 and 1, then multiplied by the last cell state. A value of 0 indicates complete forgetting, while a value of 1 signifies full retention.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

The input gate improves the necessary information for the new cell state, and its output is a sigmoid function with a range of 0 to 1, which is multiplied by the recent cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{4}$$

Then the old and new state information can be merged to construct the final new cell state.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{5}$$

The output is determined by the output gate, which uses a sigmoid function to select the information to be output, in conjunction with the final cell state and the Tanh function.

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = O_t \times \tanh(C_t) \tag{7}$$

3.4. Explainable AI methods

DL models are considered complex and operate as black-box models, making their predictions difficult to interpret. Explainable AI provides insight into these predictions, promoting transparency and trust. XAI methods can be categorised as global explanations, which explain overall model behaviour, and local explanations, which explain individual data point predictions. They can also be model-agnostic, applicable to various models, or model-specific, tailored to specific architectures.

In our research, we focus on post hoc local explanation methods like SHAP, Integrated Gradients (IG), and LIME to analyse IoT botnet predictions from an LSTM model in a FL environment. We use SHAP for server-side explanations and SHAP, LIME, and IG for client-side explanations. These techniques are vital for understanding the contributions of the features and generating interpretable alerts in FL-based IDSs for IoT networks.

3.4.1. LIME (Local Interpretable Model-Agnostic Explanations)

LIME is a XAI method that provides interpretable local explanations for black-box models by approximating the model's behaviour in the local region around a specific instance [47]. Given an instance $x \in \mathbb{R}^d$ and an explanation of the model $g \in \mathcal{G}$ where $\mathcal{G}$ is a set of interpretable models (e.g.: Linear models). provides explanations $\varphi(x)$ obtained by:

$$\underset{g \in \mathcal{G}}{\operatorname{argmin}} \{\mathcal{L}(\mathcal{M}, g, \omega_x) + \Omega(g)\} \tag{8}$$

In the case of a classification model $\mathcal{M}$, ω_x is a proximity measure or weight between the actual instance and the new instances. A higher value of ω_x signifies a stronger similarity between the new and original instances. $\mathcal{L}$ is a loss function used to measure the proximity between

the predictions made by the explanation model and the original model and $\Omega(g)$ quantifies the complexity of model g. Hence, LIME minimises $\mathcal{L}(\mathcal{M}, g, \omega_x)+\Omega(g)$ to create a locally interpretable model, which is then used to predict the instance via the explanation model $\omega(x)$.

3.4.2. SHAP (SHapley Additive exPlanations)

This method is commonly used to interpret the output of the ML model [43]. It uses Shapley values from cooperative game theory [48] to explain each feature's contribution to predictions [49]. SHAP is often used with tabular data and has key properties such as local accuracy, missing data handling, and consistency [50]. Local accuracy ensures that the explanation model matches the original model. Missingness ensures that missing features in the original input have no impact. Consistency means that increasing the impact of a feature in the model should lead to a higher SHAP value for that feature, regardless of the others.

SHAP creates simplified inputs z by mapping x to z through $x = h_x(z)$. The original model $\mathcal{M}(x)$ can be approximated using binary variables with a linear function:

$$g(z) = \varphi_0 + \sum_{i=1}^{d} \varphi_i Z_i \tag{9}$$

where $z = \{0,1\}^d$, d represents the number of input features, $\varphi_0 = \mathcal{M}(h_x(0))$, and φ_i denotes the feature attribution value

$$\varphi_i = \sum_{S \in F \setminus \{i\}} \frac{|S|!(d-|S|!-1)!}{|d|!} [\mathcal{M}_x(S \cup i) - \mathcal{M}(S)] \tag{10}$$

$$\mathcal{M}_x(S) = \mathcal{M}(h_x^{-1}(z)) = E[[\mathcal{M}(x)]|x_s] \tag{11}$$

where F is the nonzero input set in z, S is the subset of F that excludes the characteristic ith, and φ_i the SHAP value, a unified measure of the characteristics.

In our study, we employed SHAP's DeepExlainer. DeepExplainers of SHAP takes a trained deep learning model as input, along with reference input data (training data). Note that in our federated setup, for each client c_i, SHAP's DeepExplainer takes the training dataset $D_{c_i}^{\text{Train}}$ and the trained model $\mathcal{M}_{c_i}$ to compute SHAP values for the test data $D_{c_i}^{\text{Test}} \in \mathbb{R}^{m \times d}$, where m represents the number of instances and d the number of features. SHAP explainer thus returns the $\mathbb{R}^{m \times d}$ matrix of SHAP values for client c_i, where each value represents the contribution of a feature to the prediction for each instance.

3.4.3. Integrated gradients (IG)

Another XAI method we used in this work is Integrated Gradients (IG) [51], which determine the relevance R_i of the input variable x_i by approximating the integral below using the Riemann sum.

$$R_i = (x_i - \bar{x}_i) \cdot \int_0^1 \frac{\partial f_c\left(\bar{\mathbf{x}} + \alpha \cdot (\mathbf{x} - \bar{\mathbf{x}})\right)}{\partial x_i} \, d\alpha, \tag{12}$$

The variable $\bar{x}_i$ represents a baseline input that must be chosen when applying this method. The authors define this baseline input as indicating the absence of a feature input x [51]. The original authors used a zero-value baseline for the image. In our experiments, we also used a zero-valued baseline for the tabular data set.

3.5. Explaining server-side black box model in FL

In our study, we used SHAP Explainer to explain the long-short-term memory (LSTM) model trained through Federated Averaging (FedAvg) in the HFL architecture described in Section 2.1. SHAP explainer requires either access to the training data or a reference set in order to create records for analysing the impact of each feature value on the model's final prediction. For server-based learning (FL), the server requires access to the complete set of training data from the clients or needs to compute the centroids [52] of the data set resulting from the union of the training sets of all the clients to explain the learnt server model. In our research, we adopted the approach outlined in [53], where an explanation for a server model was obtained by aggregating all client explanations, thus avoiding the need for data sharing with the server model.

After completion of FL training, each client c_i receives its trained model $\mathcal{M}_{c_i}$, which is then sent to the server. The server then creates its own model $\mathcal{M}_s$ by averaging the weights of the received models using the FedAvg algorithm.

Each client of SHAP explainer returns $\varphi_{c_i}(x_j)$ SHAP values related to client c_i to the feature x_j. Each client c_i can then obtain SHAP of its data points in their model $\mathcal{M}_{c_i}$, which is strongly dependent on the training data. To derive explanations from the server's perspective, the additive nature of SHAP values was leveraged, which enables the creation of explanations for model $\mathcal{M}_s$ as an aggregation of explanations from the individual models $\mathcal{M}_c$ within U. Therefore, for a point of interest x, an explanation of the prediction performed by the server model $\mathcal{M}_s$ is obtained by the following equation.

$$\varphi_s(x) = \frac{1}{|N|} \sum_{c_i \in N} \varphi_{c_i}(x_j) \tag{13}$$

However, directly sharing the SHAP values generated by the clients with the server may violate privacy, as the server can learn excessive knowledge about the workings of the client models. This issue arises from significant vulnerabilities in post hoc explanation techniques that can be exploited by an adversary to generate classifiers whose post hoc explanations can be arbitrarily controlled [54], specifically, an attacker can deceive target classifiers and explainable methods while keeping the classifier's output consistent [55]. Thus, to ensure the privacy of SHAP explanations of client models, we employ a secure multiparty computation (SMPC) protocol based on Secret Sharing, as proposed by Google [56]. This protocol allows for the secure aggregation of SHAP values from client models without revealing individual client SHAP values to server or other clients. The protocol assumes N clients and one server. SMPC is a cryptographic protocol that enables N parties (e.g., clients and a server) to compute an aggregate function $f(x_1, x_2, \ldots, x_N)$ using their private inputs $x_1, x_2, \ldots, x_N$ without revealing them. This ensures the privacy of client data while allowing secure aggregation. In our work, we have used SMPC to securely aggregate SHAP values from client-side models. The protocol ensures that the server can compute the aggregated SHAP values $\sum_{c_i \in N} \varphi_{c_i}$ without knowing the individual client SHAP values φ_{c_i} from each client. The algorithm 1 demonstrates the steps involved in the secure aggregation of SHAP values based on clients using a multiparty secure computation. Using this secure aggregation method, we ensure that the server can obtain explanations for its model $\mathcal{M}_s$ while preserving the privacy of individual clients' SHAP values.

To check if the secure aggregation of client-side SHAP values E_{local} is sufficient for the server model, the server model $\mathcal{M}_s$ was provided with data from all clients N to obtain explanations. These server-based SHAP values were then aggregated for all clients by supplying the combined training data to the SHAP explainer, resulting in the global explanation E_{global}. Finally, to verify the correctness of the explanations, the difference d_{g-l} between E_{global} and E_{local} was calculated. i.e.,

$$d_{(g-l)} = E_{global} - E_{local}. \tag{14}$$

The difference between E_{global} and E_{local} indicates the divergence between server-based SHAP values (obtained by directly accessing clients' data) and secure aggregation from individual clients (obtained by aggregating SHAP values without sharing their data with the server). This divergence calculation helps evaluate the quality and reliability of the aggregation of clients' explanations (E_{local}) in comparison to the server-based explanations (E_{global}).

Note that We have computed the SHAP values with the same dimensions for test samples from each client. For secure aggregation of client-based SHAP values (E_{local}), each client c_i with test data $D_{c_i}^{\text{Test}} \in$

Algorithm 1 Secure Aggregation of Client-side SHAP Values

Input: SHAP values $\varphi_{c_i} \in \mathbb{R}^d$ for each client c_i, number of clients N, large prime number P
Output: Secure Aggregated SHAP values $\sum_{i=1}^{N} \varphi_{c_i}$
Step 1: Secret Sharing
for each client c_i **do**
  Split φ_{c_i} into N shares using a (t, N) threshold secret sharing scheme
  Distribute shares to all other clients and the server
end for
Step 2: Masking
for each client pair (i, j), $i \neq j$ **do**
  Generate random masking vector $S_{i,j}$
end for
for each client c_i **do**
  Compute masked SHAP value:
  $z_i = \varphi_{c_i} + \sum_{j>i} S_{i,j} - \sum_{j<i} S_{j,i} \mod P$
  Send z_i to the server
end for
Step 3: Aggregation at Server
Compute aggregated value:
$\sum_{i=1}^{N} z_i = \sum_{i=1}^{N} \left(\varphi_{c_i} + \sum_{j>i} S_{i,j} - \sum_{j<i} S_{j,i} \right) \mod P$
Simplify to: $\sum_{i=1}^{N} z_i = \sum_{i=1}^{N} \varphi_{c_i} \mod P$
Output: Secure Aggregated SHAP values $\sum_{i=1}^{N} \varphi_{c_i}$

$\mathbb{R}^{m \times d}$ has SHAP values generated by the explainer that have the same dimensions, specifically $\mathbb{R}^{m \times d}$. Consequently, after performing the secure aggregation of SHAP values on the client side (E_{local}), the resulting aggregated SHAP values also maintain the same dimensions of $\mathbb{R}^{m \times d}$. Similarly, when the server model accesses client test data, the server-based explanations (E_{global}) are also in $\mathbb{R}^{m \times d}$. The difference $d_{(g-l)}$ between E_{local} and E_{global} is calculated as the element-wise difference, resulting in a matrix of the same size, $\mathbb{R}^{m \times d}$.

3.6. Explaining client-side black box models in FL

In FL setup, each client owns their data. We have provided local explanations for each client using LIME and SHAP. These explainers utilise each client's data and trained model to provide post-hoc local explanations for every client model. Post-hoc local explanations provide contributions of individual features to a model's prediction for a specific instance. Due to the growing need for objective evaluation methods, XAI research field has shifted towards developing quantitative metrics to assess the quality and reliability of XAI method. XAI evaluation is categorised into three groups: user-focused, application-focused, and functionality-focused evaluations. The initial two categories are components of human-centred evaluation, divided into subjective and objective measures. In this work, We employed four key metrics for the local explanations of LIME and SHAP: high faithfulness, monotonicity, low complexity, and maximum sensitivity for evaluating quality of each client model's post hoc local explanations. Additionally, SHAP was used to provide global explanations for each client model, explaining how each feature contributes to the model's prediction.

3.6.1. High faithfulness

The explanation method $\mathbf{g}$ should replicate the model's behaviour. $\mathbf{g}(\mathcal{M}, \mathbf{x}) \approx \mathcal{M}(\mathbf{x})$. Faithfulness quantifies the consistency between the prediction model $\mathcal{M}$ and explanation g. For evaluating the Faithfulness of explanations, High Faithfulness correlation [57] and Monotonocity [58] metrics were used. Faithfulness Correlation is defined as

$$\mu_F(\mathcal{M}, g; x) = \underset{\mathcal{B} \in \binom{|d|}{|\mathcal{B}|}}{\text{corr}} \left(\sum_{i \in \mathcal{B}} g(\mathcal{M}, x)_i, \mathcal{M}(x) - \mathcal{M}(x_{\mathcal{B}}) \right) \tag{15}$$

where $x_{\mathcal{B}} = x_i | i \in \mathcal{B}\}$. High Faithfulness correlation metric iteratively substitutes a random subset of given attributions with a baseline value $\mathcal{B}$. Then, it measures the correlation between the sum of these attributions and the difference in the model's output.

3.6.2. Monotonocity

Let $x, x' \in \mathbb{R}^d$ be two input points where $x_i \leq x'_i$ for all i. A function $\mathcal{M}$ and a mapping g are monotonic [58] if, for any subset $S \subseteq \{1, 2, \ldots, d\}$, the sum of the attributions for features in S is nonnegative when moving from x to x'.

$$\sum_{i \in S} g(\mathcal{M}, x)i \leq \sum_{i \in S} g(\mathcal{M}, x')_i$$

imply

$$\mathcal{M}(x) - \mathcal{M}(x_{[x_s = \bar{x}_s]}) \leq \mathcal{M}(x') - \mathcal{M}(x'[x'_s = \bar{x}_s])$$

3.6.3. Complexity

For evaluating the complexity of explanations, Low complexity was used [57]. Explanations using a smaller number of features are preferred. so $\min \|\mathbf{g}(\mathcal{M}, \mathbf{x})\|_0$. Low complexity metric computes the entropy of each feature's fractional contribution to the total attribution magnitude individually. Given a prediction $\mathcal{M}(x)$, an explanation function g, and a point x, the complexity of g at x is:

$$\mu_C(\mathcal{M}, g; x) = -\sum_{i=1}^{d} P_g(i) \log P_g(i) \tag{16}$$

where,

$$P_g(i) = \frac{|g(\mathcal{M}, x)i|}{\sum j \in |d||g(\mathcal{M}, x)_j|}; P_g = P_g(1), \ldots . P_g(d) \tag{17}$$

P_g is a valid probability distribution, with $P_g(i)$ representing the fractional contribution of characteristic x_i to total attribution. If all features had equal contributions, the explanation would be complex, despite being faithful. The simplest explanation, however, would focus on one feature, with complexity defined as the entropy of P_g.

3.6.4. Max sensitivity

Robustness refers to similar inputs that should result in similar explanations. $\mathbf{g}(\mathcal{M}, \mathbf{x}) \approx \mathbf{g}(\mathcal{M}, \mathbf{x} + \epsilon)$ for small ϵ. Max sensitivity [57] ensures that nearby inputs with similar model outputs receive comparable explanations. For the explanation function g to have low sensitivity around the point of interest x, assuming the predictor function $\mathcal{M}$ is differentiable, we define a neighbourhood N_r around x as $N_r = \{z \in D_x \mid p(x, z) \leq r, \mathcal{M}(x) = \mathcal{M}(z)\}$, where D is the distance metric and p is the proximity function. Thus, the maximum sensitivity of g at point x can be defined

$$\mu_M(\mathcal{M}(x), g, r; x) = \max_{z \in N_r} D\left(g(\mathcal{M}(x), x), g(\mathcal{M}(x), z)\right) \tag{18}$$

3.7. Experimental settings

In federated learning, an LSTM model was trained using data from multiple IoT devices, with a fixed number of samples from each device. This approach minimised the impact of varying training instances and ensured a consistent dataset size across all clients while maintaining the same class proportions. The number of communication rounds for training the model was determined through iterative experimentation, starting with random initial rounds and adjusting based on observed model performance. For example, binary classification converged within 10 rounds, while 50 rounds were sufficient for botnet-type detection, and 2000 rounds were tested for attack-type classification. Each client's data is split into training and validation sets using an 80:20 ratio. During each federated training round, validation data from each client was tested exclusively on its respective client model. After processing all clients, the combined validation data from all clients was tested on the

server model. We applied min–max normalisation as a preprocessing step to scale the data to a specific range. This technique improves LSTM models' ability to capture patterns and relationships while enabling faster training convergence in resource-intensive FL settings of IoT environments.

In the evaluation of LSTM learning models in FL, we utilised a validation dataset exclusively on IoT devices. This study employed metrics Accuracy, Precision, Recall, and F1-score for evaluating both the client and server models in FL across all classification settings.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{19}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{20}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{21}$$

$$\text{F1-Score} = \frac{2 \times TP}{(2 \times TP) + FP + FN} \tag{22}$$

From the N-BaIoT dataset, where true positive (TP) is the number of network traffic statistical feature data points in the positive class that are correctly classified as positive, false positive (FP) is the number of network traffic statistical feature data points in the negative class that are misclassified as positive; true negative (TN) is the number of data points in the negative class that are correctly classified as negative, and false negative (FN) is the number of network traffic statistical feature data points in the positive class that are misclassified as negative.

Experiments were conducted on a server equipped with an AMD Threadripper 3960X processor, which features 24 cores and 48 threads. The server includes 128 GB of RAM and an NVidia 3090 GPU with 24 GB of graphics memory, and it runs on Ubuntu 20.04 LTS in HPC (High-Performance Computing Centre) cluster node using Slurm workload manager. The experiments were implemented in Python 3.9, using Flower [59] for FL with PyTorch.

4. Results

This section presents the results of our study, which are organised into three main parts. Section 4.1 discusses the performance evaluation of the LSTM model across three types of detection in FL. Section 4.2 provides the results for the explanations of the server-side LSTM model, while Section 4.3 presents the results for the explanations of the client-side LSTM model.

4.1. Performance evaluation of LSTM model

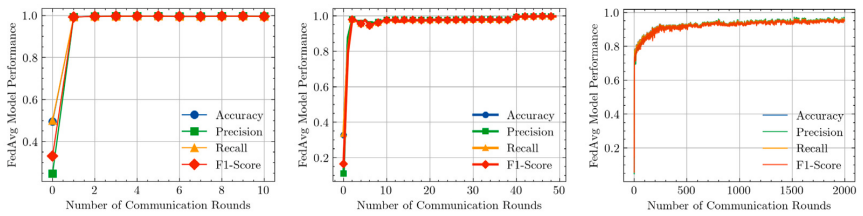
In this work, our proposal uses explainable AI to explain LSTM (Long Short-Term Memory) model learned by the FedAvg algorithm within a HFL architecture for binary and multiclass classification scenarios, specifically focusing on IoT Malware, Botnet-type, and Attack-type detection. In this study, we used the N-BaIoT dataset, which contains data from 9 IoT devices (Doorbell1, BabyMonitor, SecurityCam1, SecurityCam2, Thermostat, SecurityCam3, SecurityCam4, Doorbell2, and Webcam) infected by botnets such as Gafgyt and Mirai. Binary classification focuses on malware detection (Malware and Benign), while botnet-type (Mirai, Gafgyt, and Benign) and attack-type detection (ACK, COMBO, JUNK, SCAN, SYN, TCP, UDP, PLAIN, and Benign) were considered multiclass classification. Botnet-type detection identifies the type of malware that infects the IoT device. This detection endeavor is usually required in the incident-handling processes of organisations hosting the bots so that they can apply relevant removal procedures based on the malware type. Attack-type detection identifies the kinds of attacks launched by the bots. The organisations that are targeted by these intrusions can give more informed decisions when their security analysts know more details about the attack.

Classification performance metrics of client models the results from their evaluation with their own parameters on each client's validation data. To evaluate the FedAvg algorithm on Server side, we used test data from each IoT device-based client exclusively for its own model evaluation. In each round of training and evaluation in the federated training, 50% of clients were randomly selected to ensure diverse client updates and a more robust measure of the model's generalisation ability. All 9 IoT device-based clients participated in the evaluation round to provide a reliable estimate of the model's performance for binary classification. However, in the multiclassification setting, only 7 clients participated, as the Mirai botnet did not infect the 'Doorbell2' and 'Webcam' devices. Therefore, we excluded these two devices from the experiments for Botnet-type and Attack-type detections to maintain consistency in input feature space and label space during federated training using the FedAvg algorithm due to the horizontal nature of the FL approach. Metrics considered, including accuracy, precision, recall, and F1-score, comprehensively evaluate LSTM model in HFL setting for the FedAvg algorithm. Client models' classification performance metrics were evaluated with their own parameters on each client's validation data.

Fig. 3 illustrates LSTM model performance metrics on the server side over multiple communication rounds within an HFL setting using the FedAvg algorithm for all three classification types: binary classification, botnet-type detection and attack-type detection. Server-side LSTM model performance metrics were evaluated starting from the initial round (0th) communication round. This initial round represents the server-side LSTM model performance before client communication or parameter updates. By including the baseline performance, it can be observed how the global model improves as it includes updates from client models during the subsequent communication rounds. It can be seen in Fig. 3(a) that the LSTM model using FedAvg achieved early convergence with fewer communications for binary classification. All performance metrics (accuracy, precision, recall, and F1-score) quickly reached optimal performance with minimal communication overhead. Client-side LSTM models were evaluated using IoT device-specific data after each training communication round. For binary classification in Fig. 4, all LSTM models on device-specific data achieved almost 99% performance with metrics such as accuracy (in Fig. 4(a)), precision (in Fig. 4(b)), Recall (in Fig. 4(c)), and F-score (in Fig. 4(d)) within a few rounds, demonstrating quick convergence. The shaded areas in Fig. 4 illustrate the confidence intervals around the classification metrics (Accuracy, precision, recall, F1-score). We used fixed probabilistic error rates to estimate potential variability in the classification metrics. Error rates were used to estimate confidence intervals: Lower Bound = metric - (metric × error percentage). Upper Bound = metric + (metric × error percentage). For binary classification, the error rate was set to 0.05%.

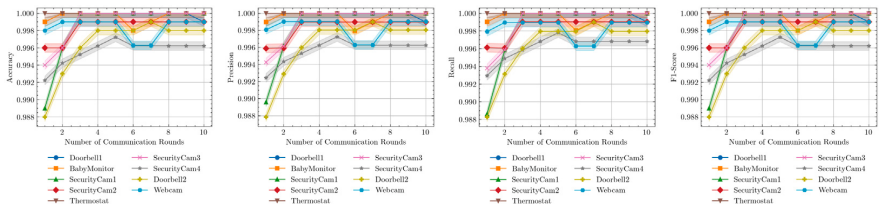
Similar to binary detection, the convergence for botnet-type detection occurred after 2 communication rounds. Fig. 3(b) demonstrates the server-side evaluation of the LSTM model's performance in detecting botnet types in IoT devices. The figure shows rapid improvement in all metrics, stabilising above 0.95 after 2 communication rounds. By the end of 50 communication rounds, the LSTM model converges close to 1 for the server-side evaluation performance of the Fedavg algorithm for Botnet-type detection. Fig. 5 presents the performance evaluation results of the client model, where the error rate was set to 0.01%. Evaluation metrics include accuracy, precision, recall, and F1-score for individual IoT devices participating in FL for Botnet-type detection. The results indicate consistently high performance across the devices. The thermostat device model exhibited more variability in the initial rounds but stabilised after 15 rounds of communication. In contrast, the BabyMonitor device model started with lower performance metrics but gradually improved to exceed 0.99 after 50 communication rounds.

Unlike Binary classification and Botnet-type detection, substantially more communication rounds (2000) were required to achieve optimal performance in attack-type detection. Metric Results for the attack type of the LSTM model were presented in Figs. 3(c) and 6 for server-side and client-side evaluation, respectively. Fig. 3(c) demonstrates the evaluation metrics of the LSTM model using FedAvg, including accuracy,



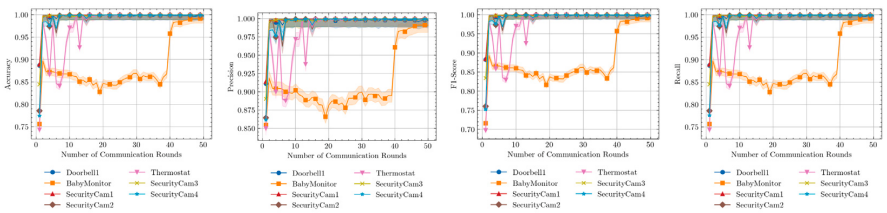
(a) Malware Detection (Binary) (b) Botnet-type Detection (c) Attack-type Detection

Fig. 3. Performance evaluation of FedAvg LSTM model on the server side across different detection tasks: (a) Malware detection (Binary), (b) Botnet-type detection, and (c) Attack-type detection.



(a) Accuracy (b) Precision (c) Recall (d) F1-score

Fig. 4. Device-specific client-side evaluation of LSTM model performance for binary classification.



(a) Accuracy (b) Precision (c) Recall (d) F1-score

Fig. 5. Device-specific client-side evaluation of LSTM model performance for botnet-type detection.

precision, recall, and F1 score. Model's performance initially improved quickly but stabilised after about 1500 communication rounds, demonstrating that additional rounds were needed for the model's consistent performance. Fig. 6 presents metrics results for client-side LSTM models across all seven participating IoT devices over 2000 communication rounds with an Error rate of 0.02%. The performance of individual IoT device-based LSTM models demonstrated significant variations. LSTM models on the client side, including Doorbell1, BabyMonitor, SecurityCam1, SecurityCam2, and SecurityCam3, illustrated constant performance above 0.95 in performance metrics after 500 communication rounds. However, for some device LSTM models, like the Thermostat, showed significant fluctuations in performance throughout the communication rounds, ranging between 0.85 and 0.95. Additionally, the SecurityCam4 model initially demonstrated lower performance during the early communication rounds but gradually improved, stabilising at higher performance levels around the 1000th round. These variations demonstrate the complexities of detecting attack types and emphasise the necessity for additional communication rounds for some devices to reach convergence.

We also evaluate the class-label-wise classification performance of the LSTM model on all client IoT-edge devices for all the classification types for each communication round. Table 3 shows the classification performance report from the final communication round for binary classification (10th), Botnet Type detection (50th), and attack type detection (200th).

In Table 3, results show that LSTM model consistently achieved high performance across all IoT client devices for both binary and multiclass classification tasks. LSTM model achieved close to 100% accuracy, precision, recall, and F1-score for the binary classification across all client devices for the "benign" and "malware" labels. For Botnet-type detection, the LSTM model also strongly achieved performance metrics 99% for labels (Benign, Gafgt, and Mirai).

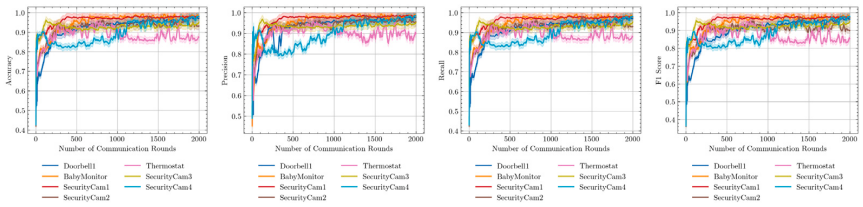
However, there is a variation in performance across different attack-type detection labels in IoT client devices. LSTM model has achieved high accuracy, f1-score, recall, and precision for attack labels such as Scan, SYN, UDPplain, and Benign. However, for labels like "ACK", "TCP", and "JUNK", the LSTM model showed some variations in performance across IoT devices. Devices like BabyMonitor and SecurityCam achieved over 90% performance, while Doorbell1 devices achieved lower performance, particularly for "ACK" and "TCP" labels, with less than 86% accuracy. For specific labels like UDP and Combo, the LSTM model achieved slightly lower performance, with Doorbell1 device achieving as low as 72% accuracy. Overall, the model performed well for the classification task of attack-type detection.

4.2. Explaining server-side LSTM model in FL

As mentioned in Section 3.5, SHAP was used to explain the LSTM model learned by the FedAvg algorithm in the case of HFL architecture to make the server model more transparent and explainable on the

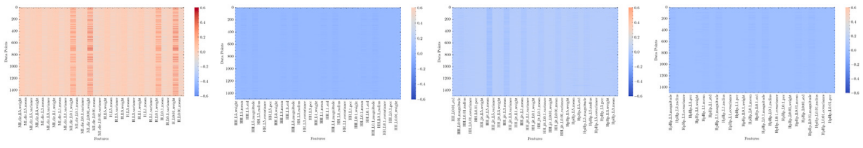
Table 3
Classification report (Accuracy (Acc), Precision (P), Recall (R), F1-Score (F1)) of client-side models from the last communication round for binary, botnet-type, and attack-type detection.

Classification type		Doorbell1				BabyMonitor				SecurityCam1				SecurityCam2				Thermostat				SecurityCam3				SecurityCam4				Doorbell2				Webcam			
		acc	P	R	F1	acc	P	R	F1	acc	P	R	F1	acc	P	R	F1	acc	P	R	F1	acc	P	R	F1	acc	P	R	F1	acc	P	R	F1	acc	P	R	F1
Binary	Benign	1.000	0.998	1.000	0.999	1.000	0.998	1.000	0.999	1.000	0.994	1.000	0.997	0.998	0.998	0.998	0.998	0.998	0.998	0.998	0.998	0.998	0.996	0.998	0.997	0.996	0.998	0.996	0.997	0.994	1.000	0.994	0.997	1.000	0.996	1.000	0.998
	Malware	0.998	1.000	0.998	0.999	0.998	1.000	0.998	0.999	0.994	1.000	0.994	0.997	0.998	0.998	0.998	0.998	0.998	0.998	0.998	0.998	0.996	0.998	0.996	0.997	0.998	0.996	0.998	0.997	1.000	0.994	1.000	0.997	0.996	1.000	0.996	0.998
Botnet type	Benign	0.996	1.000	0.996	0.998	0.996	0.998	0.996	0.997	1.000	0.998	1.000	0.999	0.990	0.996	0.990	0.993	0.996	0.996	0.996	0.996	1.000	0.996	1.000	0.998	0.994	0.996	0.994	0.995								
	Gafgyt	1.000	0.986	1.000	0.988	0.984	0.992	0.984	0.988	0.996	0.992	0.996	0.994	0.990	0.964	0.990	0.979	0.970	0.990	0.980	0.985	0.990	0.987	0.990	0.983	0.988	0.982	0.988	0.985	–				–			
	Mirai	0.990	1.000	0.990	0.990	0.996	0.986	0.996	0.991	0.990	0.996	0.990	0.993	0.982	0.993	0.982	0.980	0.984	0.974	0.984	0.979	0.985	0.994	0.985	0.974	0.984	0.988	0.984	0.986								
Attack type	ack	0.863	0.856	0.863	0.859	0.919	0.920	0.919	0.953	0.959	0.906	0.959	0.932	0.913	0.814	0.913	0.891	0.839	0.850	0.839	0.845	0.808	0.950	0.808	0.809	0.947	0.928	0.947	0.920								
	benign	0.996	0.994	0.996	0.995	0.994	0.998	0.994	0.996	1.000	0.994	1.000	0.997	0.994	1.000	0.994	0.997	1.000	0.998	1.000	0.999	0.996	0.998	0.996	0.997	0.996	0.992	0.996	0.994								
	combo	0.726	0.786	0.726	0.754	0.837	1.000	0.837	0.961	0.828	0.809	0.828	0.819	0.898	0.953	0.898	0.952	0.788	0.981	0.788	0.948	0.899	0.887	0.899	0.893	0.837	0.902	0.884	0.816								
	junk	0.720	0.767	0.720	0.793	0.992	0.781	0.992	0.808	0.931	0.860	0.931	0.884	0.972	0.853	0.972	0.875	0.992	0.846	0.992	0.884	0.859	0.870	0.859	0.865	0.912	0.871	0.912	0.934								
	scan	0.986	0.998	0.986	0.992	0.994	0.998	0.994	0.996	0.980	0.990	0.980	0.992	0.998	0.996	0.998	0.997	0.994	1.000	0.994	0.997	0.996	0.993	0.996	0.995	0.990	0.998	0.990	0.994	–				–			
	syn	1.000	0.990	1.000	0.995	1.000	0.990	1.000	0.995	0.918	0.975	0.918	0.946	0.972	0.990	0.990	0.990	0.997	0.993	0.997	0.995	0.991	1.000	0.991	0.981	1.000	0.988	1.000	0.994								
	tcp	1.000	0.882	1.000	0.897	1.000	0.869	1.000	0.869	1.000	0.911	1.000	0.944	0.998	0.809	0.998	0.829	0.998	0.818	0.998	0.835	1.000	0.803	1.000	0.832	0.998	0.857	0.998	0.839								
	udp	0.812	0.917	0.812	0.913	0.864	0.869	0.864	0.869	0.889	0.939	0.889	0.912	0.621	0.855	0.721	0.711	0.770	0.815	0.770	0.753	0.780	0.841	0.780	0.789	0.701	0.867	0.719	0.718								
	udpplain	0.998	0.987	0.998	0.992	0.992	0.998	0.992	0.995	0.994	0.984	0.998	0.988	0.993	0.991	0.992	0.993	0.991	0.989	0.991	0.986	0.987	0.993	0.987	0.989	0.992	0.981	0.992	0.968								



(a) Accuracy (b) Precision (c) Recall (d) F1-score

Fig. 6. Device-specific client-side evaluation of LSTM model performance for attack-type detection.



(a) Features from 1 to 29 (b) Features 30–59 (c) Features 60–89 (d) Features 90–115

Fig. 7. Heatmap for the difference between the secure aggregation of client models' explanations and server model explanations for binary classification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

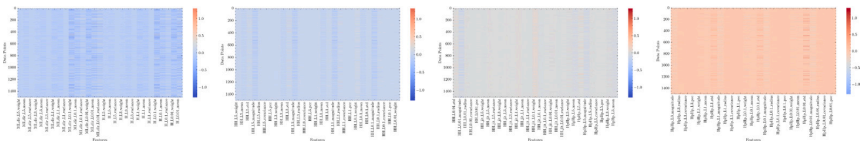


Fig. 8. Heatmap for difference between secure aggregation of client's models explanations and server model explanations for botnet-type detection. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

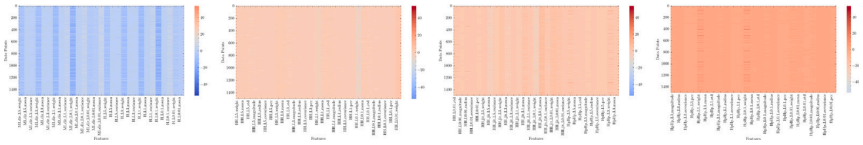
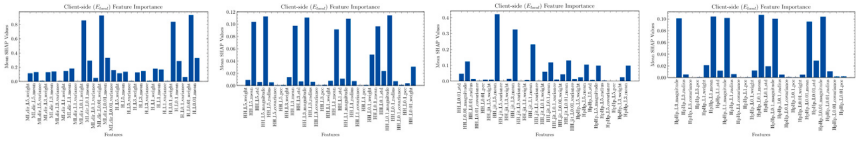


Fig. 9. Heatmap for difference between secure aggregation of client-side models explanations and server-side explanations for attack-type detection. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

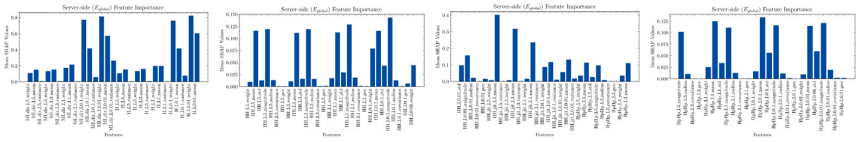
server side. Typically, SHAP explainer requires access to the training data and the trained model to compute SHAP values for the test data. However, in this work, our goal is to investigate the importance of features in the server model without providing any data to the server.

For this study, we are focusing on securely aggregating SHAP explanations from client models to be used as explanations for the server model. To ensure the privacy of SHAP value explanations from client models, a secure multi-party computation (SMPC) protocol based on Secret Sharing was employed, as mentioned in the methodology. So, the Server does not need to access the data from IoT device clients to explain the server model, and it also prevents individual client SHAP values explanations from being revealed to the Server or other clients. First, SHAP values were computed on each validation data of IoT device-based client model to find how each feature contributes to a model's output. In this context, the model's output refers to the probability distribution over the class labels obtained using the Softmax activation function at the output layer of the LSTM model. Therefore, the computation of SHAP values necessitates calculating the softmax function on the model's predictions. After completing the federated training, we used all the client models that participated and the server model from the last communication round to generate SHAP value explanations. For instance, we used the models from 10th communication round for binary and 50th for botnet-type classifications. On the other hand, we used the models from 2000th communication round for computing SHAP explanations for attack-type classification.

SHAP values were first computed for test data points for each IoT-device-based trained client model using its training data. By securely aggregating the SHAP values of each client model of all clients, we obtained secure aggregation explanations of clients-based explanations (E_{local}). We performed a comparative analysis to check the sufficiency of clients-based explanations that are good for the Server and validate their effectiveness in representing the server model's behaviour. For this analysis, we obtained the server-based explanations (E_{global}) as a baseline, assuming that the server-side model can access each client's training data and compute SHAP values of test data. SHAP values were computed and aggregated again without secure multi-party computation, resulting in the same size as (E_{local}). E_{global} acts as a comparative baseline for our proposed approach, as this option does not address any privacy restriction about accessing the client-side data. Finally, the difference (d_{g-l}) between the server-based explanations E_{global} and the client-based explanations (E_{local}) was computed, representing the divergence between the two sides. A lower difference value indicates

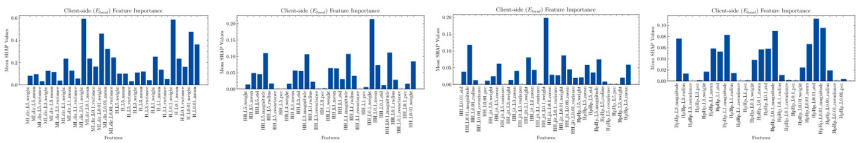


(a) Client-side absolute mean feature importance (Features 1-115)

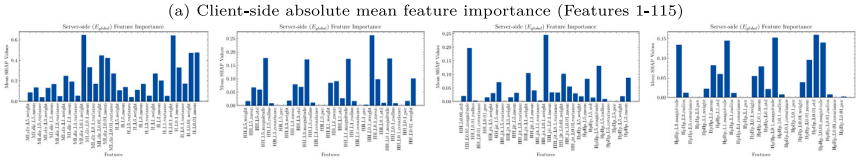


(b) Server-side absolute mean feature importance (Features 1-115)

Fig. 10. Mean SHAP values of secure aggregation matrix of client-side and server-side models for binary-type classification.

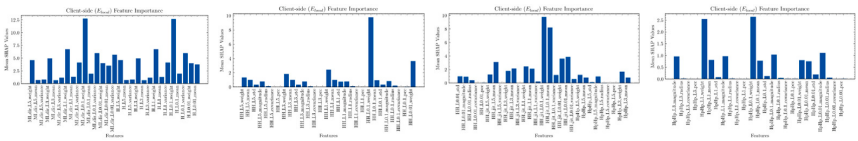


(a) Client-side absolute mean feature importance (Features 1-115)

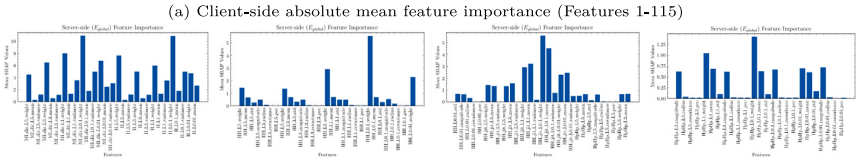


(b) Server-side absolute mean feature importance (Features 1-115)

Fig. 11. Mean SHAP values of secure aggregation matrix of client-side and server-side models for botnet-type detection.



(a) Client-side absolute mean feature importance (Features 1-115)

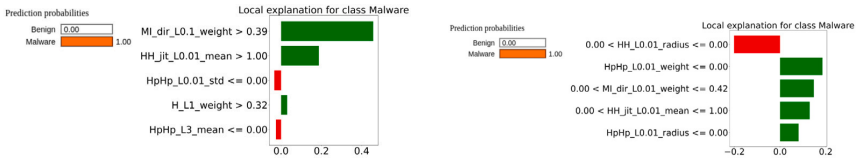


(b) Server-side absolute mean feature importance (Features 1-115)

Fig. 12. Mean SHAP values of Secure aggregation matrix of client-side and server-side models for attack-type detection.

that our proposal guarantees the privacy of the clients' data without causing intolerable distortion in the obtained explanations.

Figs. 7, 8, and 9 present heatmap graphs for the three detection types: Binary, Botnet-type, and Attack-type. To ensure the clarity and readability of the paper, we have presented a heatmap for all 115 features across four subfigures. For example, In Fig. 7, visualise the magnitude of the differences between server-based explanations and client-based explanations for each sample and feature. Each cell in the Heatmap shows the difference for a specific combination of sample and feature. The colour scale on the right side of each Heatmap shows a reference for interpreting the magnitude of the differences. Datapoints close to zero demonstrated a slight difference between the server-based and client-based explanations, demonstrating that secure SHAP values aggregation of client-based explanations closely approximate the serve model explanations when the server accesses the data. Instances further away from zero demonstrate a more significant discrepancy between the client-based and server-based explanations. Fig. 7 illustrates a Heatmap for binary detection. The overall differences between the server-based and client-based explanations were relatively small, as indicated by the predominantly close to zero across most data points. In Fig. 7(a) (features from 1–29), however, there were a few notable data points where the differences are more noticeable, as shown by instances away from zero. These instances, particularly for features like 'MI_dir_L0.1_weight', 'MI_dir_L0.01_weight', 'H_L0.1_weight', and 'H_L0.01_weight', as they suggest slight deviations in the secure aggregation of shapely values from the server-side explanation. Similarly,



(a) Lime Local Explanations for SecurityCam1 Client model over Malware instance

(b) Lime Local Explanations for Babymonitor Client model over Malware instance

Fig. 13. LIME local explanations for Client-side (SecuriytCam1 & BabyMonitor1) LSTM model over Malware instance for binary classification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Heatmap graphs for Botnet-type detection in Fig. 8 display high similarity between the server-based and client-based explanations, with most instances being close to zero. These instances highlight the specific instances and features where client-based explanations diverge from server-based ones. Heatmap for attack-type detection in Fig. 9 follows almost the same similarity, with most data points being close to zero, indicating a close alignment of the server-based explanations of SHAP values with client-based secure aggregation of SHAP explanations. However, some instances with more significant discrepancies were away from zero. For all three detection types, the secure aggregation of client-based SHAP value explanations closely approximates the server-side model explanations, even without providing the server with direct access to the IoT device-based clients' training and testing data.

We conducted a detailed analysis across 115 features by examining client-side (E_{local}) and server-based (E_{global}) explanations and comparing their differences. We computed the average SHAP values for client-based and server-based explanations. Fig. 10 shows that the feature importance indicated by the client-side mean SHAP values (Fig. 10(a)) and server-side mean SHAP values (Fig. 10(b)) were quite similar. Fig. 10(a) presents the Mean SHAP values from the secure aggregation of client-side models, while Fig. 10(b) displays the mean SHAP values from the server model, which has access to training data from IoT device-based clients. It is evident that by comparing these two sides of mean shap values, the top features influencing the model's predictions remained the same between client-side and server-side analysis. Fig. 11 displays the average SHAP values across 115 features for both client-side (Fig. 11(a)) and server-side explanations (Fig. 11(b)) related to botnet detection. The figures demonstrate a strong consistency in feature importance between the client-side and server-side explanations. Similarly, Fig. 12 compares the mean SHAP values for attack type detection, highlighting client-side feature importance in Fig. 12(a) and server-side feature importance in Fig. 12(b). This analysis shows that the key features contributing to model predictions closely align between the client side and the server side. Therefore, securely aggregating SHAP values in client-based explanations was sufficient, rather than needing to pass the data to the server for explanations of the server model.

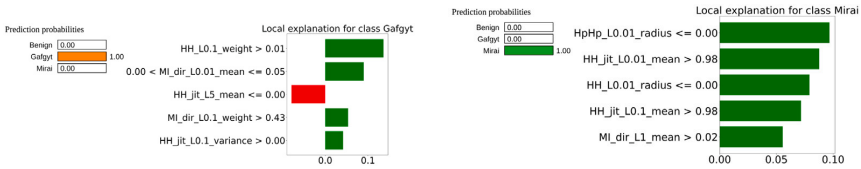
4.3. Explaining LSTM models on client-side IoT devices in FL

To explain the outcomes of the LSTM model on the client-side models, we employed three post-hoc feature-importance XAI techniques: LIME, IG and SHAP. Each client model can have its own training and test data, allowing us to present post hoc local explanations for the LSTM model. In this section, we present these post-hoc local explanations generated by the XAI methods and evaluate the quality of these explanations. To ensure the clarity and readability of the paper, we have selected only two IoT device-based models ('SecurityCam1' and 'BabyMonitor1') as use cases for illustrating the local explanations provided by these explainers.

LIME explainer explains the rationale behind assigning probabilities to each class by comparing the probability values with the actual class of the data point. LIME method provides explanations for why the probability was assigned to each class. Fig. 13 illustrates local explanations provided by LIME for the LSTM model on Client IoT devices such as SecurityCam1 and Baby monitor1 for Malware instances in Binary Classification. For Example, Fig. 13(a) shows LIME local explanations for a malware instance predicted by LSTM model on SecurityCam1 in Binay Classification. Predicted probabilities for two class labels, 'Benign' and 'Malware', in Binary classification, were shown on the left side of the figure. LSTM model predicted a 100% probability for the selected instance as Malware and 0% for the Benign class. On the right side of the figure, local explanations were provided, showing why the LSTM model classified the selected instance as Malware. Green bars indicate that features contributed positively to predicting Malware for a selected instance, while the red bars show features contributing to predicting an instance as a Benign label. For example, To predict an instance as Malware, features such as MI_dir_L0.1_weight (packet count ((weight) network traffic within in 10 s time frame window from Hos Mac & IP), HH_jit_L0.01_mean (Mean value of network traffic within 1 min from Network jitter), H_L1_weight (Packet count of network traffic within 100 ms from the host) positively contributed to malware prediction (shown in the Green bar). In Contrast, for the Benign class label, HpHp_L0.01_std (Standard deviation Network traffic within a 1 min time window from the socket), HpHp_L3_mean (mean value of network traffic from socket within 100 ms). It can be observed that Host-based and network-jitter features were more influential in predicting Malware. For the Babymonitor IoT device of instance (see in Fig. 13(b)) for the selected instance, LSTM Model predicted with 100% probability as Malware.

Similarly to Binary classification, LIME was again used to explain the outcomes of the LSTM model on IoT devices in Botnet-type and attack-type detection of multi-classification. The n-BaIoT dataset contains network traffic data from nine distinct IoT devices infected with Mirai and Gafgyt Malware and legitimate traffic. Fig. 14 show LIME explanations of the LSTM model for Botnet-type detection on IoT devices for selected instances. Fig. 14(a) shows LIME local explanations for the LSTM model's prediction of a Gafgyt instance on the SecurityCam1 IoT device. On the left, the LSTM model predicted Gafgyt with 100% accuracy. On the right, the green bars represent features (HH_L0.1_weight, MI_dir_L0.01_mean, MI_dir_L0.1_weight, HH_jit_L0.1_variance) that positively impacted the prediction of the selected Gafgyt instance, while the red bar represents the feature (HH_jit_L5_weight) that negatively impacted the prediction by contributing to other class labels (Mirai or Benign). Similarly, Fig. 14(b) shows the LSTM model's 100% accurate prediction for a Mirai Malware instance on the Babymonitory device shows the 100% accuracy.

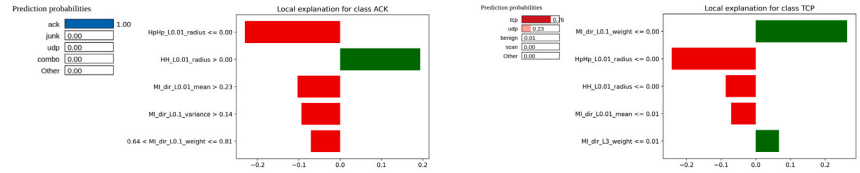
An LSTM model was used for attack type detection to identify nine different types of attacks in a FL setup. The attack types included ACK, benign, compact, junk, scan, SYN, TCP, UDP, and UDP plain. To demonstrate the use case of local explanations for LIME, an instance of an ACK attack was selected from the SecurityCAM1 device model, while a TCP attack instance was chosen from the Babymonitor device model. Fig. 15(b) illustrates LIME local explanations for the client models of BabyMonitor, focusing specifically on a TCP attack instance for the top five features in attack-type detection. On the left, the LSTM model



(a) Lime Local Explanations for SecurityCam1 Client model over Gafgyt Malware instance

(b) Lime Local Explanations for Babymonitor Client model over Mirai Malware instance

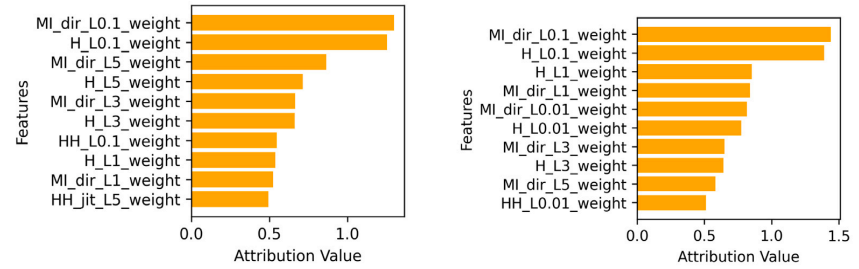
Fig. 14. LIME local explanations for client-side (SecuriytCam1 & BabyMonitor) LSTM model over Gafgyt and Mirai instance for botnet type detection. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) LIME Local Explanations for SecurityCam1 Client model over ACK attack instance

(b) LIME Local Explanations for Babymonitor Client model over TCP Attack instance

Fig. 15. LIME local explanations for SecurityCam1 and BabyMonitor client-side LSTM model over ACK and TCP attack instances for attack-type detection.



(a) IG Local Explanations for SecurityCam1 Client model over Malware instance

(b) IG Local Explanations for Babymonitor Client model over Malware instance

Fig. 16. IG local explanations for client-side (SecuriytCam1 & BabyMonitor1) LSTM model over Malware instance for binary classification.

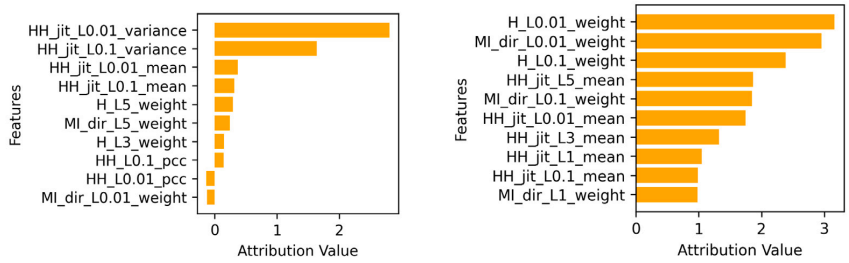
predicted the TCP attack instance with 76% accuracy, the UDP attack with 23% accuracy, and the benign class with 1% accuracy for the selected instance. On the right, the green bars represent feature importance for MI_dir_L0.1_weight and MI_dir_L3_weight, which positively impacted the prediction of the selected TCP attack instance. In contrast, the red bars indicate features that negatively impacted the prediction by contributing to other class labels, particularly the benign and UDP classes. Similarly, Fig. 15(a) shows the LSTM model's 100% accurate prediction for an ACK instance on the SecurityCam1 device.

We have also employed IG to provide feature attribution for client-side IoT device models. This model-specific post-hoc XAI method calculates attributions by integrating the gradients of the model's predictions with respect to input features along a path from a zero baseline to the actual input. Fig. 16 illustrate IG explanations for malware predictions in a binary classification context. Figs. 16(a) and 16(b) show the top 10 features for selected instances of malware class labels for SecurityCam1 and Babymonitor, respectively.

For IG local explanations of botnet-type detection in multiclass classification, Fig. 17(a) presents the top 10 feature attributions for the SecureCam1 model on the Mirai instance, while Fig. 17(b) showcases a BabyMonitor's Mirai class instance.

For IG local explanations of attack-type detection in multiclass classification, Fig. 18(a) presents the top 10 feature attributions for the SecureCam1 model on the ACK class instance, while Fig. 18(b) showcases a BabyMonitor TCP attack instance.

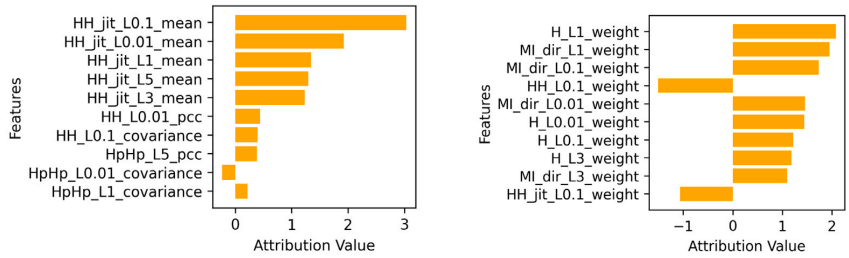
SHAP was widely used to explain models and understand how the features relate to the predictions. SHAP gives local and global explanations. In local explanations, a specific individual instance was selected, and the model prediction was explained by showing each feature's contribution to the prediction of the selected instance. SHAP calculates the Shapley values that illustrate the impact of each feature on the model's predictions. Fig. 19 shows SHAP local explanations using SHAP's Force plot for Malware instances predicted by the LSTM model on SecurityCam1, BabyMonitor in Binay Classification. For instance, in Fig. 19(a), SHAP local explanations for malware instances of the LSTM model for the SecurityCam1 device show the feature's contribution to the prediction. The plot shows the base value and features that positively impact the prediction, which is in red, and the features showing a negative impact on the prediction are in blue. The base value in the plot is the average of all prediction values. Each strip in the plot shows the impact of the features in pushing the predicted value closer or farther from the base value. Red strip features push the value to higher values, whereas blue strip features push the value to lower values. The contribution of features having wider strips was more. For Selected malware instance, the base value is 0.5037, Features such as HH_jit_L0.01_mean (network Traffic mean value within a 1 min time window from network jitter) and MI_dir_L0.1_weight (Network



(a) IG Local Explanations for SecurityCam1 Client model over Gafgyt Malware instance

(b) IG Local Explanations for Babymonitor Client model over Mirai Malware instance

Fig. 17. IG local explanations for client-side (SecuriytCam1 & BabyMonitor) LSTM model over Gafgyt and Mirai instance for botnet type detection.



(a) IG Local Explanations for SecurityCam1 Client model over ACK attack instance

(b) IG Local Explanations for Babymonitor Client model over TCP Attack instance

Fig. 18. IG local explanations for SecurityCam1 and BabyMonitor client-side LSTM model over ACK and TCP attack instances for attack-type detection.



(a) SHAP Local Explanations for SecurityCam1 Client model over Malware instance

(b) SHAP Local Explanations for BabyMonitor Client model over Malware instance

Fig. 19. SHAP local explanations for client-side of SecurityCam1 & BabyMonitor LSTM models over a Malware instance for binary classification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) SHAP Local Explanations for SecurityCam1 Client model over Gafgyt Malware instance

(b) SHAP Local Explanations for Babymonitor Client model over Mirai Malware instance

Fig. 20. SHAP local explanations for client-side (SecuriytCam1, BabyMonitor1 & Thermostat) LSTM model over Gafgyt and Mirai instance for botnet type detection. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) SHAP Local Explanations for SecurityCam1 Client model over ACK attack instance

(b) SHAP Local Explanations for Babymonitor Client model over TCP attack instance

Fig. 21. SHAP local explanations for SecurityCam1 and BabyMonitor client-side LSTM model over ACK and TCP attack instance for attack-type detection. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Results of evaluating Quality of LIME, IG & SHAP explanations using High Faithfulness (μ_f), Low Complexity (μ_c), Max Sensitivity (μ_s) & Monotonicity (μ_m) for participated client-side models in FL for three classification types.

Client	Metric \explainer	Binary			Botnet type			Attack-Type		
		Lime	IG	SHAP	Lime	IG	SHAP	LIME	IG	SHAP
BabyMonitor	μ_f	0.24 ± 0.30	0.01 ± 0.25	0.34 ± 0.23	0.47 ± 0.34	−0.03 ± 0.24	0.79 ± 0.15	0.43 ± 0.28	0.00 ± 0.22	0.62 ± 0.32
	μ_c	3.95 ± 0.07	3.29 ± 0.34	3.55 ± 0.08	4.06 ± 0.11	3.25 ± 0.39	1.26 ± 0.31	4.01 ± 0.31	3.39 ± 0.39	1.39 ± 0.75
	μ_f	0.14 ± 0.10	0.26 ± 0.11	0.04 ± 0.05	0.41 ± 0.18	0.08 ± 0.23	0.09 ± 0.23	0.26 ± 0.11	0.11 ± 0.70	0.09 ± 0.15
	μ_m	31.07%	1.00%	46.00%	97.00%	0.03%	99.00%	99.00%	0.00%	96.00%
Doorbell1	μ_f	0.38 ± 0.29	−0.01 ± 0.23	0.55 ± 0.21	0.14 ± 0.09	0.03 ± 0.28	0.25 ± 0.31	0.32 ± 0.33	0.00 ± 0.23	0.56 ± 0.32
	μ_c	4.03 ± 0.05	3.74 ± 0.36	3.55 ± 0.10	3.99 ± 0.11	2.99 ± 0.49	1.42 ± 0.35	3.89 ± 0.26	3.23 ± 0.48	1.59 ± 0.58
	μ_f	0.20 ± 0.10	0.08 ± 0.00	0.02 ± 0.02	0.65 ± 0.20	0.01 ± 0.00	0.00 ± 0.00	0.36 ± 0.13	0.01 ± 0.00	0.06 ± 0.10
	μ_m	41.40%	1%	66.54%	74%	1%	85%	91%	6%	92%
SecurityCam1	μ_f	0.31 ± 0.34	−0.00 ± 0.25	0.51 ± 0.32	0.36 ± 0.21	−0.01 ± 0.29	0.59 ± 0.22	0.39 ± 0.39	0.00 ± 0.23	0.70 ± 0.25
	μ_c	3.89 ± 0.07	3.88 ± 0.29	3.35 ± 0.23	4.13 ± 0.17	3.16 ± 0.49	1.31 ± 0.32	3.94 ± 0.43	3.23 ± 0.48	1.42 ± 0.44
	μ_f	0.10 ± 0.05	0.35 ± 0.12	0.02 ± 0.04	0.39 ± 0.17	0.10 ± 0.08	0.09 ± 0.21	0.23 ± 0.09	0.30 ± 0.14	0.05 ± 0.09
	μ_m	40.41%	5%	39.67%	96%	1%	100%	97%	20%	96%
SecurityCam2	μ_f	0.34 ± 0.37	0.01 ± 0.29	0.45 ± 0.37	0.21 ± 0.17	0.01 ± 0.19	0.36 ± 0.26	0.34 ± 0.29	−0.00 ± 0.17	0.66 ± 0.31
	μ_c	3.95 ± 0.05	3.86 ± 0.52	3.35 ± 0.07	4.13 ± 0.09	3.45 ± 0.40	1.76 ± 0.22	3.95 ± 0.20	3.50 ± 0.40	1.39 ± 0.73
	μ_f	0.18 ± 0.10	0.06 ± 0.10	0.02 ± 0.02	0.53 ± 0.21	0.36 ± 0.22	0.07 ± 0.14	0.47 ± 0.15	0.01 ± 0.00	0.06 ± 0.12
	μ_m	36.55%	4%	42.45%	70%	1%	95%	98%	2%	98%
SecurityCam3	μ_f	0.38 ± 0.34	−0.01 ± 0.19	0.59 ± 0.26	0.18 ± 0.33	−0.04 ± 0.26	0.51 ± 0.24	0.36 ± 0.31	−0.00 ± 0.22	0.62 ± 0.30
	μ_c	3.89 ± 0.08	3.50 ± 0.38	3.41 ± 0.24	4.02 ± 0.23	3.20 ± 0.48	1.40 ± 0.20	4.01 ± 0.26	3.22 ± 0.47	1.32 ± 0.72
	μ_f	0.08 ± 0.03	0.42 ± 0.15	0.01 ± 0.02	0.35 ± 0.22	0.42 ± 0.28	0.04 ± 0.14	0.23 ± 0.05	0.26 ± 0.21	0.07 ± 0.13
	μ_m	42.85%	1%	40.77%	100%	1%	100%	94%	0%	90%
SecurityCam4	μ_f	0.33 ± 0.28	0.03 ± 0.27	0.47 ± 0.27	0.46 ± 0.14	0.02 ± 0.19	0.63 ± 0.14	0.48 ± 0.36	0.01 ± 0.24	0.71 ± 0.37
	μ_c	4.00 ± 0.07	3.65 ± 0.44	3.49 ± 0.17	4.07 ± 0.19	3.39 ± 0.42	1.43 ± 0.23	4.04 ± 0.25	3.41 ± 0.43	1.36 ± 0.71
	μ_f	0.11 ± 0.07	0.01 ± 0.22	0.01 ± 0.02	0.37 ± 0.10	0.12 ± 0.05	0.02 ± 0.08	0.26 ± 0.10	0.40 ± 0.00	0.05 ± 0.09
	μ_m	44.20%	9%	37.65%	100%	8%	100%	92%	1%	99%
Thermostat	μ_f	0.37 ± 0.38	0.03 ± 0.26	0.56 ± 0.33	0.39 ± 0.13	−0.01 ± 0.19	0.61 ± 0.09	0.34 ± 0.26	−0.00 ± 0.19	0.66 ± 0.26
	μ_c	3.87 ± 0.05	3.45 ± 0.36	3.37 ± 0.11	4.10 ± 0.20	3.48 ± 0.40	1.52 ± 0.27	4.05 ± 0.14	3.48 ± 0.41	1.29 ± 0.65
	μ_f	0.10 ± 0.06	0.06 ± 0.12	0.01 ± 0.02	0.38 ± 0.19	0.01 ± 0.00	0.06 ± 0.16	0.31 ± 0.11	0.01 ± 0.00	0.05 ± 0.09
	μ_m	34.49%	3%	37.46%	90%	1%	100%	98%	3%	100%
Doorbell2	μ_f	0.24 ± 0.17	−0.01 ± 0.20	0.43 ± 0.15						
	μ_c	4.19 ± 0.16	3.57 ± 0.42	3.64 ± 0.07						
	μ_f	0.09 ± 0.03	0.04 ± 0.08	0.09 ± 0.09						
	μ_m	58.41%	8%	53.98%						
Webcam	μ_f	0.46 ± 0.42	0.01 ± 0.25	0.51 ± 0.39						
	μ_c	3.88 ± 0.10	3.35 ± 0.40	3.14 ± 0.10						
	μ_f	0.12 ± 0.02	0.04 ± 0.09	0.01 ± 0.02						
	μ_m	57.21%	9%	69.23%						

traffic packet count within 10s-time window from Host Mac & IP), have positive contribution on prediction value, and MI_dir_L1_variance have negative contribution on prediction value. Similarly, in binary classification, Fig. 19(b) show SHAP explanations for the LSTM model of BabyMonitor device

For SHAP local explanations of Botnet-type detection in Multiclass classification, Fig. 20(a) shows the LSTM model for Gafgyt instance on Securitycam1 IoT device. LSTM model output is 100% with a base value of 0.6504. Features MI_dir_L0.01_variance, HH_jit_L5_mean and MI_dir_L0.01_mean, HH_jit_L0.01_mean contributed positively to the prediction of the Gafgyt class (red strip), while features HH_jit_L0.1_variance had a negative contribution (Blue strip). Likewise, Fig. 20(b) shows the LSTM model's 0.98 output with a base value of 0.6286 for a Mirai Malware instance on the Babymonitor device.

In Fig. 21, the SHAP force plot illustrates local explanations of feature contributions for the selected class label in attack-type detection. For the BabyMonitor IoT device, Fig. 21(b) shows the features that positively influence the LSTM model's prediction towards the TCP attack class, represented by the red strip. In contrast, the features that decrease the probability of prediction (negative influence) are shown in blue. The output of the LSTM model for TCP attack instances on the Baby Monitor device is 0.79. Similarly, for the selected instance of ACK attack class labels from the SecurityCam1 device in Fig. 21(a).

We conducted a comprehensive quantitative evaluation of the effectiveness of three prominent XAI methods, LIME,IG, SHAP, for all participating clients in the FL process. Our evaluation was based on four key metrics: High Faithfulness, Monotonicity, Low Complexity, and Max Sensitivity. These metrics were chosen for their ability to provide a robust assessment of the quality of XAI methods.

In our study, we evaluated the faithfulness of explanation methods by computing the correlation between the importance assigned to attributes by XAI methods and their impact on the predictive model's probabilities. A high faithfulness correlation of the XAI method indicates that the assigned feature importance aligns closely with its impact on the model's prediction probabilities, ensuring accurate and faithful explanations. We evaluated the monotonicity of the explanation to understand how individual features influence the model's output. By incrementally adding each attribute, we evaluated the effect of individual features on the model's output and determined their importance based on their impact. The prediction probabilities of the LSTM model were obtained using the Softmax activation function. A high monotonicity score means that the explanations are consistent with the model's predictions for the given input. We calculate the low-complexity metric by computing the entropy of feature attribution derived from the explanations. The complexity of an attribution is based on the entropy of each feature's contribution to the total attribution. Max sensitivity XAI metric evaluates the robustness of the explainer's output, ensuring that nearby inputs in the feature space have similar explanations when the sensitivity value is low. To identify nearest-neighbour points related to the predicted label of the explanation, we used the Euclidean distance with a neighbourhood radius ($r = 0.1$), which helps to identify data points in the feature space that are closest to the instance and contain similar explanations for the predicted label.

We evaluated local explanations provided by LIME, IG and SHAP across 2000 test points. Table 4 presents the results of the XAI metrics, including the mean and standard deviations for both the LIME and SHAP explanations. These evaluations were conducted on 2000 data points from all participating clients in FL for three types: Binary, Botnet-type, and Attack-type.

In binary classification, the SHAP explainer shows higher mean values for faithful explanations than LIME, IG explainers across all test instances from participating clients. For example, SHAP achieved a faithfulness score of 0.34 ± 0.05 for the BabyMonitor model, compared to 0.31 ± 0.04 for IG and 0.24 ± 0.30 for LIME. This suggests that SHAP provides a more reliable feature importance, closely aligned with its effect on model prediction probabilities. When evaluating local explanations using low complexity metrics, SHAP almost achieved lower mean values than LIME and IG across all clients. However, for the Client models BabyMonitor and Doorbell2, the IG explainer outperformed the remaining explainers using lower complexity metrics. Sensitivity refers to the degree to which the explanations are stable for nearby data points. Lower sensitivity values indicate more stable

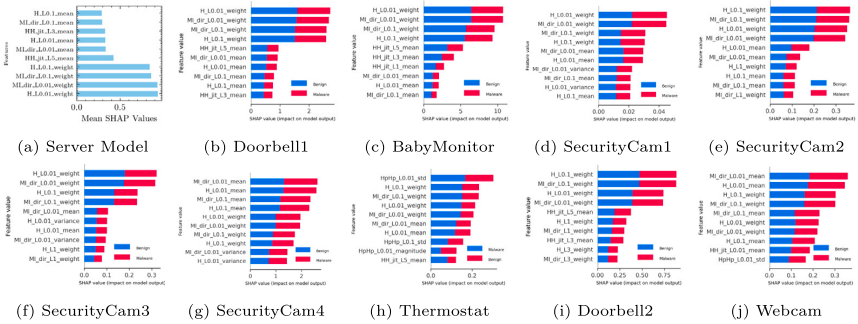


Fig. 22. Global explanations for server model feature importance and participated clients' IoT device models feature importance in binary classification type.

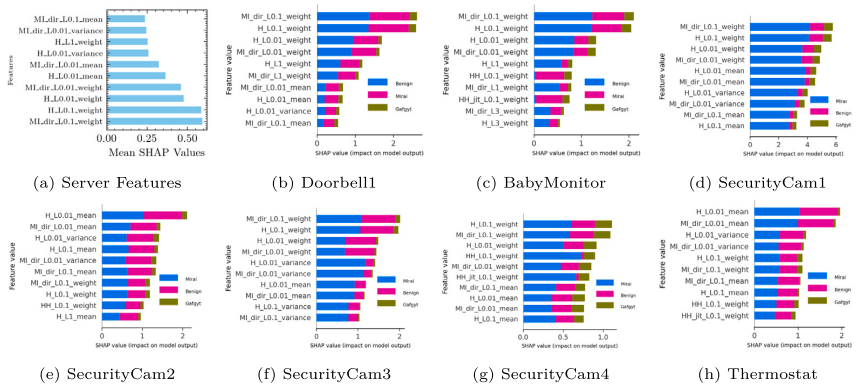


Fig. 23. Global explanations for server model feature importance and participated clients IoT devices feature importance for botnet-type detection.

explanations. On average, SHAP models, when explained using SHAP, exhibit lower sensitivity values compared to LIME and IG explanations for all clients. However, for the Doorbell2 model, IG provides more robust explanations than both LIME and SHAP. This indicates that SHAP provides more robust and stable explanations for nearby data points within the same feature space used by its client model. SHAP explanations were generally more consistent in terms of monotonicity compared to LIME, IG when evaluated using monotonicity metric across all client models. However, for the SecurityCam1, SecurityCam3, SecurityCam4 and Doorbell2 models, the LIME explanations demonstrate more monotonicity, with percentages of 40. 41%, 42. 85%, 44. 20% and 58. 41%, respectively, compared to the SHAP explanations.

Similarly to binary classification, SHAP is generally more reliable than LIME and IG for both botnet types and attack types. It shows a strong correlation between the importance of the features calculated by the SHAP explainer and the prediction probabilities of the client's model. Furthermore, SHAP exhibits lower complexity and sensitivity compared to LIME and IG. However, for the Doorbell1 and SecurityCam2 models, IG proves to be more robust than both LIME and SHAP explainers using the maximum sensitivity metric. Using the monotonicity metric, SHAP demonstrates more monotonicity compared to LIME, IG, for both botnet-type and attack-type detection across all clients. However, in the attack-type detection for the BabyMonitor, SecurityCam1 and SecurityCam3 models, the LIME explainer slightly outperforms SHAP with respect to monotonicity metrics values (see Table 4).

We have provided the results of global explanations for server and client models that participated in FL, using the SHAP explainer. In these global explanations, we explain the model's predictions using each feature's contribution. Fig. 22 shows the global explanations of feature importance for the server model and participating clients' IoT device models in binary classification, focusing on the top 10 features that impact model predictions. In the client-based model of global explanation in FL, each participating client of the SHAP explainer used its training data and trained model to calculate the Shapley values on the test data, which shows the impact of each feature on the model's predictions. Figs. 22(b)–22(j) provide global explanations from the SHAP explainer of the top ten features for client-based models participating in FL for binary classification. Fig. 22(a) presents the top 10 features of the server model for binary classification, obtained through the secure aggregation of SHAP value explanations from clients. It can be observed that features related to host IP (H), network jitter (HH_Jit), and host MAC & IP (MI) are the most impactful network categories for binary classification predictions. In particular, H_L0.1_weight and MI_dir_L0.01_weight are the most significant features across clients. Additionally, network traffic packet count (weight) features are crucial for identifying benign and malicious network traffic. Shorter time windows, specifically L0.1 (500 μs) and L1 (1.5 s), are also particularly influential regarding network time windows.

Fig. 23 illustrates global explanations for botnet type detection from SHAP explainer. It highlights the top 10 features that influence detecting different types of botnets across various LSTM client-side IoT device models and the server model. Figs. 23(b)–23(h) provide SHAP's global explanations for participating client IoT device models in FL. Fig. 23(a) shows the top ten feature importances for the server model, based on secure aggregated SHAP values from the participating client model's explanations. For detecting botnet types, features such as Host IP (H), Host MAC & IP (MI), and Network Jitter (HH_jitt) were obtained within

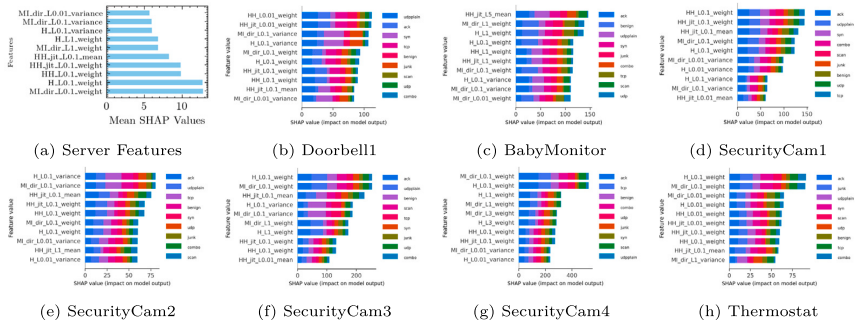


Fig. 24. Global explanations for server model feature importance and participated clients IoT devices feature importance for attack-type detection.

short to moderate time window frames were most effective in SHAP global explanations of feature importance in multiclass botnet detection within federated IoT networks. Additionally, Related to network statistical values, particularly packet count (weight), as well as their mean and variance, are vital for analysing network traffic features in botnet-type detection.

Similarly, Fig. 24 shows the global explanations for both the server model (Fig. 24(a)) and participated client-side models (Figs. 24(b)–24(h)) used in attack type detection. For detecting attack types, the Host-MAC & IP (MI) and Host-IP (H) network category features were more influential compared to other network categories among the selected top 10 features. In contrast to botnet type detection, HH_jit features were found to be less influential. Channel and socket-based features showed no influence. When considering the time window, shorter time windows, such as L0.1 (100 μs) and L0.01 (500 μs), were more influential according to the SHAP explainer in global explanations. Additionally, packet count features were more influential when it came to network traffic statistics.

5. Discussions

In this work, we propose a FedXAI-based framework designed to address the intrusion detection problem in IoT network devices within a HFL setting. Our framework incorporates XAI methods to enhance the interpretability of LSTM model predictions, enabling both client-side and server-side models to provide transparent and privacy-preserving explanations for intrusion detection decisions. Our research utilises the N-BaioT dataset, which contains statistical network traffic features for botnet detection on 9 IoT devices (Doorbell1, BabyMonitor, SecurityCam1, SecurityCam2, Thermostat, SecurityCam3, SecurityCam4, Doorbell2, and Webcam) infected by botnets such as Gafgyt and Mirai botnets. We have developed three classification techniques within a FL framework: Binary Classification, Botnet-Type Detection, and Attack-Type Detection. The hyperparameters for training the LSTM model in classification tasks are summarised in Table 5.

The effectiveness of our proposed LSTM model is evaluated and compared against recent existing models on N-BaIoT dataset within FL settings across three classification tasks. The results are summarised in Table 6. The binary classification task focuses on distinguishing between malicious and benign traffic. Our LSTM model achieved the highest performance, with an accuracy of 99.90%, precision of 99.80%, recall of 100.00%, and an F1-score of 99.90%. This model outperformed other benchmarks, including the MLP model developed by Rey et al., which achieved an accuracy of 99.60% using mini-batch and multi-epoch aggregation [30]. Additionally, Wardhan et al. proposed a DNN model using FedAvg that reached an accuracy of 98.97%, with a precision of 98.75%, recall of 99.41%, and an F1-score of 99.11% for binary classification. Zhang et al. developed a FedDetect aggregation

Table 5
Hyperparameters used for LSTM model training across three classification tasks in FL.

Hyperparameter/Classification	Binary	Botnet-type	Attack-type
Hidden layers	3	5	5
Hidden units	128	128	128
Learning rate	0.001	0.001	0.001
Optimiser	Adam	Adam	Adam
Activation function	ReLU	ReLU	ReLU
Batch size	256	256	512
Epochs	50	100	200

on deep autoencoders, which achieved an accuracy of 93.70% and a precision of 88.20% [60].

In our study, Botnet-Type Detection offers a more detailed analysis by categorising data points into three categories. Mirai, Gafgyt, and benign network traffic. Our model achieves an accuracy of 99.28%, precision of 99.63%, recall of 99.79%, and an F1 score of 99.54% in FL. In addition to preserving data privacy, the proposed FL-based LSTM model outperforms most existing centralised botnet detection models. For example, the Artificial Neural Network (ANN) model proposed by Palla and Tayeb for Mirai botnet detection achieved an accuracy of 92. 8% and an F1 score of 99%.

LSTM model achieved above 99% accuracy, precision, recall, and F1 score on the models based on the IoT device of the client, as well as the server model for binary classification and botnet-type detection. On the other hand, for attack-type detection, data points are categorised into various attack types: ACK, benign, compact, junk, scan, SYN, TCP, UDP, and UDPplain. In the attack-type detection setting, the LSTM model combined with the FedAvg aggregation algorithm has yielded impressive results. Specifically, for attack-type detection, the LSTM model achieved an accuracy of more than 94.89%, a precision of 95.71%, a recall of 94.73%, and an F1 score of 94.50% for the server model using the FedAvg algorithm. From Table 6, when comparing with other studies for the detection of attack types, our model demonstrates better performance. PH Do et al. developed a DNN model using Mini-Batch aggregation, which achieved an accuracy of 90.00%, a precision of 88.00%, and a recall of 85.00%, and an F1 score of 88. 00%. Furthermore, GDLT Parra created an LSTM model in a distributed learning setup that achieved an accuracy of 94.80%. B. Olanrewaju-George, using the FedAvgM (Federated Averaging Algorithm with Momentum), achieved an accuracy of 90.39%.

In contrast to earlier studies that did not integrate explainable AI (XAI) methods into their FL for IDS, we introduced a unique approach to securely aggregating client explanations, in order to enhance the explainability and transparency of server models in the context of FL for IoT device intrusion detection. This approach eliminates the

Table 6
Performance metrics comparison of the proposed model with existing models over N-BaIoT dataset in FL settings.

Model	Aggregation	Classification	Accuracy	Precision	Recall	F1-score	Explainability
MLP [30]	Mini-Batch, Multi-Epoch	Binary	99.60%	NA	NA	NA	No
Deep autoencoder [60]	FedDetect	Binary	93.70%	88.20%	NA	NA	No
DNN [62]	FedAvg	Binary	98.97%	98.75%	99.41%	99.11%	No
ANN [61]	Centralised	Botnet-type	92.80%	99.00%	NA	NA	No
CNN [63]	Mini-Batch	Attack-Type	90.00%	88.00%	85.00%	88.00%	No
LSTM [64]	Distributed Learning	Attack-Type	94.80%	NA	NA	NA	No
FL-DNN [65]	FedAvgM	Attack-Type	90.39%	NA	NA	NA	No
Ourwork	**FedAvg**	**Binary**	**99.90%**	**99.80%**	**100.00%**	**99.99%**	Yes
		Botnet-type	**99.28%**	**99.63%**	**99.79%**	**99.54%**	
		Attack-Type	**94.89%**	**95.71%**	**94.73%**	**94.50%**	

NA — Not Applicable.

need for client data sharing while effectively balancing transparency and privacy. To explain the server model based on HFL using the FedAvg algorithm, we proposed a privacy-preserving approach that uses securely aggregated SHAP values from the participating client models. To assess the effectiveness of these aggregated SHAP values, we compared them with the SHAP values obtained from the server model when it accessed data from each participating client model. Our findings indicate that the securely aggregated SHAP values from the client models closely approximate the SHAP values of the server model during data access. As a result, we demonstrated that the client-based aggregated SHAP explanations are sufficient for our study, enabling us to explain the server model in terms of feature importance for IoT-based IDS solutions in FL settings.

Our results send an important message to those who design IDSs. The attacks originating from the bots (i.e., as simulated in the N-BaIoT dataset) can be effectively detected by sensors that monitor the incoming and outgoing packet statistics, regardless of the traffic destination. Through the secure aggregation of client-based SHAP values for the server model and the global explanations model for clients using their own data, we have inferred the importance of various features for both server-side and client-side models. Our research suggests that the host-MAC and IP (MI), along with the host-IP (H) from pocket count network statistical features, are particularly influential in detecting IoT botnets on network devices. In contrast, network jitter (HH_jit) has been found to have significantly less impact. These findings assist SOC analysts in refining IDS configurations, prioritising essential traffic indicators, and improving botnet detection strategies in real-world deployments.

SHAP values were computed for each IoT device-based client in our approach to explaining the server model using SHAP's Deep Explainer. This step was computationally expensive due to the complexity of the game's theoretical principles, and pre-trained client models were utilised. After calculating the SHAP values, SMPC was employed for the aggregation of these values, which is more lightweight compared to the cost of generating the SHAP values. During our experiments, the integration of SMPC-based SHAP value aggregation into FL did not introduce significant bottlenecks. Although we did not explicitly measure the computational overhead of this process, our performance observations suggest that the aggregation step was computationally feasible.

We used LIME, IG and SHAP as post-hoc local explanation methods to gain insights into the behaviour of participant client models in FL. To evaluate the quality of local post-hoc explanations produced by these three methods, we employed metrics such as High Faithfulness, Low Complexity, Monotonicity, and Max Sensitivity (which indicates greater robustness). Our findings revealed that the post-hoc local explanations generated by the SHAP explainer surpassed those generated by LIME, IG across all client models in FL, achieving higher faithfulness, lower complexity, greater robustness (as indicated by max sensitivity), and higher percentages of monotonicity for the test data points.

In our evaluation of XAI methods, we found that SHAP took significantly more computational time than LIME and IG due to its complex game-theoretic principles, which involve calculating Shapley values for various feature combinations. Additionally, Evaluating the xai method using the Max Sensitivity metric was computationally intensive, as it required assessing neighbouring points (radius $r = 1$) around each instance using a Euclidean distance. These observations highlight the balance between explanation quality and computational efficiency in XAI methods like SHAP, which provide high-quality explanations but require significant computational time.

Several studies have examined the explainability of FL-IDS. For example, Oki et al. utilised SHAP for model explainability. However, the feature contributions were assessed centrally on the global model [66]. This approach requires input data from edge servers to be aggregated at a central server, which could raise privacy concerns. Fatema et al. developed a framework that focuses solely on local model explanations for the IDS task in multiclassification using CICIoT2023 dataset, without providing a clear method for explaining the server-side model [67]. Other studies [42,45,68] also utilise SHAP for explainability in FL, yet they often neglect to clarify how input data was used for server model explanations, resulting in ambiguity regarding privacy. In contrast, our work presents a clear and privacy-preserving methodology for generating explanations on both the server side and the client side. Unlike prior approaches, which may inadvertently expose sensitive data, our framework ensures that post-hoc XAI methods operate strictly on client-side data, preventing raw data from being passed to the server. This approach mitigates the risks of data exposure while preserving the interpretability of both client and server models in a federated setting.

6. Limitations and threats to validity

Our study shows that using post-hoc xai methods in an FL-based -IDS in IoT environments is effective. However, we acknowledge limitations such as the complexity of IoT systems, the computational costs of explainability techniques, and the security risks associated with FL. We also highlight challenges that may affect the generalisability and scalability of our approach.

- IoT environment is a complex network characterised by constant dynamism, resource limitations, and large data volumes. This study uses the N-baioT data set, one of the largest and most realistic IoT botnet data sets, to simulate benign and malicious network behaviour in a medium-sized IoT architecture. Our findings provide an initial evaluation of the potential of FL integrated with XAI techniques to enhance ID in IoT environments. However, more empirical studies are needed to validate and expand these results across different IoT architectures. Future work will focus on testing our approach in diverse IoT settings with varying data distributions and network configurations.
- Our study evaluates the performance of an FL-based IDS using independent and identically distributed (IID) data across edge devices. We acknowledge that this scenario is unlikely to be applicable to real applications, and we plan to conduct further

experiments on non-IID data. Nonetheless, the configuration we tested allows us to analyse the impact of FL on SHAP explanations while minimising excessive variability.

- Our approach relies on SHAP values derived from IoT device-based clients to explain the server model in FL, addressing the disadvantage of post-hoc explainers that require input data for model explanations, which could compromise privacy. However, this method becomes increasingly computationally expensive as the number of clients grows, making it less feasible for large-scale IoT environments. The high computational demands of SHAP's Deep Explainer and Kernel Explainer further exacerbate this issue. This limitation underscores the need for more efficient and scalable explainability techniques. In our future work, we plan to explore synthetic data generation to enhance the explainability of the server model, reducing reliance on client-based explanations and ensuring better scalability. Another direction in addressing scalability issues and resource constrains in IoT devices would be delegating intrusion detection function to local edge devices. This architectural decision may relax the privacy requirements as IoT devices should trust these edge devices. However, in large-scale IoT networks in which responsibilities of the edge devices are assigned to distinct entities, still, privacy can be preserved to a degree.
- This study uses SMPC to securely aggregate SHAP values; however, it remains vulnerable to inference attacks, allowing adversaries to extract sensitive information about client models. Additionally, since SHAP values are calculated post-hoc, they can be manipulated using adversarial examples, leading to misleading feature attributions while keeping the model output unchanged. Prior works, such as Fooling LIME and SHAP [54], have demonstrated that attackers can train models to provide deceptive explanations while maintaining correct predictions. Similarly, explanation-based membership inference attacks (EMIA) exploit explanation vectors like SHAP to infer whether a data sample was part of the training set [69]. In future work, we aim to improve SHAP-based secure aggregation and explore privacy-preserving techniques such as homomorphic encryption and differential privacy to enhance security in federated learning and improve system reliability in real-world applications. This integration of privacy-preserving approaches can help mitigate the risks of both inference attacks and adversarial manipulation of explanations. In FL, malicious clients can also send corrupted model updates that may compromise the integrity of the global model through model poisoning attacks. To address this issue, we plan to integrate robust aggregation techniques such as Krum, Trimmed Mean, and Median-based filtering [70] with FedAvg in future work.
- In FL settings, IoT devices typically have significantly fewer resources compared to server machines in data centres. These limitations include lower computational power, restricted communication bandwidth, limited memory, and smaller storage capacity. Training deep neural networks with millions of parameters on these resource-constrained edge devices can be extremely time-consuming and energy-intensive. Thus, achieving efficient FL, where a model is trained in a reasonable timeframe and with minimal energy consumption, can be quite challenging. In the future, we plan to explore techniques such as model pruning [71], which involves removing redundant parameters from pre-trained large models such as eliminating certain convolutional kernels or disconnecting some neurons, thereby achieving a lightweight model. In future work, we aim to implement adaptive client selection strategies [72] to effectively address the heterogeneity of client data distribution in our FedXAI framework.

7. Conclusion

In this study, we proposed an FEDXAI framework that extends the privacy guarantee of FL settings with the privacy of the post hoc local explanations that are induced for the server model. Thus, our framework achieves both transparency and privacy in FL-based solutions. We selected IoT botnet detection as a case study. One major challenge in post-hoc explainability methods, such as SHAP, is that they typically require access to raw client data to generate explanations, which poses privacy risks. To address this, our framework securely aggregates SHAP values on the client side, demonstrating that these aggregated explanations effectively approximate the feature attributions of the server model. Our findings highlight that securely aggregated client-side explanations can successfully explain the server model without depending on client data in FL settings.

We developed a high-performance LSTM-based model in an FL setting for binary classification, botnet-type detection, and attack-type detection. Our model achieved 99.90% accuracy in binary classification, 99.28% in botnet-type detection, and 94.89% in attack-type detection, outperforming existing FL-based IDS. It also exhibited high precision, recall, and F1 scores, effectively distinguishing between benign and malicious traffic, various botnet types, and multiple attack types. We evaluated LIME, SHAP, and Integrated Gradients (IG) as post hoc explanation methods for the client models. Our findings show that SHAP provided superior faithfulness, aligning closely with the predictions of the model. It also demonstrated lower complexity, and was more robust with lower Max Sensitivity for stable attributions across similar inputs, along with higher monotonicity.

In the future, we plan to explain the server-side model (aggregated model) more clearly by using an explanation with synthetic input data. We will assess whether the explanation derived from the synthetic data is adequate. Additionally, we aim to integrate differential privacy noise into the SHAP explainer without relying on any individual client's contributions.

CRediT authorship contribution statement

Rajesh Kalakoti: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Sven Nõmm:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Hayretdin Bahsi:** Writing – review & editing, Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] K.A. Da Costa, J.P. Papa, C.O. Lisboa, R. Munoz, V.H.C. de Albuquerque, Internet of things: A survey on machine learning-based intrusion detection approaches, Comput. Netw. 151 (2019) 147–157.
[2] R. Kalakoti, S. Nõmm, H. Bahsi, In-depth feature selection for the statistical machine learning-based botnet detection in IoT networks, IEEE Access 10 (2022) 94518–94535.
[3] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, ACM Trans. Inf. Syst. Secur. (TISSEC) 6 (4) (2003) 443–471.
[4] W. Ding, X. Jing, Z. Yan, L.T. Yang, A survey on data fusion in internet of things: Towards secure and privacy-preserving fusion, Inf. Fusion 51 (2019) 129–144.

[5] T. Iggena, E. Bin Ilyas, M. Fischer, R. Tönjes, T. Elsaleh, R. Rezvani, N. Pourshahrokhi, S. Bischof, A. Fernbach, J.X. Parreira, et al., Iotcrawler: Challenges and solutions for searching the internet of things, Sensors 21 (5) (2021) 1559.

[6] J. Konečnỳ, H.B. McMahan, D. Ramage, P. Richtárik, Federated optimization: Distributed machine learning for on-device intelligence, 2016, arXiv preprint arXiv:1610.02527.

[7] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, 2018, arXiv preprint arXiv:1806.00582.

[8] L.U. Khan, W. Saad, Z. Han, E. Hossain, C.S. Hong, Federated learning for internet of things: Recent advances, taxonomy, and open challenges, IEEE Commun. Surv. & Tutorials 23 (3) (2021) 1759–1799.

[9] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for internet of things: A comprehensive survey, IEEE Commun. Surv. & Tutorials 23 (3) (2021) 1622–1658.

[10] S.A. Rahman, H. Tout, C. Talhi, A. Mourad, Internet of things intrusion detection: Centralized, on-device, or federated learning? IEEE Netw. 34 (6) (2020) 310–317.

[11] Z. Abou El Houda, B. Brik, S.-M. Senouci, A novel IoT-based explainable deep learning framework for intrusion detection systems, IEEE Internet Things Mag. 5 (2) (2022) 20–23.

[12] S. Mane, D. Rao, Explaining network intrusion detection system using explainable ai framework, 2021, arXiv preprint arXiv:2103.07110.

[13] N. Capuano, G. Fenza, V. Loia, C. Stanzione, Explainable artificial intelligence in cybersecurity: A survey, Ieee Access 10 (2022) 93575–93600.

[14] I. Kök, F.Y. Okay, Ö. Muyanlı, S. Özdemir, Explainable artificial intelligence (xai) for internet of things: a survey, IEEE Internet Things J. 10 (16) (2023) 14764–14779.

[15] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, M. Seale, Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities, IEEE Access 10 (2022) 112392–112415.

[16] N. Moustafa, N. Koroniotis, M. Keshk, A.Y. Zomaya, Z. Tari, Explainable intrusion detection for cyber defences in the internet of things: Opportunities and solutions, IEEE Commun. Surv. & Tutorials 25 (3) (2023) 1775–1807.

[17] GDPR, General data protection regulation (GDPR) – official legal text, 2016, (Accessed 20 August 2024) https://gdpr-info.eu/.

[18] F. Yang, J. Xu, Privacy concerns in China's smart city campaign: The deficit of China's cybersecurity law, Asia & Pac. Policy Stud. 5 (3) (2018) 533–543.

[19] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. (TIST) 10 (2) (2019) 1–19.

[20] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.

[21] X. Hei, X. Yin, Y. Wang, J. Ren, L. Zhu, A trusted feature aggregator federated learning for distributed malicious attack detection, Comput. Secur. 99 (2020) 102033.

[22] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the KDD cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ieee, 2009, pp. 1–6.

[23] N.A.A.-A. Al-Marri, B.S. Ciftler, M.M. Abdallah, Federated mimic learning for privacy preserving intrusion detection, in: 2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), IEEE, 2020, pp. 1–6.

[24] A.K. Chathoth, A. Jagannatha, S. Lee, Federated intrusion detection for iot with heterogeneous cohort privacy, 2021, arXiv preprint arXiv:2101.09878.

[25] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, et al., Toward generating a new intrusion detection dataset and intrusion traffic characterization., ICISSp 1 (2018) 108–116.

[26] J. Li, L. Lyu, X. Liu, X. Zhang, X. Lyu, FLEAM: A federated learning empowered architecture to mitigate ddos in industrial IoT, IEEE Trans. Ind. Informatics 18 (6) (2021) 4059–4068.

[27] T.D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A.-R. Sadeghi, DÏoT: A federated self-learning anomaly detection system for IoT, in: 2019 IEEE 39th International Conference on Distributed Computing Systems, ICDCS, IEEE, 2019, pp. 756–767.

[28] T.V. Khoa, Y.M. Saputra, D.T. Hoang, N.L. Trung, D. Nguyen, N.V. Ha, E. Dutkiewicz, Collaborative learning model for cyberattack detection systems in iot industry 4.0, in: 2020 IEEE Wireless Communications and Networking Conference, WCNC, IEEE, 2020, pp. 1–6.

[29] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baIoT: Network-based detection of IoT botnet attacks using deep autoencoders, IEEE Pervasive Comput. 17 (3) (2018) 12–22.

[30] V. Rey, P.M.S. Sánchez, A.H. Celdrán, G. Bovet, Federated learning for malware detection in IoT devices, Comput. Netw. 204 (2022) 108693.

[31] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in: International Conference on Machine Learning, Pmlr, 2018, pp. 5650–5659.

[32] D.L. Marino, C.S. Wickramasinghe, M. Manic, An adversarial approach for explainable ai in intrusion detection systems, in: IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2018, pp. 3237–3243.

[33] M. Wang, K. Zheng, Y. Yang, X. Wang, An explainable machine learning framework for intrusion detection systems, IEEE Access 8 (2020) 73127–73141.

[34] L. Antwarg, R.M. Miller, B. Shapira, L. Rokach, Explaining anomalies detected by autoencoders using Shapley additive explanations, Expert Syst. Appl. 186 (2021) 115736.

[35] H. Liu, C. Zhong, A. Alnusair, S.R. Islam, FAIXID: A framework for enhancing AI explainability of intrusion detection results using data cleaning techniques, J. Netw. Syst. Manage. 29 (4) (2021) 40.

[36] D. Rao, S. Mane, Zero-shot learning approach to adaptive cybersecurity using explainable AI, 2021, arXiv preprint arXiv:2106.14647.

[37] P. Barnard, N. Marchetti, L.A. DaSilva, Robust network intrusion detection through explainable artificial intelligence (XAI), IEEE Netw. Lett. 4 (3) (2022) 167–171.

[38] P.P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, S.G. Teo, Detection and classification of botnet traffic using deep learning with model explanation, IEEE Trans. Dependable Secur. Comput. (2022).

[39] Y. Mirsky, T. Doitshman, Y. Elovici, A. Shabtai, Kitsune: an ensemble of autoencoders for online network intrusion detection, 2018, arXiv preprint arXiv:1802.09089.

[40] N. Basheer, B. Pranggono, S. Islam, S. Papastergiou, H. Mouratidis, Enhancing malware detection through machine learning using XAI with SHAP framework, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2024, pp. 316–329.

[41] R. Haffar, D. Sanchez, J. Domingo-Ferrer, Explaining predictions and attacks in federated learning via random forests, Appl. Intell. 53 (1) (2023) 169–185.

[42] T.T. Huong, T.P. Bac, K.N. Ha, N.V. Hoang, N.X. Hoang, N.T. Hung, K.P. Tran, Federated learning-based explainable anomaly detection for industrial control systems, IEEE Access 10 (2022) 53854–53872.

[43] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).

[44] E. Albini, J. Long, D. Dervovic, D. Magazzeni, Counterfactual shapley additive explanations, in: Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, 2022, pp. 1054–1070.

[45] V. Holubenko, D. Gaspar, R. Leal, P. Silva, Autonomous intrusion detection for IoT: a decentralized and privacy preserving approach, Int. J. Inf. Secur. 24 (1) (2025) 7.

[46] A. Graves, A. Graves, Long short-term memory, Supervised Seq. Label. Recurr. Neural Networks (2012) 37–45.

[47] M.T. Ribeiro, S. Singh, C. Guestrin, " why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.

[48] L.S. Shapley, A value for n-person games, Contrib. To Theory Games 2 (28) (1953) 307–317.

[49] E. Štrumbelj, I. Kononenko, Explaining prediction models and individual predictions with feature contributions, Knowl. Inf. Syst. 41 (2014) 647–665.

[50] S.M. Lundberg, G.G. Erion, S.-I. Lee, Consistent individualized feature attribution for tree ensembles, 2018, arXiv preprint arXiv:1802.03888.

[51] M. Sundararajan, A. Taly, Q. Yan, Axiomatic attribution for deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 3319–3328.

[52] A.A. Freitas, Comprehensible classification models: a position paper, ACM SIGKDD Explor. Newsl. 15 (1) (2014) 1–10.

[53] L. Longo, M. Brcic, F. Cabitza, J. Choi, R. Confalonieri, J. Del Ser, R. Guidotti, Y. Hayashi, F. Herrera, A. Holzinger, et al., Explainable artificial intelligence (XAI) 2.0: A manifesto of open challenges and interdisciplinary research directions, Inf. Fusion 106 (2024) 102301.

[54] D. Slack, S. Hilgard, E. Jia, S. Singh, H. Lakkaraju, Fooling lime and shap: Adversarial attacks on post hoc explanation methods, in: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 2020, pp. 180–186.

[55] A. Kuppa, N.-A. Le-Khac, Black box attacks on explainable artificial intelligence (XAI) methods in cyber security, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–8.

[56] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H.B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1175–1191.

[57] U. Bhatt, A. Weller, J.M. Moura, Evaluating and aggregating feature-based model explanations, 2020, arXiv preprint arXiv:2005.00631.

[58] R. Luss, P.-Y. Chen, A. Dhurandhar, P. Sattigeri, K. Shanmugam, C.-C. Tu, Generating contrastive explanations with monotonic attribute functions arXiv preprint arXiv:1905.12698 3 (2019).

[59] D.J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K.H. Li, T. Parcollet, P.P.B. de Gusmão, et al., Flower: A friendly federated learning research framework, 2020, arXiv preprint arXiv:2007.14390.

[60] T. Zhang, C. He, T. Ma, L. Gao, M. Ma, S. Avestimehr, Federated learning for internet of things, in: Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, 2021, pp. 413–419.

[61] T.G. Palla, S. Tayeb, Intelligent mirai malware detection for IoT nodes, Electron. 10 (11) (2021) 1241.

[62] A.A. Wardana, P. Sukarno, M. Salman, Collaborative botnet detection in heterogeneous devices of internet of things using federated deep learning, in: Proceedings of the 2024 13th International Conference on Software and Computer Applications, 2024, pp. 287–291.

[63] P.H. Do, T.D. Le, V. Vishnevsky, A. Berezkin, R. Kirichek, A horizontal federated learning approach to IoT malware traffic detection: An empirical evaluation with N-baIoT dataset, in: 2024 26th International Conference on Advanced Communications Technology, ICACT, IEEE, 2024, pp. 1494–1506.

[64] G.D.L.T. Parra, P. Rad, K.-K.R. Choo, N. Beebe, Detecting internet of things attacks using distributed deep learning, J. Netw. Comput. Appl. 163 (2020) 102662.

[65] B. Olanrewaju-George, B. Pranggono, Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models, Cyber Secur. Appl. 3 (2025) 100068.

[66] A. Oki, Y. Ogawa, K. Ota, M. Dong, Evaluation of applying federated learning to distributed intrusion detection systems through explainable AI, IEEE Netw. Lett. (2024).

[67] K. Fatema, M. Anannya, S.K. Dey, C. Su, R. Mazumder, Securing networks: A deep learning approach with explainable AI (XAI) and federated learning for intrusion detection, in: International Conference on Data Security and Privacy Protection, Springer, 2024, pp. 260–275.

[68] D. Attique, W. Hao, W. Ping, D. Javeed, M. Adil, EX-DFL: An explainable deep federated-based intrusion detection system for industrial IoT, in: 2024 21st International Joint Conference on Computer Science and Software Engineering, JCSSE, IEEE, 2024, pp. 358–364.

[69] H. Liu, Y. Wu, Z. Yu, N. Zhang, Please tell me more: Privacy impact of explainability through the lens of membership inference attack, in: 2024 IEEE Symposium on Security and Privacy, SP, IEEE, 2024, pp. 4791–4809.

[70] K. Pillutla, S.M. Kakade, Z. Harchaoui, Robust aggregation for federated learning, IEEE Trans. Signal Process. 70 (2022) 1142–1154.

[71] Y. Jiang, S. Wang, V. Valls, B.J. Ko, W.-H. Lee, K.K. Leung, L. Tassiulas, Model pruning enables efficient federated learning on edge devices, IEEE Trans. Neural Networks Learn. Syst. 34 (12) (2022) 10374–10386.

[72] L. Fu, H. Zhang, G. Gao, M. Zhang, X. Liu, Client selection in federated learning: Principles, challenges, and opportunities, IEEE Internet Things J. 10 (24) (2023) 21811–21819.

Appendix 10

X

R. Kalakoti, H. Bahsi, and S. Nõmm. Synthetic data-driven explainability for federated learning-based intrusion detection system. In *IEEE Internet of Things Journal*. IEEE, 2025

Synthetic Data-Driven Explainability for Federated Learning-based Intrusion Detection System

Rajesh Kalakoti, IEEE Graduate Student Member, Hayretdin Bahsi, Sven Nõmm

***Abstract*—An Intrusion Detection System (IDS) is vital for monitoring network traffic and alerting users to threats. Unlike traditional IDS, which relies on centralized data processing and raises privacy concerns, Federated Learning (FL)-based IDSs enable collaborative model training among multiple clients while keeping user data private. However, explaining model behavior in FL using Explainable AI (XAI) is challenging due to its distributed nature and lack of access to client data. Traditional XAI methods like LIME and SHAP require input data, which conflicts with FL's privacy constraints. In this work, we develop a deep neural network (DNN)-based IDS in FL setup in NoN-IID settings. Our FL-DNN model achieves high performance in binary classification for detecting malicious network traffic. In this work, we propose a novel privacy-preserving, explainable federated learning framework that uses high-quality synthetic data to enable explainability of the Global DNN model without exposing client data to the server. To generate synthetic data, we train multiple federated generative models in Non-IID settings. Among them, the Federated Wasserstein Conditional GAN with Gradient Penalty (FL-WCGAN-GP) produces synthetic samples with high data quality at the server. These synthetic samples on the server side are then used as reference inputs for post-hoc XAI methods for explaining the global DNN model. We assess the sufficiency of synthetic data-based explanations for the Global DNN model using SHAP, showing that Synthetic data-based explanations closely approximate the explanations derived from real client data. Further, we quantitatively evaluate post-local explanations of LIME and SHAP based on faithfulness and robustness. Results show that SHAP provides more faithful and robust explanations than LIME for client-side models using real data and server-side models using synthetic data, supporting privacy-preserving explainability in FL-based IDS.**

***Index Terms*—Federated Learning, Explainable AI, Intrusion detection System, Synthetic Data, Generative Adversarial Networks,**

I. INTRODUCTION

Intrusion detection systems (IDS) are vital for cybersecurity and are commonly used in IoT edge devices like routers and switches [1]. IDS helps detect malicious activities that threaten the confidentiality, integrity, and availability (CIA) of information assets [2]. The rise of IoT devices has increased opportunities for attackers, who may target these devices directly or use them as part of their attack infrastructure. Consequently, an IDS must detect attacks on IoT devices to trigger appropriate countermeasure actions.

Manuscript Submitted on March 3 (Corresponding author: Rajesh Kalakoti.)

Rajesh Kalakoti and Sven Nõmm are with the Department of Software Science at Tallinn University of Technology (TalTech), Estonia, 12616. (emails: rajesh.kalakoti@taltech.ee, sven.nomm@taltech.ee)

Hayretdin Bahsi is with the Department of Software Science at Tallinn University of Technology (TalTech), Estonia, 12616 and School of Informatics, Computing, and Cyber Systems at Northern Arizona University, USA, 86011. (email:hayretdin.bahsi@taltech.ee)

IDSs typically employ either signature-based or anomaly-based detection models [3]. Signature-based systems rely on predefined rules created by experts, whereas anomaly-based systems model normal behavior and detect deviations. Recent advances in machine learning (ML) and deep learning (DL) have enabled more effective approaches to traffic classification and ID [4, 5]. ML/DL-based IDSs analyze network traffic, system logs, or host data to identify threats, continuously adapt to evolving attacks through retraining, and issue alerts for detected anomalies [2, 4]. However, these systems often depend on centralized ML models that require access to large volumes of private data from IoT edge devices, raising significant privacy concerns [5, 6]. In privacy-sensitive domains such as healthcare and finance, transferring local data to a centralized server for model training is often infeasible due to regulatory and confidentiality constraints.

FL , introduced by Google in 2016 [7], enables collaborative training of machine learning models across distributed nodes while preserving data privacy. In FL, each client updates the model locally using its private data and shares only model parameters with a central server, which aggregates them to build a global model through an iterative, round-based process. This decentralized approach eliminates the need to share raw data, thereby enhancing privacy. Recent studies have applied FL to IDS, demonstrating strong anomaly detection performance while preserving client data confidentiality [8, 9].

Many ML approaches have demonstrated strong performance in cybersecurity tasks. However, their lack of transparency, particularly in DL models, raises concerns about trust and transparency. Security professionals often struggle to rely on the outcomes of ML-based IDSs because ML-based IDS processes are not clearly understandable. To address this, recent research has focused on Explainable Artificial Intelligence (XAI), which aims to make the reasoning of the black box nature of ML systems more explainable or transparent [10]. XAI techniques are generally classified into global explanations, which describe overall model behavior, and local explanations [11], which provide insights into specific predictions. These methods can be either model-agnostic, applicable to explain the whole behavior of a Model, or model-specific, designed for particular types of models such as decision trees [11].

Trustworthy Artificial Intelligence (AI) has become increasingly crucial in cybersecurity to enhance security and reliability, particularly in IDS. Recently, the concept of Fed-XAI has emerged, which integrates FL with XAI paradigms [12, 13]. The combination of these frameworks is essential for building trust in AI systems, as it enables the simultaneous pursuit of

TABLE I: Notation Descriptions

Notation	Description
N	Number of clients
c_i	i^{th} client in FL
$\mathbb{D}_{c_i}$	Local dataset of client c_i
G	Generator network
D	Discriminator (or critic) network
P_r	True (real) data distribution
P_g	Model (generated) data distribution
$x \sim P_r$	Real data sampled from P_r
$\tilde{x} \sim P_g$	Data generated by the generator G
$z \sim p(z)$	Latent noise input to generator
θ	Model parameters in FL
y	Auxiliary conditioning variable (e.g., class label)
$\tilde{x} \mid y$	Conditional generated sample given label y
η	Learning rate
$\mathcal{E}$	Number of training epochs
B	Mini-batch size
$\mathcal{M}$	Trained black-box model ($\mathcal{M}_s$ for server, $\mathcal{M}_{c_i}$ for client)
$L(D)$	Discriminator loss function
$L(G)$	Generator loss function
$g(\mathcal{M}, x)$	Explanation score vector returned by the explainer for input x and $\mathcal{M}$

transparency while preserving privacy. In a recent paper [14], the authors summarize the few contributions where XAI models have been trained in a federated manner, along with some real-world applications of Fed-XAI [13]. Two key paradigms that support the demands of trustworthy IDS using DL model are FL and XAI. FL focuses on maintaining data privacy by enabling the collaborative learning of an IDS system using decentralized data. In contrast, XAI ensures transparency, accountability, and trust in FL-based IDS by providing clear explanations for its predictions and decisions, which help security analysts understand and verify detected threats.

The decentralized nature of FL poses significant challenges for IDS in terms of oversight and transparency, which are both critical for trustworthy threat detection. Enabling expert understanding of model behavior, mitigating bias in detection decisions, and maintaining transparency becomes more complex in FL settings, where visibility into local training data and model behavior across distributed clients is inherently limited. Post hoc explainability methods that are model-agnostic have gained attention in research for their wider applicability. Notable methods like LIME and SHAP offer detailed explanations of given individual instances and their model outputs. Specifically, they assign importance scores to features that significantly influence model decisions. This allows experts to gain insights into the model's output by verifying whether these features are relevant within the context of the related cyber incident. In an IDS running in an FL setting, security experts operating the server-side models to analyze the intrusions to client nodes are still required to understand the model decision-making. In these scenarios, balancing the need for both privacy and explainability poses a considerable challenge. Incorporating XAI methods into FL is an area that has not yet received much attention and presents additional challenges due to FL's complex, distributed nature, where the model is trained across multiple client nodes (IoT devices, sensors). Also, post-hoc XAI methods, such as feature importance XAI methods (e.g., LIME or SHAP), and model interpretability typically require access to the complete training dataset (input reference data) and the trained model parameters, which can pose privacy risks. Creating a unified explainability framework for server-side and client models in FL is challenging due to data decentralization and limited server access to client data. Moreover, new data privacy laws, such as the General Data Protection Regulation (GDPR) [15] in the EU and China's Cybersecurity Law [16], aim to protect user privacy by restricting the sharing and central collection of data. Using data from multiple parties to train ML models while complying with privacy regulations is an urgent issue.

To address the above challenges, this work prioritizes trustworthiness, transparency, and accountability in IDS by integrating responsible AI practices and explainability into the FL paradigm. Although post-hoc explainability methods have been widely used in centralized ML settings, their application in FL has not yet been studied. This study aims to bridge this gap by proposing a novel framework that enables the explanation of IDS models in FL environments without compromising client data privacy. To the best of our knowledge, this is the first work to utilize synthetic data generated through a federated generative model that enables server-side explainability of FL-based IDS systems, marking the first application of synthetic data-driven explainability in the field of cybersecurity.

Generative modeling has recently gained attention as a practical approach to address class imbalance in cybersecurity applications [17] [18] [19]. These models typically involve a Generator (G) that produces synthetic minority class samples and a discriminator (D) that distinguishes between real and generated data. Once trained, the generator can produce synthetic instances that help balance class distributions and potentially represent unseen attack patterns. However, it is essential to note that existing literature primarily uses this synthetic data to tackle class imbalance, rather than to enhance transparency or explainability in black-box models through XAI.

Post-hoc explainability methods rely on real client data as input reference samples to furnish explanations for black-box models. However, in FL, direct access to real client data at the server level is restricted due to privacy concerns. To enhance the explainability of the global model in FL-settings for IDS, we propose the use of federated synthetic data. Instead of relying on real input reference data from participating clients, we generate federated synthetic data using a generative modelling within the FL framework. These FL-based synthetic samples serve as input reference data for post-hoc explainability methods, enabling the explanation of the global model on the server side without exposing real client data.

Specifically, in this work, we develop a FL-based deep neural network (FL-DNN) classifier to distinguish between malicious and benign network traffic in intrusion detection tasks under non-IID conditions, simulating real-world data heterogeneity across clients.

To enhance the explainability of the global DNN model on the server side while maintaining client privacy, we propose

using federated synthetic data instead of real client data. This synthetic data is generated in a decentralized way using generative modeling within FL-setup. For this purpose, we train multiple generative model variants, including FL-GAN, FL-Conditional GAN (FL-CGAN), FL-Wasserstein GAN with Gradient Penalty (FL-WGAN-GP), and FL-Wasserstein Conditional GAN with Gradient Penalty (FL-WCGAN-GP), to generate high-quality synthetic data in a federated setting. The global generator, aggregated at the server, produces federated synthetic samples that serve as the input reference data for post-hoc explainability methods. This allows for server-side interpretation of the global FL-DNN model without direct access to sensitive client data.

In addition to proposing a privacy-preserving explainability framework, this work evaluates the quality of the explanations generated for the global FL-DNN model using federated synthetic data. To this end, we employ two widely adopted post-hoc explainability methods, LIME and SHAP, which provide local explanations by identifying the most influential input features for a given prediction. These explainers are applied to the global model at the server side, using synthetic data as input reference samples in place of real client data. Our evaluation emphasizes the importance of explanation quality alongside detection performance in federated IDS systems. we evaluate the reliability of the generated explanations based on two quantitative criteria: i)Faithfulness (how accurately the explanations reflect the true behavior of the model), ii) Robustness (how consistent the explanations remain under small perturbations of the input data). By incorporating these criteria, we demonstrate that synthetic data can serve as a viable substitute for real data in evaluating the explainability of server-side models in FL.

The contribution of this work can be summarized as follows.

1) Developed an FL-DNN classifier to distinguish network traffic between Malicious and Benign and evaluated an FL-DNN classifier using accuracy, precision, recall, and F1-score over the NSL-KDD , UNSW-NB15, CIC-IoT2023 & CIC-IoMT2024 datasets.
2) Developed several federated geneative model variatns, including Standard GAN (Generative Adversarial Network), FL-CGAN (Conditional GAN), FL-WGAN-GP (Wasserstein GAN), and FL-WCGAN-GP (Wasserstein Conditional GAN) for the purpose of generating synthetic data. Evaluated server-side synthetic data generated by global generators based on these FL-GAN variants and selected the best variant for producing high-quality synthetic data.
3) Global DNN classifier of FL was explained using synthetic data as input reference for the post hoc explainability technique LIME and SHAP. We evaluated synthetic data-driven explainability by providing client-based real data to the server and analyzing the differences between real data-based explanations and those derived from synthetic data. Our findings indicate that synthetic data-driven explanations closely approximate real data-based explanations, demonstrating the viability of using high-fidelity synthetic data for model explainability while preserving data privacy in FL settings.
4) Evaluation of the quality of local explanations generated by two widely used post-hoc XAI methods, LIME and SHAP, for explaining server-side model decisions using synthetic data and client-side model decisions using real data. The evaluation is based on quantitative metrics, including faithfulness and sensitivity.

By incorporating Post hoc methods in FL, we significantly enhance transparency in decision-making, which empowers stakeholders to understand and interpret black box model outputs in FL-based IDS effectively.

The structure of the research work is outlined as follows. Section II offers a literature review on XAI for centralized IDS and XAI for FL-based IDS. Section IV details the proposed methodology. The results of the study are presented in Section V, while Section VI provides the discussions of the main findings of the research. Finally, Section VII concludes the work.

II. Related work

A. Explainable AI for Centralized Intrusion Detection Systems

Many applications require both high accuracy and clear explainability of results. To achieve this, various methods have been proposed to interpret the predictions of complex models in centralized settings for IDS task. Most works on XAI techniques for IDS primarily focus on visualization and model or prediction verification. Barnard et al. propose a two-stage network IDS, starting with a supervised XGBoost model for binary classification of network flow data and using SHAP for prediction explanations. The second stage trains an autoencoder that inputs SHAP explanations from the previous stage [20]. The central hypothesis being tested was whether the system can use the first stage to distinguish between normal and anomalous flows and the second stage to differentiate known behaviour from unknown behaviour. The proposal was evaluated on NSL-KDD dataset [21]. However, the second stage relies on the first, which requires labeled data, and does not consider the characterization of various attack behaviour clusters in the explanations.

Liu et al. proposed FAIXID, a framework designed to integrate explainability in IDS across various layers[22]. These layers encompass data cleaning, interpreting a trained, supervised model, providing local explanations for predictions, and displaying results to security analysts utilizing various visualizations relying on the role of each analyst.

Oseni et al. created an IDS for IoT-enabled transportation systems using DL and XAI[23]. They implemented a convolutional neural network (CNN) for threat detection and utilized the SHAP framework to explain the features influencing the IDS's decisions.

In a paper [24], the authors used a filter-based feature selection to select key features and they tested two models, a Deep Neural Network (DNN) and a Convolutional Neural Network (CNN), for IDS, with a focus on classifying attacks in the NSL-KDD [21] and UNSW-NB15 datasets [25]. To explain the predictions made by the DNN model, the authors utilized LIME and SHAP for local explanations and used SHAP for global explanations to identify DNN model features.

Neupane et al. focused XAI in IDS and studied AI-driven SOC (Security Operation Center) to promote transparency and accountability [26]. While the research improved the understanding of AI explainability for SOC analysts through various XAI techniques

Eriksson and Grov conducted a human-centered study on the use of XAI in SOCs [27], focusing on SHAP and LIME. They found that XAI can enhance analysts' trust in their machine-learning models, although its complexity may pose a barrier.

Kalakoti et al. proposed an active learning model aimed at identifying IoT botnet attacks by utilizing explainability methods such as LIME and SHAP [28]. Although this model improves detection capabilities, its dependence on labeled data and the complexity of integrating explainability methods may hinder its effectiveness in real-time applications.

Wang et al. propose a framework using SHAP for local and global explanations of IDS, helping security analysts interpret predictions [29]. Explanations for two supervised models trained on the NSL-KDD dataset [21] were compared. The authors demonstrate how different attack types create varied SHAP value patterns but focus mainly on visualization without analyzing these values for further information.

Sudheera et al. developed ADEPT, a framework for detecting network flow anomalies and identifying attack stages in a distributed IoT network with multiple clients and a centralized server [30]. It operates in three phases: each client detects anomalous flows and sends them to the central server, which uses Frequent Itemset Mining (FIM) to analyze the data. Although explainability is not a focus, the patterns extracted through FIM were explainable. Finally, the server classifies malicious flows into attack stages using supervised learning, requiring labeled data. While this distributed approach improves privacy and reduces bandwidth compared to centralized systems, it still transmits potentially sensitive anomalous data to the central server. In contrast, FL architectures can enhance privacy and data reduction while enabling client collaboration.

The above-mentioned all studies focus on using post-hoc explainability techniques for centralized IDS models, where the server has direct access to actual client data for training and explanation purposes. However, these centralized approaches do not consider privacy constraints in collaborative learning environments. Despite recent advancements, significant gaps remain in the literature on XAI for IDSs, particularly regarding standardized evaluation. A major limitation is the lack of quantitative metrics to assess the effectiveness and utility of XAI methods in the intrusion detection domain. While prior studies have initiated efforts to incorporate explainability into IoT botnet detection, there is still a notable absence of rigorous, quantitative evaluation of XAI methods in cybersecurity. This underscores the need for systematically assessing the quality of generated explanations to build trust in XAI-based systems [20, 22, 24, 26, 27, 29, 30].

B. Explainable AI for FL-based Intrusion Detection Systems

FedXAI (Federated Learning of Explainable AI) concept aims to enhance user trust in AI systems by addressing privacy preservation through FL and explainability through XAI models and techniques. Research in this area is expanding, focusing on both post-hoc methods [31, 32] and explainable-by-design models [33]. The adoption of post-hoc explanations in FL context is far from trivial. Recent advances have been made in both FL and XAI for IDS tasks. However, very few studies have explored the integration of XAI into FL settings. Furthermore, some studies do not address privacy concerns when explaining the global model (server-side model) using post-hoc explainable AI methods.

Huang et al. propose an FL architecture for anomaly detection in industrial control systems, using SHAP for model explanation and visualization for domain experts[8]. However, the SHAP model explainer is not trained federated way, and while Post-hoc explainer like SHAP requires background data as a baseline, the authors do not explain how this baseline is obtained, which is crucial given the data's distributed nature in FL. Oki et al. utilized SHAP for model explainability. However, the feature contributions were assessed centrally on the global model [34].

Kalakoti et al. proposed a method to explain the server model in HFL using FedAvg without direct access to client data [9]. Their approach aggregates SHAP values from individual client models to approximate explanations for the server model and was evaluated on the N-BaIoT dataset by comparing aggregated client-based explanations with server-based ones derived from actual client data. While effective, this method is computationally expensive, as it requires computing SHAP values at each client, which becomes increasingly costly as the number of participating IoT devices increases.

Haffar et al. use random forests (RF) as substitutes for the supervised FL model[35]. Each client trains an RF with its local data. When the FL model misclassifies a sample, they analyze the RF trees to compute feature importance values, which help detect and explain attacks on the FL training process. The explanations are performed at the client level and require labeled training data. Each client has its own explainer model, which might differ from the rest as they are trained independently and not in a federated way, making the interpretation of the explanations for the global model difficult. They focus on detecting potential attacks against the FL training process rather than explaining and characterizing predictions.

Some studies apply explainable AI methods in FL to detect adversarial attacks [35] or validate models [36]. However, explainer models are usually not developed federated, which may violate FL assumptions or require different explainers for each client, complicating network-wide interpretation. None of this works using SHAP in FL. Current research on using SHAP in FL lacks a thorough discussion on how to extract a baseline, which is essential for generating explanations. This selection of a baseline is critical because the explanations depend on it [37, 38]. One paper employs [39] SHAP and LIME in federated settings but fails to address the privacy risks associated with explaining the server-side global model. Previous research has not sufficiently addressed the privacy risks related to providing explainability for the global (server) model. Since post-hoc XAI methods depend on data points for generating explanations, the absence of privacy protections

raises significant concerns about potential data exposure.

Most existing studies do not clearly define how baseline reference data is used for post-hoc XAI methods in FL and often rely on client-side SHAP explainers, which are computationally expensive and challenging to scale. Additionally, many of these studies do not quantitatively evaluate the quality of the generated explanations or address the privacy risks associated with accessing real client data for explanation purposes. Our study advances explainability by utilizing federated synthetic data, which eliminates the need to rely on client data. Unlike previous research that assumes centralized access to feature attributions, we present a synthetic data approach for explaining the global model in a federated setting. Our method shows that post-hoc explanations can be effectively generated without compromising privacy, thereby ensuring a secure and explainable FL-based IDS.

III. Overview of Generative Adversarial Networks

Generative Adversarial Networks (GANs), introduced by Goodfellow et al. in 2015, are deep generative models that operate as a two-player game [40]. They consist of two neural networks that update their weights to enhance data generation and discrimination capabilities.

Let z represent the random noise input, and $z \mid y$ denote the generator's input conditioned on the label y. Similarly, x and $x \mid y$ refer to the actual training data and the actual training data conditioned on the label y, respectively. The symbol $\tilde{x}$ represents the data generated by a standard GAN, while $\tilde{x} \mid y$ indicates the data generated by a conditional GAN conditioned on the label y. Fig. 1 shows an overview of both the standard GAN and the conditional GAN (CGAN).

Weight updates are performed through backpropagation, with the discriminator fixed during generator updates. The key difference between Wasserstein GANs (WGAN) and standard GANs is in loss calculation and weight updates. In our work, we have developed four different types of generative model variants in FL settings.

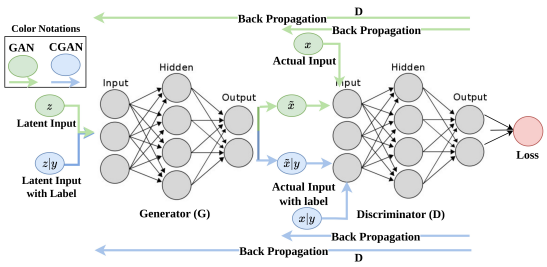


Fig. 1: Architecture of Basic GAN and CGAN

A. GAN & CGAN

The Generative Adversarial Network (GAN) training procedure involves a game played between two competing neural networks: the generator network G and the discriminator network D. The generator G transforms a source of random noise into samples that resemble the input data space. Meanwhile, the discriminator D aims to differentiate between samples that come from the actual data distribution and those generated by G

These two networks engage in a two-player minimax game with the following equation

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1-D(x))] \quad (1)$$

Here, P_r is the true data distribution, and P_g is the generative data distribution, which is implicitly defined by $x = G(z)$ where z is sampled from a simple noise distribution $p(z)$ (such as uniform, normal, or Gaussian distribution). The discriminator D is optimized to maximize the probability of correctly classifying both training samples and samples generated by G. Conversely, the generator G is optimized to minimize $\mathbb{E}_{x\sim P_g}[\log(1-D(x))]$ or $\mathbb{E}_{x\sim P_g}[-\log D(x)]$.

The Conditional Generative Adversarial Network (CGAN) is a variation of the GAN that incorporates additional information, y, into both the generator and discriminator. This information y can represent a class label or any other form of auxiliary data. Furthermore, the training process for CGAN is the same as that used in GAN.

Formally, the objective function for the generator G and the discriminator D is defined by the following minimax equation:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_r}[\log D(x|y)] + \mathbb{E}_{x\sim P_g}[\log(1-D(x|y))] \quad (2)$$

In this equation, P_r and P_g retain the same meanings as in GAN, and y is combined with the prior noise as inputs to the hidden layer. The optimization process in CGAN is almost similar to that of GAN.

Since CGANs are an extension of the original GAN variants, they share some of the same challenges, such as mode collapse and unstable training due to vanishing gradients, among other issues. Additionally, researchers have noted that when both GANs and CGANs utilize the Jensen–Shannon (JS) divergence as a metric for generative samples, they are limited to generating only continuous data, rather than discrete data [41].

$$\mathrm{JS}(P_r, P_g) = \mathrm{KL}(P_r \| P_m) + \mathrm{KL}(P_g \| P_m) \quad (3)$$

where KL denotes the Kullback–Leibler divergence, and P_m is the mixture distribution defined as $P_m = \frac{1}{2}(P_r + P_g)$.

B. WGAN & WGAN-GP

Unlike GAN and CGAN, WGAN and WGAN-GP utilize the Earth-Mover (EM) distance, also known as Wasserstein-1, instead of the JS divergence to measure the distance between the true data distribution and the generative data distribution, which is because the EM distance offers better smoothness compared to the JS divergence. Theoretically, WGAN addresses the vanishing gradient problem commonly

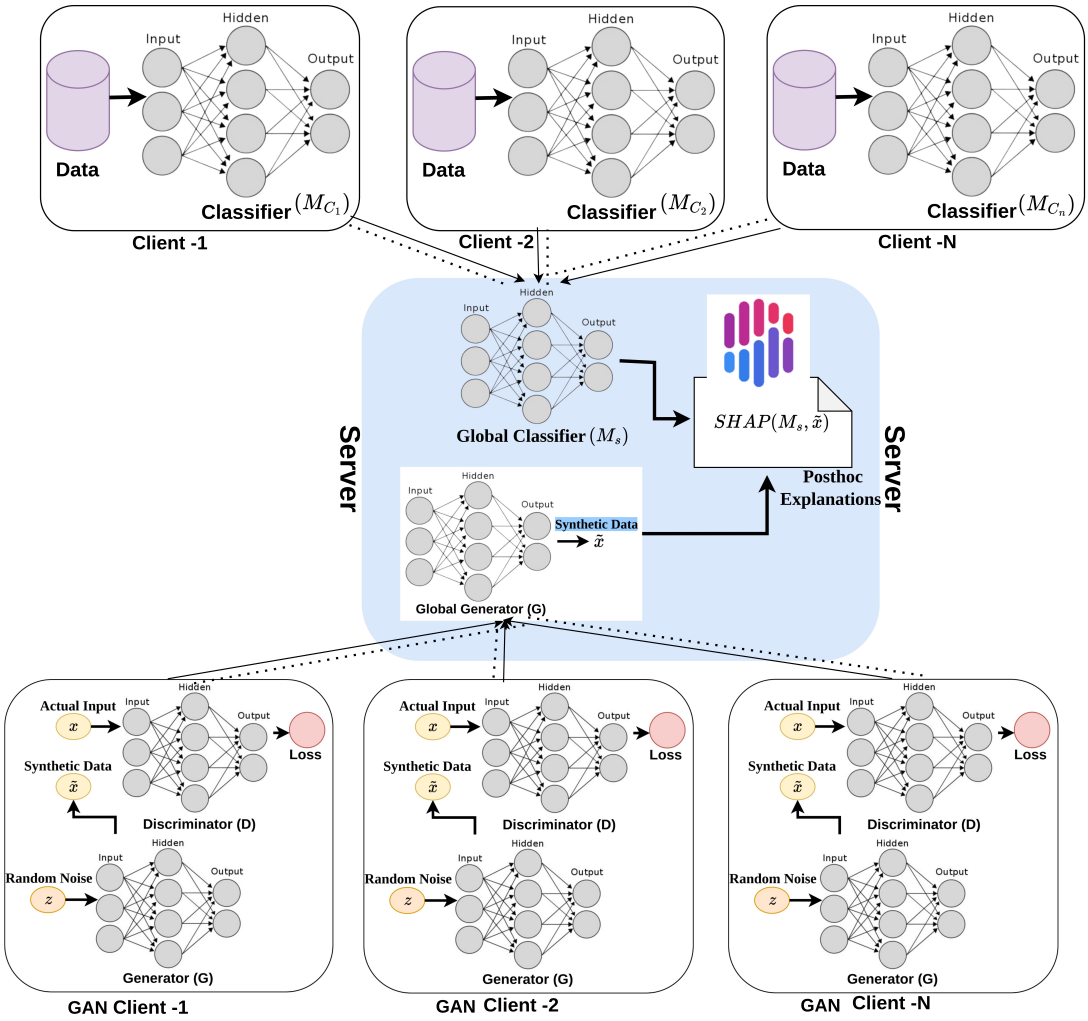


Fig. 2: Synthetic Data-Driven Explainability Framework for Explaining Blackbox Classifier in FL settings

observed in GAN and CGAN. Moreover, studies have demonstrated [41, 42] that replacing the JS divergence with the EM distance also helps mitigate the issue of mode collapse.

$$W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[\|x - y\| \right] \tag{4}$$

Where $\Pi(P_r, P_g)$ is the complete set of feasible joint distributions $\gamma(x, y)$ for the true data distribution P_r and the generative data distribution P_g. The term $W(P_r, P_g)$ is defined as the minimum cost required to transport mass in order to transform the distribution P_r into the distribution P_g. Additionally, under mild assumptions, $W(P_r, P_g)$ is continuous and differentiable almost everywhere. However, Equation 4 is highly intractable; therefore, the EM distance can be reconstructed using the Kantorovich-Rubinstein duality [43].

$$W(P_r, P_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{x\sim P_g}[f(x)] \tag{5}$$

Here, the supremum is over all K-Lipschitz functions $f : x \to \mathbb{R}$, with K being the Lipschitz constant (set to 1 for the original WGAN). This forms the minimax objective for the generator and critic shwon in Equation 4

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{x\sim P_g}[D(x)] \tag{6}$$

WGAN often struggles with producing high-quality samples and can fail to converge due to weight clipping, which is an ineffective method for enforcing a Lipschitz constraint on the discriminator. To solve this, Gulrajani et al. proposed

WGAN-GP, which uses a penalty on the gradient norm of the discriminator instead of weight clipping. Their results showed that WGAN-GP outperforms the standard WGAN and allows stable training of various GAN architectures with minimal hyperparameter tuning, represented by the minimax formulation in Equation 7

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_r}[D(x)] - \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x})] - \lambda\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2\right] \quad (7)$$

where λ is the gradient penalty coefficient, and $\hat{x}$ is sampled along straight lines between the true data distribution P_r and the generative data distribution P_g: $\hat{x} = \epsilon x + (1-\epsilon)\tilde{x}$, where $\epsilon \sim \text{Uniform}[0,1]$, $x \sim P_r$, and $\tilde{x} \sim P_g$.

C. WCGAN-GP

The development of WGAN-GP can extend to WCGAN-GP by conditioning both the discriminator and generator on auxiliary information y, which represents class labels in this study.

In the discriminator, we concatenate the real data distribution P_r and the generated data distribution P_g with y to form a joint hidden representation. The generator also concatenates y with $p(z)$ in the same way. The objective function for the generator and discriminator is defined as a minimax problem below

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim P_r}[D(x|y)] - \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x}|y)] - \lambda\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x}|y)\|_2 - 1\right)^2\right] \quad (8)$$

Here, λ denotes the gradient penalty coefficient, and the sampling strategy for $\hat{x}$ follows that of WGAN-GP. The discriminator and generator loss functions are defined as:

$$L(D) = -\mathbb{E}_{x\sim P_r}[D(x|y)] + \mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x}|y)] + \lambda\, \mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x}|y)\|_2 - 1\right)^2\right] \quad (9)$$

$$L(G) = -\mathbb{E}_{\tilde{x}\sim P_g}[D(\tilde{x}|y)] \quad (10)$$

IV. Proposed Methodology

Fig. 2 shows the proposed framework of this study. In this work, we propose a dual FL framework that integrates two key components: 1) a federated DNN for IDS(FL-DNN), and 2) federated synthetic data generation for explaining global DNN model in FL.

FL-DNN is designed to classify network traffic into benign or attack categories while preserving client data privacy. Each client trains a local DNN on its traffic data, and model updates are aggregated at the server using the FedAvg algorithm to construct a global FL-DNN.

For explainability, federated generative models are employed to generate synthetic data at the server. Each client participates in training a GAN-based model, and their updates are used to build a global generator. The synthetic data generated by this global generator serves as reference input for post-hoc explainability methods such as SHAP, which are applied at the server to interpret the decisions made by the global DNN model without accessing real client data.

A. Deep Neural Network

To achieve accurate classification of network traffic in IoT edge devices in FL, we propose a deep neural network (DNN) architecture that learns hierarchical representations through multiple layers of abstraction. The model consists of an input layer, multiple densely connected hidden layers, and an output layer. The number of input neurons, d, equals the number of features representing a single network traffic packet. The number of hidden layers and their corresponding neurons is determined empirically through experimentation to obtain an optimal DNN architecture.

The transformation at the first hidden layer for an input x is defined as:

$$h_1 = \sigma_h(W_1 x + b_1) \quad (11)$$

where σ_h is the activation function (ReLU), W_1 is the weight matrix, and b_1 is the bias vector of the first hidden layer.

For each subsequent hidden layer, the output h_{i+1} is computed as:

$$h_{i+1} = \sigma_h(W_i h_i + b_i) \quad (12)$$

where W_i and b_i are the weight matrix and bias vector of the i^{th} hidden layer, respectively. All hidden layers use the rectified linear unit (ReLU) activation function due to its training efficiency and effectiveness. The weight matrices are initialized using the He uniform initialization method, which is specifically designed for ReLU activations.

The final output $\tilde{y}$ is computed from the last hidden layer output h_j as:

$$\tilde{y} = \sigma_y(h_j) \quad (13)$$

For binary classification, the output layer contains two neurons ($C = 2$), corresponding to the two class labels. A softmax activation function is applied to convert the raw outputs into a probability distribution over the classes:

$$\sigma_y(h)_\alpha = \frac{e^{h_\alpha}}{\sum_{\beta=1}^{C} e^{h_\beta}} \quad (14)$$

where h_α is the activation corresponding to class α, and C is the number of class labels.

The model is trained using the Adam optimizer [44] with categorical cross-entropy as the loss function, enabling efficient first-order gradient-based stochastic optimization.

B. Federated Deep Neural Network

FL trains ML models across distributed devices without sharing raw data. Clients send model parameters or gradients to a central server, which combines the local updates by averaging them. The updated global model is then sent back to the clients, allowing each device to learn collaboratively while keeping sensitive data securely stored on the client side.

We denote N as the number of clients, where each client is represented by $c_1, c_2, \ldots, c_N$. The set of samples owned by client c_i is denoted as U_{c_i}, the feature space as X_{c_i}, the label space as Y_{c_i}, and the dataset as $\mathbb{D}_{c_i} = \{(u_{c_i}^{(j)}, x_{c_i}^{(j)}, y_{c_i}^{(j)})\}_{j=1}^{|\mathbb{D}_{c_i}|}$. Each data point $(u_{c_i}^{(j)}, x_{c_i}^{(j)}, y_{c_i}^{(j)})$ indicates that client c_i owns the sample $u_{c_i}^{(j)}$, with features $x_{c_i}^{(j)}$ and label $y_{c_i}^{(j)}$. In a network traffic ID scenario, U represents individual network flows, X includes features like packet size and protocol type, and Y indicates if traffic is benign or malicious. Based on how the data (X, Y, U) is partitioned across clients, FL can be categorized into three types by Yang et al. [45].

- Horizontal FL (HFL): In HFL, clients share the same feature and label space ($X_{c_i} = X_{c_j}$, $Y_{c_i} = Y_{c_j}$) but have different sample spaces ($U_i \neq U_j$).
- Vertical FL (VFL): In VFL, some common data samples are shared among clients ($U_{c_i} \cap U_{c_j} \neq \emptyset$), but the space between features and labels differs ($X_{c_i} \neq X_{c_j}$, $Y_{c_i} \neq Y_{c_j}$).
- Federated Transfer Learning (FTL): FTL does not impose restrictions on the sample, feature, and label space, allowing arbitrary differences.

Algorithm 1: FedAvg for FL-DNN based IDS

Input: N, $\theta^{(0)}$, η, $\mathbb{D}_{c_i}$, $\mathcal{E}$
Output: $\theta^{(R)}$

Initialize: $\theta^{(0)} \rightarrow \{c_1, c_2, \ldots, c_N\}$
for $r = 1$ ***to*** R **do**
  for $c_i \in \{c_1, \ldots, c_N\}$ **do**
    Receive $\theta^{(r-1)}$
    for $e = 1$ ***to*** $\mathcal{E}$ **do**
      Sample $B \subset \mathbb{D}_{c_i}$
      /*Local optimization */
      $\theta_{c_i}^{(r)} \leftarrow \theta_{c_i}^{(r-1)} - \frac{\eta}{|B|} \sum_{(x,y)\in B} \nabla_\theta l(x, y; \theta_{c_i}^{(r-1)})$
    Send $\theta_{c_i}^{(r)}$ to server
  Server aggregation:
  $\theta^{(r)} \leftarrow \sum_{i=1}^{N} w_{c_i} \theta_{c_i}^{(r)}$ where $w_{c_i} = \frac{|\mathbb{D}_{c_i}|}{\sum_{j=1}^{N} |\mathbb{D}_{c_j}|}$
  Distribute $\theta^{(r)} \rightarrow \{c_1, c_2, \ldots, c_N\}$
return $\theta^{(R)}$

In our work, a federated DNN model is proposed for classifying network traffic as malicious or benign in an HFL setup, as described in Algorithm 1. The server initializes the global DNN model $\mathcal{M}_s$ with parameters $\theta^{(0)}$ and distributes them to N clients. Each client c_i receives the parameters $\theta^{(r-1)}$ at round r, samples a mini-batch $B \subset \mathbb{D}_{c_i}$, and trains a local DNN model $\mathcal{M}_{c_i}$ over $\mathcal{E}$ epochs using the Adam optimizer. The locally updated parameters $\theta_{c_i}^{(r)}$ are then sent back to the server. The server aggregates the updates using the FedAvg algorithm [7], computing a weighted average to obtain $\theta^{(r)}$, which is then sent back to all N. This process is repeated for R communication rounds until convergence.

C. Federated Generative model

To enable explainability without accessing real client data in FL settings, we rely on synthetic data generated through a federated generative model. Specifically, we train multiple variants of federated generative models, where each client maintains a local model that learns to generate synthetic network traffic samples based on its local dataset. The local generator parameters are then sent to the server, where they are aggregated using FedAvg to construct a global generative model. This global model is subsequently used to generate synthetic data at the server, serving as reference input for a post-hoc XAI method to explain the global FL-DNN model.

Algorithm 2: Federated generative model

Input: N, $\mathbb{D}_{c_i}$, G_{c_i}, $\theta_{G_s}^{(0)}$, D_{c_i}, η_G, η_D, R, $\mathcal{E}$
Output: $\theta_{G_s}^{(R)}$

Initialize: $\theta_{G_s}^{(0)} \rightarrow \{c_1, c_2, \ldots, c_N\}$
for $r = 1$ ***to*** R **do**
  for $c_i \in \{c_1, \ldots, c_N\}$ **do**
    $\theta_{G_{c_i}}^{(r)} \leftarrow \theta_{G_s}^{(r-1)}$
    for $e = 1$ ***to*** $\mathcal{E}$ **do**
      Sample $\{x\} \subset \mathbb{D}_{c_i}$
      $z \sim p(z)$
      /*Local discriminator update*/
      $\theta_{D_{c_i}} \leftarrow \theta_{D_{c_i}} - \eta_D \nabla_{\theta_D} \mathcal{L}_D(x, z)$
      /*Local generator update */
      $\theta_{G_{c_i}}^{(r)} \leftarrow \theta_{G_{c_i}}^{(r)} - \eta_G \nabla_{\theta_G} \mathcal{L}_G(z)$
    Send $\theta_{G_{c_i}}^{(r)}$ to server
  Server aggregation:
  $\theta_{G_s}^{(r)} \leftarrow \sum_{i=1}^{N} w_{c_i} \theta_{G_{c_i}}^{(r)}$, where $w_{c_i} = \frac{|\mathbb{D}_{c_i}|}{\sum_{j=1}^{N} |\mathbb{D}_{c_j}|}$
  /*Generate synthetic samples on server */
  $z \sim p(z)$
  $\tilde{x}_s^{(r)} \sim G_s(z; \theta_{G_s}^{(r)})$
return $\theta_{G_s}^{(R)}$

In Algorithm 2, the server begins by initializing the global generator $\theta_{G_s}^{(0)}$ with parameters θ_0, and distributes them to all N participating clients. At each communication round r, each client c_i receives the latest global generator parameters $\theta_{G_s}^{(r-1)}$. Locally, each client updates its generator G_{c_i} and discriminator D_{c_i} using its local client dataset $\mathbb{D}_{c_i}$. Specifically, a mini-batch of real samples $x \subset \mathbb{D}_{c_i}$ is drawn, along with a corresponding batch of latent noise vectors $z \sim p(z)$. The discriminator D_{c_i} is trained to maximize its ability to distinguish between real and generated samples, while the generator G_{c_i} is trained to minimize the discriminator's ability to do so, thereby improving the realism of the generated data. This min-max training procedure is performed locally over $\mathcal{E}$ epochs using the Adam optimizer. It is important to note that the discriminator parameters, $\theta_{D_{c_i}}$, remain local to each client and are never shared with the server. Only the generator parameters, $\theta_{G_{c_i}}$, are updated for aggregation.

After local training, only the updated generator parameters $\theta^{(r)}_{G_{c_i}}$ are sent back to the server. The server performs weighted aggregation using the FedAvg algorithm to compute the updated global generator parameters $\theta^{(r)}_{G_s}$, where the weights w_{c_i} are proportional to the size of each client's dataset, i.e., $w_{c_i} = \frac{|\mathbb{D}_{c_i}|}{\sum_{j=1}^{N} |\mathbb{D}_{c_j}|}$. Importantly, after each aggregation step, the global generator G_s with parameters $\theta^{(r)}_{G_s}$ is used to generate synthetic samples $\tilde{x}_s^{(r)} \sim G_s(z; \theta^{(r)}_{G_s})$, where the latent noise $z \sim p(z)$ is sampled from a predefined prior distribution. This process is repeated for R communication rounds.

In this study, we explore four types of generative model variants within the FL framework: standard GAN, CGAN, WGAN-GP, and WCGAN-GP. A detailed description of these variants is provided in Section III. Algorithm 2 illustrates the federated training procedure for the standard GAN case, referred to as FL-GAN. For other variants (FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP), the training follows the same procedure, with only the loss functions modified according to the specific generative model variant used.

D. Explainable AI Methods

Deep learning models are inherently complex and often considered black boxes due to the difficulty in interpreting their predictions. Explainable Artificial Intelligence (XAI) seeks to improve model transparency and trust by offering interpretability of predictions [46]. XAI techniques are typically classified as either global explanations, which describe overall model behavior, or local explanations, which focus on individual predictions [11]. These methods can be either model-agnostic, applicable across various architectures, or model-specific. In this work, we adopt a model-agnostic approach by employing the LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive Explanations) methods to interpret the FL-DNN model.

Consider a dataset $\mathbb{D}$ consisting of input-output pairs $\mathbb{D} = \{(x^i, y^i)\}$, where each input $x \in \mathbb{R}^d$ represents a d-dimensional feature vector, and the corresponding output is generated by a black-box model $\mathcal{M}$ such that $\mathcal{M}(x) \in \mathcal{Y}$. A post-hoc explanation method is defined as a mapping $\mathbf{g}$, which, for a given model $\mathcal{M}$ and an input instance x, produces a feature attribution vector $\varphi_x \in \mathbb{R}^d$, denoted as $\mathbf{g}(\mathcal{M}, x) = \varphi_x$. Let $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ denote a distance metric in the space of explanations, and $S : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ a metric in the input space. The evaluation criterion μ is defined as a function that takes the model $\mathcal{M}$, the explainer $\mathbf{g}$, and the input x, and returns a scalar score that quantifies the quality of the explanation provided by $\mathbf{g}$ for that instance.

1) LIME: LIME is a post-hoc XAI method that generates local explanations for black-box models by fitting an interpretable model $g \in \mathcal{G}$ (e.g., linear models) around a specific instance $x \in \mathbb{R}^d$ [47]. The explanation $\varphi(x)$ is obtained by

$$\underset{g \in \mathcal{G}}{\operatorname{argmin}} \{\mathcal{L}(\mathcal{M}, g, \omega_x) + \Omega(g)\} \tag{15}$$

Here, $\mathcal{L}$ measures the fidelity of the explanation model g to the original model $\mathcal{M}$ using a locality-aware weighting function ω_x, and $\Omega(g)$ penalizes model complexity. By minimizing this objective, LIME constructs a locally faithful and interpretable surrogate for the black-box model.

2) SHAP: SHAP is a widely used XAI method for interpreting ML model outputs [37]. It is grounded in Shapley values from cooperative game theory [48], which quantify each feature's contribution to a model's prediction [49]. SHAP is particularly well-suited for tabular data and satisfies key properties such as local accuracy, missingness, and consistency [50]. Local accuracy ensures that the explanation aligns with the model's prediction, missingness implies that missing features have no impact, and consistency guarantees that a feature with higher influence receives a higher SHAP value.

SHAP approximates the original model $\mathcal{M}(x)$ using a simplified binary input representation $a \in \{0,1\}^d$, where $x = h_x(a)$. The explanation model $g(a)$ is expressed as an additive linear function:

$$\mathcal{M}(x) \approx g(a) = \varphi_0 + \sum_{i=1}^{d} \varphi_i a_i \tag{16}$$

Here, $\varphi_0 = \mathcal{M}(h_x(0))$ is the model output with all features absent, and φ_i is the attribution value for feature i, calculated as:

$$\varphi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(d - |S| - 1)!}{d!} [\mathcal{M}_x(S \cup \{i\}) - \mathcal{M}_x(S)] \tag{17}$$

$$\mathcal{M}_x(S) = \mathcal{M}(h_x^{-1}(a)) = \mathbb{E}[\mathcal{M}(x)|x_S] \tag{18}$$

In this formulation, F is the full feature set, S is a subset excluding feature i, and $\mathcal{M}_x(S)$ denotes the model's expected output conditioned on the known subset S. The result is a unified measure of additive feature attribution.

In our study, we employed SHAP's `DeepExplainer`, which takes as input a trained deep learning model and its corresponding dataset $\mathbb{D} \in \mathbb{R}^{m \times d}$, where m is the number of instances and d is the number of features. It returns an $m \times d$ matrix of SHAP values, quantifying the contribution of each feature to the model's prediction for each instance.

E. Evaluation of Explainable AI methods

Recently, the field of XAI has shifted toward using quantitative metrics to assess explanation quality. These metrics are generally categorized into user-focused, application-focused, and functionality-focused evaluations. In this work, we employ two key metrics to evaluate LIME and SHAP: High Faithfulness and Max Sensitivity.

1) High Faithfulness: The explanation method $\mathbf{g}$ should replicate the model's behaviour. $\mathbf{g}(\mathcal{M}, \mathbf{x}) \approx \mathcal{M}(\mathbf{x})$. Faithfulness quantifies the consistency between the prediction model $\mathcal{M}$ and explanation g. For evaluating the Faithfulness of explanations, High Faithfulness correlation [51] was used.

$$\mu_F(\mathcal{M}, g; x) = \underset{\mathcal{B} \in \binom{|d|}{|\mathcal{B}|}}{\operatorname{corr}} \left(\sum_{i \in \mathcal{B}} g(\mathcal{M}, x)_i, \mathcal{M}(x) - \mathcal{M}(x_{\mathcal{B}}) \right) \tag{19}$$

where $x_{\mathcal{B}} = x_i | i \in \mathcal{B}\}$. High Faithfulness correlation metric iteratively substitutes a random subset of given attributions with a baseline value $\mathcal{B}$. Then, it measures the correlation between the sum of these attributions and the difference in the model's output.

2) Max Sensitivity: Robustness refers to similar inputs that should result in similar explanations. $\mathbf{g}(\mathcal{M}, \mathbf{x}) \approx \mathbf{g}(\mathcal{M}, \mathbf{x} + \epsilon)$ for small ϵ. Max sensitivity [51] ensures that nearby inputs with similar model outputs receive comparable explanations. For the explanation function g to have low sensitivity around the point of interest x, assuming the predictor function $\mathcal{M}$ is differentiable, we define a neighborhood $\mathcal{V}$ around x as $\mathcal{V}_r = \{v \in D_x \mid p(x, v) \leq r, \mathcal{M}(x) = \mathcal{M}(v)\}$, where D is the distance metric and p is the proximity function. Thus, the maximum sensitivity of g at point x can be defined

$$\mu_M(\mathcal{M}(x), g, r; x) = \max_{v \in \mathcal{V}_r} D\left(g(\mathcal{M}(x), x), g(\mathcal{M}(x), v)\right) \tag{20}$$

F. Synthetic Data-Driven Explanation for Global Model in FL

In this study, we propose to utilize the post hoc SHAP XAI method to explain the black box nature of the DL classification model trained using the FedAvg algorithm. We note that SHAP requires access to either the training data $\mathbb{D}^{\text{train}}$ or a "reference set" that is similar to the training set used by the model. This is necessary to create records (q) that determine the impact of each feature value on the final prediction. To expedite the explanation process, a medoid of the dataset is sometimes used, or a small set of centroids [52] is utilized to represent $\mathbb{D}^{\text{train}}$, capturing the main characteristics with a few records of feature importance [53]. Consequently, in server-based FL settings, explaining the learned global model requires the server to have access to the complete set of training data $\mathbb{D}^{\text{train}}$ from its clients. Alternatively, the server should be able to compute the centroids of a dataset formed by combining the training sets of all clients. However, this approach depends on access to client training data, making it unsuitable in scenarios where privacy regulations prevent data sharing with the server.

After completing the training process in the FL setting, each client c_i holds its local model $\mathcal{M}_{c_i}$, which is then transmitted to the server. The server aggregates the received models using the FedAvg algorithm to construct the global model $\mathcal{M}_s$. Traditionally, explanations for a given instance x_j are computed locally by each c_i using a post-hoc explanation method $\mathbf{g}$, yielding attribution scores $\varphi_{c_i}(x_j) = \mathbf{g}(\mathcal{M}_{c_i}, x_j)$, which reflect the contribution of each feature toward the prediction made by $\mathcal{M}_{c_i}$.

To enable post-hoc explainability in FL without compromising data privacy, we propose an approach that leverages synthetic data generated by the global federated generator G_s to interpret the server-side black-box model $\mathcal{M}_s$. For each communication round r, the server updates the global generator G_s to generate synthetic samples $\tilde{x}_s^{(r)} \sim G_s(z; \theta_{G_s}^{(r)})$.

After all communication rounds, the final global generator $G_s^{(R)}$ defines a synthetic distribution P_g^s that closely approximates the true data distribution P_r. This synthetic data is used as the input reference to interpret the black-box classifier $\mathcal{M}_s$ without relying on client data. We use a model-agnostic post-hoc explainer $\mathbf{g}$ to obtain the feature attributions $\varphi_s(\tilde{x}) \in \mathbb{R}^d$ for any synthetic instance $\tilde{x} \in \mathbb{R}^d$, defined as:

$$\varphi_s(\tilde{x}) = \mathbf{g}(\mathcal{M}_s, \tilde{x}), \quad \tilde{x} \sim P_g^{s,(R)} \tag{21}$$

In our study, we use two post-hoc XAI methods, LIME and SHAP, to explain the global DNN model using synthetic data without relying on real client data.

Analytical Methodology: In our experiments, we aim to evaluate whether synthetic data-driven SHAP explanations are sufficient to approximate real data-based explanations for the global model $\mathcal{M}_s$ in a federated setting. To this end, we propose an analytical methodology that compares explanations derived from two settings:

(i) *Real data-based explanation:* The server accesses the real training data $\mathbb{D}_{c_i}^{\text{train}}$ from each client and computes SHAP values.

(ii) *Synthetic data-based explanation:* The server computes SHAP values using synthetic data generated by the global generator G_s, without access to real client data.

To perform the comparison, the following steps are carried out using each client's test data $\mathbb{D}_{c_i}^{\text{test}}$:

- For each client c_i, Server side SHAP explainer takes test data $\mathbb{D}_{c_i}^{\text{test}}$ and server model $\mathcal{M}_s$, SHAP values are computed, and computes explanation matrices $E_{\text{real}}^{(c_i)} \in \mathbb{R}^{m \times d}$, where m is the number of instances and d the number of features.
- A global explanation matrix $E_{\text{real}} \in \mathbb{R}^{m \times d}$ is derived by averaging SHAP values across all clients:

$$e_{ij}^{\text{real}} = \frac{1}{N} \sum_{c_i=1}^{N} e_{ij}^{(c_i)}.$$

- Similarly, The server samples m synthetic instances from the distribution P_g^s using the final global generator $G_s^{(R)}$, and computes SHAP values for the global model $\mathcal{M}_s$, computing the synthetic-data based SHAP explanation matrix $E_{\text{syn}} \in \mathbb{R}^{m \times d}$.
- A difference matrix $\Delta = E_{\text{syn}} - E_{\text{real}}$ is computed between the synthetic data-based and real data-based explanation matrices. If $\Delta_{ij} \approx 0$ for all i, j, this indicates that the synthetic data-driven explanations are sufficiently close to real data-based explanations, for explaining the global classifier $\mathcal{M}_s$.

G. Dataset

The proposed methodology is evaluated on 4 datasets. A description of the datasets is provided below.

1) NSL-KDD: NSL-KDD dataset is an improved version of the KDD Cup 1999 dataset, which is commonly used in experiments related to IDS tasks. This enhanced dataset addresses the issues of redundant records found in the original KDD Cup 1999 dataset [54]. Additionally, it ensures that the number of records in both the training and testing datasets is more reasonable. NSL-KDD dataset consists of network traffic of five different classes DoS, Prob, R2L (Remote to Local), U2R (User to Root), and Normal. Network traffic is collected

at fixed intervals and is classified as time series data. Each network unit comprises multiple data packets, which contain 41 features.

2) UNSW-NB15: This dataset was generated at the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) using the IXIA PerfectStorm tool [25]. It contains raw network packets that combine real-world normal activity with synthetic attack behaviors. The dataset includes nine different types of attacks, such as Denial of Service (DoS), exploits, generic attacks, reconnaissance, shellcode, and worms. In total, the dataset features 46 attributes, which were extracted using the Argus and Bro-IDS tools.

3) CIC-IoT2023: The CIC-IoT2023 dataset [55], introduced in 2023, is the largest publicly available IoT-specific ID dataset to date. It was collected in a controlled smart environment and comprises network traffic from 105 real-world IoT devices, spanning various categories such as smart TVs, sensors, cameras, and home automation appliances. A key distinguishing feature of this dataset is the inclusion of 33 distinct attack types, making it one of the most comprehensive resources for evaluating IDS models in IoT contexts. Notably, the attacks in CIC-IoT2023 are launched from compromised IoT devices targeting other IoT devices, which closely mirrors modern IoT threat landscapes.

4) CIC-IoMT2024: CIC-IoMT 2024 dataset [56] was also created by the Canadian Institute for Cybersecurity, similarly to CIC IoT 2023. This dataset provides a realistic benchmark dataset to enable the development and evaluation of IoMT security solutions. Eighteen attacks were executed on a testbed of 40 IoMT devices (25 real and 15 simulated), using protocols common in healthcare like Wi-Fi, MQTT, and Bluetooth. These attacks fall into five categories: DDoS, DoS, Recon, MQTT, and spoofing. This dataset establishes a baseline complementary to existing contributions, aiding researchers in creating new security solutions for healthcare systems through ML mechanisms. The dataset features 18 cyber attacks on 40 IoMT devices with diverse healthcare protocols such as Wi-Fi, MQTT, and Bluetooth. This dataset significantly aids the healthcare industry.

H. Experimental Setup

In the FL setup, both the DNN for classification and the generative models for synthetic data generation were trained under non-iid settings. Four network traffic-related datasets (NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024) were utilized in the experiments for a federated-based IDS task. Each dataset was distributed across multiple clients in a cross-silo FL setting, where the number of clients was varied as $N \in \{5, 10, 15\}$. For each client, the local dataset was partitioned into training and validation sets in an 80:20 ratio.. Notably, each client used the same local data to train both the DNN classifier and the generative model.

To simulate non-IID data distribution across clients in our FL experiments, we utilized the Dirichlet distribution (DD) for dataset partitioning. The Dirichlet distribution is a probability distribution that generates a set of non-negative values summing to one, which can be interpreted as proportions for allocating data samples among clients. In the federated setting, the DD is employed to assign class-wise sampling weights, thereby enabling controlled heterogeneity across client datasets. The level of statistical skewness is governed by the concentration parameter α: smaller values of α lead to more heterogeneous (non-IID) distributions, where each client may predominantly receive samples from a limited number of classes. In this work, we set $\alpha = 0.1$, which induces a high degree of non-IID-ness, closely mimicking real-world federated scenarios where client data distributions are highly imbalanced.

1) Data Preprocessing: In the preprocessing step, the client-side data in the FL-set setup was individually cleaned by removing null and infinite values. Each client's data $\mathbb{D}_{c_i}$ was transformed and normalized using min-max feature scaling, standardizing it within the range $[0, 1]$ to enhance the model's accuracy and performance. For each feature f where $f = 1, 2, \ldots, d$, the min-max normalization is defined as:

$$x'^{(f)}_{c_i} = \frac{x^{(f)}_{c_i} - \min\left(x^{(f)}_{c_i}\right)}{\max\left(x^{(f)}_{c_i}\right) - \min\left(x^{(f)}_{c_i}\right)} \in \mathbb{R}^d$$

where $x^{(f)}_{c_i}$ is the actual value of feature f for client c_i, $\max\left(x^{(f)}_{c_i}\right)$ and $\min\left(x^{(f)}_{c_i}\right)$ are the maximum and minimum values of feature f in client data $\mathbb{D}_{c_i}$, respectively. The value $x'^{(f)}_{c_i}$ is the new normalized value for feature f in client data $\mathbb{D}_{c_i}$, which lies in the range of 0 to 1.

We conducted experiments to determine the appropriate number of communication rounds for training the model. This was achieved through iterative experimentation, starting with random initial rounds and adjusting based on the observed performance of the model. For instance, when training federated DNN classifiers for binary classification, we employed 100 communication rounds. In contrast, 1000 rounds were used for training generative model variants in FL for both datasets. During each round of federated training, the validation data from each client is tested only on its corresponding client model. After all clients have been processed, the combined validation data from all clients is tested on the server model.

2) Evaluation of FL-DNN: To accurately classify network traffic characteristics, including flow-based, packet-based, and protocol-specific features from the NSL-KDD , UNSW-NB15, CIC-IoT2023, and CICIoMT2024 datasets, we proposed an FL-based deep neural network (FL-DNN). FL-DNN model consists of an input layer, multiple densely connected hidden layers, and an output layer containing neurons. The number of neurons in the input layer d, corresponds to the number of features representing a single network traffic packet in the training data. For the NSL-KDD dataset, $d = 41$, for the UNSW-NB15 dataset, $d = 46$, for CIC-IoT2023 dataset $d = 39$, for CIC-IoMT2024 dataset $d = 45$.

The number of hidden layers and the number of neurons in each hidden layer were determined empirically through randomized configuration searches to identify the architecture that provides optimal classification performance. All hidden layers employ the rectified linear unit (ReLU) activation function, which improves convergence during training and enhances classification accuracy. The output layer is designed for binary

classification, distinguishing between benign and malicious traffic. It consists of two neurons and uses the softmax activation function, which ensures a normalized probabilistic output for the two-class (benign vs. attack) problem

In evaluating the FL-DNN model in FL, we utilized a validation dataset exclusively. In this study, we employed classification metrics like accuracy, precision, recall, and F1-score to evaluate both the client and server models in FL for binary classification.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{22}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{23}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{24}$$

$$\text{F1-Score} = \frac{2 \times TP}{(2 \times TP) + FP + FN} \tag{25}$$

In the context of binary classification for 4 network traffic datasets, a true positive (TP) denotes the number of malicious traffic instances correctly classified as malicious. A false positive (FP) refers to benign traffic instances that are incorrectly classified as malicious. A true negative (TN) is the number of benign instances correctly identified as benign, while a false negative (FN) represents malicious instances that are incorrectly classified as benign.

3) Evaluation of Federated Generative Models: In this study, we evaluated four different generative model variants in a HFL setting such as GAN, CGAN, WGAN-GP, and WCGAN-GP. Each client independently trained a local generator and discriminator as part of the federated process. Both the generator and discriminator were implemented as fully connected neural networks, with the ReLU activation function used in all hidden layers.

For all generative models, the generator takes as input a latent noise vector and produces a synthetic network traffic instance with the same dimensionality as the original data. Specifically, the input dimension d corresponds to the number of features in each dataset: for NSL-KDD, $d = 41$, for UNSW-NB15, $d = 46$, for CIC-IoT2023, $d = 39$, and for CIC-IoMT2024, $d = 45$. For the conditional generative variants (CGAN and WCGAN-GP), the generator additionally receives a condition vector corresponding to the class label. Since the task involves binary classification, the conditional input vector is of size 2.

On the server side, the global generator produces synthetic data at each communication round. The quality of this synthetic data is evaluated by comparing it with the real data using the Wasserstein distance. The real data used for comparison consists of the combined validation datasets from all clients.

To evaluate the quality of synthetic data generated by the global generator G_s corresponding to each generative model variant in the FL setting, we employ the Wasserstein Distance (W_D), denoted as $W_D(P_r, P_g)$, as shown in Eq. 26. This distance quantifies the divergence between the true data distribution P_r, derived from the real dataset, and the generative distribution P_g, produced by the global generator based on latent noise $z \sim p(z)$. A lower value of $W_D \in [0, \infty)$ indicates greater similarity between real and synthetic data distributions, whereas a higher value reflects a larger discrepancy. This metric is used to evaluate the convergence and generation capability of FL-GAN, FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP over multiple communication rounds.

$$W_D(P_r, P_g) = \inf_{\gamma \in \Gamma(P_r, P_g)} \int_{\mathbb{R}^n \times \mathbb{R}^n} \|x - \tilde{x}\| \, d\gamma(x, \tilde{x}) \tag{26}$$

Here, P_r denotes the true data distribution aggregated from all participating clients, and P_g represents the synthetic data distribution generated by the global generator G_s. The set $\Gamma(P_r, P_g)$ includes all joint distributions $\gamma(x, \tilde{x})$ with marginals P_r and P_g. The term $\|x - \tilde{x}\|$ indicates the Euclidean distance between a real sample $x \sim P_r$ and a generated sample $\tilde{x} \sim P_g$, while $\gamma(x, \tilde{x})$ defines the optimal coupling that describes how much probability mass is transported from x to $\tilde{x}$.

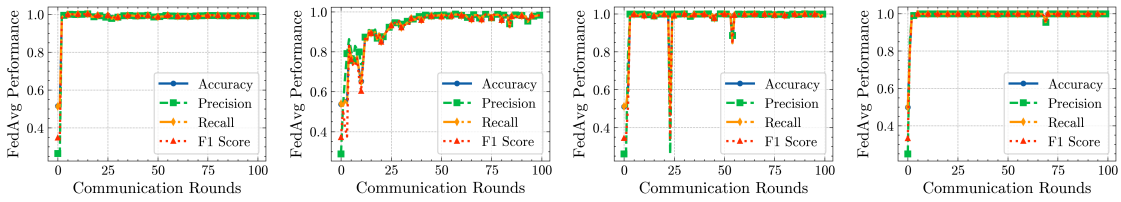
Experiments were conducted on a server equipped with an AMD Threadripper 3960X processor, which boasts 24 cores and 48 threads. The server features 128 GB of RAM and an NVIDIA 3090 GPU with 24 GB of graphics memory. It operates on Ubuntu 20.04 LTS as part of a high-performance computing (HPC) cluster node, utilizing the Slurm workload manager. The experiments were implemented in Python 3.9, using Flower [57] for FL in conjunction with PyTorch.

V. Results

This section presents the results of the experimental evaluation. Section V-A discusses the performance of the FL-DNN-based IDS using classification metrics. Section V-B focuses on the evaluation of federated generative models for synthetic data generation, highlighting data quality and distributional similarity to real data. Section V-D & V-C presents post-hoc explainability results for both client-side and server-side DNN models in FL settings.

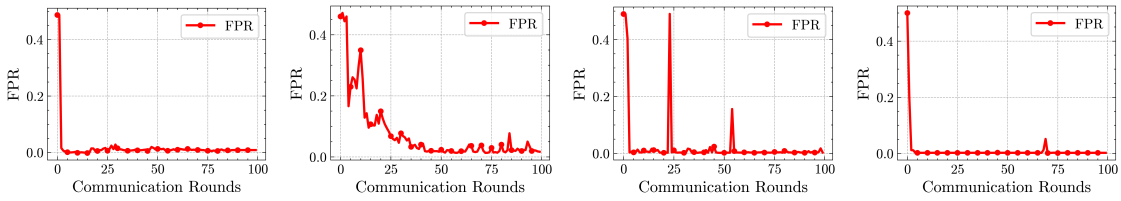
A. Performance Evaluation of FL-DNN Based IDS

We evaluated the experimental results using four network traffic-based ID datasets: NSL-KDD, UNSW-NB15, CIC-IoMT2024, and CIC-IoT2023. These datasets contain both packet-level and flow-based network traffic features relevant for detecting malicious activity. The classification task was formulated as a binary classification to distinguish between benign and malicious traffic. The FL training was performed in an HFL environment, with a central server aggregating model updates from multiple clients(N). Experiments were carried out under non-IID conditions across client configurations with $N \in \{5, 10, 15\}$. To simulate realistic deployment scenarios, we used the Dirichlet distribution-based sampling to introduce statistical heterogeneity among clients. To provide clarity of results, we report the results specifically for the configuration with N=5 clients, as model performance remained consistent across other client configurations for both FL-DNN classifier and Federated generative models.



(a) NSL KDD (b) UNSW-NB15 (c) CIC-IoT2023 (d) CIC IoMT2024

Fig. 3: Global DNN model performance using FedAvg aggregation across communication rounds for NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets



(a) NSL KDD (b) UNSW-NB15 (c) CIC-IoT2023 (d) CIC IoMT2024

Fig. 4: False positive rate of the global DNN model across communication rounds on NSL KDD, UNSW NB15, CIC IoT2023, and CIC IoMT2024 datasets

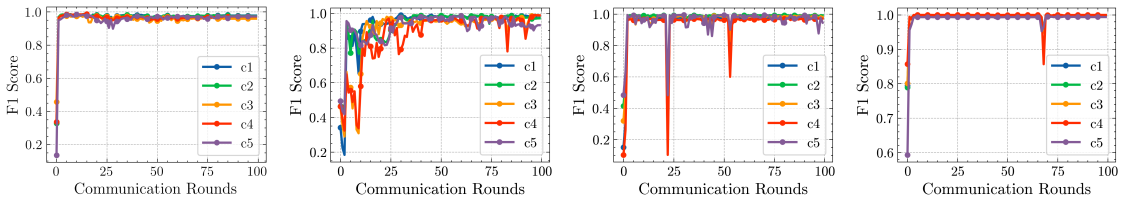
FL-DNN classifier was designed to detect malicious traffic while maintaining data privacy in a non-IID federated setting. The global model on the server-side was trained using the FedAvg aggregation algorithm across multiple communication rounds. In each communication round, all N clients were actively involved in both the training and evaluation phases to ensure diverse model updates and allow a thorough evaluation of the global model generalisation in non-IID settings. The global model was evaluated using a test set comprising the combined testing data from all the participating clients. On the other hand, each local model on the client side was evaluated separately using its respective client data. Fig. 3 and Fig. 4 illustrate the performance of the global DNN model trained using the FedAvg algorithm over $R = 100$ communication rounds with $N = 5$ clients. The results are presented for all four datasets, including NSL KDD, UNSW NB15, CIC IoT2023, and CIC IoMT2024, using classification evaluation metrics accuracy, precision, recall, F1 score, and FPR. For FPR, macro averaging was used based on the binary class labels 'Benign' and 'Attack' to ensure consistency across imbalanced distributions.

For all four datasets, the global DNN model displayed rapid convergence, with considerable improvements in classification performance metrics within the first ten communication rounds, followed by stabilization. For example, in Fig. 3a on the NSL KDD dataset, accuracy and recall exceeded 99% after round 15 with minimal fluctuations in subsequent rounds. Besides, as shown in Fig. 3b, the UNSW NB15 dataset demonstrated stable performance, with the DNN model attaining precision and recall values above 98% within 20 rounds. Similar performance was observed in CIC IoT2022 (Fig. 3c and CIC IoMT2024 (Fig. 3d) datasets, where the global DNN model consistently achieved high performance with classification metrics.

Fig. 4 illustrates the performance of the global DNN model, evaluated by FPR across communication rounds for the NSL KDD, UNSW NB15, CIC IoT2023, and CIC IoMT2024 datasets. FPR exhibited a significant decline during the initial training rounds and stabilized below 2%, showing that the model became increasingly effective at correctly identifying benign network traffic without mistakenly classifying it as malicious, thus reducing false alarms.

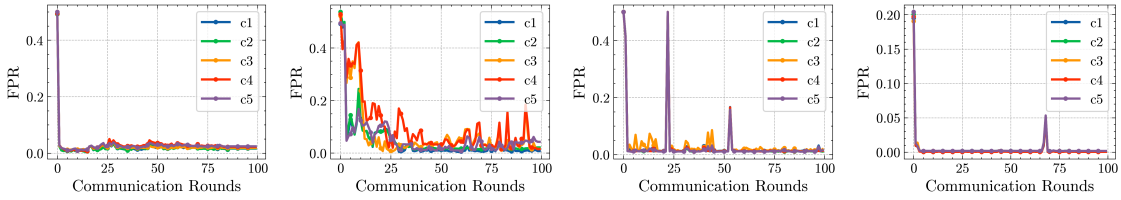
To evaluate the classification performance of Local DNN models, we report the F1 score over communication rounds for all four datasets in Fig. 5. Across most clients in datasets, local models show steady improvements in F1 score, particularly within the first 20 rounds. In NSL KDD (Fig. 5a) and CIC IoMT2024 (Fig. 5d) datasets, all clients quickly converged to high F1 scores, demonstrating a steady and consistent classification performance. However, the UNSW NB15(Fig. 5b) and CIC IoT2023(Fig.5c) datasets demonstrated slightly higher variability among clients, with some clients requiring more rounds to achieve stable performance.

Fig. 6 shows FPR for individual client DNN models across communication rounds for four different datasets. Most clients maintained low FPR values, particularly in the NSL KDD and CIC IoMT2024 datasets, where clients consistently achieved FPRs below 5%. In the UNSW NB15 and CIC IoT2023 datasets, some clients initially showed higher FPR. However, as training progressed, the FPR rates diminished substantially, demonstrating that local models became more effective at reducing misclassifications and accurately distinguishing between benign and malicious traffic.



(a) NSL KDD (b) UNSW-NB15 (c) CIC-IoT2023 (d) CIC IoMT2024

Fig. 5: Local DNN models performance across communication rounds for the NSL KDD, UNSW NB15, CIC IoT2023, and CIC IoMT2024 datasets



(a) NSL KDD (b) UNSW-NB15 (c) CIC-IoT2023 (d) CIC IoMT2024

Fig. 6: FPR of Local DNN models across communication rounds for the NSL KDD, UNSW NB15, CIC IoT2023, and CIC IoMT2024 datasets

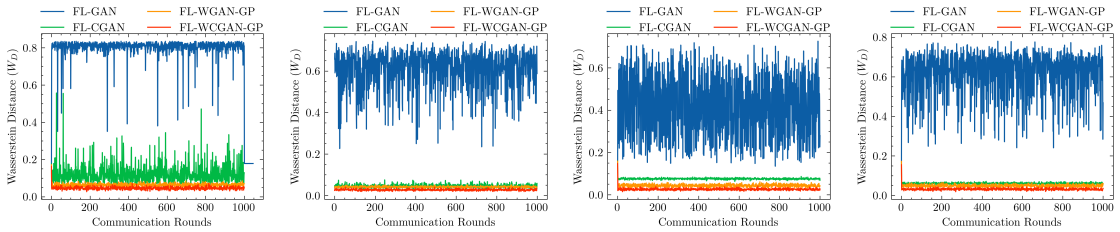
B. Evaluation of Federated Synthetic Data

In this section, we evaluate the performance of federated generative models in producing high-quality synthetic data within the FL settings. The synthetic data is later used in post-hoc explainability methods tasks to explain the trained global DNN model on the server without depending on client data. We implemented four variants of generative models for generating synthetic data on the server side in an FL setup: FL-GAN, FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP. These models were trained under non-IID data conditions using the FedAvg algorithm. During training, only the generator parameters were aggregated at the server, while the discriminator parameters remained local to each client, thus preserving data privacy. To evaluate the performance of federated generative models, we employed Wasserstein Distance (W_D) as a performance metric to quantify the discrepancy between the distributions of real and synthetic data. W_D metric ranges from $[0, \infty)$, where A value close to zero indicates that the synthetic data and real data distributions are highly similar. In contrast, a larger value signifies a greater discrepancy between the synthetic data distributions and the real data. In our experiments, W_D was computed between real test data and synthetic data generated by the global generative model across $R = 1000$ communication rounds. The evaluation was performed exclusively using test data at each round to ensure an unbiased evaluation of the synthetic data quality produced by the global generative model on the server side.

Fig. 7a illustrates the W_D performance of four global generative model variants on the server in FL using FedAvg on the NSL-KDD dataset. Basic FL-GAN demonstrates the highest W_D values, fluctuating between 0.6 and 0.8, indicating a significant discrepancy between the real and synthetic data distributions over several communication rounds. FL-CGAN model displays better stability over the rounds, but noticeable fluctuations were still shown, especially in the early rounds. In contrast, the Wasserstein-based generative models in FL perform better in generating synthetic data that was much closer to the real data distribution. Both FL-WGAN-GP and FL-WCGAN-GP achieved lower W_D values across the communication rounds, demonstrating their effectiveness.

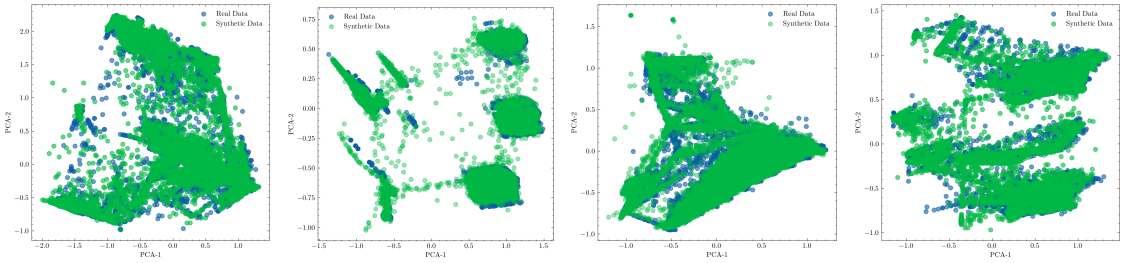
Similarly, Fig. 7b shows W_D performance of four GAN variants in FL using FedAvg on the UNSW-NB15 dataset over 1000 communication rounds. Basic FL-GAN has the highest W_D values, whereas the FL-CGAN consistently performs better with lower W_D values, despite some minor fluctuations over rounds. Wasserstein GAN variants (FL-WGAN & FL-WCGAN-GP) outperform other GAN variants in producing high-quality synthetic data. Both FL-WGAN-GP and FL-WCGAN-GP consistently exhibit lower W_D values throughout the communication rounds. Notably, FL-WCGAN-GP achieved the lowest W_D value, successfully generating high-quality synthetic samples from the global generator for the UNSW-NB15 dataset.

Similar W_D performance was observed across generative models for the CIC-IoT2023 (Fig. 7c) and CIC-IoMT2024 datasets (Fig. 7d). Among the federated generative models, the FL-GAN model produced the least accurate synthetic data, as indicated by its higher W_D values. While the FL-CGAN model showed improved performance, it showed noticeable fluctuations in W_D across communication rounds on both datasets. In contrast, the Wasserstein-based models, FL-WGAN-GP and FL-WCGAN-GP, consistently outperformed the other variants. Notably, FL-WCGAN-GP achieved the lowest W_D values and demonstrated high stability, approaching zero, which high-



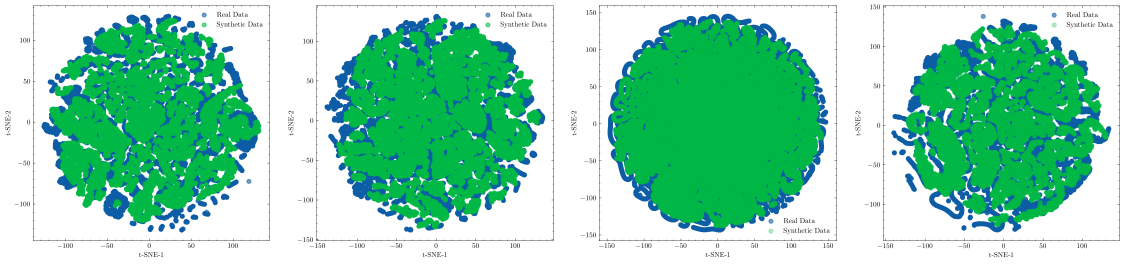
(a) NSL KDD (b) UNSW-NB15 (c) CIC-IoT2023 (d) CIC IoMT2024

Fig. 7: Performance of federated generative models across communication rounds for the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets.



(a) NSL KDD (b) UNSB-nb15 (c) CIC-IoT2023 (d) CIC IoMT2024

Fig. 8: 2-D visualization using PCA for real and synthetic data generated by FL-WCGAN-GP for all four datasets



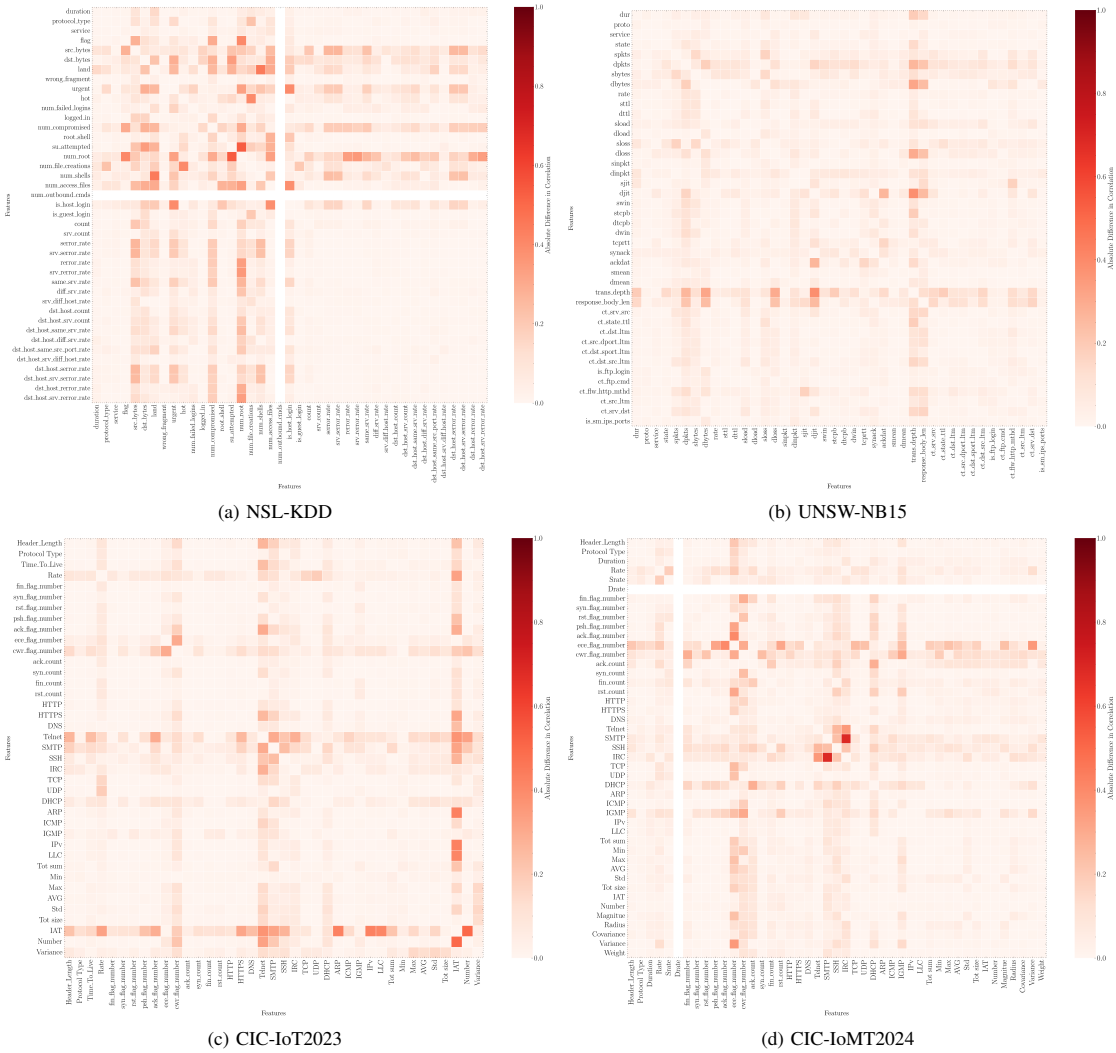
(a) NSL KDD (b) UNSW-NB15 (c) CIC-IoT2023 (d) CIC IoMT2024

Fig. 9: 2-D visualization using t-SNE for real and synthetic data generated by FL-WCGAN-GP for all four datasets

lights its effectiveness in producing high-fidelity synthetic data in federated settings.

Among the evaluated generative models (FL-GAN, FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP), FL-WCGAN-GP demonstrated superior performance in minimizing the Wasserstein Distance (W_D) across communication rounds for all four datasets. The FL-WCGAN-GP model consistently achieved the lowest W_D values, close to zero, demonstrating its effectiveness in generating high-quality synthetic data that closely approximates the distribution of real data. Based on these results, we selected the FL-WCGAN-GP global generative model to produce synthetic data on the server side. This synthetic data was subsequently used as the reference input for the post-hoc XAI method employed to explain the global FL-DNN classifier.

We further investigated the differences between synthetic data distribution generated by FL-WCGAN-GP global generative model and real data using two-dimensional representations from Principal Component Analysis (PCA) [58] and t-distributed Stochastic Neighbor Embedding (t-SNE) [59]. PCA is a statistical method used for linear dimensionality reduction. It transforms high-dimensional data into a lower-dimensional form while retaining the most important information. PCA accomplishes this by identifying new axes, known as principal components, along which the data exhibits the greatest variance. These components are orthogonal and uncorrelated, making PCA an effective tool for reducing dimensionality. The original dimensionality was reduced from 41 features (NSL-KDD), 45 features (UNSW-NB15), 39 features (CIC-IoT2023), and 45 features (CIC-IoMT2024) to 2-D (dimen-



(a) NSL-KDD (b) UNSW-NB15

(c) CIC-IoT2023 (d) CIC-IoMT2024

Fig. 10: Feature-pairwise Absolute differences in Pearson correlations between real and synthetic data generated by the FL-WCGAN-GP model for the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets

sional) for visualization of PCA and t-SNE.

Fig. 8 illustrates the PCA projections for all four datasets in 2-D. For each dataset, the synthetic data generated by the global FL-WCGAN-GP generative model on the server closely aligns with the distribution of the real test data in the principal component space. FL-WCGAN-GP generative model on the server side effectively captures the underlying variance structure of the original datasets that overlap with synthetic data, despite slight deviations in high-density areas. Overall, the consistency in distributions demonstrates the FL-WCGAN-GP's ability to generate high-fidelity synthetic data.

On the other hand, t-SNE is a nonlinear dimensionality reduction method designed for high-dimensional data visualization. t-SNE is an effective method for visualizing high-dimensional datasets in lower-dimensional spaces. It preserves the structure of the data, ensuring that points which are close together in the high-dimensional space remain close in the low-dimensional representation. Fig. 9 displays the t-SNE visualizations for four datasets, illustrating the 2-D representations of both real data and synthetic data generated by the FL-WCGAN-GP generative model variant. Across all datasets, the synthetic samples show a high degree of overlap with real test data, indicating that the FL-WCGAN-GP model also preserves local structures in the data. While some minor discrepancies can be observed, particularly at the boundaries of dense clusters, the overall the generated synthetic is closely

aligned with real test data.

To further evaluate the quality of the synthetic data generated by the selected global FL-WCGAN-GP model on the server side in the FL setup, we analyzed the pairwise feature relationships by comparing Pearson correlation values between the real and synthetic data. Specifically, we computed Pearson correlations on the real test data and the synthetic data generated by FL-WCGAN-GP, then calculated the absolute differences in bivariate correlations for each feature pair, as shown in Fig. 10 across all four datasets. These absolute difference values are bounded within the range $[0, 1]$, where values closer to zero indicate that the feature relationships in the synthetic data are well-aligned with features in the real data. Conversely, values farther from zero represent the larger discrepancies between the two datasets.

Fig. 10a presents a heatmap that illustrates the absolute differences in Pearson correlation coefficients between real and synthetic data for the NSL-KDD dataset, which comprises 41 features. In this heatmap, most feature pairs show minimal absolute differences, indicating that the server-side generator from the FL-WCGAN-GP model effectively produces synthetic data that closely resembles real data for each feature. However, a few feature pairs exhibit moderately larger differences, implying that the synthetic data diverges slightly from the real dataset. Similarly, Fig. 10b exhibits the absolute differences in correlation, as measured by the Pearson correlation coefficient, between real and synthetic data for the UNSW-NB15 dataset, which consists of 45 features. This correlation-based evaluation indicates that most pairs of features exhibit low absolute differences close to zero which suggests that the FL-WCGAN-GP model successfully captures and retains the key correlations present in the UNSW-NB15 dataset. However, there were a few feature pairs that showed relatively higher discrepancies, with correlation differences significantly deviating from zero, which implies that the synthetic data for these features was not aligned closely with the real data.

Fig. 10c & Fig. 10d illustrate the absolute differences in Pearson correlations between real and synthetic data for the CIC-IoT2023 and CIC-IoMT2024 datasets, containing 39 and 45 features, respectively. In both datasets, the heatmaps show that the majority of feature pairs display low absolute differences close to zero, demonstrating that the FL-WCGAN-GP model effectively generates synthetic data that resembles real test data. While a few feature pairs demonstrate moderate deviations, indicating minimum discrepancies, the feature correlation values in the synthetic data remain highly consistent with the feature correlation values of the real test data.

Table II presents the classification performance of the global FL-DNN model evaluated on both real and synthetic test data generated by the FL-WCGAN-GP model across four datasets, including NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024. The evaluation includes classification metrics such as accuracy, precision, recall, F1-score, and FPR, where FPR represents the proportion of benign samples incorrectly classified as malicious.

Across all datasets, the FL-DNN model achieves high performance on real test data, with accuracy and F1-scores consistently above 98%, and FPR values remaining low. Notably, the model achieves the highest performance on the CIC-IoMT2024 dataset, with almost perfect accuracy and an F1-score of 99.94%, as well as an extremely low FPR of 0.06%. A similar strong performance was achieved for CIC-IoT2023 and UNSW-NB15 datasets as well.

When evaluated on synthetic data generated by the FL-WCGAN-GP model, the FL-DNN classifier also demonstrates strong performance in metrics. Although there is a slight performance decrease compared to real data across all metrics, the accuracy, precision, and F1-scores remain relatively high, particularly for CIC-IoMT2024 and CIC-IoT2023, where the F1-scores reach 98.86% and 95.90%, respectively. The false positive rate on synthetic data is slightly higher than on real data, demonstrating minor distributional differences between real and synthetic-driven samples. However, the overall performance of the synthetic data is sufficiently high quality to support model evaluation and explainability without direct access to real data from clients.

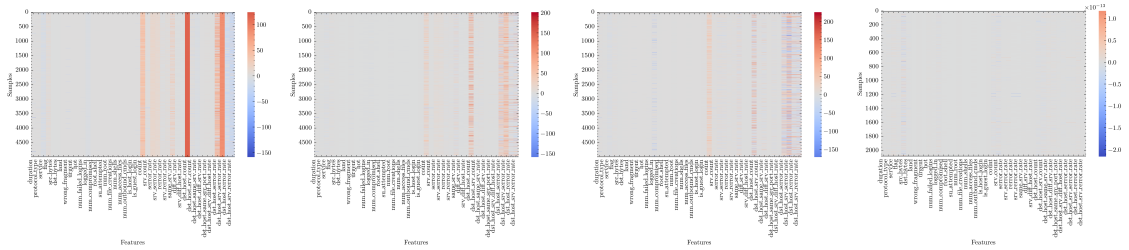
TABLE II: Performance of global FL-DNN model on Real Data and Synthetic data generated by FL-WCGAN-GP generative model

Data Set	Data-type	Accuracy	Precision	Recall	F1-score	FPR
NSL-KDD	Real	99.31%	99.50%	98.88%	99.18%	1.12%
	Synthetic	96.32%	96.27%	93.11%	94.57%	6.89%
UNSW-nb15	Real	98.46%	98.45%	98.48%	98.46%	1.52%
	Synthetic	95.12%	90.81%	96.10%	93.09%	3.90%
CiC-IoT2023	Real	99.24%	99.25%	99.24%	99.24%	0.76%
	Synthetic	97.11%	97.20%	97.11%	95.90%	2.89%
CIC-IoMT2024	Real	99.94%	99.94%	99.94%	99.94%	0.06%
	Synthetic	98.86%	98.88%	98.85%	98.86%	1.15%

C. Explaining global DNN model in FL using synthetic data

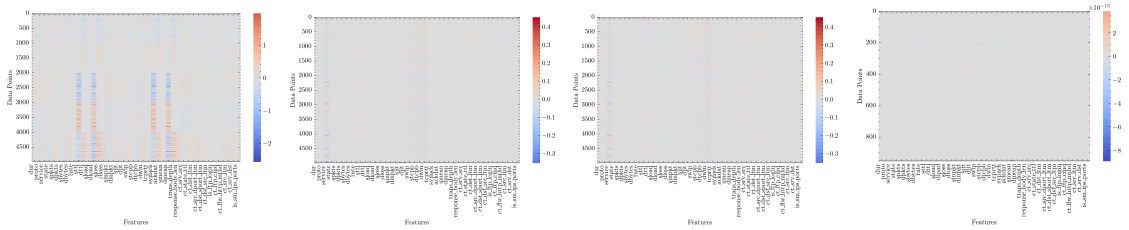
In FL-setup, explaining local models on the client side is straightforward, as each client keeps its data locally, allowing the use of post-hoc explainability methods like SHAP or LIME. However, explaining the global model on the server is challenging due to the unavailability of training data or reference input data, which post-hoc methods require. Direct access to client data would violate the privacy principles of FL. To overcome this, we propose using synthetic data generated by a federated generative model as a privacy-preserving alternative to real data. This synthetic data serves as an input reference data for post-hoc XAI methods, such as LIME and SHAP, on the server, allowing explainability for the global DNN model without exposing client data.

To explain the global DNN model on the server side, we utilized the SHAP explainer with synthetic data generated by different federated generative models. SHAP calculates the Shapley value, which indicates the influence of features on model predictions. Specifically, SHAP was applied to the global DNN model obtained at the 100th communication round, using synthetic data as the input reference to compute Shapley values. To assess whether synthetic data is sufficient



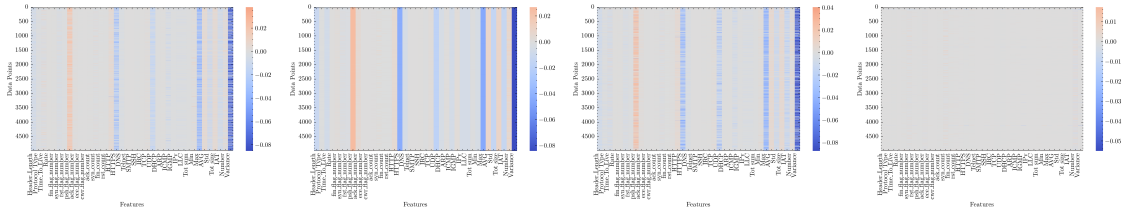
(a) FL-GAN (b) FL-CGAN (c) FL-WGAN (d) FL-WCGAN

Fig. 11: Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on NSL-KDD dataset.



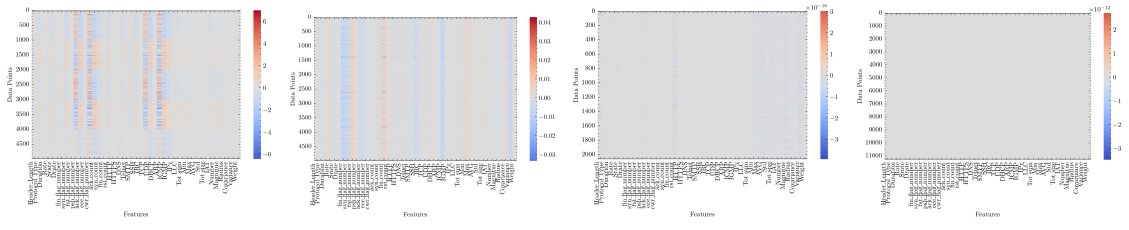
(a) FL-GAN (b) FL-CGAN (c) FL-WGAN (d) FL-WCGAN

Fig. 12: Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the UNSW-NB15 dataset.



(a) FL-GAN (b) FL-CGAN (c) FL-WGAN (d) FL-WCGAN

Fig. 13: Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the CIC-IoT2023 dataset



(a) FL-GAN (b) FL-CGAN (c) FL-WGAN (d) FL-WCGAN

Fig. 14: Heatmaps showing the magnitude of SHAP value differences between synthetic data-based and real data-based explanations for each sample on the CIC-IoMT dataset

for constructing reliable explanations for a global DNN model, the server accessed real client data to compute SHAP values again on the same model. We then calculated the difference $\Delta = E_{\text{real}} - E_{\text{syn}}$, where E_{real} is the SHAP values derived from real data-based explanations, E_{syn} is synthetic data-based SHAP explanations.

Figs. 11–14 present SHAP delta matrices as heatmaps for the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT datasets. Each heatmap visualizes the difference (Δ) in feature attributions between synthetic data-based explanations and real data-based explanations across all features. When the difference closes to zero ($\Delta \approx 0$), this indicates a close alignment between E_{real} and E_{syn}, implying that synthetic data can effectively approximate real data for explaining the global DNN model. Conversely, when Δ values deviate significantly away from zero, it implies that the synthetic data is insufficient to produce reliable explanations.

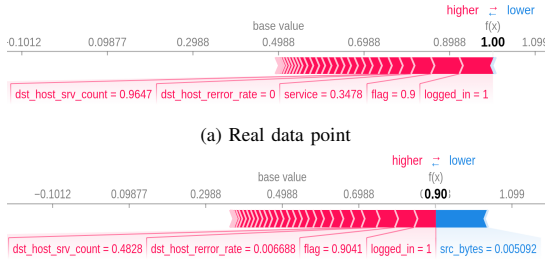
SHAP values difference (Δ) shown in Figs. 11–14 reveal significant variation in explanation fidelity across different generative models in four datasets. FL-GAN, FL-CGAN, and FL-WGAN-GP show varying levels of divergence between E_{real} and E_{syn}. In particular, FL-GAN shows the highest discrepancy, with SHAP values in difference (Δ) in the heatmaps deviating substantially from zero, displaying noticeable variability in feature attributions, for all four datasets.

FL-CGAN and FL-WGAN-GP show moderate improvements, with several features exhibiting smaller differences in Δ closer to zero, which suggests a better alignment between synthetic data-based SHAP values and real data-based SHAP explanations. Nevertheless, considerable variation remains, particularly in CIC-IoT2023 and CIC-IoMT2024, where feature attribution differences are still evident.

Among all the evaluated federated generative models, the FL-WCGAN-GP generative model on the server side consistently produced the smallest differences in Δ , which are close to zero in all four datasets. This implies that the synthetic data-based SHAP explanations are closely aligned with explanations derived from real data, highlighting FL-WCGAN-GP's effectiveness in generating high-fidelity synthetic data suitable for server-side explainability of the global FL-DNN model.

Unlike GANs and WGANs, Conditional GAN models (CGANs and WCGAN variants) take both a noise vector and a class label as inputs. These Conditional GAN variant generative models ensure that the generator produces synthetic samples corresponding to the specified target label, which is crucial for post-hoc explainability methods that rely on class-specific synthetic data. To ensure reliable explanations of the global DNN model, only high-fidelity synthetic samples that closely resembled real data were used. Specifically, we selected synthetic instances with minimal deviation from real samples by computing pairwise Euclidean distance and applying a threshold of 0.1 to exclude outliers from the synthetic data distribution.

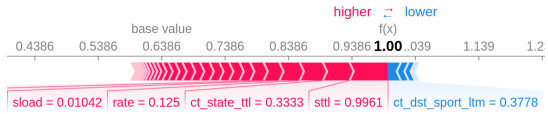
To demonstrate the use case of SHAP for local explainability of a global DNN model on a server, we randomly selected a real data-based instance and a synthetic data-based instance from the NSL-KDD dataset, both of which were classified as "attack." To obtain prediction probabilities for each class label, we applied the softmax activation function at the output layer of the DNN model. This function is essential for interpreting the model's confidence, as it transforms output logits into a normalized probability distribution over the class labels (attack and benign).



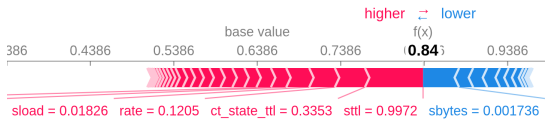
(a) Real data point

(b) Synthetic data point

Fig. 15: SHAP local explanations for the global DNN model on an attack instance from the NSL-KDD dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.



(a) Real data point

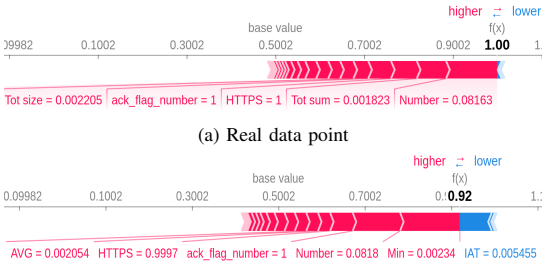


(b) Synthetic data point

Fig. 16: SHAP local explanations for the global DNN model on an attack instance from UNSW-nb15 dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.

Fig. 15 presents SHAP local explanation plots for a single instance classified as an attack in the NSL-KDD dataset, using both real and synthetic data to illustrate each feature's contribution to the prediction of the global DNN model on the server. The plot illustrates the base value, with features that positively influence the prediction represented in red and those that negatively impact the predictions shown in blue. The base value is the average of all prediction values. Each strip in the plot illustrates how different features impact the predicted value, either moving it closer to or further away from the base value. Red strips indicate features that push the predicted value higher, while blue strips show features that push it lower. The width of each strip reflects the strength of the feature's contribution have wider strips indicate a greater impact on the predicted value.

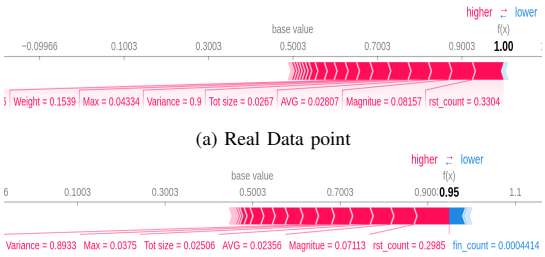
For example, Fig. 15a illustrates the SHAP local explanation for a real data point of an attack class from the NSL-



(a) Real data point

(b) Synthetic data point

Fig. 17: SHAP local explanations for the global DNN model on an attack instance from CIC-IoT2023 dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.



(a) Real Data point

(b) Synthetic Data point

Fig. 18: SHAP local explanations for the global DNN model on an attack instance from the CIC-IoMT2024 dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.

KDD dataset. The global DNN model predicts this instance with a score of 1.0. The base value is 0.4988, and key features such as user login status (logged_in), connection status flag (flag), network service on the destination (service), percentage of connections to the same host that have “REJ” errors (dst_host_rerror_rate), and number of connections to the same service as the current connection in the past 100 (dst_host_srv_count) positively contribute to the final prediction by pushing the output away from the base value toward the attack class.

Similarly, Fig. 15b presents the SHAP local explanation for a synthetic data point of the attack class, generated by the global FL-WCGAN-GP model. The predicted value of the global DNN model using this synthetic data point is 0.90. Notably, the same influential features (including logged_in, flag, dst_host_rerror_rate, and dst_host_srv_count) appear again, contributing positively to the attack prediction. However, the feature number of bytes sent from source to destination (src_bytes) negatively contributes to the prediction in the synthetic case.

Fig. 16a presents the SHAP local explanation of the global DNN model on the server for a real data point labeled as an attack in the UNSW-NB15 dataset. Both the real and synthetic data points have the same base value of 0.6386. In this real instance, the most influential features that positively contributes the model’s output toward the attack class (with a prediction value of 1.0) include the source to destination time-to-live value (sttl), number of connection records per state based on TTL range (ct_state_ttl), source bits per second (sload), and connection packet rate (rate). Conversely, the feature number of connections sharing the same destination IP and source port in a recent time window (ct_dst_sport_ltm) contributes negatively to the prediction, slightly lowering the prediction value.

Fig. 16b shows the SHAP explanation for a synthetic data point generated by the FL-WCGAN-GP model, with a prediction value of 0.84 from the global DNN model. As with the real data point, the most influential features for predicting the "attack" class label by the global DNN model include the sttl, ct_state_ttl, rate, and sload. However, in this case, the number of bytes sent from source to destination (sbytes) negatively impacted the prediction.

Fig. 17 shows SHAP local explanations for the CIC IoT2023 dataset, using real data points and synthetic data points for the global DNN Model, each having the same base value of 0.5002. In Fig. 17a, the global DNN model predicts real data points as an attack class label with a prediction value of 1. The most influential features contributing positively to this prediction include the total number of packets in the flow (Number), sum of packet lengths in the flow (Tot sum), use of HTTPS protocol at the application layer (HTTPS), ACK flag value (ack_flag_number), and the packet length (Tot size), all of which push the model’s output toward the attack class.

In Fig. 17b, the SHAP explanation shows that the DNN model has a slightly lower prediction value of 0.92 on the synthetic data point generated by global FL-WCGAN-GP generative model. The most influential features contributing positively to the attack class prediction include the minimum packet length in the flow (Min) and Number, followed by the ACK flag value ,HTTPS, and the average packet length in the flow (AVG). In contrast, the inter-arrival time between packets (IAT) contributes negatively, slightly reducing the predictive value of classifying the attack.

Fig. 18 illustrates SHAP local explanations for the global DNN model based on an attack instance from the CIC-IoMT2024 dataset, using a real data point shown in Fig. 18a and a synthetic data point in Fig. 18b, both with a base value of 0.5003. In both cases, the model correctly classifies the instance as an attack, with prediction scores of 1.0 for the real point and 0.95 for the synthetic data point. The most influential features contributing positively to the attack class prediction in both explanations include the count of RST flag occurrences in packets (rst count), the average of incoming and outgoing packet lengths in the window (Magnitude), the mean of packet lengths within the aggregated packets (AVG), the average packet size per window (Tot size), and the maximum packet length in the window (Max).

We have also provided local explanations using the LIME explainer. LIME method explains the rationale behind assigning probabilities to each class by comparing the probability values with the actual class of the data point. As with SHAP,

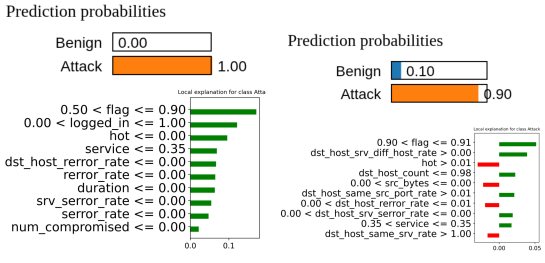
to illustrate local explanations for the global DNN model using LIME, we selected both a real data point and a synthetic data point generated by the FL-WCGAN-GP model for all four datasets.

Fig. 19 illustrates the local explanations generated by LIME for the global DNN model on an attack instance from the NSL-KDD dataset. In these visualizations, green bars represent features that contribute positively toward predicting the data point as belonging to the attack class, while red bars indicate features that influence the prediction toward the benign class label.

In Fig. 19a, LIME explanations of the global DNN model predict the real data point as an attack with 100% confidence. Features such as flag,logged_in, hot, service, dst_host_rerror_rate, and rerror_rate are among the top contributors, as shown by the green bars, indicating strong influence on the attack class prediction.

In Fig. 19b, the synthetic data point generated by the FL-WCGAN-GP model yields a 90% prediction for the attack class and 10% for the benign class. features shown in green bar color in like flag, dst_host_srv_dff_host_rate , dst_host_count and so on contributes to predicting the class attack, on other hand features shown in red bars like hot, src_bytes, dst_host_rerror_rate, dst_host_same_srv_rate contribute to predicting the Benign class label.

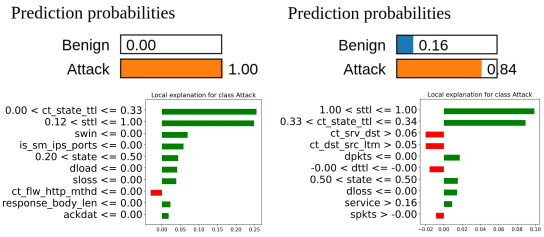
Similar to the LIME explanation for the global DNN model on the NSL-KDD dataset, Fig.20, Fig. 21 and Fig. 22 present LIME-based local explanations of global DNN model for the UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets, respectively, using both real and synthetic data points labeled as attacks. In these figures, features shown in green bars contribute positively toward attack class prediction, while features shown in red bars support the benign class prediction. Overall, across both SHAP and LIME local explanations, the top contributing features for synthetic data points generated by the FL-WCGAN-GP model are mostly similar to the features identified in explanations based on real data points.



(a) Real Data point (b) Synthetic Data point

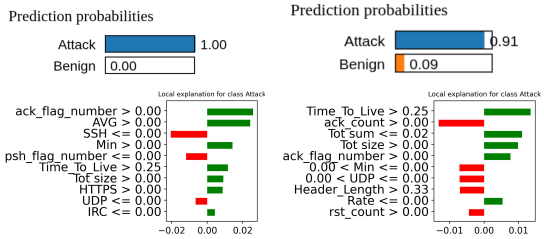
Fig. 19: LIME local explanations for the global DNN model on an attack instance from the NSL-KDD dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.

We have also evaluated post hoc local explanations quantitatively using synthetic data for the global model, on the server, using two criteria, faithfulness and robustness. To evaluate the



(a) Real data point (b) Synthetic data point

Fig. 20: LIME local explanations for the global DNN model on an attack instance from UNSW-nb15 dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.



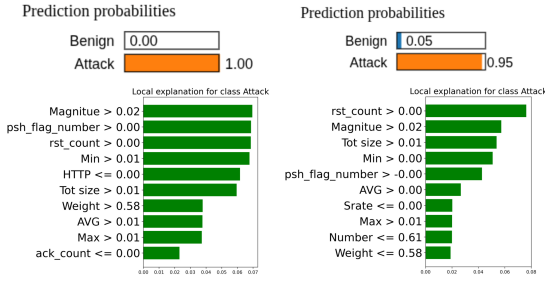
(a) Real data point (b) Synthetic data point

Fig. 21: LIME local explanations for the global DNN model on an attack instance from CIC-IoT2023 dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model.

faithfulness of post hoc local explanations, a High faithfulness correlation metric was used. High faithfulness metrics measure how well the feature importance scores provided by an explanation method (e.g., SHAP or LIME) align with the XAI method's actual influence on the model's prediction. For each feature, its value is replaced with a baseline value, and the resulting change in the predicted class probability is measured. Zero baseline values were used. The Pearson correlation coefficient is then calculated between explainer feature importance scores and the change in model output across all features. A high correlation value indicates that the explanation accurately reflects the model's decision-making behavior, suggesting that the explanation for a given local instance is highly faithful. The DNN model prediction probabilities were obtained using the Softmax activation function.

To evaluate robustness, we employed the max-sensitivity metric. This metric measures the stability of the explanation by analyzing whether nearby instances in the input space have similar explanations. In this metric, nearest neighbors were identified using Euclidean distance within a radius of $r = 0.1$. A low sensitivity value implies that minor perturbations in the input do not cause significant changes in the explanation, thereby indicating that the explainer is robust.

Table III presents the quantitative evaluation of post-hoc



(a) Real data point (b) Synthetic data point

Fig. 22: LIME local explanations for the global DNN model on an attack instance from CIC-IoMT2024 dataset, using (a) a real data point and (b) a synthetic data point generated by the FL-WCGAN-GP model

TABLE III: Evaluation of SHAP and LIME local explanations on synthetic data for the global DNN model on server using Faithfulness (μ_f) and Max Sensitivity (μ_s) metrics across four datasets.

Dataset	Metric	LIME	SHAP
NSL-KDD	μ_f	0.2087 ± 0.1731	0.6628 ± 0.3006
	μ_s	0.1723 ± 0.0509	0.0170 ± 0.0341
UNSW-nb15	μ_f	0.2853 ± 0.3931	0.5949 ± 0.4048
	μ_f	0.0389 ± 0.0107	0.0206 ± 0.0697
CIC-IoT2023	μ_f	0.3372 ± 0.3682	0.6113 ± 0.1802
	μ_s	0.1314 ± 0.0397	0.0512 ± 0.0455
CIC-IoT2024	μ_f	0.2635 ± 0.2165	0.7884 ± 0.3415
	μ_s	0.2052 ± 0.0560	0.0406 ± 0.0395

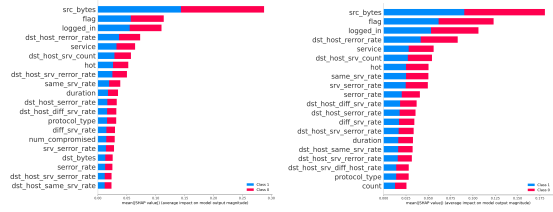
local explanations generated using SHAP and LIME for the global DNN model on the server, based on synthetic data points generated by a global generative FL-WCGAN-GP model. For all four datasets, the mean and standard deviation were computed over 1000 synthetic instances of post hoc local explanations using two XAI metrics, high faithfulness (μ_f) and max sensitivity (μ_s). The better-performing XAI method for each metric and dataset is highlighted in gray in the table.

For the NSL-KDD dataset, SHAP explanations demonstrate significantly higher faithfulness compared to LIME. SHAP achieves the highest mean faithfulness score of $\mu_f = 0.6628 \pm 0.3006$, indicating that the feature attributions produced by SHAP are closely aligned with the actual influence of input features on global DNN model's predictions. This high correlation indicates that SHAP explanations are highly faithful when applied to the global DNN model using synthetic data, making them more accurate and trustworthy than LIME explanations. In terms of robustness for NSL-KDD , SHAP outperforms LIME. It yields the lowest max sensitivity value of $\mu_s = 0.0170 \pm 0.0341$, which means its explanations are more stable when small changes are made to the input data. This low sensitivity suggests that SHAP generates consistent explanations for nearby synthetic samples.

Similarly, in Table III, for the UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets, SHAP explanations consistently outperform LIME, exhibiting greater fidelity and more robustness in local explanations for the global DNN model using synthetic data generated by the global FL-WCGAN-GP generative model.

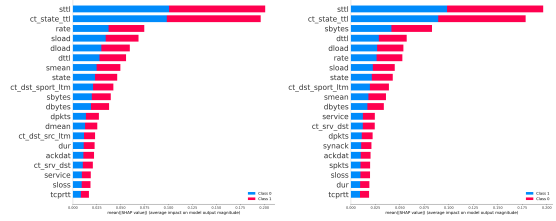
We also present SHAP global feature importance for the global DNN model on the server by comparing results using real data and synthetic data generated by the FL-WCGAN-GP model. This comparison helps to see how closely the feature importance from synthetic data reflects that of the real data. Fig. 23 shows SHAP-based global explanations for the NSL-KDD dataset, using both real and synthetic data generated by the global FL-WCGAN-GP generative model. Global feature importance indicates how much each feature contributes to the model's predictions across multiple instances, capturing its overall impact on the model's output.

In Fig. 23a, the global feature importance for global DNN model using real data accessed by server is shown, while Fig. 23b shows the SHAP global feature importance based on synthetic samples generated by the FL-WCGAN-GP model . The most influential features in both the real-data-based and synthetic-data-based global feature importance rankings are quite similar. Top influential Features such as src_bytes, flag, logged_in, and dst_host_rerror_rate exhibit the highest mean SHAP values in both cases, suggesting that the synthetic data effectively preserves the feature attribution behavior of global DNN model observed with real data.



(a) Real data (b) Synthetic data

Fig. 23: SHAP global feature importance for the global DNN model on the server using the NSL-KDD dataset, based on (a) real data and (b) synthetic data generated by the FL-WCGAN-GP model.



(a) Real Data (b) Synthetic data

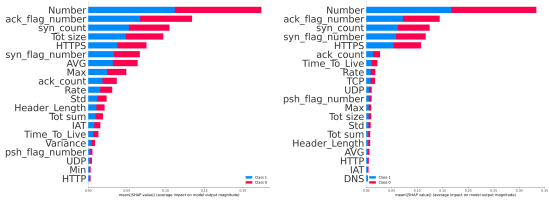
Fig. 24: SHAP global feature importance for the global DNN model on the server using the UNSW-nb15 dataset, based on (a) real data and (b) synthetic data generated by the FL-WCGAN-GP model.

Similarly, Fig. 24 presents post hoc SHAP-based global explanations for the global DNN model on the UNSW-NB15 dataset, using both real data and synthetic data generated by the FL-WCGAN-GP generative model on the server. The figure compares the global feature importance of influential features obtained from real data (Fig. 24a) and synthetic data (Fig. 24b) to assess whether synthetic data-based feature importance can reliably approximate the global feature attributions of real data. Most influential features are consistent between the real data and synthetic data points, with 'sttl' and 'ct_state_ttl' appearing as the top influential features in the global DNN model's predictions.

Fig. 25 presents the SHAP-based global feature importance for the global DNN model on the CIC-IoT2023 dataset. It compares the top influential features obtained from real data (see Fig. 25a) and synthetic data generated by the FL-WCGAN-GP model (see Fig. 25b). The most influential features, such as "Number," ack_flag_number", Number of packets with syn flag set in the same flow (syn_count) & HTTPS show strong alignment in both the SHAP mean values from real data and mean SHAP values from synthetic data.

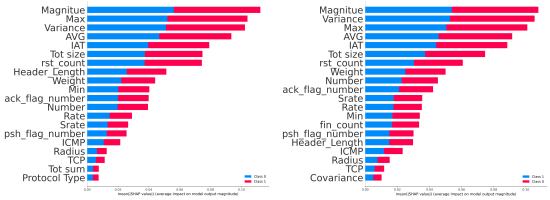
Similarly, Fig. 26 presents the SHAP global feature importance for the global DNN model trained on the CIC-IoMT2024 dataset, using both real data(Fig. 26a) and synthetic data generated by the FL-WCGAN-GP model (Fig. 26b). Features such as Magnitude, Variance of the packet lengths in the window (Variance) , Max, AVG, Interval mean between the current and previous packet in the window (IAT), TotSize, and rst_count appear as the most influential in both the real-data-based and synthetic-data-based explanations.

Overall, the top influential features identified in both real data-based explanations (when the server has access to real client data) and synthetic data-based explanations (generated by the FL-WCGAN-GP model) are significantly consistent in both post hoc local explanations and global explanations for the global DNN model on the server. As a result, post-hoc explanations derived from synthetic data closely approximate the explanations obtained when the server accesses real data, offering a privacy-preserving alternative to real data for providing explanations to a global DNN model on the server without compromising privacy in FL.



(a) Real Data (b) Synthetic data

Fig. 25: SHAP global feature importance for the global DNN model on the server using CIC-IoT2023 dataset, based on (a) real data and (b) synthetic data generated by the FL-WCGAN-GP model.



(a) Real Data (b) Synthetic data

Fig. 26: SHAP global feature importance for the global DNN model on the server using CIC-IoMT2024 dataset, based on (a) real data and (b) synthetic data generated by the FL-WCGAN-GP model.

D. Explaining local DNN models in FL

In FL settings, post-hoc XAI methods can be directly applied to local DNN models on the client side without violating privacy constraints, as each client retains access to its data. XAI methods can utilize the client's training data as reference or background data to generate explanations for local DNN models on the client side. However, providing post-hoc explanations for all client models becomes impractical as the number of clients increases. Therefore, we report experimental results for N=5 clients, as shown in Table IV. We observed that the top features identified by the global DNN model were mostly the same as features found in local client models. We evaluated the post-hoc local explanations generated by LIME and SHAP across all clients and all four datasets, using two quantitative metrics, high faithfulness and max sensitivity. The evaluation metrics were computed over the test data of each client. As shown in Table IV, SHAP consistently performed better than LIME in terms of both high faithfulness (μ_f) and low max sensitivity (μ_s) across all datasets and clients when explaining the local DNN models using its client data. SHAP produces higher mean values μ_f, indicating that its feature attributions are more strongly correlated with the actual influence of features on the model's predictions. Similarly, for all datasets, SHAP produces lower max sensitivity mean values for all client DNN models, demonstrating that it provides more robust and consistent explanations for nearby data points in the feature space.

VI. Discussions

In this study, we proposed a privacy-preserving and trustworthy FL framework that integrates XAI to enable a transparent FL-based IDS through synthetic data-driven explainability. The proposed framework is composed of two key components. Firstly, a federated DNN model is designed for classifying network traffic as either benign or malicious. Secondly, a federated synthetic data generation module, which enables the explanation of the global DNN model on the server side without requiring access to real client data, thereby preserving privacy while allowing for transparent DNN model decisions in FL-based IDS deployments. To evaluate the effectiveness of our approach, we conducted experiments using four network traffic datasets, 1) NSL-KDD,2) UNSW-NB15, 3)CIC-

TABLE IV: Evaluation of SHAP and LIME local explanations on real data for local DNN models on client-side using Faithfulness (μ_f) and Max Sensitivity (μ_s) metrics across four datasets

Data Set	Clients	C1		C2		C3		C4		C5	
	Metric/Explainer	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP	LIME	SHAP
NSL-KDD	μ_f	0.19 ± 0.15	0.61 ± 0.30	0.23 ± 0.18	0.69 ± 0.33	0.23 ± 0.22	0.66 ± 0.34	0.20 ± 0.16	0.66 ± 0.28	0.19 ± 0.15	0.61 ± 0.29
	μ_s	0.18 ± 0.05	0.02 ± 0.05	0.17 ± 0.05	0.02 ± 0.04	0.17 ± 0.05	0.02 ± 0.03	0.17 ± 0.05	0.02 ± 0.03	0.18 ± 0.05	0.02 ± 0.03
UNSW-nb15	μ_f	0.26 ± 0.35	0.64 ± 0.39	0.24 ± 0.38	0.65 ± 0.41	0.24 ± 0.35	0.62 ± 0.42	0.24 ± 0.32	0.63 ± 0.40	0.23 ± 0.39	0.63 ± 0.41
	μ_s	0.04 ± 0.01	0.03 ± 0.07	0.04 ± 0.01	0.03 ± 0.07	0.04 ± 0.01	0.03 ± 0.06	0.04 ± 0.01	0.04 ± 0.07	0.04 ± 0.01	0.03 ± 0.06
CIC-IoT2023	μ_f	0.25 ± 0.32	0.57 ± 0.20	0.24 ± 0.34	0.71 ± 0.21	0.21 ± 0.30	0.71 ± 0.19	0.33 ± 0.39	0.76 ± 0.22	0.22 ± 0.31	0.73 ± 0.18
	μ_s	0.13 ± 0.05	0.04 ± 0.04	0.13 ± 0.04	0.04 ± 0.04	0.15 ± 0.05	0.05 ± 0.04	0.14 ± 0.04	0.04 ± 0.05	0.14 ± 0.05	0.05 ± 0.05
CIC-IoMT2024	μ_f	0.23 ± 0.22	0.80 ± 0.37	0.14 ± 0.18	0.88 ± 0.32	0.16 ± 0.22	0.84 ± 0.36	0.22 ± 0.24	0.89 ± 0.39	0.23 ± 0.23	0.80 ± 0.38
	μ_s	0.19 ± 0.06	0.03 ± 0.04	0.19 ± 0.05	0.03 ± 0.03	0.19 ± 0.05	0.03 ± 0.03	0.20 ± 0.06	0.03 ± 0.03	0.19 ± 0.05	0.02 ± 0.02

IoT2023, and 4) CIC-IoMT2024. Each dataset contains various statistical features and types of cyberattacks relevant to real-world IoT environments. The following discussion is based on the experimental findings across these four datasets.

TABLE V: Hyper Parameters used for training Federated DNN model

Hyper Parameter	Value
Hidden Layers	5
Input size	d
Output size	C=2
Activation	SeLU
Learning Rate	0.001
Optimizer	Adam
Batch Size	256
Epochs per client	100

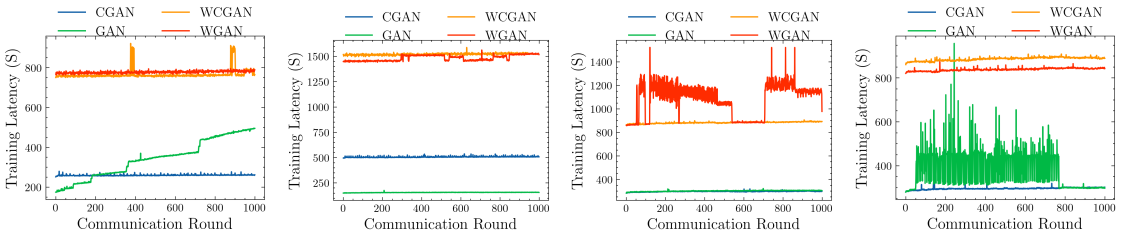
In this study, we developed an HFL-based DNN model using the FedAvg algorithm to perform binary classification of network traffic as either benign or malicious. To simulate real-world data heterogeneity, we employed a Dirichlet distribution to create non-IID data partitions across clients. The hyperparameters used for training the FL-DNN model across all four datasets are shown in Table V. The input layer size d, which corresponds to the number of features in each dataset, varied depending on the dataset. Specifically, $d = 41$ for NSL-KDD, $d = 46$ for UNSW-NB15, $d = 45$ for CIC-IoMT2024, and $d = 39$ for CIC-IoT2023. The model architecture consisted of five hidden layers, and the output layer ($C = 2$) was configured for binary classification. We used the SeLU (Scaled Exponential Linear Unit) activation function in the hidden layers, along with the Adam optimizer, a batch size (B) of 256, and 100 training epochs (e) per client. Across all four datasets, both the global and local DNN models achieved performance above 98% in terms of accuracy, precision, recall, and F1-score, demonstrating the effectiveness of the proposed model for detecting malicious network traffic in federated edge environments.

A key challenge in applying post-hoc explainability methods such as SHAP and LIME in FL is their dependence on background data taken from the training distribution. Since FL prevents server-side access to client data, using these XAI methods directly raises privacy concerns and violates the core principles of the FL framework. To address this issue, we propose a synthetic data-driven explainability approach for interpreting the global model on the server. Instead of accessing real client data, we train federated generative models to produce synthetic samples, which serve as input references for SHAP and LIME, allowing privacy-preserving interpretation of the global DNN model. To generate synthetic data for this purpose, we implemented several federated generative model variants, including FL-GAN, FL-CGAN, FL-WGAN-GP, and FL-WCGAN-GP using the FedAvg algorithm in HFL settings under non-IID conditions. The training setup and hyperparameters are summarized in Table VI. All models consisted of five fully connected hidden layers and employed either ReLU or Sigmoid activation functions, depending on the variant. The input and output dimensions were set to match the number of features ddd in each dataset, excluding the class label. For the conditional variants (FL-CGAN and FL-WCGAN-GP), class labels were embedded and appended to the input of both the generator and discriminator to enable class-conditional data generation. We used the Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.9$, and each client trained its local model for $e = 500$ with a $B = 256$. In the conditional generative variant models, the discriminator received both feature vectors and their corresponding labels, allowing it to better distinguish class-specific patterns in the generated data.

TABLE VI: Hyperparameters used for training federated generative model variants (GAN, CGAN, WGAN, WCGAN)

Hyper parameter	value
Hidden Layers	5
Activation function	ReLU/Sigmoid
Input/Output size	d features
Conditional Input	Label embedding used in CGAN and WCGAN-GP (C = 2)
Optimizer	Adam (β_1 = 0.5, β_2 = 0.9)
Batch Size	256
Epochs per client	500

Fig. 27 illustrates the training latency of the generative models, measured as the average communication time per round. Training latency refers to the time required for clients and the server to complete one round of training. Among the models, FL-WCGAN-GP demonstrates the highest latency, followed by FL-WGAN-GP, FL-CGAN, and FL-GAN across all four datasets. Notably, FL-GAN shows the lowest latency. In contrast, FL-WCGAN-GP has a high latency, which is



(a) NSL-KDD (b) UNSW-NB15 (c) CIC-IoMT (d) CIC-IoT2023

Fig. 27: Training latency (communication time per round) of FL with GAN, CGAN, WGAN-GP, and WCGAN-GP models across four datasets: NSL-KDD, UNSW-NB15, CIC-IoMT, and CIC-IoT2023.

primarily due to the computational complexity introduced by the Wasserstein loss and gradient penalty.

To evaluate the quality of the generated synthetic data, we used the Wasserstein distance (W_D). Among the federated generative models, FL-WCGAN-GP consistently achieved the lowest W_D across all datasets, indicating high fidelity to the original data distribution. As a result, FL-WCGAN-GP was selected to generate synthetic samples on the server for post-hoc explainability. To further assess synthetic data quality, we used Pearson correlation to identify feature-wise deviations from real data, while PCA and t-SNE were applied to visualize distributional differences between synthetic and real samples. When tested on the trained global DNN model, the synthetic data yielded classification accuracies of 96.32% on NSL-KDD, 95.12% on UNSW-NB15, 97.11% on CIC-IoT2023, and 98.86% on CIC-IoMT2024.

To explain the global DNN model on the server, SHAP and LIME were applied using synthetic data generated by the FL-WCGAN-GP model. To assess the adequacy of these explanations, we compared those derived from synthetic data with SHAP explanations generated using real data (available to the server for evaluation only). The results showed that synthetic data-driven SHAP explanations closely approximated with SHAP explanations based on real data, indicating that the proposed approach enables effective model explainability without compromising client privacy. Moreover, the top influential features identified using synthetic data were consistent with those from real data, validating the reliability of the synthetic data-driven explanations. In future work, we plan to extend this analysis to systematically evaluate global explanations across the federated client population.

We further evaluated the post-hoc local explanations generated using synthetic data for the global DNN model by applying SHAP and LIME. The evaluation considered two key criteria. The first was faithfulness, measured through correlation with the model output, and the second was robustness, assessed using the max sensitivity metric. The results show that SHAP provided more faithful and robust explanations compared to LIME for global DNN model based on synthetic data. Similarly, when local explanations were generated using real data for individual client-side DNN models, SHAP again outperformed LIME in terms of both faithfulness and robustness, highlighting its reliability in interpreting FL-based IDS models and suggesting that security analysts can have a degree of trust in its explanations. Overall, the results show that synthetic data is an effective, privacy-preserving alternative to real client data for explaining FL-based IDS models, especially black-box models like DNNs.

IoT applications operate in multi-vendor ecosystems, where IoT devices or services (e.g., remote patient monitoring) deployed to the clients are managed by different vendors than the underlying IT resources, such as computers or home modem routers. It poses a significant risk for vendors to deploy an intrusion detection sensor (e.g., a network sensor at the modem router in our application), collect all data, and relay it to a central location, as this data can contain sensitive medical information that is managed by other vendors. In such situations, rigorous data filtering should be applied to remove information from third-party devices to protect privacy, which is practically difficult to do effectively. Our work primarily targets edge devices with greater computational capabilities, rather than on ultra-constrained IoT nodes.

However, in FL settings, models deployed locally do not need to share the data with the vendor, instead, they share the model parameters that only lead to the intrusion detection decision without leaking additional information. On the other side, the security experts working at the vendor side still need insights into whether the model works as intended and understand the intrusion details. The explanations identified by our proposed method under the FL setting are instrumental in addressing these requirements while protecting privacy, especially in multi-vendor environments.

A. Limitations and Threats to validity

1) In this study, we use federated synthetic data to explain the global FL-DNN model without accessing real client data. While our results show that synthetic data can approximate real data for generating explanations, this may not always be the case in real-world scenarios. If synthetic data doesn't accurately capture the statistical properties and feature distributions of real data, the resulting explanations could differ significantly from Explanations based on actual client data, potentially limiting the reliability of post-hoc explainability methods. Conversely, if the Generator replicates client dis-

tributions too closely, synthetic data may leak feature distributions that enable inference attacks.

2) Training high-quality synthetic data in a federated setting is computationally intensive. Wasserstein-based GAN variants like FL-WGAN-GP and FL-WCGAN-GP require longer training times due to additional constraints such as weight clipping and gradient penalty regularization.
3) Our study is vulnerable to inference attacks that can expose sensitive client model information due to the absence of differential privacy [60], secure multi-party computation, and homomorphic encryption [61]. SHAP values, computed post-hoc, are also susceptible to adversarial manipulation [62], which can mislead security analysts. In future work, we aim to incorporate encryption techniques and adversarial training to enhance data privacy in FL and improve the reliability of SHAP-based explanations.
4) In this study, we assumed that the data features used by IDS models are understandable to security experts. Therefore, we did not evaluate how specific feature attributions, particularly in global explanations, influence decision-making in practice. Our main goal was to develop and evaluate a federated explainable framework for IDS, rather than examine analyst interactions or incident-handling workflows in SOC. While we recognize the importance of these aspects, they were outside the scope of this study. In the future, we plan to involve SOC analysts in user studies to determine if synthetic data-driven explanations offer actionable insights for cybersecurity operations.

VII. Conclusion

In this study, we propose a federated synthetic data-driven explainability framework for FL-based IDS. Our approach addresses the challenge of explaining complex DL models while preserving client data privacy. By using FL-based generative models to generate high-quality synthetic data as reference input for a post-hoc XAI method, enabling the server-side model to be explained without exposing sensitive client information. We developed a privacy-preserving FL-based IDS using an FL-DNN classifier trained with the FedAvg algorithm within a HFL framework. The model achieved high accuracy, precision, recall, and F1-score in detecting malicious and benign network traffic across the NSL-KDD, UNSW-NB15, CIC-IoT2023, and CIC-IoMT2024 datasets, demonstrating both rapid convergence and stable performance while preserving data privacy.

In our study, we employed several generative model variants for federated synthetic data generation, including FL GAN, FL CGAN, FL WGAN GP, and FL WCGAN GP, each trained using the FedAvg algorithm. Among these, FL WCGAN GP consistently outperformed the others by achieving the lowest Wasserstein Distance, indicating a high similarity between the generated synthetic data and the real client data. We used the synthetic data generated by the global FL WCGAN GP model to explain the global FL DNN model using a post-hoc explainer, specifically LIME and SHAP.

We quantitatively evaluated LIME and SHAP post-hoc local explanations based on two key metrics: faithfulness and robustness. The results show that SHAP provides more faithful and robust explanations than LIME for both client-side models using real data and server-side models using synthetic data, supporting privacy-preserving explainability in FL-based IDS.

References

[1] F. Skopik, M. Landauer, and M. Wurzenberger, "Blind spots of security monitoring in enterprise infrastructures: a survey," *IEEE Security & Privacy*, vol. 20, no. 6, pp. 18–26, 2022.

[2] P. Svenmarck, L. Luotsinen, M. Nilsson, and J. Schubert, "Possibilities and challenges for artificial intelligence in military applications," in *Proceedings of the NATO Big Data and Artificial Intelligence for Military Decision Making Specialists' Meeting*, vol. 1, 2018.

[3] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[4] Z. Wang, Z. Li, D. He, and S. Chan, "A lightweight approach for network intrusion detection in industrial cyber-physical systems based on knowledge distillation and deep metric learning," *Expert Systems with Applications*, vol. 206, p. 117671, 2022.

[5] R. Kalakoti, H. Bahsi, and S. Nõmm, "Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection," *IEEE Internet of Things Journal*, 2024.

[6] R. Kalakoti, S. Nõmm, and H. Bahsi, "In-depth feature selection for the statistical machine learning-based botnet detection in iot networks," *IEEE Access*, vol. 10, pp. 94 518–94 535, 2022.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[8] X. Huang, J. Liu, Y. Lai, B. Mao, and H. Lyu, "Eefed: Personalized federated learning of execution&evaluation dual network for cps intrusion detection," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 41–56, 2022.

[9] R. Kalakoti, H. Bahsi, and S. Nõmm, "Explainable federated learning for botnet detection in iot networks," in *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2024, pp. 01–08.

[10] R. Kalakoti, R. Vaarandi, H. Bahşi, and S. Nõmm, "Evaluating explainable ai for deep learning-based network intrusion detection system alert classification," in *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, INSTICC. SciTePress, 2025, pp. 47–58.

[11] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[12] R. López-Blanco, R. S. Alonso, A. González-Arrieta, P. Chamoso, and J. Prieto, "Federated learning of explainable artificial intelligence (fed-xai): a review," in *International Symposium on Distributed Computing and Artificial Intelligence*. Springer, 2023, pp. 318–326.

[13] R. Kalakoti, S. Nõmm, and H. Bahsi, "Federated learning of explainable ai (fedxai) for deep learning-based intrusion detection in iot networks," *Computer Networks*, p. 111479, 2025.

[14] J. L. C. Bárcena, M. Daole, P. Ducange, F. Marcelloni, A. Renda, F. Ruffini, and A. Schiavo, "Fed-xai: Federated learning of explainable artificial intelligence models." in *XAI. it@ AI* IA*, 2022, pp. 104–117.

[15] GDPR, "General data protection regulation (gdpr) – official legal text," https://gdpr-info.eu/, April 2016, (Accessed on 20/12/2025).

[16] F. Yang and J. Xu, "Privacy concerns in china's smart city campaign: The deficit of china's cybersecurity law," *Asia & the Pacific Policy Studies*, vol. 5, no. 3, pp. 533–543, 2018.

[17] J. Kim, K. Jeong, H. Choi, and K. Seo, "Gan-based anomaly detection in imbalance problems," in *European Conference on Computer Vision*. Springer, 2020, pp. 128–145.

[18] G. Andresini, A. Appice, L. De Rose, and D. Malerba, "Gan augmentation to deal with imbalance in imaging-based intrusion detection," *Future Generation Computer Systems*, vol. 123, pp. 108–127, 2021.

[19] A. Tabassum, A. Erbad, W. Lebda, A. Mohamed, and M. Guizani, "Fedgan-ids: Privacy-preserving ids using gan and federated learning," *Computer Communications*, vol. 192, pp. 299–310, 2022.

[20] P. Barnard, N. Marchetti, and L. A. DaSilva, "Robust network intrusion detection through explainable artificial intelligence (xai)," *IEEE Networking Letters*, vol. 4, no. 3, pp. 167–171, 2022.

[21] R. Zhao, "Nsl-kdd," 2022, dataset. [Online]. Available: https://dx.doi.org/10.21227/8rpg-qt98

[22] H. Liu, C. Zhong, A. Alnusair, and S. R. Islam, "Faixid: A framework for enhancing ai explainability of intrusion detection results using data cleaning techniques," *Journal of network and systems management*, vol. 29, no. 4, p. 40, 2021.

[23] A. Oseni, N. Moustafa, G. Creech, N. Sohrabi, A. Strelzoff, Z. Tari, and I. Linkov, "An explainable deep learning framework for resilient intrusion detection in iot-enabled transportation networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 1000–1014, 2022.

[24] B. Sharma, L. Sharma, C. Lal, and S. Roy, "Explainable artificial intelligence for intrusion detection in iot networks: A deep learning based approach," *Expert Systems with Applications*, vol. 238, p. 121751, 2024.

[25] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[26] S. Neupane, J. Ables, W. Anderson, S. Mittal, S. Rahimi, I. Banicescu, and M. Seale, "Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities," *IEEE Access*, vol. 10, pp. 112 392–112 415, 2022.

[27] H. S. Eriksson and G. Grov, "Towards xai in the soc–a user centric study of explainable alerts with shap and lime," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 2595–2600.

[28] R. Kalakoti, S. Nõmm, and H. Bahsi, "Enhancing iot botnet attack detection in socs with an explainable active learning framework," in *2024 IEEE World AI IoT Congress (AIIoT)*. IEEE, 2024, pp. 265–272.

[29] M. Wang, K. Zheng, Y. Yang, and X. Wang, "An explainable machine learning framework for intrusion detection systems," *IEEE Access*, vol. 8, pp. 73 127–73 141, 2020.

[30] K. L. K. Sudheera, D. M. Divakaran, R. P. Singh, and M. Gurusamy, "Adept: Detection and identification of correlated attack stages in iot networks," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6591–6607, 2021.

[31] P. Chen, X. Du, Z. Lu, J. Wu, and P. C. Hung, "Evfl: An explainable vertical federated learning for data-oriented artificial intelligence systems," *Journal of Systems Architecture*, vol. 126, p. 102474, 2022.

[32] J. Fiosina, "Explainable federated learning for taxi travel time prediction." in *VEHITS*, 2021, pp. 670–677.

[33] J. L. C. Bárcena, P. Ducange, A. Ercolani, F. Marcelloni, and A. Renda, "An approach to federated learning of explainable fuzzy regression models," in *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2022, pp. 1–8.

[34] A. Oki, Y. Ogawa, K. Ota, and M. Dong, "Evaluation of applying federated learning to distributed intrusion detection systems through explainable ai," *IEEE Networking Letters*, 2024.

[35] R. Haffar, D. Sanchez, and J. Domingo-Ferrer, "Explaining predictions and attacks in federated learning via random forests," *Applied Intelligence*, vol. 53, no. 1, pp. 169–185, 2023.

[36] T. T. Huong, T. P. Bac, K. N. Ha, N. V. Hoang, N. X. Hoang, N. T. Hung, and K. P. Tran, "Federated learning-based explainable anomaly detection for industrial control systems," *IEEE Access*, vol. 10, pp. 53 854–53 872, 2022.

[37] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[38] E. Albini, J. Long, D. Dervovic, and D. Magazzeni, "Counterfactual shapley additive explanations," in *Proceedings of the 2022 ACM conference on fairness, accountability, and transparency*, 2022, pp. 1054–1070.

[39] V. Holubenko, D. Gaspar, R. Leal, and P. Silva, "Autonomous intrusion detection for iot: a decentralized and privacy preserving approach," *International Journal of Information Security*, vol. 24, no. 1, p. 7, 2025.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural infor-*

mation processing systems, vol. 27, 2014.
[41] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.
[42] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.
[43] C. Villani *et al.*, *Optimal transport: old and new*. Springer, 2008, vol. 338.
[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[45] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.
[46] D. Castelvecchi, "Can we open the black box of ai?" *Nature News*, vol. 538, no. 7623, p. 20, 2016.
[47] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.
[48] L. S. Shapley *et al.*, "A value for n-person games," 1953.
[49] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowledge and information systems*, vol. 41, pp. 647–665, 2014.
[50] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*, 2018.
[51] U. Bhatt, A. Weller, and J. M. Moura, "Evaluating and aggregating feature-based model explanations," *arXiv preprint arXiv:2005.00631*, 2020.
[52] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Pearson Education India, 2016.
[53] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.
[54] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.
[55] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.
[56] S. Dadkhah, E. C. P. Neto, R. Ferreira, R. C. Molokwu, S. Sadeghi, and A. Ghorbani, "Ciciomt2024: Attack vectors in healthcare devices-a multi-protocol dataset for assessing iomt device security," 2024.
[57] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.
[58] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.
[59] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html
[60] M. Asif, S. Naz, F. Ali, A. Salam, F. Amin, F. Ullah, and A. Alabrah, "Advanced zero-shot learning (azsl) framework for secure model generalization in federated learning," *IEEE Access*, 2024.
[61] A. Salam, M. Abrar, F. Ullah, I. A. Khan, F. Amin, and G. S. Choi, "Efficient data collaboration using multi-party privacy preserving machine learning framework," *IEEE Access*, vol. 11, pp. 138 151–138 164, 2023.
[62] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 180–186.



Rajesh Kalakoti is a Ph.D. student and early-stage researcher in XAI within the cybersecurity domain at Tallinn University of Technology. He received his master's degree in Information Technology from JntuK, Kakinada, in 2015. Rajesh has worked as a lecturer in Information Technology and as a Software Engineer in India.

His research interests focus on IoT Security, network traffic, Malware, HTTP-Based Botnet, Machine Learning, Federated Learning, Deep Learning, and XAI. Rajesh is a member of the IEEE Computer Society.



Hayretdin Bahsi is a research professor at the Center for Digital Forensics and Cyber Security at Tallinn University of Technology in Tallinn, Estonia, and assistant professor at School of Informatics, Computing, and Cyber Systems, Northern Arizona University, United States. He received his Ph.D. and MSc degrees in Computer Engineering from Sabanci University and Bilkent University, in 2002 and 2010, respectively.

His research interests include cyber-physical system security and the application of machine learning methods to cyber-security problems.



Sven Nõmm is the tenured associate professor at Talinn University of Technology -TalTech. He received his Ph.D. degree jointly from Tallinn University of Technology, Estonia, and École Centrale de Nantes et L'Université de Nantes, France, in 2004.

His research interests include human-machine interaction, analysis of human motions, applications of artificial intelligence (AI) to cybersecurity problems, and geoscience.

Appendix 11

XI

M. U. Rehman, R. Kalakoti, and H. Bahşi. Exploring the impact of feature selection on non-stationary intrusion detection models in iot networks. In *Accepted for 22nd Annual International Conference on Privacy, Security, and Trust (PST2025)*, 2025

Exploring the Impact of Feature Selection on Non-Stationary Intrusion Detection Models in IoT Networks

Muaan Ur Rehman[1], Hayretdin Bahşi[1,2], and Rajesh Kalakoti[1]

[1]Department of Software Science, Tallinn University of Technology, Tallinn, Estonia
[2]School of Informatics, Computing and Cyber Systems, Northern Arizona University, United States
Email:{muaan.ur, hayretdin.bahsi, rajesh.kalakoti }@taltech.ee

***Abstract*—The proliferation of Internet of Things (IoT) devices has increased the attack surface of networks, necessitating robust and adaptive security mechanisms such as machine learning (ML)-based intrusion detection systems (IDS). However, the effectiveness of these systems can degrade over time due to concept drift, where patterns in data evolve as attackers develop new techniques. This study investigates the role of feature selection in enhancing the long-term performance of non-stationary IDS models in IoT networks. Specifically, we apply a filter-based feature reduction technique, Mutual Information, in conjunction with XGBoost models, to evaluate two learning paradigms i.e. static (trained once) and dynamic (periodically retrained). Using the CICIoMT2024 dataset, which includes 18 attack variants across five major categories, we conduct multiclass classification to provide a granular analysis of security threats. Our results demonstrate how selected features perform under different drift conditions and highlight critical network features for evolving attack detection. The study offers new insights into the interplay between feature selection and model adaptability in dynamic IoT environments, aiming to inform the development of more resilient IDS solutions.**

***Index Terms*—Concept Drift, Non-stationary Models, Feature Selection, Machine Learning, Intrusion Detection System**

I. Introduction

The rapid growth of the IoT has led to an increase in connected devices, making it a vital part of daily life while simultaneously heightening security risks. IoT security is critical, given the potential for financial damage or threats to human safety [1]. IDS play a key role in monitoring malicious activities and, thus, ensuring the reliability and security of these networks. ML has been applied to IDS [2], including in IoT networks, as it can identify complex attack patterns and adapt to evolving threats without relying on the creation of specific rules.

ML is widely incorporated into cybersecurity for tasks such as identifying malware, detecting intrusions, detecting online abuse, and recognizing malicious events. However, ML models in real-world applications can encounter "concept drift", which means the patterns they learned from data change over time [3]. This can occur because the underlying data evolves or the initial data used for training were unstable or poorly defined. Furthermore, as cyber threats continuously evolve and new attack vectors emerge in cyberspace due to the continuous attack-defense game between attackers and defenders, it becomes essential to employ non-stationary IDS models that can be regularly retrained or updated to maintain robust detection performance over time. However, IoT devices have limited computational resources, making it essential to reduce data dimensions and select the most relevant features to ensure that ML-based IDS operates efficiently without overburdening the network.

There is a line of research on feature selection for stationary ML-based intrusion detection in IoT devices [4]–[6]. Although handling concept drift through non-stationary models has been studied in network intrusion detection [7], the impact of feature selection on the long-term performance of these models has not been elaborated. It is essential to understand how feature selection operates in the context of continuous threat evolution and identify the most effective features over time.

In this paper, we apply a filter-based feature selection method to non-stationary intrusion detection model alternatives to understand their long-term performance with varying numbers of feature sets and explore how network features evolve under different drift conditions. We induced two typical non-stationary models, namely static and dynamic, for the intrusion detection task in IoT networks. The static model represents the case where a model is trained in the first time interval and is not updated throughout the whole timeline, whereas the dynamic model is updated in each time interval with the newly introduced attack types.

We utilized a recent and well-developed dataset in IoT networks, CICIoMT2024 [8], for training and testing our models. In our previous work [6], filter-based methods were benchmarked in a stationary setting using the same dataset. Since XGBoost and mutual information demonstrated strong performance in that study, we adopt them in this work as the classifier and feature selection method, respectively.

We create an evolution timeline and various drift scenarios using benign samples and attack variants from CICIoMT2024. We evaluate the static and dynamic models in terms of accuracy score by focusing on multiclassification, categorizing traffic into specific attack categories. We consider

that multiclass classification enables a granular understanding about the security threats and guides the security incident handling experts better about deciding on the relevant defensive actions. The CICIoMT2024 dataset includes attacks in five main categories: DDoS, DoS, Reconnaissance, MQTT and Spoofing (i.e., additionally dataset has benign samples). In each category, there are attack variants (e.g., DoS-TCP, DoS-UDP, and DoS-ICMP in the DoS category), totaling 18 attack variants. The CICIoMT2024 dataset uses network flow features extracted from benign and malicious traffic. This study examines key network features within the datasets that are essential for classifying attacks in a non-stationary model setting. Our multiclass model differentiates between the main categories, while new attack variants can emerge over time.

By highlighting critical features across IoT datasets, this study contributes to more robust, feature-driven methods for evolving attack variants in IoT environments, ultimately aiming to strengthen the security and reliability of these networks. Our work compares static and dynamic non-stationary models under drift scenarios and analyzes feature selection dynamics in the dynamic setting. We put a particular emphasis on feature selection in multi-class classification settings, which has not been elaborated on well in the literature.

The rest of the paper is organized as follows. Section II reviews the related work and gives background information about impact of feature selection in non-stationary models. Section III presents the methodology used in our paper. In Section IV , we show our results and discuss them in Section V. Finally, Section VI concludes the paper and discusses future directions.

II. Related Work

Most learning models are static and assume that the input data remains stable over time, a concept known as stationary data [9]. However, a challenging aspect of real-world systems is their time-varying behaviour, as many problem domains involve non-stationary data distributions. Addressing Concept Drift and Feature Drift in IDS is essential for creating resilient and adaptive security systems that can effectively respond to the evolving landscape of cyber threats.

A paper [10] presents a feature selection approach in a stationary model using Mutual Information (MI) and XGBoost to optimize performance and reduce computational costs in IoMT networks. A mathematical intersection was applied to efficiently extract common features, enabling effective ID during data transfer and enhancing healthcare data analysis at the network's edge. Evaluated on the CICIDS2017 dataset [11], the model achieved 98.79% accuracy with a low false alarm rate of 0.007 FAR, focusing on binary classification for intrusion detection. In a paper [4], the authors proposed a feature selection technique that primarily uses filter and wrapper methods for the statistical ML detection of IoT botnet attacks. They reported higher detection rates with a smaller set of selected features. Both multi-class and binary classification models were evaluated on the N-BaIoT and MedBIoT datasets. The wrapper models achieved better performance with a limited number of features. A paper [12] presented a lightweight method for the early detection of IoT botnets, utilizing feature selection prior to training a one-class KNN classifier. This approach achieved an F1-score ranging from 98% to 99% across various IoT botnet datasets. Some studies have also applied static model-based feature selection methods prior to using explainable techniques in IoT botnet detection problems [13]. There are numerous methodologies in the literature regarding concept drift-aware IDS methods and models aimed at identifying attacks in streamed data. A paper [14] introduced a method for detecting concept drift and DDoS attacks in networks, aimed at enhancing data representation and improving the prediction of cybersecurity threats. It employs a secure adaptive windowing and website data authentication protocol (SAW_WDA) to analyze the network's intrusion detection capabilities. A study [7] discussed a drift detection mechanism that employs Principal Component Analysis (PCA) to monitor changes in feature variance within intrusion data streams. Concept drift occurs when the relationship between input and output data in a ML model changes over time.

A study [15] proposed a drift detection technique using principal component analysis (PCA) to monitor variance changes in intrusion detection data streams and an online outlier detection method for identifying deviations from both historical and nearby data points. To tackle these drifts, an online deep neural network (DNN) was developed, which dynamically adjusts its hidden layer sizes using a Hedge weighting mechanism. This allows the model to adapt to new intrusion data continuously. Experiments on an IoT-based intrusion detection dataset showed that this approach stabilizes performance for both training and testing, compared to conventional static DNN models. A paper [16] introduces a framework designed to adapt deep learning models for use in online IDS that can handle dynamic attack patterns and benign traffic variations. This framework integrates continuous deep anomaly detection with federated learning to address traffic concept drift, which includes emerging attack patterns and evolving benign behavior. Additionally, it incorporates sequential packet labeling for analyzing network flows, providing updated probabilities of attack through continuous packet analysis. A paper [17] presented a detailed network analytics model for adapting to concept drift in industrial IoT systems. This model encompasses dynamic feature selection, dynamic model training, and a probability-based ensemble approach. The study [18] introduces CLEAR (Concept Learning for Intrusion Event Aggregation), a system that aggregates intrusion events in real-time using concept learning. Without requiring pre-existing data, CLEAR dynamically generates and refines 'concepts' or clusters of statistically similar alerts as they arise. These evolving concepts then help quickly and accurately cluster new alerts.

While prior work has addressed feature selection or concept drift separately, our study examines their combined impact in a non-stationary, multiclass IoT intrusion detection setting. We compare static and dynamic adaptation strategies and track

feature relevance over time, offering new insights into long-term IDS performance.

III. Methodology

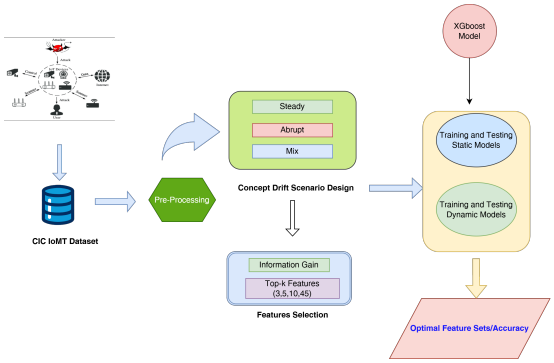


Figure 1: Methodology

This methodology provides a comprehensive framework for understanding model robustness and adaptability in non-stationary environments under evolving attack landscapes (Concept Drift), while also considering the role of feature selection. We applied a ML workflow that includes dataset preprocessing, concept drift scenario design, feature selection, and static and dynamic model training and testing, as illustrated in Fig 1. In the static model setup, the model is trained only once on the initial training set at t_0 and subsequently tested across each time window (t_0, ..., t_n). In contrast, for dynamic models, before testing at each time window t_i (t_2, ..., t_n), the model is retrained at every step using the updated training set from the previous window, i.e., t_{i-1}. In the data pre-processing stage, we eliminated the the null values, converted the string values to numerical values, labeled the files, and constructed train and test sets. After pre-processing the dataset, we designed various attack drift scenarios. We applied filter-based feature selection methods, specifically Mutual Information, to prioritize features at each time window. In the final stage, we trained and tested the ML algorithm (XGBoost) both statically and dynamically, using varying numbers of top-ranked features selected based on Mutual Information.

A. CICIoMT2024 Dataset

We apply the Concept Drift analysis to the CICIoMTDataset 2024 dataset [8], which focus on Internet of Medical Things devices (IoMT) in the healthcare sector. This is a rich dataset (Multiple attacks variants), designed to assess and improve the cybersecurity of IoMT devices through IDS. The dataset encompasses traffic generated by 40 devices (25 real and 15 simulated) across various protocols, including Wi-Fi, MQTT, and Bluetooth. The authors simulated 18 different cyberattacks, grouped into five main categories: DDoS, DoS, Reconnaissance, MQTT-specific attacks, and Spoofing as shown in Fig 2. Table I further illustrate the attacks sub categories in each major attack. The DDoS and DoS categories include traditional flooding attacks such as SYN, TCP, UDP, and ICMP floods aimed at overwhelming device resources. Reconnaissance attacks like Port Scans, OS Scans, Ping Sweeps, and Vulnerability Scans were performed to simulate information-gathering phases. MQTT-specific attacks involved Connect and Publish Floods, as well as sending malformed MQTT packets to disrupt broker communication. Spoofing attacks, such as ARP spoofing, targeted network layer vulnerabilities to manipulate or intercept device communications. The original dataset comprises 45 total features which including header and flow metadata, TCP/IP flag-related indicators, protocol types, statistical metrics,and other network-related characteristics.

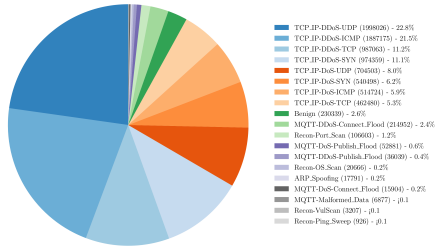


Figure 2: Pie chart for attacks distribution in CICIoMT2024 dataset

Table I: Attack categories and attack variants in each category

Attack Categories	Attack Variants
MQTT	MQTT-Malformed_Data MQTT-DoS-Connect_Flood MQTT-DoS-Publish_Flood MQTT-DDoS-Publish_Flood MQTT-DDoS-Connect_Flood
DoS	TCP_IP-DoS-TCP TCP_IP-DoS-ICMP TCP_IP-DoS-UDP
DDoS	TCP_IP-DDoS-UDP TCP_IP-DDoS-ICMP TCP_IP-DDoS-SYN TCP_IP-DDoS-TCP
Recon	Recon-Port_Scan Recon-OS_Scan Recon-VulScan Recon-Ping_Sweep

B. Constructing Drift Scenarios

The evaluation of our framework is conducted across multiple scenarios, each simulating a different drift pattern over a sequence of time windows. In order to construct different scenarios, we illustrate the impact of introducing different attacks in the CICIoMT2024 dataset. For this purpose, we train an XGBoost ,the best model in ML based intrusion detection in the IoMT Networks [6], on six sub-attack categories, ensuring representation for each main attack category

(i.e., MQTT, DoS, DDoS, Recon, and Spoofing) . Then, we introduce the remaining subcategories (Table I) of a particular attack at time window t_n and test the model. Time windows are defined based on the staged introduction of new attack subcategories, rather than fixed time or size intervals. In order to understand to what extend the main attack categories create performance drops, we created a non-stationary model that is initially trained with some sample attack variants and then, at each time interval, all remaining attack variants of a specific category are introduced. The initial model is not updated in the subsequent intervals. The performance of this model is shown in Fig 3. DDoS and Recon exhibit a significant performance drop; therefore, we consider them highly impactful when constructing drift scenarios. We designed three experimental scenarios (Table II) based on a six-class classification task.

- In Scenario 1 (Gradual Drifts), new sub-attack types are progressively introduced across time windows to simulate a gradual concept drift.
- In Scenario 2 (Abrupt Drifts), large and sudden drifts are simulated by introducing multiple sub-attack categories simultaneously, creating an abrupt change in the data distribution.
- In Scenario 3 (Mixed Drifts), a combination of gradual and hard drifts is introduced, involving varying intensities and frequencies of new attack types over time. Each scenario is crafted to assess how well static and dynamic models adapt to evolving threats by continuously monitoring performance across successive time windows.

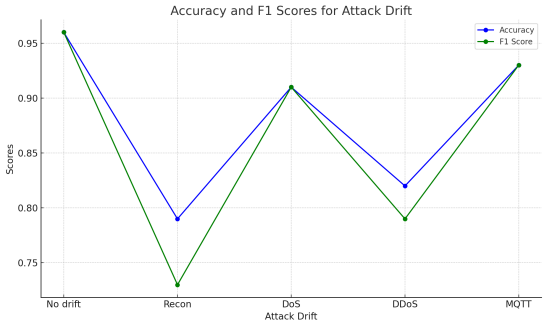


Figure 3: Analysing drift happened in accuracy and f1 score of XGBoost Model due to introduction of different attack sub categories. See Table (Table I)

C. Feature Selection

Irrelevant features for classification problems are reduced to decrease the running time and improve the classification accuracy of the dynamic models ML algorithms in handling concept drift. Feature selection methods are divided into three categories: wrapper, filter, and embedded techniques [19]. Wrapper methods iteratively evaluate subsets of features using a ML algorithm, but they can be computationally intensive for high-dimensional data. In contrast, filter methods rank features independently of the learning algorithm, which may result in suboptimal selections due to the lack of guidance. To reduce computational complexity, we opted for filter-based methods, which are highly efficient and well-suited for resource-constrained IoMT environments. As shown in our previous study [6], Mutual Information was the most effective filter method and is therefore used in our non-stationary experiments. Throughout the scenarios, feature subsets of varying sizes (3, 5, 10, and 45) are used to assess the impact of feature selection under drift condition, and allow us to assess how much feature reduction is tolerable in dynamic settings.

1) Mutual Information: Mutual Information (MI) measures the degree of dependency between two variables [20]. For continuous variables, MI is defined as:

$$I(X,Y) = \int\int p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \tag{1}$$

MI is given by the following equation for discrete variables:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{2}$$

Here, $p(x,y)$ is the joint probability, and $p(x)$, $p(y)$ are the marginal probabilities.

D. Training and Testing

To evaluate the effectiveness of the dynamic non-stationary model in handling various types of concept drift, we train and test two types of models (Static and Dynamic) on a six-class classification task using XGBoost with key hyperparameters including random search over 10 iterations with 5-fold cross-validation, and model performance was evaluated based on accuracy, as it closely aligns with F1-score, precision, and recall in our balanced test setting.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Here, TP and TN denote the number of true positives and true negatives, respectively, while FP and FN denote the number of false positives and false negatives.

In each scenario (Table II), both the static and dynamic models are initially trained on representative sub-attacks at t_0 from each major attack category (Table I), establishing a stable baseline. As the timeline progresses, new sub-attack types are introduced into the testing datasets at t_i to simulate different drift intensities, including gradual, abrupt, and mixed drifts . The dynamic model is retrained at each time window using the previously tested dataset t_{i-1} as the new training set, and evaluated on the current dataset t_i, while the static model is evaluated on the drifted datasets $(t_0, \ldots, t_n)$ without any retraining. The training and testing sets consist of 900 randomly sampled instances per class using stratified sampling. This setup ensures a consistent comparison between the two models' ability to adapt to evolving attacks. Throughout the scenarios, accuracy is monitored across time windows, and feature subsets of varying sizes (3, 5, 10, and 45) are used to assess the impact of feature selection under drift conditions.

Table II: Training and Testing sub-attack sets in various scenarios. "↓" shows the first appearance of a sub-attack category at time window t_i.

Attack Sub-categories	**Scenario 1**						**Scenario 2**					**Scenario 3**					
	t_0	t_1	t_2	t_3	t_4	t_5	t_0	t_1	t_2	t_3	t_4	t_0	t_1	t_2	t_3	t_4	t_5
MQTT-Malformed	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DoS-TCP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DDoS-SYN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Recon-Ping sweep	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Spoofing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Benign	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MQTT-DoS Connect		✓↓	✓	✓	✓	✓		✓↓	✓	✓	✓			✓↓	✓	✓	✓
MQTT-DoS Publish		✓↓	✓	✓	✓	✓		✓↓	✓	✓	✓			✓↓	✓	✓	✓
MQTT-DDoS Publish		✓↓	✓	✓	✓	✓		✓↓	✓	✓	✓					✓↓	✓
MQTT-DDoS Connect								✓↓	✓	✓	✓					✓↓	✓
DoS-ICMP			✓↓	✓	✓	✓				✓↓	✓			✓↓	✓	✓	✓
DoS-SYN			✓↓	✓	✓	✓				✓↓	✓			✓↓	✓	✓	✓
DoS-UDP			✓↓	✓	✓	✓				✓↓	✓			✓↓	✓	✓	✓
DDoS-ICMP				✓↓	✓	✓			✓↓	✓	✓			✓↓	✓	✓	✓
DDoS-UDP				✓↓	✓	✓			✓↓	✓	✓			✓↓	✓	✓	✓
DDoS-TCP				✓↓	✓	✓			✓↓	✓	✓			✓↓	✓	✓	✓
Recon-VulScan					✓↓	✓			✓↓	✓	✓					✓↓	✓
Recon-OS Scan					✓↓	✓			✓↓	✓	✓					✓↓	✓
Recon-Port Scan					✓↓	✓			✓↓	✓	✓		✓↓	✓	✓	✓	✓

IV. Results

In this study, we aim to evaluate the feature impact of a non-stationary model on the detection of attacks in the IoMT networks. We established an evolution timeline and created various drift scenarios using benign samples and attack variants from the CICIoMT2024 dataset. We assessed both static and dynamic models based on their performance, with a particular focus on accuracy. Our emphasis was on multiclass classification to categorize traffic into specific attack categories.

Figures (4, 5 and 6) show the performance of dynamic model (solid lines) versus traditional static models (dotted lines) for 6-class classification, evaluated across multiple time windows (t_0–t_5). At t_0, both training and testing sets contain representative attacks from every major attack category, and both models perform similarly well with high accuracy, indicating a stable environment.

A. Dynamic Vs Static Model Performance

a) ***Scenario 1: Gradual Drift.****:* In the Scenario 1 shown in Table II (Fig 4), we illustrate the concept of **gradual drift** through a 6-class classification task using XGBoost. At each time window t_i, new sub-attack types are introduced to simulate drift, except t_5, which represents a phase with no drift. Initially, at t_0, the training and testing sets are identical, each containing one representative sub-category from every major attack type. As time progresses, three new attack subcategories from the following categories are gradually introduced: *MQTT* at t_1, *DoS* at t_2, *DDoS* at t_3, and *reconnaissance* at t_4. This setup enables a comprehensive evaluation of model performance under evolving threat conditions.

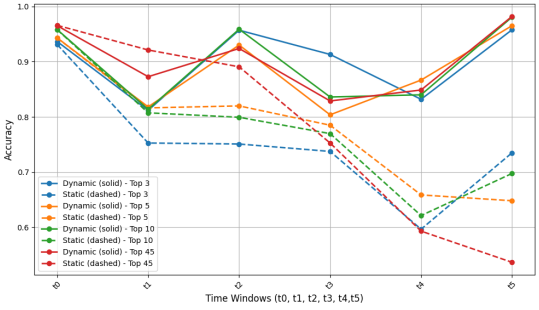


Figure 4: Gradual Drifts: Accuracy score dynamic model (solid lines) vs static models (dotted lines) for multi-class classification (6-classes) considering six-time windows having a steady drift of attacks in each interval.

The dynamic model (Fig 4), which is retrained at every step using the most recent dataset, consistently outperforms the static model, trained only at t_0. The static model performance with 45 features does not degrade significantly in t_1 and t_2, where the drifts are relatively minor. However, in t_3, the performance drops considerably—falling even below that of other feature sets. This suggests that the 45-feature configuration is more resilient under mild drift conditions, but its

Table III: Static vs Dynamic accuracy scores report for different top k features from Mutual Information feature selection for the CICIoMT2024 dataset, using the XGBoost model for 6-class classification.

Scenario	Top-k features	Static							Dynamic						
		t0	t1	t2	t3	t4	t5	Average	t0	t1	t2	t3	t4	t5	Average
Scenario 1 (Steady)	3	0.931	0.751	0.737	0.597	0.597	0.735	0.71	0.947	0.957	0.913	0.832	0.832	0.966	**0.878**
	5	0.942	0.82	0.785	0.659	0.659	0.648	0.77	0.955	0.93	0.804	0.867	0.867	0.975	0.855
	10	0.957	0.799	0.77	0.621	0.621	0.697	0.749	0.963	0.959	0.836	0.84	0.84	0.981	0.862
	45	0.965	0.891	0.753	0.593	0.593	0.536	**0.79**	0.968	0.924	0.829	0.849	0.849	0.985	0.868
Scenario 2 (Hard)	3	0.935	0.785	0.686	0.677	0.676	—	0.752	0.921	0.813	0.847	0.96	0.954	—	**0.90**
	5	0.935	0.785	0.616	0.611	0.61	—	0.711	0.946	0.812	0.852	0.865	0.959	—	0.887
	10	0.949	0.807	0.669	0.667	0.661	—	0.751	0.947	0.791	0.824	0.962	0.963	—	0.898
	45	0.955	0.946	0.689	0.648	0.649	—	**0.778**	0.959	0.96	0.668	0.961	0.966	—	**0.90**
Scenario 3 (Mixed)	3	0.93	0.838	0.735	0.711	0.706	0.601	**0.753**	0.927	0.803	0.736	0.952	0.957	0.965	0.89
	5	0.936	0.809	0.676	0.67	0.674	0.71	0.746	0.94	0.775	0.817	0.957	0.957	0.966	0.902
	10	0.947	0.829	0.694	0.67	0.646	0.734	**0.753**	0.956	0.788	0.83	0.964	0.964	0.985	**0.915**
	45	0.959	0.802	0.566	0.58	0.601	0.619	0.688	0.966	0.796	0.759	0.939	0.966	0.986	0.902

performance deteriorates rapidly when exposed to substantial drift, as seen in t_3 and t_4. On the other hand, as expected, the dynamic model maintains relatively high accuracy across time, demonstrating its ability to adapt to evolving threats. Furthermore, as shown in Table III, the average accuracy achieved using only 3 features is higher than that of larger feature sets, suggesting that a significantly reduced feature subset can yield comparable or even superior performance.

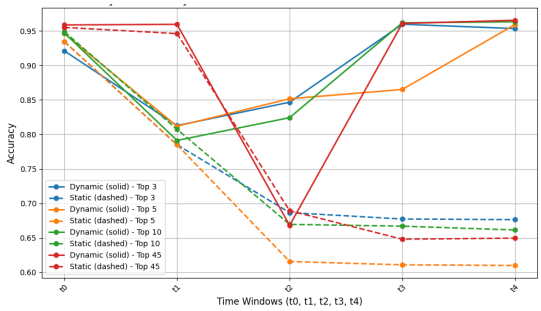


Figure 5: Scenario 2: Abrupt drift: Accuracy score of the dynamic model (solid lines) vs static models (dotted lines) for 6-class classification considering five-time windows

*b) **Scenario 2: Abrupt Drift**:* Fig 5 illustrates how the dynamic model effectively handles big or hard concept drifts that static models struggle with. At t_1, we introduced MQTT attack subcategories, resulting in a moderate performance drop in both models; however, the dynamic model consistently outperforms the static one across all feature subsets, showing its adaptability. A hard drift occurs at t_2 due to the simultaneous introduction of DDoS and Recon attack subcategories, leading to a significant drop in accuracy for all models. As shown in III, the dynamic model performance at t_1, for the 3,5, and 10 feature decreased more than that of the 45-feature configuration. However, at t_3, while the 45-feature setup experienced a significant performance drop, the performance with 3, 5, and 10 features actually showed an improvement during this dip. These results indicate that the performance of low-dimensional alternatives, when compared to the 45-feature alternative, is impacted more by mild drifts but is more resistant to hard drifts. Overall the average accuracy using 3 features and 45 features is similar, which suggests that a significantly reduced feature set can achieve similar performance.

*c) **Scenario 3: Mixed Drift**:* Fig 6 illustrates how the dynamic model effectively handles mixed concept drifts that static models often struggle with. At t_1, we introduced the Recon-Port Scan attack into the test set without modifying the training set. As a result, both dynamic and static models experience a comparable drop in accuracy, since neither had prior exposure to this specific drift. A more substantial drift is introduced at t_2, where two subcategories from each of MQTT, DoS, and DDoS are added (i.e., MQTT-DoS Connect, MQTT-DoS Publish, DoS-ICMP, DoS-SYN, DDoS-ICMP, and DDoS-UDP). This leads to a sharp drop in accuracy, especially for the static model, while the dynamic model, though impacted, shows more stability and manages a quicker recovery. At t_3, the addition of DoS-UDP and DDoS-TCP allows the dynamic model to recover significantly and regain high accuracy. After t_2, the performance of the dynamic models consistently improves. Up to t_2, the system is exposed to new classes such as MQTT, reconnaissance, DoS, and DDoS. Although additional sub-categories are introduced after t_2, the model appears to leverage the knowledge gained from previously seen sub-classes to effectively classify the new ones. Furthermore, unlike Scenario 1 and 2 , static model gives best performance with 3 features set.

These results highlight that dynamic model is significantly more robust and reliable, consistently outperforming static models by effectively adapting to evolving attack patterns, as well as steady and hard concept drifts commonly encountered in real-world environments.

B. Feature importance analysis

Table III illustrates the accuracy scores achieved across the entire timeline (t_0–t_5) for both static and dynamic models

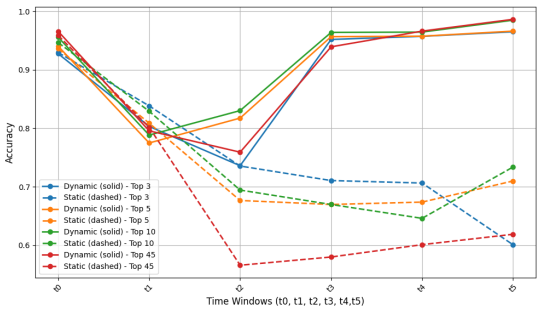


Figure 6: Mixed drifts: Accuracy scores of dynamic model (solid lines) versus static models (dotted lines) for 6-class classification, evaluated across six-time windows.

using top 3, 5, 10, and 45 feature sets. The table highlights (in **bold**) the highest average accuracy achieved across the windows. We observe that, in most scenarios, high performance is achieved with very few features (i.e., 3 or 5) in the case of dynamic models, in contrast to static models, where the highest performance is typically achieved with 10 or more features, often with 45 features. For this reason, we analyze the importance of the top 3 and top 5 features in dynamic models for handling concept drift in each scenario in the following paragraphs.

*a) **Top 3 Feature Importance Analysis Across Scenarios.**:* Fig 7 depicts the impact of the top 3 features in dynamic models in handling concept drifts. Since retraining is performed at t_i using the updated training set from t_{i-1}, the relevance of a feature corresponds to its ability to distinguish attacks introduced at t_{i-1}. For instance, MQTT variants in Scenario 1 are introduced in t1, creating a huge drop in that time interval as shown in Fig 7. The feature changes introduced by these variants are demonstrated at t2 in the figure so that `Tot size` is replaced by `Min` in this case. It is important to consider this while correlating the performance result with the changes in feature in Figures 7, 8. At t_1, we have introduced attack variants in the test set (and not in training). Therefore, the top features remain similar to those at t_0, showing no major shift.

Across all scenarios, the feature `IAT` exists in each time window in every scenario. `Tot_size` is also more prevalent across the scenarios, suggesting their crucial role in handling concept drifts. `Header_Length`, appeared in first three windows in every scenario. Other features (`Magnitude`, `Min`, `Max`, or `Tot sum`) vary depending on the drift type: dynamically emerge based on newly introduced attacks. In Scenario 1 (Gradual Drift), feature relevance shifts smoothly, reflecting the model's stable adaptation. Scenario 2 (Hard Drift) shows a rapid pivot towards `Tot sum` following the sudden introduction of DDoS and Recon attacks, demonstrating quick retraining capabilities. In Scenario 3 (Mixed Drift), the model flexibly alternates among multiple features, evidencing robustness against both gradual and abrupt changes. Overall, the combination of stability in core features and flexibility in auxiliary features enables effective drift handling.

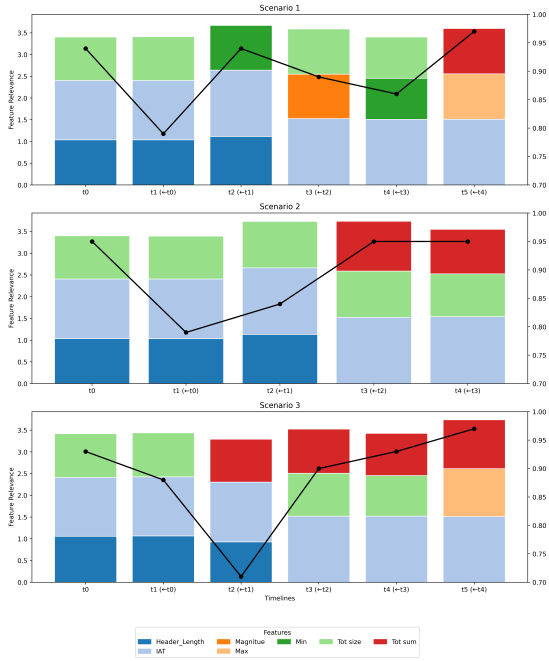


Figure 7: Top 3 Features impact in each scenario in XGboost dynamic model while considering mutual information feature selection method.

*b) **Top 5 Feature Importance Analysis Across Scenarios.**:* Fig 8 presents the top-5 most relevant features (by information gain) at each timeline t_n for the dynamic model in all three scenarios. `IAT` and `Tot size` is common across all time lines and scenarios. The feature `Header_Length` disappeared at t_3 and onward in Scenario 1, at t_4 and onward in Scenario 2, and again at t_3 and onward in Scenario 3. This suggests that the feature is likely irrelevant to the DDoS and reconnaissance sub-categories introduced in the later stages of the timelines.

In Scenario 1 (Gradual Drift), where new attacks are introduced progressively, the feature set evolves accordingly. Early time windows (t_0, t_1) highlight features like `IAT`, `Magnitude`, and `AVG`, which are effective in distinguishing foundational attack types (baselines). At t_2, with the introduction of MQTT attacks in the previous window, we observe increased importance of features like `Min` while disappearance of `Magnitude` . At t_3, we notice the re-appearance of `Magnitude`, which may be relevant to identifying DoS-type attacks. `Header_Length` disappeared at t_3 and onward. At t_4, we see the addition of `AVG`, which suggests its relevance to the DDoS attack introduced at t_3. By t_5, `Srate` and `Tot`

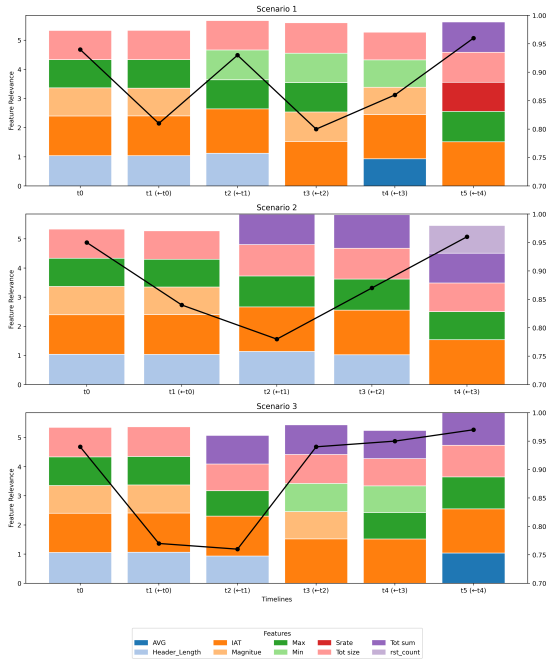


Figure 8: Top 5 Features impact in each scenario in XGboost dynamic model while considering mutual information feature selection method.

`sum` become relevant, possibly due to their effectiveness in identifying reconnaissance traffic introduced at t_4. The relatively stable and rotational presence of features demonstrates the manageable nature of gradual drift.

In Scenario 2 (Hard Drift), where a severe drift occurs at t_2 due to the simultaneous introduction of DDoS and Recon attacks, the feature landscape shifts sharply. At t_2 and t_3, features like `Tot sum` can be linked to DDoS and Recon attacks. `rst_count` at t_4 might be relevant to DoS subcategories added in t_3. This scenario shows rapid feature fluctuation, reflecting the complex nature of hard drifts and the model's adaptability through retraining. The reappearance of features like `IAT` and `Header_Length` shows that temporal and structural features remain crucial. Furthermore, `Header_Length` disappeared at t_4.

In Scenario 3 (Mixed Drift), which blends aspects of both gradual and abrupt drift, feature importance varies more dynamically. At t_2, where Recon-Port Scan was introduced at t_1, features like `Tot sum` become dominant—indicating the model's focus on packet volume. By t_3, `Min` and `Magnitude` become highly relevant, reflecting the prior introduction of DoS, DDoS, and MQTT variants. At t_4, we see the reappearance of `Max`, which seems relevant to DoS-UDP or Recon-VulScan. By t_5, we notice the appearance of `AVG`, which may be linked to the addition of MQTT and Recon variants. The feature `Header_Length` disappeared at t_3 and onward again in this scenario.

C. Comparison of detection overhead

In Table IV, we compare the detection overhead between static and dynamic models in Scenario 3. The table reports average latency (in μs per sample), false alarm rate (FAR, the percentage of benign samples misclassified as attacks), total training time, inference time, and training memory usage. While static models achieve lower latency and FAR with minimal training time and memory usage, dynamic models incur significantly higher training costs to maintain better robustness, particularly as the number of selected features increases. This underscores the trade-off between detection performance and resource overhead in evolving data environments.

Table IV: Comparison of average latency, FAR , training time, inference time, and memory usage in both static and dynamic models in mixed drifts (Scenario 3)

	Top-k Features	Avg. Latency (μs)	Avg. FAR (%)	Total Training Time (s)	Total Inference Time (s)	Total Training Memory (MB)
Static	3	1.19	10	120	0.037	98
	5	0.89	10	134	0.027	25
	10	1.75	9	115	0.054	65
	45	1.99	9	116	0.062	83
Dynamic	3	1.17	7	593.37	0.035	458
	5	1.35	7	606.5	0.041	453
	10	1.44	8	610.2	0.043	317
	45	2.16	8	633.2	0.067	433

V. Discussions

In this study, we observed that selecting just three features for dynamic models provides the best long-term performance in Scenario 1 and achieves similar performance to the 45-feature alternative in Scenario 2. 10-feature alternative is the best in Scenario 3 (see Table III). As illustrated in Fig 7, the dynamic model consistently preserves two out of the three top features across consecutive time intervals during nearly all drift scenarios, contributing to its overall high accuracy in two scenarios. This finding mostly aligns with a study, where the same mutual information-based feature selection method achieved performance close to 0.99 using only three features on the same dataset [6]. These results demonstrate the effectiveness of informative features in stationary and non-stationary models.

Another significant finding is that models trained with 45 features demonstrate smaller performance drops during lighter drifts compared to models with a limited number of features in both static and dynamic settings. However, under hard drifts, these models suffer severe performance degradation, impacting their long-term performance in two scenarios. In contrast, models that use fewer but more stable features yield larger drops during lighter drifts, but can preserve their

performance during hard drifts. The ability to preserve a small set of consistently important features over consecutive time intervals becomes a long-term advantage to maintain detection performance.

It is important to note that, despite the suitability of CICIoMTDataset 2024 for our purpose, this dataset still has varied subcategories for DoS, DDoS, and Recon, which share common characteristics of high traffic volume, repetitive connection patterns, and protocol-level anomalies. These similarities may simplify the maintenance of long-term detection performance compared to scenarios involving more heterogeneous attack categories.

In this study, we focus on multiclass classification, as it provides more detailed and actionable information for cyber incident investigators compared to binary classification tasks. Identifying the specific attack category can significantly reduce the decision-making time when determining an appropriate course of action, which is the ultimate aim of deploying IDS. Although a variety of attack datasets are available [21], [22], we selected the CICIoMT 2024 dataset as it offers comprehensive coverage across multiple attack categories and sub-categories, making it particularly suitable for our analysis. Nevertheless, this study can be further extended to include binary classification tasks to examine the impact of feature selection on model performance under the corresponding non-stationary conditions.

While formulating the non-stationary model, we assume that the initial training phase includes at least one attack sub-category from each major attack category (e.g., using DoS-TCP to represent the DoS category). We consider this assumption feasible, as major network attack categories are well-established, and obtaining representative samples for initial model development is relatively straightforward. However, during the model's operational lifecycle, it may encounter previously unseen variations of known attack categories (e.g., DoS-SYN, DoS-ICMP). The model should demonstrate adaptability to these variations through its non-stationary capability. Additionally, future work could explore alternative non-stationary scenarios, such as the introduction of entirely new attack categories at later stages, to further evaluate the model's ability to adapt to an evolving network threat landscape.

Another key aspect of our non-stationary model formulation is the introduction of new concepts (i.e., in the form of new subcategories) at each upcoming time interval, while preserving the previously existing subcategories. In the concept drift literature, various types of drift have been identified, such as sudden, gradual, and incremental drift [3]. These drift types typically assume that older concepts either disappear immediately or fade over time. In malware detection, a domain where concept drift has been extensively studied [23], [24], it is common that certain malware families evolve into new variants while older families may eventually vanish. However, we argue that in network intrusion detection, threat evolution tends to be less dynamic, making it impractical to assume that certain attack categories would completely disappear over time. In this setting, most types of network attacks given in datasets are still encountered in real-world scenarios, and thus, maintaining all existing subcategories remains a realistic assumption.

Feature selection in ML-based intrusion detection has been extensively studied in the literature, particularly in the context of IoT [4]–[6]. In this study, we aim to investigate the impact of feature selection and the feature evolvement within a non-stationary setting, an area that has received limited attention in the literature. It is important to note that our objective is not to propose a novel concept drift handling method. Instead, we examine two commonly used modelling approaches (i.e., static and dynamic) to represent non-stationary models and analyze feature space dynamics under three different drift scenarios. Our findings can guide further research in developing lightweight intrusion detection models that maintain long term detection performance by incorporating advanced drift detection mechanisms.

Our dynamic model benefits from the whole data prior to the corresponding time interval, assuming that all training data with proper labels and necessary computational resources are readily available. However, in cybersecurity, labeling large volumes of data instances is often expensive and time-consuming. Consequently, learning from a minimally labeled dataset becomes a critical requirement, particularly in real-time deployment settings. It is shown that active learning can be applied in IoT botnet detection to minimize the size of labeled data while providing higher detection performances [25]. Integrating this learning paradigm with feature selection techniques offers a promising direction for non-stationary models. While active learning optimizes training costs by minimizing labeling efforts, feature selection reduces both training and inference costs by focusing on the most informative features.

VI. Conclusion and Future Work

In this work, we performed filter-based feature selection methods (Mutual Information) to identify the best features in nonstationary models in the CICIoMT2024 dataset. We compared the performance of static models with the dynamic models by utilizing a machine learning algorithm (XGBoost) for the said dataset. We evaluate the performance for multi-class classifications (6-classes).

We presented a comprehensive framework to evaluate the robustness of machine learning-based intrusion detection systems (IDS) under concept drift in IoT networks. Using the CICIoMT2024 dataset, which contains diverse attack types across 40 devices, the framework includes, construction of drift scenarios (gradual, hard, and mixed), filter-based feature selection (Mutual Information), and training/testing of static and dynamic XGBoost models. Static models are trained once and tested over time, while dynamic models are updated at each step. The framework assesses model adaptability using varying feature set sizes across evolving attack conditions.

A key finding is that dynamic models maintain strong detection performance even with highly reduced feature sets (e.g., top 3 or 5 features), offering significant efficiency gains for

resource-constrained IoT environments. Our paper highlights key features for multiple attack detection over time . It revealed valuable insights into feature behavior under evolving threats. Certain features, such as `IAT` and `Tot size`, consistently appeared across all time windows and drift scenarios, indicating their foundational role in effective detection. Meanwhile, other features like `Header_Length`, `Magnitude`, and `Tot sum` exhibited scenario-specific relevance, adapting to the nature of the introduced attacks.

Another key finding is that low-dimensional models still give reasonable performance in hard drift scenarios when compared to the high dimensional (45 features) alternative. However, the latter one maintains higher performance in mild drift scenarios.

Future work will explore combining Mutual Information with wrapper-based methods like Recursive Feature Elimination (RFE) to improve feature relevance under concept drift. Developing a benchmarking framework for evaluating nonstationary models that can dynamically adapt to new features, previously unseen attacks, and limited labeling scenarios would further improve the robustness of intrusion detection systems in IoT networks. Future work should consider comparisons of the current XGBoost model with online and semi-supervised learning models that are better suited for real-time non-stationary environments. Furthermore, extending this work to incorporate more diverse and complex drift-based attack datasets could broaden the applicability of the findings. Future research may also focus on integrating active learning with dynamic feature selection to reduce labeling effort while maintaining adaptability to emerging threats.

ACKNOWLEDGEMENT

This study was co-funded by the European Union and Estonian Research Council via project TEM-TA5.

REFERENCES

[1] M. M. Lopez, S. Shao, S. Hariri, and S. Salehi, "Machine learning for intrusion detection: Stream classification guided by clustering for sustainable security in iot," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, 2023, pp. 691–696.

[2] R. Kalakoti, H. Bahsi, and S. Nõmm, "Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection," *IEEE Internet of Things Journal*, vol. 11, no. 10, pp. 18 237–18 254, 2024.

[3] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[4] R. Kalakoti, S. Nõmm, and H. Bahsi, "In-depth feature selection for the statistical machine learning-based botnet detection in iot networks," *IEEE Access*, vol. 10, pp. 94 518–94 535, 2022.

[5] H. Bahşi, S. Nõmm, and F. B. La Torre, "Dimensionality reduction for machine learning based iot botnet detection," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 1857–1862.

[6] M. U. Rehman, R. Kalakoti, and H. Bahşi, "Comprehensive feature selection for machine learning-based intrusion detection in healthcare iomt networks," in *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 2: ICISSP*, INSTICC. SciTePress, 2025, pp. 248–259.

[7] O. A. Wahab, "Intrusion detection in the iot under data and concept drifts: Online deep learning approach," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19 706–19 716, 2022.

[8] S. Dadkhah, E. C. P. Neto, R. Ferreira, R. C. Molokwu, S. Sadeghi, and A. A. Ghorbani, "Ciciomt2024: A benchmark dataset for multi-protocol security assessment in iomt," *Internet of Things*, vol. 28, p. 101351, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660524002920

[9] R. Kalakoti, S. Nõmm, and H. Bahsi, "Explainable transformer-based intrusion detection in internet of medical things (iomt) networks," in *2024 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2024, pp. 1164–1169.

[10] G. Balhareth and M. Ilyas, "Optimized intrusion detection for iomt networks with tree-based machine learning and filter-based feature selection," *Sensors*, vol. 24, no. 17, p. 5712, 2024.

[11] "Ids 2017 — datasets — research — canadian institute for cybersecurity — unb," [Online; accessed 2025-04-30]. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[12] K. Malik, F. Rehman, T. Maqsood, S. Mustafa, O. Khalid, and A. Akhunzada, "Lightweight internet of things botnet detection using one-class classification," *Sensors*, vol. 22, no. 10, p. 3646, 2022.

[13] R. Kalakoti, S. Nõmm, and H. Bahsi, "Enhancing iot botnet attack detection in socs with an explainable active learning framework," in *2024 IEEE World AI IoT Congress (AIIoT)*. IEEE, 2024, pp. 265–272.

[14] E. Alsuwat, S. Solaiman, and H. Alsuwat, "Concept drift analysis and malware attack detection system using secure adaptive windowing." *Computers, Materials & Continua*, vol. 75, no. 2, 2023.

[15] A. Adnan, A. Muhammed, A. A. Abd Ghani, A. Abdullah, and F. Hakim, "Hyper-heuristic framework for sequential semi-supervised classification based on core clustering," *Symmetry*, vol. 12, no. 8, p. 1292, 2020.

[16] M. Soltani, K. Khajavi, M. Jafari Siavoshani, and A. H. Jahangir, "A multi-agent adaptive deep learning framework for online intrusion detection," *Cybersecurity*, vol. 7, no. 1, p. 9, 2024.

[17] L. Yang and A. Shami, "A multi-stage automated online network data stream analytics framework for iiot systems," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 2107–2116, 2022.

[18] G. Werner, S. J. Yang, and K. McConky, "Near real-time intrusion alert aggregation using concept-based learning," in *Proceedings of the 18th ACM International Conference on Computing Frontiers*, 2021, pp. 152–160.

[19] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 1200–1205.

[20] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on neural networks*, vol. 20, no. 2, pp. 189–201, 2009.

[21] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[22] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "Medbiot: Generation of an iot botnet dataset in a medium-sized iot network." in *ICISSP*, 2020, pp. 207–218.

[23] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting concept drift in malware classification models," in *26th USENIX security symposium (USENIX security 17)*, 2017, pp. 625–642.

[24] A. Guerra-Manzanares, M. Luckner, and H. Bahsi, "Android malware concept drift using system calls: detection, characterization and challenges," *Expert Systems with Applications*, vol. 206, p. 117200, 2022.

[25] A. Guerra-Manzanares and H. Bahsi, "On the application of active learning for efficient and effective iot botnet detection," *Future Generation Computer Systems*, vol. 141, pp. 40–53, 2023.

Curriculum Vitae

1. Personal data

Name	Rajesh Kalakoti
Date and place of birth	16 July 1992 Guntur, India
Nationality	Indian

2. Contact information

Address	Tallinn University of Technology, School of Information Technologies, Department of Software Science, Ehitajate tee 5, 19086 Tallinn, Estonia
E-mail	rajesh.kalakoti@taltech.ee

3. Education

2021–...	Tallinn University of Technology, School of Information Technologies, Course name, PhD studies
2018–2021	Tallinn University of Technology & University of Tartu, Faculty of Information Technologies, Cyber Security, MSc
2013–2015	Velagapudi Ramakrishna Siddhartha Engineering College, Department of Computer Science, Computer Science, M.Tech
2009–2013	Vasireddy Venkatadri Institute of Technology, Department of Computer Science, Computer Science, B.Tech

4. Language competence

Telugu	Native
English	Fluent
Estonian	Basic

5. Professional Employment

2016, Jan–2017, April	Mic College of technology, Lecturer
2017, May–2018, August	Excellar Info Services LLP, Software Engineer

6. Honours and awards

- 2024, Author of Research article, Taltech.

7. Defended theses

- 2015, Avoiding Attacks Using Node Position Verification in Mobile Ad Hoc Networks, Velagapudi Ramakrishna Siddhartha Engineering College, India
- 2013, RESEARCH AND DESIGN OF CHATING ROOM SYSTEM BASED ON ANDROID BLUETOOTH, Vasireddy Venkatadri Institute of Technology, India.

8. Scientific work
Papers

1. R. Kalakoti, S. Nõmm, and H. Bahsi. In-depth feature selection for the statistical machine learning-based botnet detection in iot networks. *IEEE Access*, 10:94518–94535, 2022

2. M. U. Rehman, R. Kalakoti, and H. Bahşi. Comprehensive feature selection for machine learning-based intrusion detection in healthcare iomt networks. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 2: ICISSP*, pages 248–259. INSTICC, SciTePress, 2025

3. R. Kalakoti, H. Bahsi, and S. Nõmm. Improving iot security with explainable ai: Quantitative evaluation of explainability for iot botnet detection. *IEEE Internet of Things Journal*, 11(10):18237–18254, 2024

4. R. Kalakoti, S. Nõmm, and H. Bahsi. Improving transparency and explainability of deep learning based iot botnet detection using explainable artificial intelligence (xai). In *2023 International Conference on Machine Learning and Applications (ICMLA)*, pages 595–601. IEEE, 2023

5. R. Kalakoti, R. Vaarandi, H. Bahşi, and S. Nõmm. Evaluating explainable ai for deep learning-based network intrusion detection system alert classification. In *Proceedings of the 11th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, pages 47–58. INSTICC, SciTePress, 2025

6. R. Kalakoti, S. Nõmm, and H. Bahsi. Explainable transformer-based intrusion detection in internet of medical things (iomt) networks. In *2024 International Conference on Machine Learning and Applications (ICMLA)*, pages 1164–1169. IEEE, 2024

7. R. Kalakoti, S. Nõmm, and H. Bahsi. Enhancing iot botnet attack detection in socs with an explainable active learning framework. In *2024 IEEE World AI IoT Congress (AIIoT)*, pages 265–272. IEEE, 2024

8. R. Kalakoti, H. Bahsi, and S. Nõmm. Explainable federated learning for botnet detection in iot networks. In *2024 IEEE International Conference on Cyber Security and Resilience (CSR)*, pages 01–08. IEEE, 2024

9. R. Kalakoti, S. Nõmm, and H. Bahsi. Federated learning of explainable ai (fedxai) for deep learning-based intrusion detection in iot networks. *Computer Networks*, page 111479, 2025

10. R. Kalakoti, H. Bahsi, and S. Nõmm. Synthetic data-driven explainability for federated learning-based intrusion detection system. In *IEEE Internet of Things Journal*. IEEE, 2025

11. M. U. Rehman, R. Kalakoti, and H. Bahşi. Exploring the impact of feature selection on non-stationary intrusion detection models in iot networks. In *Accepted for 22nd Annual International Conference on Privacy, Security, and Trust (PST2025)*, 2025

Elulookirjeldus

1. Isikuandmed

Nimi	Rajesh Kalakoti
Sünniaeg ja -koht	16.07.1992, Guntur, India
Kodakondsus	India

2. Kontaktandmed

Aadress	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Ehitajate tee 5, 19086 Tallinn, Estonia
E-post	rajesh.kalakoti@taltech.ee

3. Haridus

2021–…	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, arvutiteadus, doktoriõpe
2013–2015	Velagapudi Ramakrishna Siddhartha Engineering College, arvutiteaduse osakond, MSc
2009–2013	Vasireddy Venkatadri Institute of Technology, Department of Computer Science, arvutiteaduse osakond, B.Tech

4. Keelteoskus

telugu keel	emakeel
inglise keel	kõrgtase
eesti keel	algtase

5. Teenistuskäik

2016, Jan–2017, April	Mic College of technology, lektor
2017, May–2018, August	Excellar Info Services LLP, Tarkvarainsener

ISSN 2585-6901 (PDF)
ISBN 978-9916-80-402-5 (PDF)