TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Johan Valdemar Leoste 232665IVCM

Comparative Analysis of Deep Learning and Machine Learning for Network Intrusion Detection Using Data from the Largest Live-Fire Cyber Defence Exercise

Master's thesis

Supervisor: Risto Vaarandi, PhD
Co-Supervisor: Allard Dijk, MSc

Tallinn 2025

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Johan Valdemar Leoste 232665IVCM

Süvaõppe ja masinõppe võrdlev analüüs võrgu sissetungide tuvastamiseks kasutades andmeid maailma suurimalt küberkaitseõppuselt

magistritöö

Juhendaja: Risto Vaarandi, PhD
Kaasjuhendaja: Allard Dijk, MSc

Tallinn 2025

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Johan Valdemar Leoste

18.05.2025

Abstract

Intrusion detection systems face ongoing challenges in accurately identifying malicious activities, particularly due to the increasing complexity of cyber threats. Despite widespread research on machine learning and deep learning for intrusion detection, real-world applicability of the findings remains limited due to limitations inherent to outdated or synthetic training datasets. This thesis addresses this gap through a comparative analysis of a random forest model and a custom one-dimensional convolutional neural network, trained and evaluated using the large-scale Locked Shields 2023 dataset, derived from NATO CCDCOE's live-fire cyber defence exercises. The results, validated on over 16 million network flows, demonstrate that both models achieve high detection accuracy with F1 scores over 99%, with the neural network marginally outperforming the random forest in classifying malicious traffic, while the latter offers significant advantages in computational efficiency and interpretability. Cross-year validation using data from Locked Shields 2024 highlights a notable decrease in performance for both models, emphasizing the inherent difficulty of maintaining high accuracy amidst continuously evolving threats and network configurations. This thesis delivers a practical workflow and detailed insights into the comparative strengths and limitations of both traditional machine learning and advanced deep learning techniques, equipping cybersecurity researchers with clear guidance for implementing resilient intrusion detection systems. Additionally, the models implemented in this thesis provide a strong baseline for future expansions, including multiclass detection or adaptation to upcoming Locked Shields datasets.

This thesis is written in English and is 112 pages long, including 6 chapters, 17 figures and 11 tables.

Annotatsioon

Süvaõppe ja masinõppe võrdlev analüüs võrgu sissetungide tuvastamiseks kasutades andmeid maailma suurimalt küberkaitseõppuselt

Sissetungituvastuse süsteemid (IDS) seisavad silmitsi üha keerukamate väljakutsetega pahatahtliku võrguliikluse täpsel tuvastamisel küberohtude kasvava komplekssuse tõttu. Kuigi masinõppe ja süvaõppe kasutamist sissetungituvastuses on laialdaselt uuritud, piiravad vananenud või sünteetilised treeningandmestikud tulemuste rakendatavust reaalsetes oludes. Käesolev magistritöö käsitleb nimetatud probleemi, võrreldes otsustusmetsa (RF) algoritmi spetsiaalselt häälestatud ühemõõtmelise konvolutsioonilise tehisnärvivõrguga (1-D CNN). Mõlemad masinõppe mudelid on treenitud ja hinnatud NATO CCDCOE küberkaitseõppuse Locked Shields 2023 mahukal ja realistlikul andmestikul. Rohkem kui 16 miljoni võrguliikluse voo põhjal valideeritud tulemused näitavad, et mõlemad mudelid saavutavad kõrge klassifitseerimise täpsuse, saavutades ~99% F1-skoori. Tehisnärvivõrk ületas otsustusmetsa napilt tuvastustäpsuga, seevastu otsustusmets pakkus olulisi eeliseid arvutusliku efektiivsuse ja tulemuste tõlgendatavuse seisukohalt. Samas järgneva aasta Locked Shields 2024 andmetega ristvalideerimine demonstreeris märkimisväärset mudelite jõudluse langust, mis viitab raskustele mudelite üldistusvõimes ajas muutuvate võrgustruktuuride ja arenevate küberohtude korral. Magistritöö tulemusteks on praktiliselt rakendatav töövoog ning detailne ülevaade nii traditsiooniliste masinõppe meetodite kui ka süvaõppetehnikate tugevustest ja piirangutest, pakkudes seeläbi küberkaitse ekspertidele selgeid juhiseid töökindlate sissetungituvastuse süsteemide loomiseks. Lisaks moodustavad käesolevas töös valideeritud mudelid tugeva aluse edasistele uuringutele, võimaldades näiteks mitmeklassilise tuvastuse integreerimist või rakendamist järgnevate Locked Shieldsi õppuste andmestikega.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 112 leheküljel, 6 peatükki, 17 joonist, 11 tabelit.

List of abbreviations and terms

AE	Autoencoder
ANN	Artificial Neural Network
APT	Advanced Persistent Threat
CCDCOE	Cooperative Cyber Defence Centre of Excellence
CNN	Convolutional Neural Network
CRISP-DM	Cross-Industry Standard Process for Data Mining
DL	Deep Learning
DT	Decision Tree
EXPO	Central Logging Platform for Locked Shields
Gamenet	Virtualised Exercise Network in Locked Shields
GPU	Graphics Processing Unit
ICS	Industrial Control Systems
IDS	Intrusion Detection System
IoT	Internet of Things
KNN	K-Nearest Neighbours
LS	Locked Shields (Cyber Defence Exercise)
LSPR23	Locked Shields Partners Run 2023 (Dataset)
LSPR24	Locked Shields Partners Run 2024 (Dataset)
LSTM	Long Short-Term Memory
MITRE ATT&CK	MITRE Adversarial Tactics, Techniques & Common Knowledge Framework
ML	Machine Learning
NATO CCDCOE	NATO Cooperative Cyber Defence Centre of Excellence
NB	Naïve Bayes
NIDS	Network-based Intrusion Detection System
PLC	Programmable Logic Controller
RF	Random Forest
RFE	Recursive Feature Elimination
ROC-AUC	Receiver Operating Characteristic – Area Under the Curve
RNN	Recurrent Neural Network
SCADA	Supervisory Control and Data Acquisition
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine

Table of contents

List of figures

List of tables

1 Introduction

Intrusion Detection Systems (IDS) are used for monitoring network traffic and detecting suspicious or malicious activity. As such, IDS are a critical element of an effective cybersecurity strategy in the constantly evolving threat landscape [1]. Despite ongoing improvements in intrusion detection systems, accurately detecting increasingly evolving attacks remains challenging. The primary difficulty is balancing the timely detection of novel threats while minimising false alarms [1], [2]. Existing machine learning-based intrusion detection systems are often limited by the quality and the scale of their training datasets. This restricts the model's effectiveness in recognising patterns in the malicious network data, and thus worsening the detection accuracy [2], [3].

1.1 Background

The escalating frequency and sophistication of cyber threats pose a significant challenge to the security of digital infrastructures and assets. Organisations across various sectors are increasingly vulnerable to malicious activities that can compromise the integrity, confidentiality, and availability of their critical systems and data. This evolving threat landscape compels the deployment of robust security mechanisms capable of detecting and mitigating these risks. [1], [4]

1.1.1 Network intrusion detection systems

Intrusion Detection Systems serve as a critical component of an organisation's defence strategy. While preventative measures such as firewalls aim to block malicious traffic before it enters a network, IDS focus on identifying and alerting security personnel to malicious activities that may have bypassed these initial defences or originated from within the network itself [4]. The primary function of an IDS is to detect suspicious behaviour, providing timely alerts that enable security teams to respond effectively and minimise potential damage. [1], [5]

Network-based Intrusion Detection Systems (NIDS) are deployed at various points within a network infrastructure to monitor network traffic for suspicious activity [5]. These systems operate by capturing and analysing network packets as they traverse the network, looking for patterns that match known attack signatures or behaviours that deviate from established baselines. Key characteristics of NIDS include real-time monitoring capabilities, data preprocessing to filter irrelevant information, feature extraction to identify potential threats, and detection processing and classification using various algorithms [4], [6].

In the operation of an IDS, two common error types are false positives and false negatives. A false positive occurs when an IDS incorrectly identifies benign or normal activity as malicious, triggering an alert when no actual intrusion has taken place [6]. A high rate of false positives can lead to alert fatigue among security personnel, causing them to potentially miss genuine threats amidst the noise of numerous false alarms [6], [7]. Conversely, a false negative occurs when an IDS fails to detect an actual intrusion or malicious activity, classifying it as normal [6]. False negatives are particularly dangerous as they allow attackers to operate undetected within a system or network, potentially leading to significant data breaches or other harmful consequences [8]. Therefore, these two error types will be used as performance metrics for model validation in subsequent sections of this thesis.

As the nature of cyber threats evolved, so did the sophistication and deployment of IDS, adapting to new attack vectors and technological advancements. This is why machine learning (ML) techniques have become increasingly popular in the field of IDS, offering the potential to significantly enhance their capabilities in detecting both known and novel cyber threats [9]. Supervised ML algorithms such as random forests are trained on labelled datasets and then classify new traffic accordingly [9]. These methods can achieve high accuracy given sufficiently representative training data [2]. Unsupervised ML methods detect outliers or anomalies in unlabelled data and thus can potentially better detect previously unseen attacks [10]. Nevertheless, the scope of this thesis is restricted to supervised learning approaches.

In recent years, deep learning (DL) techniques have gained prominence in network intrusion detection research as a subset of machine learning that can automatically learn complex feature representations. Traditional ML methods typically rely on manually

crafted features as input, whereas deep learning models (e.g. deep neural networks) can ingest raw or high-dimensional data and learn informative features during training [11].

Despite their effectiveness, DL models can be computationally intensive and may face challenges related to interpretability and handling imbalanced datasets, which are common in intrusion detection scenarios [9], [11]. These different strengths and weaknesses highlight the need for a systematic comparative analysis of DL techniques against traditional ML methods, assessing not only classification performance but also practical considerations such as resource efficiency and generalisation to unseen data. This thesis aims to address this gap by evaluating and comparing these approaches using realistic datasets derived from the Locked Shields cyber defence exercise, thereby providing meaningful insights into the optimal deployment of machine learning methodologies for network intrusion detection in operational environments.

1.1.2 Background of the Locked Shields cyber defence exercise

Locked Shields is the world's largest international live-fire cyber defence exercise, held annually since 2010 by the NATO Cooperative Cyber Defence Centre of Excellence (CCDCOE) in Tallinn [12]. It brings together thousands of cybersecurity professionals from dozens of nations to practice defending critical national information systems in a high-pressure, realistic scenario. The exercise comprises multiple simultaneous scenarios, each mimicking different real-world APTs (Advanced Persistent Threats), complete with distinct indicators of compromise. These scenarios challenge teams to maintain services and rapidly respond to real-time threats [3], [12], [13].

The training exercise is primarily focused on evaluating blue team performance, with the red team serving as adversaries and trainers, coordinating attacks across all blue team networks. The blue teams act as national cyber defence incident response units, tasked with securing and maintaining their assigned virtual infrastructures (or "Gamenets") while under attack, and serve as the primary audience for the exercise. In contrast, the red team centrally coordinates cyber-attacks against all blue team networks, employing pre-planned operations to ensure a challenging, but level playing field. [2], [12], [14]. There are several supporting teams: the green team maintains the technical infrastructure, the yellow team offers real-time situational awareness, the user simulation team mimics everyday user behaviour with imperfect cyber hygiene to add realism, and the white team oversees overall coordination, including non-technical challenges. [14]

Blue teams operate as a rapid-reaction force in an already compromised network, identifying active threats, preventing escalation, and eradicating or mitigating malicious activity while maintaining critical services. In the past few scenarios, including 2023 and 2024, the story has centred on the fictional nation of Berylia coming under cyber-attack amid geopolitical conflict [3], [12]. Blue teams are working to preserve critical services like power grids, telecommunications, banking systems and other critical infrastructure from the onslaught of red team attacks [13]. It is an interdisciplinary exercise, and as such the defending teams must also be able to handle parallel challenges such as strategic decision making, forensic analysis and legal issues. [12]

The Locked Shields cyber defence exercise provides an unmatched opportunity for IDS research, as it uniquely captures contemporary threats within realistic scenarios, thereby allowing for thorough evaluation of ML and DL models under conditions closely mirroring real-world cyber operations. A more technical overview of the exercise is described in the sections 3.2 and 3.3.

1.2 Research overview

1.2.1 Research problem

Even with the advancements in IDS technology, there remains a critical gap in understanding how traditional machine learning models compare to deep learning models in real-world intrusion detection scenarios. Many existing studies on network intrusion detection have relied on outdated or synthetic benchmark datasets, which limits the practical applicability of their findings [2]. Consequently, it remains unclear which type of approach, traditional ML or modern DL, is more effective at detecting malicious activities in contemporary network environments. The recent availability of a large, high-quality intrusion detection dataset from the Locked Shields exercise [3] offers a new opportunity to evaluate these approaches under realistic conditions. However, it also raises new questions: will a conventional ML model or a deep neural network perform better when trained on such state-of-the-art data, and how well can models trained on one year's attack data generalise to novel threats emerging in the following year? Addressing this uncertainty defines the research problem of this thesis: to determine the comparative effectiveness and practicality of traditional machine learning versus deep learning

methods for network intrusion detection, using modern large-scale data and considering the challenge of evolving cyber threats.

1.2.2 Research questions

The thesis is structured into 3 distinct parts. There is the literature review to identify the best ML and DL models, the experimental setup using CRISP-DM, and validating the trained models on the next years dataset, to test the generalisation capabilities of the models when encountering novel threats. The RQ1 provides the conceptual foundation on model selection and is addressed by the literature review. RQ2 provides direct performance comparisons and is addressed by the ML experiments done on the LSPR23 dataset. RQ3 addresses model robustness over time, generalisation, and it is addressed by cross-year validation using LSPR24.

To investigate the research problem, this research is structured around three key questions (RQ1, RQ2, and RQ3). A systematic literature review is conducted to identify the leading traditional ML model and deep learning model for network intrusion detection, an experimental evaluation of these selected models on the LSPR23 dataset is performed following a structured methodology (CRISP-DM), and a cross-year validation using the subsequent year's dataset (LSPR24) is used to test the models' robustness to new threats. Accordingly, each research question aligns with one part of the thesis: RQ1 lays the conceptual foundation through model selection in the literature review, RQ2 involves the performance comparison on the LSPR23 dataset, and RQ3 examines model performance over time via validation on the LSPR24 dataset. The research questions are formulated as follows:

RQ1. Which industry-standard traditional machine learning model and state-of-the-art deep learning model are most effective and widely used for network intrusion detection?

RQ2. How do the traditional learning techniques compare to the deep learning techniques in classification performance in classifying malicious network traffic?

RQ3. How effective are traditional machine learning models compared to deep learning models in transfer learning across different databases?

1.2.3 Research objectives

To address the above research questions, the following research objectives have been defined for this research, revolving around the central question: How can the best ML/DL algorithms be reliably identified, trained, and validated for real-world IDS deployments, using strong methodological standards and large-scale, up-to-date datasets?

O1: Conduct a comprehensive literature review to identify one representative traditional machine learning model and one state-of-the-art deep learning model that are considered most effective and widely used for network intrusion detection (addressing RQ1).

O2: Develop, implement, and evaluate the selected ML and DL models on a large-scale real-world intrusion detection dataset from Locked Shields 2023. This evaluation will compare the models' classification performance in detecting malicious network traffic under realistic conditions (addressing RQ2).

O3: Assess the generalisation capability of the trained models by validating their performance on the following year's dataset from Locked Shields 2024 (LSPR24). This objective will determine how well each model handles novel attack patterns, thereby evaluating the robustness of traditional ML vs. DL approaches to evolving cyber threats (addressing RQ3).

1.3 Contribution

This thesis makes several contributions to the field of network intrusion detection systems and advances the state of knowledge in network intrusion detection. In practical terms, the findings of this thesis can help improve IDS performance, which has significant implications for enhancing network security and reducing the impact of cyber-attacks. The actionable insights gained from this comparative analysis will assist cybersecurity professionals in selecting and deploying the most suitable machine learning techniques for protecting critical systems against intrusions. In addition to these theoretical contributions, the thesis introduces a fully reproducible and complete evaluation framework which is directly applicable in future research or real-world scenarios, such as live-fire cybersecurity exercises. The major contributions are listed as follows:

1.3.1 Extensive literature review

The thesis provides an up-to-date overview of current intrusion detection approaches by performing a systematic literature search of machine learning and deep learning methods for NIDS. This review condenses the landscape of techniques and identifies the most effective and widely used ML and DL models reported in the literature (addressing RQ1). Additionally, due to its comprehensive and structured presentation, the literature review can effectively guide future research directions, both on the LSPR23 dataset and others in the domain of intrusion detection.

1.3.2 Comparative experimentation

The thesis presents an empirical comparison of a traditional ML model and a modern DL model on a realistic, large-scale IDS dataset, unlike most intrusion detection research relying on outdated or artificial datasets. By leveraging the LSPR23 dataset, which depicts real-world network traffic from a live-fire cyber exercise, the research bridges the gap between theoretical model performance and practical deployment. The results offer insights into each model's detection capabilities and limitations in an operational environment (addressing RQ2).

1.3.3 Evaluation of model robustness

The thesis evaluates the temporal robustness of ML and DL models through cross-dataset validation on LSPR24. This analysis demonstrates how the chosen models perform when faced with new and evolving attack vectors, providing evidence on whether advanced deep learning techniques maintain their advantage and practical utility over time or if their performance degrades relative to simpler models (addressing RQ3).

1.3.4 Reproducible methodology

A critical methodological contribution is the development of a fully reproducible experimental framework based on the CRISP-DM methodology, explicitly tailored for evaluating NIDS. All stages of the process, from data preparation and feature selection to model training and evaluation, are documented and conducted in a transparent manner. This provides a methodological best-practice and enables other researchers to replicate the research or adapt it to future datasets. Moreover, the work presented in this thesis is not only reproducible but directly applicable, as the developed models and procedures are structured with operational use in mind. This means cybersecurity professionals or

researchers can deploy the methods described here in subsequent Locked Shields exercises or similar live-fire cybersecurity scenarios. By clearly outlining computational requirements, dataset handling strategies, and model configurations, this thesis enables immediate practical application, enhancing its value beyond theoretical contribution alone.

1.3.5 Model architecture

This thesis contributes an innovative and customised 1-dimensional Convolutional Neural Network (1D CNN) architecture, specifically tailored and optimised for NIDS applications on the LSPR23 dataset. Unlike standard CNN models typically used for intrusion detection, this approach includes dedicated embedding layers for categorical variables addressing the challenge of efficiently handling high-cardinality categorical data, and thus significantly improving both computational performance and model accuracy. Furthermore, this research incorporates a novel GPU-accelerated implementation of the Random Forest algorithm using the recently introduced cuML library [15], a feature not yet widely adopted within the cybersecurity research community. This cutting-edge deployment has demonstrated significant efficiency and computational time advantages, significantly reducing both training and inference durations compared to conventional scikit-learn CPU-based implementations.

1.4 Thesis roadmap

The thesis is structured around the following key chapters. Chapter 2 provides an overview of the existing literature on machine learning and deep learning for intrusion detection, addressing RQ1. Chapter 3 describes the research methodology, including data analysis following CRISP-DM methodology, preprocessing, feature selection, and model implementation. Chapter 4 presents experimental results on the LSPR23 dataset and cross-year validation utilising LSPR24, addressing RQ2 and RQ3. Chapter 5 discusses findings, limitations, and future research directions. Chapter 6 summarizes the thesis, highlighting key contributions and implications of the conducted research for future intrusion detection systems.

2 Literature review

The objective of this literature review is to systematically identify and critically analyse the current industry-standard ML and DL models utilised for NIDS. The review addresses the first research question which was:

RQ1. Which industry-standard traditional machine learning model and state-of-the-art deep learning model are most effective and widely used for network intrusion detection?

The review process starts with mapping the current landscape of ML- and DL-based intrusion detection research through database searches. The selected studies are screened based on the inclusion and exclusion criteria to eliminate unsuitable or irrelevant research and the remaining candidates are evaluated according to their real-world relevance, methodology reproducibility, and any potential compatibility issues with the LSPR23 dataset. As the outcome of the literature review, one representative industry-standard traditional ML model and one state-of-the-art DL model are selected to be compared on the LSPR23 dataset.

2.1 Literature review protocol

Given that this research area is continually evolving due to rapid technological advancements, it is important that the literature review is conducted in a structured and transparent manner to be reproducible. This approach ensures the reliability of the results, allows future researchers to replicate and extend the analysis, and offers clarity on the decisions made during the review process. Consequently, systematic literature search methodology was adopted for this literature review following established guidelines such as PRISMA. [16], [17].

2.1.1 Data sources and timeframe

To minimise selection bias and enhance the robustness of the review, three databases were chosen based on their relevance and coverage in computer science and machine learning: IEEE XPLORE Digital Library, ACM Digital Library, and Scopus [18]. These databases collectively encompass a wide variety of peer-reviewed journals, conference proceedings, and technical magazines. The timeframe for the literature review is from

January 2020, until March 2025, thus covering recent advancements in machine learning and deep learning for intrusion detection.

2.1.2 Search strategy

The search query was constructed by combining relevant keywords from the fields of traditional machine learning and deep learning, alongside specific terms related to intrusion detection systems. Keywords were selected based on common terminology found in existing literature and included terms indicating comparative studies, benchmarks, and performance evaluations. The goal was to retrieve literature directly evaluating or comparing different ML or DL methods applied to network intrusion detection. One search query, SQ1, was constructed from the identified keywords:

SQ1. ("intrusion detection system" OR "network intrusion detection" OR "IDS" OR "network anomaly detection") AND ("machine learning" OR "ML" OR "deep learning" OR "DL") AND ("supervised learning") AND ("review" OR "survey" OR "comparative study" OR "benchmark" OR "evaluation")

The search query was applied to the titles, abstracts and keywords of the articles. Including abstracts ensured that relevant studies were identified even if the specific search terms were absent from the titles, as abstracts typically summarise the objectives, methods, and key contributions of the papers, providing a more comprehensive basis for identifying relevant literature. For example, the Scopus database allowed the use of the *TITLE-ABS-KEY()* search operator, applying the search query specifically to the title, abstract, and keywords. Similar search options were also available and utilised in searches conducted within the ACM Digital Library and IEEE Xplore databases, ensuring a consistent approach across all selected sources.

The retrieved papers were sorted automatically by relevance on Scopus and IEEE Xplore, and by recency on ACM Digital Library, to prioritise the most relevant and recent papers prior to manual review. Following the automatic retrieval and prioritisation, a manual two-stage screening process was employed to further filter these results, as detailed in Section 2.2.

2.2 Selection process

All papers were systematically selected from the previously defined databases. The literature review was conducted at the beginning of 2025 to ensure the inclusion of most up to date papers. The search produced a considerable number of papers, which were subjected to a two-staged screening process in accordance with PRISMA 2020 guidelines. [17]

2.2.1 Preliminary screening

The first stage involved screening the titles and abstracts of the retrieved papers to assess their relevance to the research objectives. This preliminary screening aimed to exclude studies that did not directly address the application of supervised machine learning or deep learning techniques in intrusion detection systems or those that focused on unrelated aspects of cybersecurity. During this phase, only papers that did not trigger any exclusion criteria were advanced to full-text evaluation.

2.2.2 exclusion criteria

A study was excluded if it matched any of the following criteria:

1) The study was published prior to 2020, therefore increasing the risk of using outdated ML methods.

2) The study was written in any language other than English, introducing potential language barriers.

3) The study was not published in reputable, peer-reviewed journals or conference proceedings.

4) The study was a duplicate of an already selected study from a different database.

5) The methods proposed were clearly incompatible or impractical given the nature, size, or other characteristics of the dataset targeted by this research.

2.2.3 Full text evaluation

In the second stage, the full texts of the selected papers were examined to confirm their suitability and quality. This stage was used to validate that each paper clearly described its experimental setup, including model design, dataset specifics, and evaluation metrics,

so that the results could be independently reproduced. Only the studies that met all the inclusion criteria were included in the final review.

2.2.4 Inclusion criteria

A study was included if it met all the following criteria:

1) The study primarily addresses network intrusion detection or cybersecurity anomaly detection using supervised machine learning or deep learning methods.

2) The study provides sufficient details on the model's architecture, evaluation metrics, and dataset characteristics to allow for replication.

3) The proposed method in the study is reproducible using widely available libraries (e.g., TensorFlow, PyTorch, scikit-learn).

4) The study includes empirical results or performance evaluations, preferably using recognised benchmarks or datasets.

5) The dataset used to train the model/s is similar enough to LSPR23 to ensure likely compatibility.

2.3 Selected papers

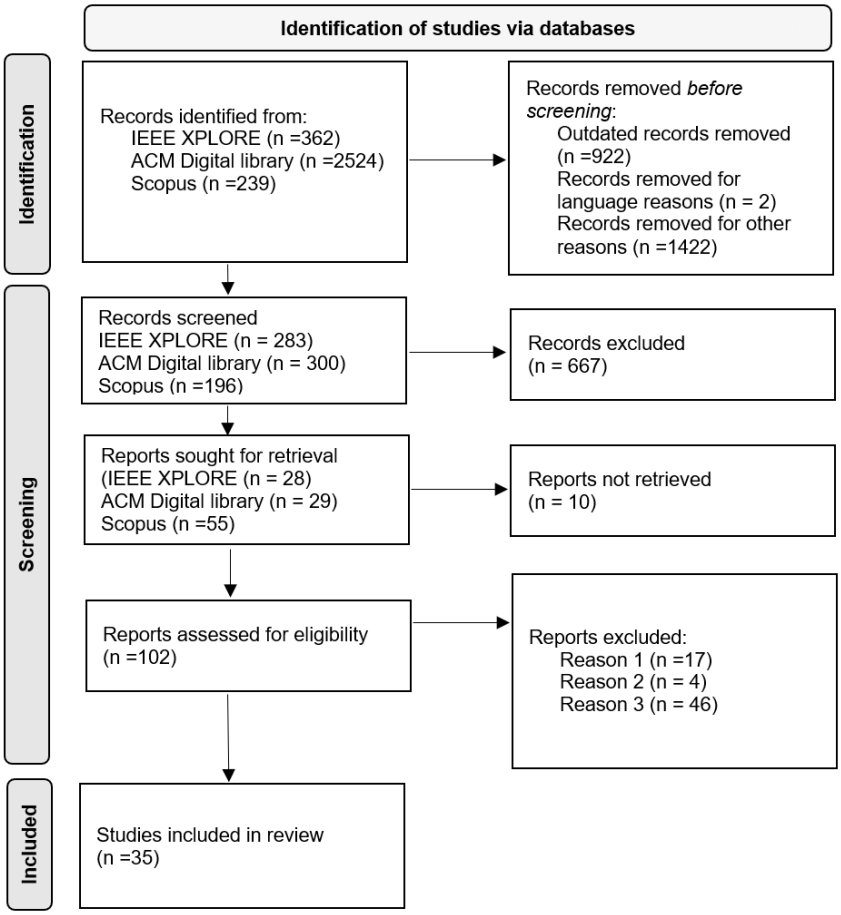

Figure 1. PRISMA systematic search flow diagram, adapted from Page et al. [17]

As can be seen from the above PRISMA flow diagram [17] on figure 1, the search query returned a grand total of 3125 records. With ACM Digital Library contributing 2,524 records, IEEE Xplore 362 records, and Scopus 239 records. After applying the language and publishing date restrictions, 924 records were dropped from the results. The ACM search returned a substantial number of records with a substantial proportion identified as irrelevant to the scope of the review upon preliminary screening. To optimise the allocation of resources and limit the assessment of unrelated literature, the abstract screening process for ACM was confined to the first 300 records. This amounted to discarding an extra 1422 records resulting in the removal of 2,347 records overall before the screening. This left 779 records eligible for further screening: 283 from IEEE Xplore, 300 from the ACM Digital Library, and 196 from Scopus.

The next phase involved a detailed screening based on title, abstract, and keywords to determine the relevance of each study in the context of the research topic. After this

assessment 667 of these records failed the exclusion criteria, e.g. the models used were unsupervised or otherwise clearly incompatible with the characteristics targeted by this research. Thus, 112 potentially relevant records, 28 from IEEE Xplore, 29 from ACM Digital Library, and 55 from Scopus, were sought for retrieval. Of these, 10 were not successfully retrieved as the full text was not available with the Taltech library access. Consequently, a total of 102 reports were advanced to full-text evaluation to determine their compliance with the inclusion criteria. Out of these reports, 17 were excluded due to including no empirical testing (Reason 1), 4 reports were duplicates of already selected papers (Reason 2) and 46 reports were dropped due to other issues (Reason 3), such as not including enough details to be reproducible.

Ultimately, 35 papers were selected for inclusion in the final literature review, categorised as follows: 19 studies addressing machine learning approaches, and 16 studies focusing on deep learning methodologies. Specifically, the ACM Digital Library contributed 7 ML and 5 DL papers, IEEE Xplore produced 5 ML and 3 DL papers, and Scopus provided 7 ML and 8 DL papers. After the full text analysis, four of the selected papers were identified to have sufficient depth to be included both in the ML and DL sections, resulting in a total of 21 papers for ML section and 18 for DL. Additionally, several papers addressing related topics, such as feature selection frameworks or explainability of machine learning methods within the context of NIDS were identified. While these papers were not included in the primary literature review, they were incorporated into subsequent sections of the thesis to provide complementary insights.

2.3.1 Publication trends

Out of the 21 reviewed machine learning studies, 15 appeared as conference papers and 6 as journal articles. Similarly, the deep learning category comprised 13 conference papers and 5 journal articles, indicating a clear preference for rapid dissemination through conferences. Geographically, the research is global in scope: notable contributions originated from Asia, particularly China [19], [20], [21], [22] and India [23], [24], [25], USA [26], [27], [28], as well as Europe [29], [30], [31], [32] and the Middle East [33], [34], [35]. This diverse provenance highlights widespread international engagement in network intrusion detection research, with active cybersecurity research communities prominently represented: USA, China, India, Australia [36]. Overall, the reviewed

literature reflects a broad cross-section of global academic interest in ML- and DL-based intrusion detection methodologies.

2.3.2 Reproducibility of ML vs. DL approaches

Most NIDS papers outline their methodology in detail, aiding reproducibility. Many studies clearly describe data preprocessing, feature selection, and model tuning. For instance, Gnanasivam et al. [26] explicitly detail using recursive feature elimination and cross-validation in training ML models. Several works list hyperparameters: e.g., a decision tree's settings or an LSTM's architecture (4 layers, learning rate 0.0001, dropout 0.2). Such transparency allows researchers to replicate or extend experiments. Almost all studies specify the dataset used (NSL-KDD, CIC-IDS2017, etc.), ensuring that evaluation benchmarks are well-defined. Both ML and DL papers report standard metrics: accuracy, precision, F1, etc., enabling result comparisons across studies.

A high proportion of papers explicitly mention the libraries/frameworks employed, which further supports reproducibility. Traditional ML studies often used scikit-learn, Apache Spark MLlib, or MATLAB toolboxes. This indicates authors built on well-tested implementations. Deep learning studies frequently note using Keras/TensorFlow or PyTorch for neural networks. Knowing the framework and version (e.g., TensorFlow 2.x) helps others achieve consistent training behaviour. Additionally, some works utilise platform environments like Google Colab or CUDA acceleration, which are mentioned to inform the computational setup.

Despite methodological clarity, only a minority of works publicly released code. In the surveyed papers, Code Repository links were rarely provided. Notable exceptions include Bridges et al. who shared a GitHub repository with code associated with their malware detector evaluation [28], and Alotaibi and Maffeis [32] who released the Mateen framework code. The lack of easy access to the code in the majority of papers means that reproducibility often relies on reimplementing models based on descriptions.

Traditional ML papers generally achieve higher reproducibility due to simpler model structures and reliance on standard algorithms. Many ML studies evaluate well-known classifiers with default or easily tuned parameters (e.g. C4.5 decision trees, SVM with RBF kernel), which anyone with the same library can replicate. In contrast, DL papers involve custom neural network architectures that are sometimes not fully specified. For

instance, some DL works omit certain hyperparameters, e.g. number of neurons per layer, or only describe them qualitatively, for example “CNN with convolutional, pooling, and fully connected layers” without specifying the exact layer counts. This can make reimplementation tricky without additional guidance. However, several deep learning studies do enhance transparency by reporting training times or giving partial architecture details. Both ML and DL research in NIDS strive for methodological openness, but ML approaches tend to be easier to reproduce given their use of well-documented algorithms and fewer tuneable parameters. Deep learning approaches show very high performance yet require careful reimplementation of neural networks to replicate results, especially when code isn’t shared.

2.3.3 Data extraction

Key information from each reviewed paper was systematically extracted, including authors, publication year, algorithm category (classical machine learning or deep learning), models tested, datasets used, best-performing model, top reported metrics, with focus on accuracy and F1 score, utilised tools/libraries, and strengths or weaknesses highlighted by the original authors. Terminology and measurement units were standardised across studies to ensure consistency; for example, all decision tree-based methods (CART, C4.5, ID3) were consolidated under "DT (decision tree)." For the metrics, accuracy and F1 were chosen, as accuracy provides a straightforward evaluation of overall detection correctness across different literature due to its widespread use, while the F1 score balances precision and recall, useful for assessing performance in imbalanced datasets typical in NIDS literature.

Uncertainties regarding the top-performing methods were resolved using authors' explicit conclusions or highest reported metric values. Tools and frameworks such as scikit-learn and TensorFlow were noted explicitly when stated by authors. Primary metrics, the accuracy and F1 score for binary supervised classification, were consistently extracted to enable uniform comparison.

Although multiple data extraction tables were created, only the most essential and summarised data is presented here due to length constraints. Appendix 2 table provides a comprehensive reference summary of the key metrics and findings for comparative analysis.

2.4 Overview of ML-based models in intrusion detection

A wide range of classical machine learning algorithms has been applied to network intrusion detection, with tree-based classifiers and support vector machines among the most prevalent approaches. This distribution suggests that researchers gravitate toward models known for high accuracy in classification tasks, such as trees and SVM for intrusion detection, while simpler models serve as baselines or for comparison. Table 1 summarises the usage and average performance metrics of the main ML models across the literature.

Table 1. Overview of Machine Learning methods used in studies

ML model	Studies (n)	Avg. acc.	Avg. F1	References
Random Forest (RF)	18	~95% (80.66–100%)	~93% (87.00–99.82%)	[26] [37] [33] [20] [38] [39] [29] [28] [21] [40] [41] [42] [43] [22] [25] [44] [23] [36]
Decision Tree (DT)	11	~93% (70.33–100%)	~92% (76.00–99.90%)	[26] [37] [20] [39] [40] [19] [42] [22] [25] [44] [23]
Support Vector Machine (SVM)	12	~89% (51.75–100%)	~91% (82.00–99.97%)	[26] [33] [20] [38] [39] [30] [29] [40] [41] [43] [25] [36]
K-Nearest-Neighbours (KNN)	11	~91% (64.70–100%)	~91% (79.00–99.52%)	[26] [33] [20] [29] [40] [19] [41] [43] [25] [44] [23]
Naïve Bayes (NB)	5	~77% (51.75–100%)	~88% (83.00–98.00%)	[20] [38] [43] [25] [23]
Logistic Regression (LR)	5	~83% (63.96–93.00%)	~88% (85.00–93.60%)	[26] [33] [20] [43] [25]
LDA (Linear Disc. Anal.)	3	~94% (89.30–98.10%)	N/A	[37] [29] [41]
Extra Trees (Ext. Randomised Trees)	3	~97% (95.87–99.03%)	~98% (95.96–99.82%)	[30] [22] [36]

XGBoost (Gradient Boosted DT)	5	~90% (65.08–99.30%)	~97% (93.00–99.40%)	[33] [21] [35] [22] [25]
Ensemble (Bagging/Stacking)	4	~97% (94.50–99.00%)	~96% (94.90–96.90%)	[20] [38] [43] [22]

Random Forest is frequently highlighted for its robustness to overfitting and ability to handle high-dimensional feature spaces. Because RF aggregates many decision trees via bagging, it significantly reduces variance and can maintain strong performance even when the input feature set is large or noisy [25]. Indeed, several studies note that RF yields stable, high accuracy across different network environments and attack types, making it a reliable choice for intrusion detection [25], [41]. For example, Li et al. [20] found tree-based models, such as DT, RF, and Bagging ensembles, consistently outperformed linear models in both accuracy and efficiency, and Leon et al. [41] showed that RF can accurately detect all classes of network attacks across multiple datasets. The primary drawback of Random Forest is its computational cost: training an ensemble of hundreds of trees can be slower and more memory-intensive than training simpler models [29], [41]. In one experiment on UNSW-NB15, an RF classifier attained the highest accuracy with the incurred cost in significantly longer training time, on the order of minutes, compared to lightweight methods like linear discriminant analysis or clustering, which had lower accuracy [29]. Nevertheless, RF often remains computationally feasible. RF model, even when trained on the large CIC-IDS-2017 dataset, was still faster in execution than an SVM or kNN classifier, indicating that well-optimised ensemble implementations can strike a good balance between accuracy and speed [41].

Decision Trees (DT), on the other hand, are praised for their simplicity, interpretability, and low latency. While a single DT might not always match the raw accuracy of an ensemble, it can be highly efficient and, if properly tuned, still achieve excellent detection rates. Li et al. [20] and Disha & Waheed [42] both demonstrate that a plain decision tree can reach around 92–99% accuracy in intrusion detection tasks when enhanced with appropriate preprocessing. In a comparative study focusing on resource-constrained IoT settings, a CART decision tree was identified as the best performer, providing over 99% accuracy with the shortest training and prediction time among 15 algorithms tested [37]. Likewise, using UNSW-NB15, a tuned decision tree achieved about 92.8% accuracy with

the lowest false positive rate of 11.7%, thus outperforming more complex models in that experiment. This was made possible by applying feature selection and class-balancing techniques [42]. The strengths of DTs include very fast inference and the ability to naturally handle categorical features, which are beneficial for real-time IDS deployment [20], [42]. However, a known weakness is the tendency to overfit if the tree grows too complex or if noisy features are present. Simpler trees may also have slightly lower precision than recall in skewed datasets, meaning they might produce more false alarms unless carefully pruned or balanced. Overall, studies suggest that when computational efficiency is a priority, such as in IoT or edge devices, a well-tuned decision tree can offer a favourable trade-off between accuracy and speed [20], [37], [42].

Other classical classifiers have been explored to varying success. Support Vector Machines (SVMs) with non-linear kernels often attain high accuracy on binary classification tasks and have been used in numerous NIDS works. For instance, SVM reached about 96–97% accuracy in some evaluations on KDD and LAN simulation data [33], [41]. Nonetheless, SVMs tend to be memory- and time-intensive for large datasets, as training complexity grows with the number of samples, and their performance can degrade if the data is not scaled or if optimal hyperparameters are not found. In one multi-dataset study, an RBF-kernel SVM achieved excellent accuracy on simpler datasets, such as KDD99, but was outperformed by ensemble methods on more complex data [41]. In another study, SVM was outperformed by even a basic tree [25]. Moreover, the SVM required longer runtime than RF to train on a large dataset, in this case the CIC-IDS-2017 [41].

k-Nearest Neighbours classifiers have also shown strong detection capability in some cases. A comprehensive IoT-focused analysis by Saif et al. [23] found that kNN was among the top-performing algorithms with approximately 97% average accuracy across a collection of 15 IoT and network intrusion datasets. This reflects kNN's strength in adapting to varied data distributions without an explicit training phase. However, kNN's major drawback is computational inefficiency at scale: storing and comparing large volumes of network traffic instances leads to high memory usage and slow lookup times for detection. The same study noted that kNN's classification process became computationally expensive for large datasets, making it less practical despite its accuracy.

Meanwhile, Naïve Bayes (NB) and linear models such as LR and LDA generally exhibit much faster training and prediction times but consistently underperform in detection accuracy on complex datasets. [25], [38] For example, Garg and Mukherjee [25] report NB barely reached 52% accuracy on NSL-KDD under certain realistic train-test conditions, which was the lowest among the algorithms examined. Because of such limitations, NB and simple linear classifiers are mostly used as lightweight baseline models or as components in ensemble strategies, rather than stand-alone solutions for high-security scenarios [25], [38].

Beyond individual algorithms, feature engineering and evaluation practices play a critical role in classical ML-based NIDS performance. Many researchers apply feature selection or dimensionality reduction techniques prior to modelling, in order to remove redundant features and mitigate the curse of dimensionality. For instance, Recursive Feature Elimination (RFE) with an RF estimator was used by Gnanasivam et al. [26] to identify an optimal subset of network features in UNSW-NB15, which improved the accuracy and balance of their classifiers by reducing overfitting. A similar approach using RFE was adopted in an ensemble study on NSL-KDD, where selecting the top 13 features contributed to a notable jump in detection accuracy for the combined SVM+RF model versus using all features [38]. Other studies have employed statistical measures such as mutual information, chi-square, or model-based importance scores to guide feature selection [25], [42]. Disha and Waheed [42] demonstrated that a backward elimination of less significant features by using chi-square tests significantly enhanced the performance of DT, GBT, and MLP models on UNSW-NB15. In in fact, the decision tree's success in their work was partly attributed to this feature reduction, which pruned irrelevant attributes that had been hindering the more complex models. Interestingly, they observed that RF's performance did not improve with feature elimination. This is likely because RF inherently benefits less from external feature selection, as it can internally ignore uninformative features due to averaging across trees [42]. This suggests that feature selection benefits are more pronounced for simpler models, whereas ensemble methods already mitigate some effects of high dimensionality.

In summary, classical machine learning methods continue to be prominent in NIDS research. Tree ensemble models, particularly Random Forest and boosted trees, are the most frequently endorsed due to their high detection accuracy and resilience against overfitting [21], [37], [41]. In many cases, these ensembles have demonstrated

performance on par with or even exceeding more complex methods, especially when combined with proper feature selection and parameter tuning [21], [38]. Simpler algorithms such as decision trees, kNN, and SVM remain important as baseline or niche solutions, offering advantages in speed or simplicity that can be valuable in constrained environments, for example, real-time IoT devices [20], [23], [42]. The literature also highlights trade-offs between detection performance and computational cost: ensemble models yield superior accuracy but may introduce greater training or inference latency, whereas lightweight classifiers are faster but generally less accurate [23], [38]. Researchers have addressed these trade-offs by employing techniques such as dimensionality reduction, parallel processing, and adaptive sampling to ensure that even high-performing models can meet operational efficiency requirements [26], [29], [37], [45]. Finally, extensive evaluations on diverse datasets, including the classic KDD and NSL-KDD as well as newer benchmarks such as UNSW-NB15, CIC-IDS2017, and domain-specific IoT datasets, have built a thorough understanding of classical ML capabilities. These studies collectively demonstrate that with careful tuning and evaluation, traditional machine learning-based NIDS can achieve robust and reliable intrusion detection, forming a strong baseline against which newer deep learning approaches are often compared, as discussed in the next section.

2.5 Overview of DL-based models in intrusion detection

Feed-Forward Deep Neural Networks (DNN/ANN) surface frequently as effective classifiers as in multiple studies, a fully connected ANN with several hidden layers achieved top accuracy and F1 scores. For instance, a comprehensive survey by Gamage and Samarabandu [46] found that a deep feed-forward network attained accuracy up to ~99% and the best F1-score across four benchmark datasets. Its strength lies in fast training/inference and high accuracy even on large datasets, outperforming other DL models in that survey. Similarly, Saif et al. [23] report an ANN classifier as the most reliable model for IoT intrusion detection, noting its adaptability to complex and structured data with robust recall and F1. Attia et al. [27] observed that compared to XGBoost an ANN -based approach could generalise better to previously unseen attacks in some cases. These results illustrate that a well-tuned deep MLP/ANN can provide strong baseline performance in NIDS tasks.

Convolutional Neural Networks (CNN) have been widely applied, especially for capturing spatial or sequential features in traffic data. Nour and Said [34] evaluated CNNs against DNNs and RNNs on the CICIDS2017 dataset and found CNN achieved the highest accuracy and F1-score. The CNN's ability to extract local feature patterns contributed to its superior true positive and true negative rates. Several other works corroborate CNN effectiveness: CNN variants are also used in hybrid models; one study combined CNN with LSTM to leverage both spatial and temporal features, reporting that the CNN+LSTM hybrid consistently outperformed standalone RNNs on unbalanced and balanced data, but a standalone CNN outperformed the hybrid model when using balanced datasets [47]. However, CNNs typically require more training time and computational resources; for example, a CNN in one experiment needed substantially longer training time than a simpler DNN on the same data [34]. Despite this, their high accuracy on complex multi-class traffic patterns makes them a popular deep learning choice for NIDS.

Recurrent Neural Networks (RNN), particularly Long Short-Term Memory networks (LSTMs), are effective when sequential dependencies in network flows are considered. They have been applied to model time-series aspects of packet streams or event logs. LSTM was the best performer in Layeghy and Portmann's cross-domain NIDS evaluation [36], achieving the highest F1-score on certain NetFlow-based datasets. Its strength was noted in handling sequential data and maintaining high accuracy across different network domains. Pawlicki et al. [31] also found that a LTSM model yielded the highest balanced accuracy and F1 among deep models on CIC-IDS2017, effectively detecting multiple attack types. RNNs excel at temporal pattern recognition, for example Shaffi et al. [48] demonstrated a RNN surpassing a CNN on NSL-KDD, attributing this to the RNN's ability to capture temporal features in the connection sequences. The downside is computational cost: RNN/LSTM models can be slower to train and inference, as noted by Pawlicki et al. and others, particularly due to high inference latency. They may also struggle with rare attack classes if not enough sequence data is available. Nonetheless, in scenarios where the order of events is important, such as detecting slow-moving attacks over time, LSTM-based NIDS have a clear advantage in capturing context that static models might miss.

Table 2. Overview of Deep Learning methods used in studies

DL model	Studies (n)	Avg. acc.	Avg. F1	References
RNN/LSTM	11	~92% (81.3–100%)	~88% (73–99.92%)	[34] [36] [47] [46] [48] [40] [19] [31] [49] [50] [51]
CNN	9	~92% (77.25–98.71%)	~93% (84–98.67%)	[34] [24] [47] [48] [52] [19] [31] [49] [51]
DNN	4	~96% (90.77–98.05%)	~95% (93–98.33%)	[34] [24] [31] [49]
ANN	5	~98% (95.99–99.17%)	~96% (86.43–99.76%)	[27] [23] [36] [46] [50]
AE	5	~95% (92.66–99%)	~90% (76.28–99%)	[23] [36] [46] [53] [32]
MLP	2	~90% (80.3–99.99%)	~83%	[54] [40]
Other (e.g. Transformer, LAD, GRU, Mateen)	7	N/A	N/A	[34] [24] [47] [46] [53] [19] [32]

While the focus of this thesis is on supervised learning, a few studies integrated unsupervised components to the supervised approach to enhance detection of novel attacks. For example, Uddin et al. [53] proposed a dual-tier system using an Autoencoder (AE) in the first tier for one-class learning. Their unsupervised model (usfAD) retrains on anomalies via clustering, yielding an accuracy of ~98.9% and notably high detection of zero-day attacks. This highlights that even within supervised contexts, AEs can assist by learning normal traffic patterns and flagging outliers. Another study by Alotaibi & Maffeis [32] used Deep Autoencoders as part of an adaptive ensemble (Mateen) to handle concept drift in network traffic. While autoencoders alone were not the top-performing supervised classifiers as they were mainly used to reconstruct inputs, their value in NIDS is seen in hybrid or adaptive frameworks that maintain performance over time.

Across the DL literature, certain performance trends emerge. Most deep models, when trained and tuned properly, achieve very high binary classification accuracy, often 95–99%, on standard datasets. For example, various IoT-focused studies report deep model F1-scores above 95% on average. However, these impressive results sometimes come with caveats: class imbalance can inflate accuracy as models may just learn the majority class, so recall and F1 are more reliable indicators in such cases. Indeed, many authors emphasise F1-score and recall for evaluation, and some note the need for techniques like oversampling to balance training data. Another trend is that no single DL architecture is universally best, as performance can depend on the dataset characteristics. CNNs topped the rankings on flow-based datasets with rich feature structure, whereas LSTMs excelled when temporal sequence mattered. Strengths noted for DL models include their ability to handle complex, high-dimensional data and discover intricate patterns e.g. subtle attack signatures without manual feature engineering. Reported weaknesses include high training time. the need for large, labelled datasets, and especially poor explainability, which is critical when taking decisions based on the models output. Researchers are actively addressing these issues by exploring hybrid models, transfer learning, and more explainable DL in NIDS [22], [36], [55], [56].

2.6 Literature review findings

This section presents the primary findings derived from the systematic literature search conducted to identify effective and widely utilised machine learning and deep learning models for network intrusion detection. It identifies existing research gaps and establishes the rationale for selecting Random Forest and Convolutional Neural Networks as the representative models for subsequent experimental evaluation on the LSPR23 dataset.

2.6.1 Summary

Both classical machine learning and deep learning methods have achieved notable success in binary network intrusion detection. Among traditional ML approaches, tree-based ensemble models stand out. Random Forest and boosted trees (e.g., XGBoost, LightGBM) consistently rank as top performers, often attaining accuracy above 99% on benchmark datasets, as can be seen in table 1. Multiple studies showed RF or gradient-boosted trees detecting intrusions with near-perfect true positive rates and minimal false alarms [21], [22], [33], [37], [41], [43]. Simpler algorithms like K-Nearest Neighbours

and Naïve Bayes sometimes lag behind in complex scenarios but can excel in optimised settings or certain data domains [19], [23], [38]. For example, Yu et al. showed KNN outperforming more complex deep learning models across multiple log-based anomaly detection datasets, demonstrating that simpler methods can be more effective in optimised, well-suited data domains, particularly benefiting from simplicity and computational efficiency [19].

Deep learning models, meanwhile, demonstrate an ability to automatically learn features and handle complex attack patterns. Several works reported deep neural networks, especially CNNs and LSTMs, reaching 95–99% F1-scores in detecting attacks ranging from DoS to web intrusions [24], [31], [34], [36], [46], [47], [50]. Notably, DL methods proved adept at low signal-to-noise situations: e.g., an LSTM-based system maintained ~99% F1 across different network environments [36]. However, the literature also highlights trade-offs. Classical ML models are generally faster to train and easier to interpret, for example decision trees offer transparency in decisions, whereas DL models, though powerful, can be computationally intensive and act as black boxes [19]. For instance, an AlexNet-based IDS achieved excellent accuracy with low latency, but its complexity makes it harder to interpret decisions [52]. Furthermore, some deep models did not significantly outperform well-tuned ML models on tabular NIDS datasets, suggesting that simpler algorithms remain competitive, especially with proper feature engineering.

2.6.2 Importance of the training dataset for ML algorithm selection

Performance outcomes for classical ML-based NIDS vary notably with the choice of dataset and algorithm. On legacy benchmarks like KDD99 and its refined version NSL-KDD, most classifiers can achieve very high accuracy due to the relative simplicity or redundant patterns in these datasets [29], [37], [43]. In contrast, on more recent and complex datasets such as the UNSW-NB15 or CIC-IDS 2017, reported accuracies are generally lower, in the range of roughly 85–95% for the best classical models [26], [40], [42]. For instance, using the UNSW-NB15 dataset, which contains contemporary attack traffic and is more challenging, Gnanasivam et al. [26] observed the highest accuracy with a neural network model. 88.62%, while the best traditional classifier, RF, achieved about 87.39% accuracy. Notably, the RF in that thesis still produced a high recall and F1-score, indicating balanced detection capability, and it did so with a faster runtime than the

neural network approach [26]. Chindove and Brown [40] report that while several algorithms showed nearly 100% overall accuracy on CIC-IDS owing to majority of traffic being benign, their F1-scores for attack detection were much lower; among the tested models, RF obtained the highest F1 of 87% on CIC-IDS-2017, outperforming other classifiers like multi-layer perceptrons, SVMs and RNNs under the same conditions.

2.6.3 Research gaps

Detecting unknown or zero-day attacks remains a significant challenge, as traditional supervised models typically underperform on novel threats [27]. While some deep learning models demonstrate improved generalisation, no unified solution currently exists, prompting exploration of hybrid or advanced models such as combining RF with anomaly detection or enhancing CNN with outlier detection mechanisms [34], [53].

Another persistent issue is the performance of models across different domains, with models often experiencing reduced accuracy across different datasets due to shifts in feature distribution [36], [44]. This indicates the need for robust models adaptable to varying network contexts, to be addressed through evaluations on multiple datasets and possible domain adaptation techniques. This and the previous point will be tested by using the 2024 LS dataset.

Class imbalance of the datasets further complicates intrusion detection tasks, as it can lead to high overall accuracy masking poor minority-class detection [40], [47]. This thesis plans to address class imbalance by using the synthetic minority over-sampling technique (SMOTE) proposed by Chawla et al. [57], and utilised by Meliboev et al. [47], Chindove and Brown [40], Shaffi et al. [48], Altamimi and Abu Al-Haija [44].

Reproducibility remains limited within the field of NIDS due to inconsistent evaluation practices and lack of shared code [28], [46]. By employing established models (RF, CNN) under uniform testing conditions, this research provides reproducible baseline results, validating previous claims. In summary, RF and CNN are confirmed as effective models for NIDS, and the experimental approach of this thesis is guided by addressing the highlighted gaps: particularly unknown attacks, cross-domain robustness, class imbalance, and reproducibility.

2.6.4 Answer to research question RQ1

RQ1: Which industry-standard traditional machine learning model and state-of-the-art deep learning model are most effective and widely used for network intrusion detection?

Based on performance and reproducibility considerations, the selections made for experimentation stage were Random Forest as the representative ML model and a Convolutional Neural Network as the representative DL model. RF was chosen because it repeatedly emerged as a top traditional classifier with robust accuracy and low false alarm rates across various studies. It offers advantages in reproducibility, as it is implemented in standard libraries, with relatively few hyperparameters (trees count, depth) to tune and many studies found RF results easy to replicate and stable across datasets [58]. On the deep learning side, CNNs have demonstrated high efficacy in capturing complex patterns in network traffic, yielding F1-scores around 97–99% in several evaluations. CNNs were frequently favoured for their balance of performance and implementation maturity: CNN architectures for intrusion detection are well-documented in frameworks like TensorFlow, and their design (convolution + pooling layers) is easier to customise for tabular or flow data than more opaque architectures. While other DL models also performed strongly, CNNs offer a good compromise between accuracy and training speed and have been successfully reproduced by multiple independent studies while providing enough details (layer counts, training epochs, etc.) [34], [46], [47]

Random Forests and Convolutional Neural networks were selected based on literature consensus highlighting ensemble trees and deep neural networks as highly effective methods in NIDS, forming the basis of the experimental work.

3 Experimental design and implementation

This chapter details the experimental design and methodology employed to systematically compare traditional machine learning and deep learning models for network intrusion detection. Specifically, it addresses the second research question (RQ2):

RQ2: How do traditional machine learning techniques compare with deep learning techniques in terms of classification performance for malicious network traffic?

The experimental approach integrates applied experimentation with the Cross-Industry Standard Process for Data Mining, providing a structured, transparent, and replicable framework for data preparation, model implementation, and evaluation. CRISP-DM was chosen due to its rigorous yet flexible process, which provides a clear understanding of data-centric steps required for robust model development. [59]

Given that the primary goal is evaluating the practical efficacy of ML and DL models, an applied experimentation approach, specifically validation testing, was selected. Validation testing ensures a structured and controlled evaluation environment that closely mirrors real-world operational conditions, and thus enables a direct and fair comparison between competing models [60].

The experimental environment is established using the Locked Shields Partners Run 2023 (LSPR23) dataset, which provides high-quality, accurately labelled, and realistic network traffic data captured from actual attacker-defender interactions during the world's largest live-fire cyber defence exercise [3]. Although real-time traffic is not used for the validation, the realism and scale of the LSPR23 dataset ensure appropriateness for the validation testing approach on the ML and DL algorithms. Consequently, this approach provides a solid foundation to determine whether state-of-the-art deep learning methods outperform, match, or underperform industry-standard traditional machine learning techniques in detecting malicious network activities.

3.1 Overview of CRISP-DM methodology

The Cross-Industry Standard Process for Data Mining methodology was chosen as the methodological backbone for this thesis because of its structured approach and proven effectiveness in handling data-related projects from the idea to the deployment. While

alternative methodologies offering more exploratory or iterative workflows have emerged, CRISP-DM remains highly regarded for projects that demand goal-oriented processes and clear, documented procedures. [59], [61], [62]. In the context of this thesis, CRISP-DM ensures that each phase, from understanding the business problem (or, in this case, the research objectives) to deploying and monitoring the model, can be conducted and reported systematically.

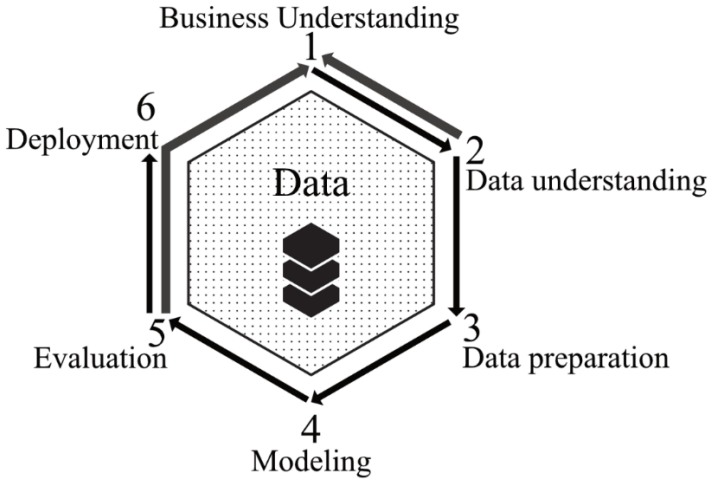

Figure 2. CRISP-DM process model

CRISP-DM contains six main phases, outlined on Figure 2. While the phases are commonly presented linearly, projects often loop back to earlier stages as new findings emerge, on the figure this is illustrated by the thicker dark grey arrows. For instance, additional data cleaning or feature engineering might be required if the evaluation phase reveals inadequate performance [62]. The six phases of CRISP-DM, as applied within the scope of this thesis, are detailed below:

1. Business understanding

   Although originally designed with business contexts in mind, the CRISP-DM framework generalises to accommodate a broader spectrum of data science applications, including academic research [62]. In the context of this thesis this phase translates into defining the research goals, identifying the core problems to be solved, and specifying success criteria.

2. Data understanding

This phase involves familiarisation with the dataset, exploration of its key characteristics, and initial identification of any quality or representativeness issues. For the LSPR23 dataset, data understanding entails examining its network traffic features, identifying the different types of malicious activity, and confirming that labelling is accurate and consistent.

3. Data preparation

The LSPR23 dataset is already well prepared and labelled by Dijk et al. [3]. Therefore, the data preparation required in this thesis primarily involves validating data integrity, selecting the relevant features, and performing minor preprocessing steps to optimise data compatibility with the selected machine learning and deep learning algorithms.

4. Modelling

This phase focuses on selecting and configuring appropriate algorithms to meet the research objectives: comparing traditional ML and deep learning approaches. Key activities include hyperparameter tuning, model architecture design, and resource usage, e.g. GPU vs CPU based approach considerations. Each model is built upon the refined dataset resulting from the earlier steps.

5. Evaluation

Evaluation involves measuring how well each model performs against established criteria, such as accuracy, F1-score, or ROC-AUC, and determining whether it meets the requirements defined during the initial phases. Because intrusion detection entails high-stakes decision-making in cybersecurity, this step also involves an in-depth look at false positives and false negatives, along with any potential trade-offs between model outcome explainability, detection efficacy and computational cost.

6. Deployment

Although the models evaluated in this thesis are not intended for immediate deployment in a production environment, the CRISP-DM framework emphasises the importance of planning for future deployment and monitoring. Therefore, this section

describes the experimental environment in detail, including hardware and software configurations, as well as providing clear instructions to ensure reproducibility of the conducted experiments.

3.2 Business understanding

In the context of this thesis, the 'Business Understanding' phase from CRISP-DM translates into clearly defining research objectives and establishing precise evaluation criteria. Although this thesis is conducted within an academic setting, it retains practical relevance by addressing real-world needs in network intrusion detection.

3.2.1 Objectives

The primary objective of this section, aligned directly with the second and third research questions (RQ2, RQ3), is to systematically evaluate and compare the effectiveness of traditional machine learning and deep learning models in classifying malicious network traffic. This involves not only measuring raw classification performance but also examining trade-offs in terms of computational efficiency, resource demands, and interpretability of results.

Secondary objectives are to explicitly identify the operational constraints and requirements relevant to cybersecurity stakeholders, quantify the impact of resource limitations on model deployment feasibility, and offer clear guidance on selecting suitable intrusion detection models tailored to practical scenarios and real operational environments.

3.2.2 Stakeholders

From a practical standpoint, the stakeholders for this research would be the infrastructure defending cybersecurity teams, in the context of the LS cyber-exercise, the so-called blue teams. The blue teams are interested in detecting intrusions with high reliability, minimal false alarms, and manageable resource consumption [2], [63]. Incorrectly classified intrusions (false negatives) may lead to substantial operational disruption, loss of sensitive data, and potential reputational damage. On the flip side, excessive false alarms (false positives) divert critical human resources from real threats [8]. By conducting the comparative analysis, the goal of this section of the thesis is to inform these teams on the potential trade-offs between advanced DL models and more established ML counterparts.

3.2.3 Resources and constraints

The operational environment within the LS cyber-defence exercise places substantial resource limitations on the participating Blue Teams. Teams predominantly rely on virtual machines (VMs) with restricted computational capacities and limited connectivity through a low-bandwidth VPN tunnel. Consequently, any intrusion detection systems employed must be optimised for computational efficiency and minimal bandwidth consumption. [63]

In this research, experimental analyses are conducted using hardware of moderate computational capacity, comprising components that are several generations behind the current state-of-the-art as of publishing: an Intel Core i5-13500 CPU with 14 cores, an Nvidia RTX 3090 GPU, and 64GB of DDR4 RAM. This hardware configuration was deliberately selected due to its cost-effectiveness and realistic ML performance, making it a practical option for deployment by Blue Teams during the exercise [64]. Thus, the findings and models derived from this thesis maintain their practical relevance and applicability to environments subject to similar operational constraints.

3.2.4 Success criteria

In an typical operational environment, success is measured by how effectively an Intrusion Detection System can distinguish between benign and malicious traffic while minimising false negatives [8]. In this thesis, the primary performance indicators were chosen during the literature review. The chosen indicators are accuracy, F1-score, ROC-AUC, and confusion matrix analyses. Secondary factors include training time, inference speed, power usage and memory usage, which can significantly impact deployment feasibility. Both sets of metrics, detection accuracy and resource footprint, are needed to justify the adoption of the given IDS model in practice.

This stage of the thesis research has been considered a success if it results in clear, evidence-based conclusions regarding the relative effectiveness and practical trade-offs between traditional machine learning and deep learning models. Thus, providing stakeholders with actionable insights to support informed decision-making in operational intrusion detection contexts.

3.2.5 Risks and mitigation

1. Data imbalance in the training dataset leading to biased models

As discussed in the upcoming Data understanding section, the LSPR23 dataset is imbalanced, featuring approximately a 10:1 ratio of benign to malicious network flows. Such imbalance poses a risk of producing machine learning models biased towards the majority class, adversely affecting detection accuracy, particularly regarding malicious traffic. This risk can be mitigated by assigning weights to each class that are inversely proportional to their frequency in the dataset during training. Additionally, sampling methods specifically designed for imbalanced datasets, such as the SMOTE dataset balancing technique proposed by Chawla et al. [57], are considered to enhance model robustness if the model performance is not satisfactory.

2. Computational resource limitations impeding deep learning model training

Although the selected deep learning model was chosen with computational constraints in mind, there remains a potential risk that its resource demands, in particular GPU memory usage and processing power, could exceed the available hardware capacity. If such resource constraints impede effective model training, mitigation measures would include employing incremental or batch training strategies. A more substantial mitigation strategy would be reverting to an alternative, less resource-intensive yet adequately performing, runner-up model identified during the literature review.

3.3 Data understanding

3.3.1 Dataset selection

The LSPR23 dataset was selected because it aligns best with the research objectives of this thesis. Established and popular datasets such as KDDCUP-99 and CICIDS-17 exhibit significant limitations, including outdated attack scenarios, unrealistic network infrastructure, restricted scale, and imbalanced data representation. Recent datasets like Rosid-23 and ETFD-22, while addressing some contemporary requirements, still lack sufficient scale for robust, generalisable machine learning model development. [2], [3]

In contrast, LSPR23, derived from the Locked Shields exercise, the largest live-fire cybersecurity exercise globally, features realistic attacker-defender interactions,

sophisticated and contemporary attack types, and an extensive, balanced representation of network activities [3]. Unlike many of the network datasets which are unlabelled [65], the LSPR23 dataset provides accurately labelled data, which is essential given that this thesis exclusively employs supervised machine learning methods [66]. All these characteristics enhance the dataset's applicability for advanced machine learning-based intrusion detection, making it the most suitable choice for this research among the publicly available datasets [3].

3.3.2 Data access

The LSPR23 dataset utilised in this research is publicly accessible and can be retrieved from the Zenodo repository [67], thereby supporting reproducibility and transparency within the academic community. Additionally, the more recent LSPR24 dataset has now been published on Zenodo [68], with the accompanying paper scheduled to appear at CyCon2025. Early access to the LSPR24 dataset was granted through direct collaboration with the dataset creators, who permitted its use within the scope of this thesis.

3.3.3 Overview of the Locked Shields cyber exercise

Locked Shields (LS) is an annual international live-fire cyber defence exercise organised by the NATO Cooperative Cyber Defence Centre of Excellence (CCDCOE) since 2010 [12]. It is structured as a red team vs. blue team competition: multiple national blue teams act as defenders, while a centralised red team plays the adversary launching cyber-attacks [2]. Within a compressed timeframe, typically two days of full-scale operations, red team attacks progress through all stages of sophisticated cyber campaigns. From initial intrusion and privilege escalation to data exfiltration and infrastructure sabotage, all of which blue teams must detect and mitigate in real time [2], [3]. Each blue team operates in isolation on its own identical network environment and is scored on its ability to maintain service availability and contain attacks [14].

The attacking red team in LS is a coordinated group of cybersecurity professionals who execute a broad range of offensive tactics against each blue team's network. Their operations are pre-planned to meet exercise objectives but are performed in a live, dynamic manner. This means that the red team is not limited to a any specific attack and can adjust strategies, making the exercise unpredictable for defenders [14]. They also take advantage of the exercise's user simulation: for instance, red team may trick "benign"

users, played by the user simulation team, into executing malicious commands or opening infected files, thereby bypassing some defences [3], [14]. From a defender's perspective, this means the network traffic includes everything from phishing emails and drive-by downloads to SQL injection attacks on web servers and large DDoS bursts; all happening in parallel. This multi-faceted threat environment produces a uniquely diverse network data stream, and the many features of this environment (e.g., many simulated users, diverse protocols, and coordinated multi-stage attacks) are directly reflected in the LSPR23 dataset captures.

3.3.4 Network environment

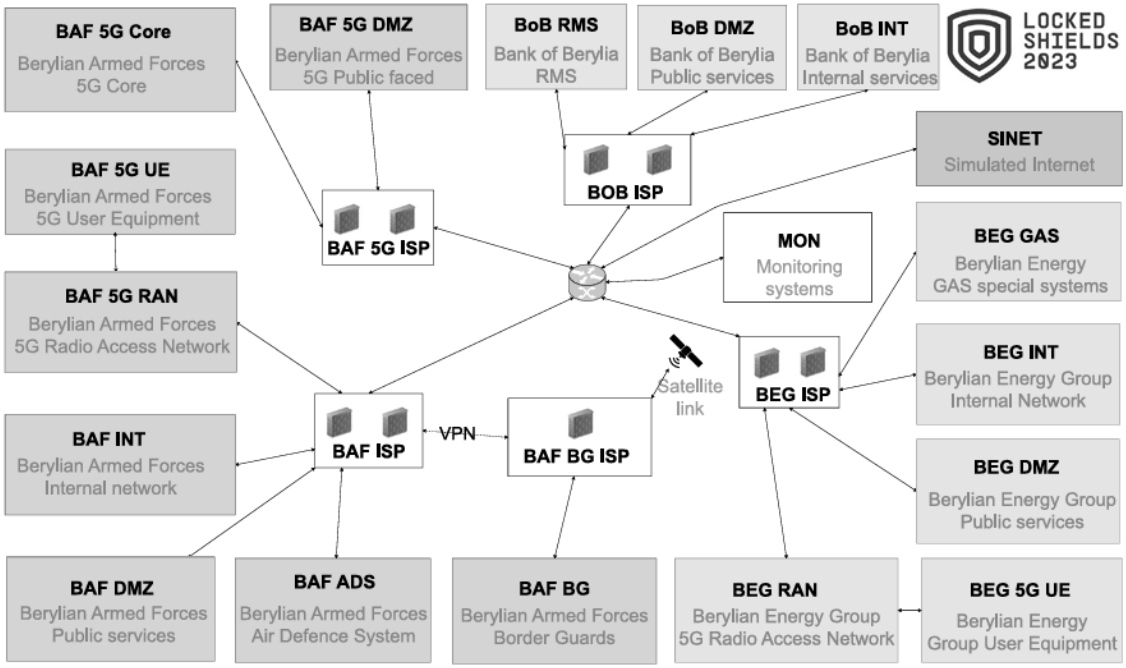

Figure 3. Locked Shields 2023 network map, adapted from Dijk et al. [3]

Each Blue Team in Locked Shields is provided a dedicated virtual network, known as a "Gamenet", which replicates the IT infrastructure of a fictional country's critical systems [14]. The Gamenet architecture is highly elaborate and segmented, combining conventional enterprise IT components with specialised operational technology. For example, the LS 2023 environment included typical enterprise networks: internet-facing services, internal corporate LANs, ISP infrastructure with routers, switches, firewalls, and VPNs. In addition critical infrastructure systems such as a 5G mobile network, industrial control systems (SCADA/ICS) managing a power grid, a military air defence network, a central bank payment system (SWIFT), border guard systems, and even satellite communication link were also included as illustrated in figure 3, adopted from Dijk at al.

[3]. Therefore blue teams must be able to defend a wide range of devices and platforms within this network, including Linux and Windows servers, user workstations, FreeBSD-based firewalls, PLCs and other industrial controllers (for water treatment plants, power management, etc.), and specialised military systems [3].

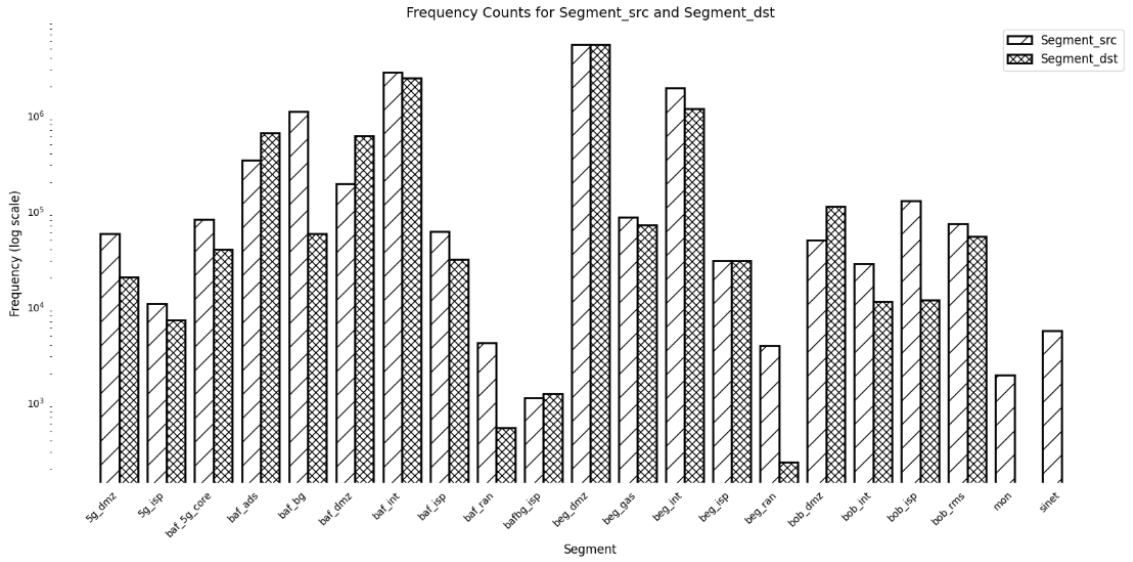

Figure 4. Overview of the traffic per network segment

An overview of network traffic by comparing the frequency of network flows originating from (bars with diagonal hatching, Segment_src) and destined to (cross-hatched bars, Segment_dst) each network segment, presented on a logarithmic scale is provided in figure 4. Most network segments exhibit comparable counts for inbound and outbound flows, though certain segments, notably ISP backbones, the 5G core network, and internal corporate LANs, demonstrate significantly higher traffic compared to specialised segments like satellite links, monitoring (mon), and remote management systems (RMS). Several segments, such as the sinet and radio access networks, display noticeable asymmetry between inbound and outbound flows. This distribution underscores that network flow volumes vary widely across segments.

3.3.5 Dataset characteristics and composition

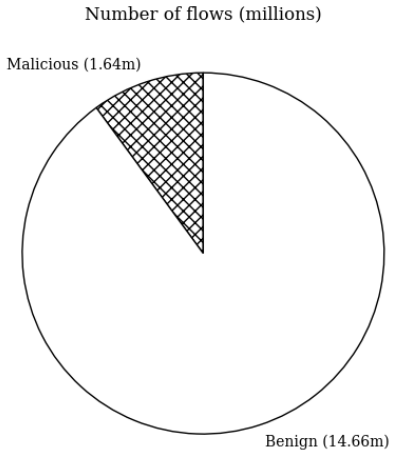

Figure 5. Distribution of malicious / benign network flows in LSPR23

The LSPR23 dataset is a collection of network traffic data captured from one of the blue team networks during the LS 2023 exercise. Specifically, a special virtual blue team deployed by the researchers to gather data. The raw packet captures from the exercise were processed into flow records, resulting in 16.3 million network flows spanning the two days of operations. Each flow represents a sequence of packets sharing common endpoints and protocol parameters, with bidirectional traffic combined into a single record for analysis. Importantly, every flow in LSPR23 is labelled to indicate whether it is part of malicious activity or just benign background traffic. The dataset creators assigned binary labels ("malicious" or "benign") to the network flows based on their connections to red-team infrastructure. Specifically, they classified a flow as "malicious" if either its source or destination was associated with known red-team infrastructure. As can be seen from figure 5, using this approach 1.64 million flows or around 10% of the dataset, were labelled as "malicious," while the remaining flows were labelled "benign." [3] This proportion of malicious traffic is much higher than in most real-world network traces, reflecting the high intensity of attacks during LS and providing a balanced dataset for machine learning in contrast to imbalanced legacy datasets [3].

In terms of data format and features, LSPR23 is extremely rich. For each network flow record, a total of 101 features have been extracted or computed. These include the usual IP/TCP header fields and flow statistics (e.g. durations, byte counts, packet counts in each direction) as well as higher-level attributes. The CICFlowMeter tool was modified to recognise 26 additional network protocols present in LS, such as industrial and military protocols not handled by default, so that application-layer traffic could be properly identified in the flows. Custom features were added to characterise the LS environment, for example an “Internal/External” flag to denote if a flow crosses the network boundary and an “L3/L4 service” label indexing which service (by port/protocol) is in use. Each flow is timestamped and indexed, and the dataset provides lookup tables for any coded values. [3]

Beyond basic flow metrics, security-specific annotations are included: the team merged Suricata IDS output into the flow records, this means if a flow triggered any Suricata alert, the corresponding signature IDs, alert categories, and severity levels are attached to that flow entry. Additionally, the dataset authors provide a high-level attack narrative that runs in parallel to the flow data. This narrative is essentially a list of documented attack events, extracted from the exercise's central logging platform, EXPO. With fields describing each attack's goal, method, tools (mapped to MITRE ATT&CK tactics), the affected victim segment, and timestamps for when the attack occurred. This acts as a glossary of all significant red team actions during LS 2023, and it can be used to pinpoint which flows correspond to which specific incidents. Finally, the dataset includes host availability logs from the exercise's scoring system: these indicate when critical services went down or recovered, providing ground truth for denial-of-service or service outage events. [3]

The structure of the released dataset is organised into multiple CSV files/tables: a primary table for flow features and labels, and supplementary tables for attack events and service availability, with relational keys that allow cross-reference. For example, an attack event entry has an ID that can be found in the flow records that were part of that attack. Overall, the dataset is delivered in a machine-learning-friendly format: numerical feature vectors with ground truth labels, plus rich metadata to enable domain-specific analyses, like focusing on specific attack techniques or network segments of interest.

3.3.6 Data quality and limitations

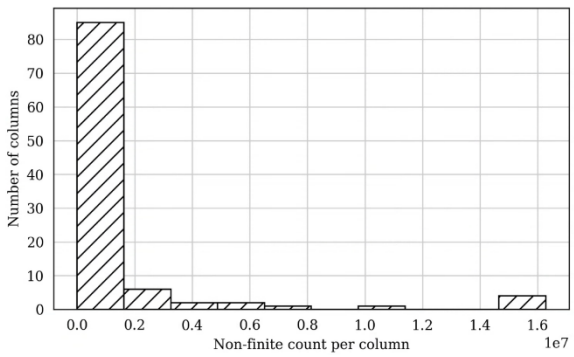

Figure 6. Distribution of non-finite values in the LSPR23 dataset

A throughout data quality assessment of LSPR23 has already been conducted by its creators to identify any shortcomings or idiosyncrasies in the collected data. They reported that the dataset is comprehensive in scope as it captures traffic from all relevant parts of the LS network and includes a broad spectrum of protocols and systems [3]. In particular, the network configuration used for data capture was very complete: all key segments: internal LANs, external connections, and even the backbone ISP and special

infrastructure; were monitored, and even typically hard-to-collect traffic like internal routing updates and inter-segment traffic was recorded [3]. The histogram shown in figure 6 confirms the overall completeness of the dataset, with the majority of the columns containing no missing data. A total of 16 columns had NaN or infinite values, but only 8 of these columns had NaN or infinite values accounting for more than 33% of their total data points.

Several limitations and minor data issues were acknowledged by Dijk et al., mostly relating to the accuracy of labels for certain edge-case attack scenarios. First, the authors highlight a challenge with "stepping-stone" attacks, i.e. when a red team attacker gains control of a blue team machine and then uses that machine to attack other blue systems. In the current ground truth labelling, such internal attacks are not marked as malicious because the labelling logic relied on identifying traffic involving red team IP addresses as malicious. Consequently, if an attacker pivoted and launched an attack from an already compromised blue host, the resulting network flows appear as legitimate blue-to-blue communication and were incorrectly labelled as benign. The impact is that a learning model trained on these labels might not recognise those internal attack patterns as malicious since they are in the benign category. [3]

Beyond labelling nuances, the dataset's quality review also notes some inherent complexity that does not detract from data integrity but is important for analysis. For instance, many hosts in the LS network have multiple network interfaces and IP addresses for redundancy or to sit in multiple network segments [3]. This can complicate tracing an attack end-to-end, since the same physical server might appear under different IPs in different parts of the traffic. To mitigate this, the dataset includes host metadata such as segment identifiers in addition to IPs so that analysts can potentially recognise when flows on different IPs involve the same machine. Another point is that some benign traffic in the exercise is generated by automated scoring bots: green team systems that continually check service health, by making periodic requests to services. These bots' traffic is labelled benign and included in the dataset; it is legitimate activity, though it might look artificial compared to human user traffic. The presence of scoring bot traffic is a feature of the exercise environment and could be considered an artifact, but it is clearly distinguishable by IP address and behaviour, and it provides useful information on service availability [3]. Overall, the LSPR23 dataset is considered high-quality and representative of a real, large-scale cyber defence scenario.

3.4 Data preparation

Data preparation involved feature selection, handling missing values, and performing transformations on the raw dataset. Feature selection aimed to remove redundant and irrelevant features to improve model generalisation and reduce complexity. Missing values were identified and systematically addressed through appropriate imputation techniques. The following subsections describe these processes in detail, outlining the rationale behind the inclusion and exclusion of specific features.

3.4.1 Feature selection

Initially, the dataset contained a total of 101 features, spanning various aspects of network traffic. To enhance model effectiveness and prevent overfitting, irrelevant or redundant features, particularly those tied explicitly to flow identification and network addressing, were excluded. For instance, features such as IP addresses were dropped to promote model generalisability beyond the specific dataset context, as their inclusion could negatively affect transferability and robustness over time. The selection process involved careful consideration of the nature and redundancy of each feature, balancing the risk of information loss against the benefits of dimensionality reduction. The table below summarises the included and excluded features.

Table 3. Summary of feature selection decisions

Category	Included/Total	Selected feat.	Excluded feat.
Flow identification & network address columns	7/16	SrcPort, DstPort, Protocol, service, conn_state, External_src, External_dst, L3/L4 Protocol	Flow ID, SrcIP, DstIP, Int/Ext Dst IP, Segment_src, Segment_dst, Expoid_src, Expoid_dst
Timestamp & duration features	1/7	Flow Duration	mTimestampStart, mTimestampLast, Flow Bytes/s, Flow Packets/s, Fwd Packets/s,Bwd Packets/s
Packet count & byte count features	16/16	Tot Fwd Pkts, Tot Bwd Pkts, Total Length of Fwd Packet, Total Length of Bwd Packet, Packet Length Min / Max / Mean / Std, Fwd Packet Length Min / Max / Mean / Std, Bwd Packet Length Min / Max / Mean / Std	

Inter-arrival time (IAT) features	14/14	Flow IAT Mean / Min / Max / Stddev, Fwd IAT Mean / Min / Max / Std, Fwd IAT Tot, Bwd IAT Mean / Min / Max / Std, Bwd IAT Tot	
TCP flags & protocol-specific indicators	10/13	FIN Flag Cnt, SYN Flag Cnt, RST Flag Cnt, ACK Flag Cnt, CWR Flag Cnt, ECE Flag Cnt, Fwd PSH flags, Bwd PSH flags, Fwd URG flags, Bwd URG flags	URG Flag Cnt, PSH Flag Cnt, Down/Up Ratio
Size, volume, and window features	20	Average Packet Size, Fwd Bytes/Bulk Avg, Fwd Packet/Bulk Avg, Fwd Bulk Rate Avg, Bwd Bytes/Bulk Avg, Bwd Packet/Bulk Avg, Bwd Bulk Rate Avg, Subflow Fwd Packets/Bytes, Subflow Bwd Packets/Bytes, Fwd Init Win Bytes, Bwd Init Win Bytes, Fwd Act Data Pkts, Fwd Seg Size Min	Fwd/Bwd Segment Size Avg
Active/idle times	8/8	Active Min / Mean / Max / Std, Idle Min / Mean / Max / Std	
Suricata alert features	0/4	-	SigID_revision, Category, Severity, Anomaly_event
Label columns	0/3	-	Label_src, Label_dst, Label

3.4.2 Flow identification features

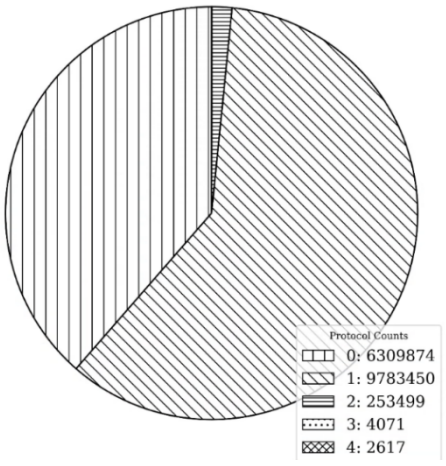

Figure 7. L3/L4 protocol value distribution in the LSPR23 dataset

The L3/L4 protocol feature, initially defined by Känzig et al. [63] and later adopted in the LSPR23 dataset by Dijk et al. [3] represents Layer 3 (Network Layer) and Layer 4 (Transport Layer) protocols within the OSI reference model [69]. This feature is supposed to distinguish among three primary protocols: TCP (coded as 0), UDP (coded as 1), and ICMP (coded as 2). However, exploratory analysis, shown on figure 7, uncovered unexpected protocol identifiers

(4071 instances coded as 3, and 2617 instances coded as 4). These are undocumented both in the works of Känzig et al. and Dijk et al, therefore these 2 values were treated as NaN values and encoded into “unknown”. These anomalies should be corrected in the most recent release of the LSPR24 dataset, eliminating the need for manual recoding.

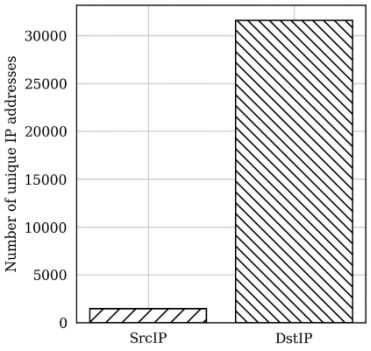

Figure 8. Distribution of source and destination IP addresses in the LSPR23 dataset

Regarding the inclusion of IP addresses as features, scholarly opinions diverge significantly. Dube [70] has criticised the CIC-IDS 2017 dataset for omitting IP addresses, highlighting their potential usefulness for accurately identifying malicious or misbehaving nodes within network environments when training ML models. Conversely, from the perspective of cybersecurity exercises and dataset generalisation, the inclusion of specific IP addresses raises concerns about limited applicability beyond the immediate context of a particular exercise, potentially reducing transferability and robustness of findings, e.g. beyond the LSPR23 exercise. In practical environments, IP addresses might also be dynamic or temporary. As can be seen from figure 8, there was 1466 unique source IP addresses in the dataset, but 31598 destination IP addresses. Including them as static features, for example by one-hot encoding, may gradually degrade model performance as the network structure changes over time, leading to frequent retraining. Relying on explicit IP addresses as identifying features may also facilitate adversarial evasion by permitting attackers to merely alter these addresses, thereby undermining the resilience and security efficacy of the detection models. Merkli et al. demonstrated that adversaries could compromise network flow classifiers by introducing minimal, strategically targeted perturbations to key traffic features, ultimately subverting decision boundaries and exposing inherent vulnerabilities in such feature-dependent systems [71].

Meliboev et al. [47] used IP header-based higher level data instead of specific IP addresses when training their CNN deep learning models, and both Gehri et al. [58] & Känzig et al. [63] also utilised an abstracted feature derived from IP addresses when training their RF machine learning models, Int/Ext Dst IP. This feature indicates whether destination IPs reside within internal or external network ranges. As the LSPR23 features separate External_src, External_dst features, these were picked as they also contain information about the source address.

Gehri et al. has noted that the LS network infrastructure changes annually [58]. Therefore, exercise-specific features Segment_src, Segment_dst, Expoid_src and Expoid_dst were also excluded to improve the model's ability to generalise.

3.4.3 Timestamp and duration features

Start and last timestamps were dropped because in an offline ML context, raw timestamps could inadvertently allow the model to cheat. For instance, if attacks primarily occurred after a specific timestamp, the model might wrongly infer a simplistic rule such as "timestamp > X implies malicious," limiting its ability to generalise beyond the specific exercise context. Instead, the flow duration was chosen as a more useful feature because it represents intrinsic temporal properties of network flows without relying on absolute timing. Flow duration thus provides a robust, transferable feature that enhances the model's ability to generalise effectively across different scenarios.

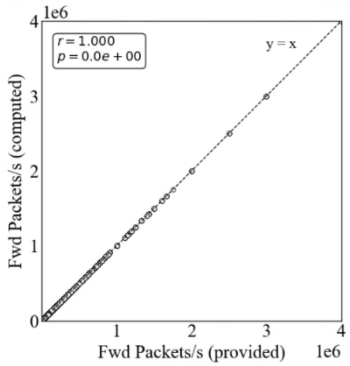

Figure 9. Comparison of calculated vs computed Fwd Packets/s feature

Flow Bytes/s, Flow Packets/s, Fwd Packets/s and Bwd Packets/s were dropped, as they consider no additional information. These can directly be derived from time and other features. For example, the Fwd Packets/s can be perfectly calculated from Flow duration and Tot Fwd Pkts features, as can be seen from figure 9.

3.4.4 Packet count and byte count features

All features in this category were considered valuable because they characterise the nature of network communication, including the number and sizes of packets in each direction. For instance, a benign flow might exhibit a balanced exchange consisting of a request followed by a response. In contrast, a malicious scan could involve only forward-directed packets without any responses, or a large data transfer may feature numerous large forward packets accompanied by fewer and smaller response packets.

3.4.5 Inter-arrival time features

Inter-arrival time (IAT) features were all selected, as they provide useful information about the arrival times and timing information of the packets, which can reveal patterns in traffic flows. However, Gehri et al. noted that if network infrastructure changes, the

time dependant features can lead to incorrect classifications [58]. For this reason, these features might be excluded from the training data when evaluating models on the 2024 dataset.

3.4.6 TCP flags and protocol-specific indicators

URG Flag Cnt and PSH Flag Cnt were dropped, as there are separate forwards and backwards flag counts for both which provide more information. Down/up ratio is also derived from total forward and total backward packets, so it is redundant.

3.4.7 Feature reduction

Disha and Waheed showed that while simpler algorithms such as DT benefited in performance from an optimised feature set, RF and MLP (simple neural net) didn't see notable improvements from the reduced feature amount, and in fact RF saw a slight performance decrease [42]. This is why for this thesis recursive feature elimination was not done, instead being limited to manual removal of strongly correlating and irrelevant features. Preliminary tests done both with the CNN and RF models showed that neither model performed better with a reduced feature set, even if trained on the 2023 dataset and validated on the 2024 dataset.

3.4.8 Data cleaning

In preparing the raw network-traffic dataset for analysis, a critical initial step involved addressing the issue of missing data. Missing data, represented in the dataset as NaN (Not a Number) or infinite values, arise naturally in real-world measurements, often due to sensor malfunctions, measurement limitations, incomplete logging, or conditions inherent to network traffic, such as flows with insufficient packets to compute certain statistics.

Table 4 shows that non-finite values are not uniformly distributed but cluster in a handful of columns. Four Suricata alert fields exhibit more than 95 % missing values; however, these variables had already been omitted from the training feature set on conceptual grounds. In addition, several network-address descriptors (Conn_state, Service, Segment_*, Expoid_*) are missing roughly one-third to two-thirds of all flows. Among the numerical features, the entire family of inter-arrival-time statistics (Flow IAT *) lacks data for about 18 % of the rows, and the rate-style metrics Flow Bytes/s and Flow Packets/s contain the same proportion of Inf values caused by zero-duration flows

or out of range readings. In total these 16 columns account for all the non-finite entries in the raw file, making them the primary targets for the imputation strategies discussed below.

Table 4. LSPR23 features with non-finite values

Feature	Category	dtype	NaN values	Inf values	Nonfinite %
Anomaly_event	Suricata alert features	Float	16277465	0	99.535%
Category	Suricata alert features	Float	15639501	0	95.634%
Severity	Suricata alert features	Float	15639501	0	95.634%
SigID revision	Suricata alert features	Float	15639501	0	95.634%
Conn_state	Flow identification & network address columns	Category	10055421	0	61.488%
Service	Flow identification & network address columns	Category	7218172	0	44.138%
Segment_dst	Flow identification & network address columns	Category	5531411	0	33.824%
Expoid_dst	Flow identification & network address columns	Category	5531411	0	33.824%
Segment_src	Flow identification & network address columns	Category	3859355	0	23.600%
Expoid_src	Flow identification & network address columns	Category	3859355	0	23.600%
Flow IAT Min	Inter-arrival time (IAT) features	Float	3042190	0	18.603%
Flow IAT Mean	Inter-arrival time (IAT) features	Float	3042190	0	18.603%
Flow IAT Max	Inter-arrival time (IAT) features	Float	3042190	0	18.603%
Flow IAT stddev	Inter-arrival time (IAT) features	Float	3042190	0	18.603%
Flow Bytes/s	Timestamp & duration features	Float	1205	3052847	18.675%
Flow Packes/s	Timestamp & duration features	Float	0	3054052	18.675%

Handling these missing values is essential because most machine learning algorithms, including most gradient-boosted decision trees, random forests, and neural networks, cannot directly process NaN values. If left untreated, missing data either causes errors during model training or significantly degrades model performance. Therefore, a process known as imputation was employed. Imputation refers to the substitution of missing data points with reasonable estimates, derived from the observed dataset, thus enabling complete and analysable data. Two separate imputation strategies were adopted, corresponding to the types of features (numerical and categorical):

The categorical features included connection states (Conn_state), protocols (L3/L4, Protocol), and port numbers (SrcPort, DstPort). For these features, the presence of missing data was addressed by explicitly inserting a new categorical level, labelled "unknown". Introducing a dedicated category for missing values is considered best practice for categorical data, particularly when the absence of data itself may carry meaningful patterns or predictive significance [72]. Unlike replacing missing values with the most frequently occurring category (mode), assigning a dedicated category avoids introducing biases by clearly differentiating missing data from valid observations. It preserves the integrity of the original distribution and allows machine learning models to explicitly learn from the absence of information rather than implicitly merging missing data into an existing category.

Numerical features in this dataset primarily represent measurements related to network-flow characteristics such as packet sizes, byte counts, and inter-arrival times (Flow IAT features). These numeric features exhibited missingness typically due to insufficient number of packets within certain flows, making it impossible to compute statistics like mean or standard deviation. To handle these gaps, a median-imputation strategy was chosen. Median imputation replaces missing numeric values with the median of the observed data, ensuring robustness against outliers which are common in network traffic. To ensure unbiased and representative median estimates across the large dataset of 16 million flows, the median was computed from a randomly selected subset of about four million observations, including about ~25% of the values in the dataset. Random sampling was used to prevent potential biases associated with the dataset's temporal ordering, ensuring the medians reflect the overall distribution rather than any localised or temporal effects.

The actual imputation of numeric values was performed using a chunk-based approach due to the large dataset size. Rather than loading the entire dataset into memory simultaneously, data was processed incrementally in smaller chunks, each containing two million rows. This method optimised memory usage, computational speed, and efficiency. Once the median was calculated from the random subset, it was consistently applied across all chunks, effectively filling in all missing numeric values in a computationally manageable manner.

Following imputation, categorical features were further processed using embeddings. Embeddings are dense vector representations of categorical variables, transforming sparse, discrete categories into continuous numerical vectors suitable for neural networks. Embeddings significantly enhance model performance by capturing semantic relationships between categories and reducing dimensionality compared to traditional one-hot encoding.

To determine the optimal number of dimensions of each embedding vector, the heuristic from the FastAI library, commonly known as the "fastai tabular rule", was employed [73]. This rule calculates the embedding size using the formula:

$$Embedding\_size = min(600, \lfloor 1.6 \times cardinality^{0.56} \rfloor)$$

Here *cardinality* represents the number of distinct categories present in each categorical feature. The embedding size increases sub-linearly with the number of categories, ensuring that features with a very large number of unique values, such as port numbers, do not result in prohibitively large embedding matrices. The upper bound of 600 dimensions prevents excessively large embeddings, thus balancing representational power and computational feasibility. Using this approach, each categorical feature was embedded into a numerical space optimised for capturing latent structures in the data, significantly aiding the predictive capabilities of the neural network models used later in this analysis.

For illustration, the `SrcPort` column in the training split alone contained 61 079 distinct values. A naïve one-hot encoding would expand this single attribute into over 60 thousand binary columns, ballooning the input matrix, wasting memory with ~99 % zeros, and pushing the number of trainable weights well beyond practical limits. With the fastai rule, the same column is mapped to a *600-dimensional* embedding vector, as the value computed by is capped at 600, shrinking the representation by two orders of magnitude while still allowing the network to learn meaningful similarities between ports.

3.4.9 Dataset partitioning strategy

The partitioning of datasets into training and testing subsets is an important methodological step, significantly impacting the reliability and validity of experimental outcomes. Traditional machine learning research typically employs randomised splits,

with an 80/20 ratio widely recognised as an industry standard, balancing ample training data with sufficient testing coverage for unbiased performance estimation [59]. Nonetheless, recent research highlights that this random approach may be inadequate for evaluating models in cybersecurity contexts, where temporal and spatial characteristics of network traffic are highly relevant [74], [75].

Ring et al. [75] emphasise that conventional randomised splits or k-fold cross-validation strategies can inadvertently introduce data leakage, undermining realistic assessments of model generalisability. They suggest either a temporal split, where training data precedes test data chronologically, or a host-based split, where subsets are separated by source IP addresses. These splits would preserve realistic network structure and correct temporal patterns in both training and test subsets. Similarly, Vaarandi et al. [74] advocate for chronological splits as they better reflect the evolving nature of cyber threats over time. By training models exclusively on earlier network flows, temporal splits evaluate the model's capacity to generalise effectively to future, potentially unseen attack patterns, a condition closely resembling real-world intrusion detection scenarios.

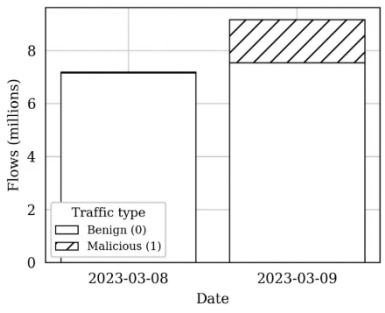

Figure 10. Distribution of malicious traffic over the exercise duration

Initially, a time-based partitioning strategy analogous to that employed by Vaarandi et al. was considered, dividing the LSPR23 dataset chronologically into distinct training and testing intervals. However, preliminary analysis revealed significant drawbacks: the dataset exhibited marked asymmetry between days, with the first day containing minimal malicious activity and the majority of attacks occurring on the second day, as can be seen from figure 10. This uneven distribution would severely limit model exposure to malicious patterns in the training phase, resulting in poor generalisation during evaluation, and thus, this chronological approach was ultimately deemed unsuitable.

An alternative host-based partitioning, recommended by Ring et al. [75] was also assessed, splitting the dataset based on IP addresses. In theory, such an approach could prevent leakage of information between training and test sets by ensuring distinct network hosts and attack targets. However, as discussed earlier, the Locked Shields exercise features dynamically changing host roles and IP addresses assigned to multiple interfaces, significantly complicating host-based separation. Additionally, splitting by IP addresses

risked creating subsets that did not reflect realistic traffic distribution, as certain IP addresses were involved disproportionately in either benign or malicious activities, leading to artificially biased subsets.

Considering these challenges, the final solution was a straightforward, randomised 80/20 split. This widely adopted method ensured a representative distribution of malicious and benign flows across both subsets, enabling robust and unbiased evaluation of the models' predictive performance. To minimise concerns regarding potential data leakage or temporal bias, careful cross-checking confirmed that malicious traffic and benign traffic characteristics remained comparably distributed between training and testing datasets. Furthermore, the large scale of the LSPR23 dataset significantly mitigated the risk of inadvertent biases typically associated with smaller datasets.

While advanced split strategies proposed by Ring et al. [75] and Vaarandi et al. [74] are theoretically superior in cybersecurity contexts, practical constraints specific to the LSPR23 dataset made these approaches infeasible in this case. Thus, the adopted randomised 80/20 split represents the most pragmatic and methodologically sound option, providing a balanced and robust foundation for subsequent comparative analyses of traditional and deep learning models.

3.4.10 Data balancing

Class imbalance is a common challenge in machine learning-based intrusion detection, often resulting in biased models that perform poorly when identifying minority class instances, typically malicious traffic [57]. The LSPR23 dataset, although relatively balanced compared to legacy intrusion detection datasets, still exhibited a notable imbalance, with benign network flows outnumbering malicious flows at an approximate ratio of 10:1. Given this class distribution, careful consideration was given to implementing class balancing techniques to mitigate potential bias towards the majority (benign) class.

A comprehensive review of the literature (see chapter 2) indicated that Synthetic Minority Over-sampling Technique (SMOTE), introduced by Chawla et al. [57], is widely recognised as an effective method to address class imbalance in cybersecurity datasets. SMOTE combines oversampling of minority class instances by synthetically generating new examples with undersampling of majority class instances, aiming to improve the

classifier's sensitivity to minority class predictions without significantly compromising overall accuracy. Many prior studies employing similar intrusion detection datasets have effectively utilised SMOTE to improve classifier performance [40], [44], [47], [48].

However, preliminary experimentation with the LSPR23 dataset revealed that the employed traditional and deep learning models achieved strong predictive performance without applying explicit data balancing. Despite the inherent imbalance, the selected models demonstrated high classification accuracy, satisfactory F1-scores, and robust ROC-AUC metrics, indicating minimal bias towards the majority class. The results suggested that the models successfully captured the distinctive patterns characterising malicious traffic within the extensive and detailed feature set provided by the LSPR23 dataset, diminishing the necessity for additional balancing techniques such as SMOTE.

Given these findings, the decision was made not to apply SMOTE or any other data balancing method for the final experiments. This decision simplified the preprocessing pipeline and avoided potential complications introduced by synthetic data generation or random undersampling, such as the risk of overfitting, loss of genuine data variability, or reduced generalisability [57]. Thus, the dataset was employed in its original distribution, and any residual imbalance was accounted for by carefully selecting and tuning model hyperparameters, as well as performing rigorous model validation to confirm that minority class performance was maintained at a high level.

The decision to forego explicit balancing is discussed further in the evaluation phase (Section 4), where detailed comparative results illustrate the effectiveness of both traditional machine learning and deep learning techniques in accurately classifying malicious traffic despite inherent class imbalance.

3.5 Machine learning model: Random Forest

The traditional machine learning approach adopted in this research utilises the Random Forest algorithm, which has consistently demonstrated robust performance in network intrusion detection tasks across diverse cybersecurity datasets, as indicated by the literature review in Chapter 2. RF was selected primarily due to its inherent strengths: excellent classification accuracy, interpretability through feature importance metrics, resilience to overfitting, and strong capability for handling high-dimensional and

heterogeneous feature spaces, which are characteristics particularly relevant to network intrusion datasets, such as LSPR23.

Specifically, the RF implementation used for this thesis leverages GPU acceleration via the cuML library, significantly enhancing training and inference speed compared to traditional CPU-based implementations such as those provided by scikit-learn. GPU acceleration, as quantitatively demonstrated in the results section, provides crucial efficiency improvements, enabling faster analysis and classification of network traffic flows, which is essential for practical deployment in real-time or near-real-time cybersecurity scenarios. Although a detailed computational comparison between cuML's GPU-accelerated Random Forest and scikit-learn's CPU-based implementation is provided in Chapter 4, the choice of cuML here directly aligns with operational requirements for timely intrusion detection.

In contrast to neural network-based approaches, Random Forest also offers additional practical advantages, notably its transparency and interpretability. RF models inherently support explainability through built-in metrics such as Gini importance scores, which can provide cybersecurity professionals with actionable insights into feature significance and model decision-making processes. Moreover, because the model aggregates many decorrelated decision trees, its predictions are highly stable. The stable and reproducible predictions are an essential attribute for reliable cybersecurity operations and audits. When tested across 10 independent runs, the classifier’s performance metrics (F1, accuracy, precision, recall) were extremely stable, with standard deviations on the order of 10^{-4} or smaller. Further details are reported in section 4.

The Random Forest hyperparameters were selected based on optimisation results previously established by Känzig et al. [63], who optimised these parameters specifically for the detection of malicious network activity using similar datasets, for efficient yet accurate detection. The suitability and robustness of the optimised hyperparameters have been independently validated in subsequent studies employing comparable network intrusion datasets, notably by Merkli et al. [71], Gehri et al. [58], and Dijk et al. [3].

The consistent and successful deployment of these hyperparameters across multiple recent studies conducted between 2023 and 2024 demonstrates their transferability and effectiveness for comparable intrusion detection tasks. Leveraging previously validated

hyperparameter values mitigates the need for extensive experimental tuning such as grid search. This accelerated the experimental workflow and allocated more time to focus on model performance and implications. Table 5 summarises the configuration parameters used in this research:

Table 5. Random forest model hyperparameters

Device	N-estimators	Max depth	N features in	Bootstrap	Outputs
GPU	128	10	72	True	[0,1]

The chosen configuration effectively balances complexity and computational demands: using 128 estimators provides diversity within the ensemble, ensuring reliable classification performance without imposing high computational costs. Setting the maximum tree depth to 10 helps control the model complexity and mitigates potential overfitting, thus maintaining generalisability across varying intrusion scenarios [63]. All 72 features selected during preprocessing (Section 3.4) were fed into the model, allowing the RF to utilise the rich feature space provided by the LSPR23 dataset. Additionally, enabling bootstrap sampling further increases tree diversity, leading to enhanced stability and accuracy. Bootstrap sampling is a statistical resampling technique that generates new training subsets by randomly selecting observations from the original dataset with replacement, thereby ensuring each tree within the Random Forest is trained on slightly different data and reducing correlations between individual tree predictions [76]. The model outputs probabilistic predictions in the range of [0, 1], which are subsequently converted into binary classification outcomes (malicious or benign) using a sigmoid decision threshold.

3.6 Deep learning model: 1D Convolutional neural network

The deep learning approach employed in this research consists of a custom-designed one-dimensional Convolutional Neural Network, tailored specifically for detecting malicious network traffic within the LSPR23 dataset. The model's architecture was inspired by prior state-of-the-art research on applying 1D CNNs for network intrusion detection tasks, notably the works of Kilichev and Kim in 2023 on hyperparameter optimisation for 1D CNNs [77], and Singh et al. in 2021 who developed a robust 1D CNN for the classification and analysis of network attacks [78]. Building upon these foundational

studies, the CNN architecture has been adapted and optimised specifically for the characteristics and scale of the LSPR23 dataset. The detailed structural breakdown of the CNN and associated parameter counts is summarised in table 6.

Numeric and embedded categorical features were combined into a unified input vector of length 953, subsequently passed through two convolutional layers. The first convolutional layer transforms the input to 16 channels, and the second further increases it to 32 channels, employing kernels with a size of 3 with padding. This captures both localised and broader contextual feature interactions across the input features. Fully connected layers combined with dropout with probability of 0.5 and ReLU activations are utilised downstream, concluding with in a sigmoid-activated neuron for binary classification: malicious or benign. Training made use of an early stopping criterion based on the validation F1-score to mitigate overfitting and ensure generalisability to novel network conditions.

3.6.1 Input features and feature encoding

A one-dimensional convolutional neural network was specifically chosen over a two-dimensional CNN due to the inherent nature of the input data. Unlike image data, which inherently contains two spatial dimensions, the height and width, and thus naturally aligns with a 2D CNN, the structured tabular network traffic data used here is inherently sequential and one-dimensional. Each network flow is represented by a flat numeric vector rather than a spatial matrix, making 1D convolutions a logical choice.

The network structure integrates both numeric and categorical data types, latter of which require encoding to transform them from textual inputs into numeric inputs for the machine learning models. In the example code provided with the LSPR23 dataset, Dijk et al. [79] used SHA256 hashing for encoding the categorical features. This approach was not considered as the structured categorical data such as ports, protocols, and service types used here benefit more effectively from learned embeddings. Unlike hashing, embeddings capture latent semantic relationships inherent in the data, thus providing richer feature representations for the neural network model [80].

Kilichev and Kim instead used 1-hot encoding for handling the categorical features in their model, producing 197 novel features from just 3 categorical variables [77]. This approach was also unsuitable for the LSPR23 dataset, as some of the categorical variables

were high cardinality, e.g. as the src port with over 60 thousand distinct values. Using 1-hot encoding would have resulted in a sparse and high-dimensional input space, leading to computationally inefficient CNN, with an increased risk of overfitting.

As a novel contribution, dedicated embedding layers for categorical feature transformation were used. Embedding categorical variables has been demonstrated as an effective technique in deep learning for tabular data, capturing latent semantic relationships and improving predictive performance compared to traditional encoding methods [80]. These embedding layers convert categorical features into dense, low-dimensional numeric representations, enabling the CNN to capture latent semantic relationships effectively. Eight categorical features were embedded with heuristically determined dimensions based on feature cardinality: Source Port, Destination Port, Protocol, L3/L4 Protocol, Service, Connection State, External Source, and External Destination.

Each categorical feature is represented as distinct embedding layers labelled from 2-1 to 2-8 in the architecture overview table (Table 6) below. Embedding 2-1 corresponds to the Source Port (600-dimensional), embedding 2-2 to the Destination Port (256-dimensional), embedding 2-3 to the Protocol (5-dimensional), and embedding 2-4 to the L3/L4 Protocol (4-dimensional). Further embeddings include 2-5 representing the Service (11-dimensional), 2-6 corresponding to Connection State (7-dimensional), 2-7 for External Source (3-dimensional), and finally, embedding 2-8 representing External Destination (3-dimensional). These embedding sizes were heuristically determined using a rule based on feature cardinality, as recommended in the FastAI library [73], balancing representational power with computational efficiency. By assigning each categorical variable to an appropriately sized embedding space, the model is capable of effectively capturing semantic relationships inherent in network traffic data, substantially improving predictive performance while minimising computational overhead compared to traditional categorical encoding approaches, such as 1-hot.

3.6.2 Model architecture

Table 6. Detailed CNN model architecture

Layer (depth-idx)	Input Shape	Output Shape	Param #	Mult-Adds
CNN	[1, 64]	[1]	--	--
├─ModuleDict: 1-1	--	--	--	--
│ └─Embedding: 2-1	[1]		[1, 600]	36,648,000
│ └─Embedding: 2-2	[1]		[1, 256]	2,202,624
│ └─Embedding: 2-3	[1]		[1, 5]	45
│ └─Embedding: 2-4	[1]		[1, 4]	24
│ └─Embedding: 2-5	[1]		[1, 11]	330
│ └─Embedding: 2-6	[1]		[1, 7]	98
│ └─Embedding: 2-7	[1]		[1, 3]	9
│ └─Embedding: 2-8	[1]		[1, 3]	9
├─Conv1d: 1-2	[1, 1, 953]	[1, 16, 953]	[1, 16, 953]	64
├─ReLU: 1-3	[1, 16, 953]	[1, 16, 953]	[1, 16, 953]	--
├─Conv1d: 1-4	[1, 16, 953]	[1, 32, 953]	[1, 32, 953]	1,568
├─ReLU: 1-5	[1, 32, 953]	[1, 32, 953]	[1, 32, 953]	--
├─Dropout: 1-6	[1, 30496]	[1, 30496]	[1, 30496]	--
├─Linear: 1-7	[1, 30496]	[1, 64]	[1, 64]	1,951,808
├─ReLU: 1-8	[1, 64]	[1, 64]	[1, 64]	--
├─Dropout: 1-9	[1, 64]	[1, 64]	[1, 64]	--
├─Linear: 1-10	[1, 64]	[1, 1]	[1, 1]	65
Total params: 40,804,644				

Two Conv1D layers are utilised in the model, as this depth was found sufficient for feature extraction in malicious traffic by both Singh et al. and Kilichev & Kim [77], [78]. Each Conv1D layer uses the ReLU activation function, following the practice of used in the previously mentioned 2 studies. For the convolutional layers, Singh et al. adopted a small kernel size of 1 in their 1D-CNN, effectively treating each input feature in isolation [78], whereas Kilichev and Kim recommended a larger kernel size of 9 to capture broader interactions among network features [77]. For this CNN, a kernel size of 3 was selected as a balanced intermediate choice. This kernel size is large enough to detect meaningful local feature interactions inherent in network traffic data, but small enough to maintain computational efficiency, avoid overfitting, and retain flexibility in modelling more complex patterns through multiple convolutional layers.

Following the architectural approach established by Singh et al [78], the model includes a dropout layer following the convolutional layers, which serves to mitigate overfitting.

While Singh et al [78] used a single fully connected (FC) layer, Kilichev and Kim [77] identified two as the optimal amount of FC layers after their hyperparameter optimisation. This model utilises 2 linear FC layers with 64 neurons each. The output layer consists of a single neuron with a sigmoid activation, this is in line with the approach employed by Kilichev and Kim for binary classification [77].

The model's total parameter count of approximately 40.8 million is predominantly due to the embedding layers used for categorical features, particularly due to high-cardinality categorical variables such as Source and Destination Ports, which require substantial embedding dimensions to capture meaningful representations. This CNN architecture, implemented using PyTorch, employed binary cross-entropy loss optimised via the Adam optimiser with a learning rate of 0.001, following Singh et al. [78]. This decision is further supported by Kilichev and Kim's optimisation results, which also converged on learning rates on the order of 10^{-3} for optimal performance [77]. Additionally Kilichev and Kim [77] tested their optimised hyperparameters across multiple distinct network intrusion detection datasets, demonstrating the broader applicability and robustness of these architectural choices beyond the specific context of the LSPR23 dataset.

3.7 Evaluation

The evaluation phase of this research systematically assesses the classification performance of traditional machine learning and deep learning models for network intrusion detection. To ensure comprehensive and unbiased comparisons, multiple evaluation metrics have been selected based on their appropriateness for cybersecurity applications, particularly in the context of imbalanced datasets. Additionally, computational performance metrics have been incorporated to quantify the practical implications of model deployment.

3.7.1 Evaluation metrics and methodology

The models were evaluated using a set of performance metrics widely utilised in intrusion detection literature. The metrics selected for assessing classification performance were accuracy, precision, recall, F1-score, and confusion matrix analysis. Accuracy generally refers to the proportion of correctly classified instances. Previously in this thesis, accuracy has been used broadly, including references to both classification and detection contexts. To avoid ambiguity, from this point forward, the term accuracy specifically denotes

classification accuracy, formally defined with the formula provided in Table 7. This is distinct from detection accuracy, a broader concept specifically referring to the correct identification of threats, often represented through metrics exclusively focused on threat detection, such as precision and recall. However, accuracy alone may not sufficiently describe model performance in scenarios characterised by class imbalance, which is prevalent in cybersecurity data. Therefore, precision and recall metrics were included to offer deeper insight into the models' predictive capabilities with respect to false positives and false negatives. The F1-score was specifically included as a balanced metric that harmonises precision and recall into a single indicative value. The confusion matrix complements these metrics by explicitly representing counts of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), thus enabling nuanced analysis of the types of errors models are prone to make. Table 7 provides detailed definitions and calculation formulas of these evaluation metrics:

Table 7. Definitions and formulas for the evaluation metrics

Metric	**Formula**	**Explanation**
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Proportion of correctly classified instances
Precision	$\frac{TP}{TP + FP}$	Proportion of positive identifications that are correct
Recall	$\frac{TP}{TP + FN}$	Proportion of actual positives correctly identified by the model
F1-score	$\frac{2 \times Precision \times Recall}{Precision + Recall}$	Harmonic mean of precision and recall; balances both metrics
Confusion Matrix	n/a	Matrix summarising TP, FP, TN, FN to detail model classification errors

These metrics were computed consistently across all ML and DL models, using standard libraries such as scikit-learn, thereby ensuring reproducibility and consistency in evaluation.

3.7.2 Computation performance metrics

In addition to assessing classification effectiveness, computational efficiency metrics were measured to address practical considerations relevant to real-world deployments,

particularly in environments with limited resources. Specifically, training time, inference speed, memory usage, and power consumption were monitored and reported.

For measuring training and inference time, python time library was used. Training time, defined as the duration required to fully train a model to convergence or until CNNs early stopping criteria are met, was recorded to reflect computational resource demands. Inference speed was measured by calculating the time taken to classify batches of test data, thus providing an indicator of suitability for deployment in scenarios requiring timely response, such as in real-time NIDS usage during the exercise.

Memory usage was measured by recording the peak GPU memory allocated during training phases. This metric is relevant because graphics card memory constraints often limit model deployment in real-world cybersecurity environments. The peak memory allocation was measured using PyTorch's built-in CUDA functionality.

Power consumption was measured using the codecarbon library which accurately captures the CPU GPU energy usage (in kilowatt-hours, kWh) during both model training and inference phases. Internally, codecarbon leverages the pyNVML library, an established interface providing direct access to power consumption metrics from Nvidia graphics cards [81], [82]. The CPU energy usage was captured by the Intel Power Gadget tool, which presents real-time data for Intel processors [82], [83]. Since codecarbon currently lacks reliable methods for accurately measuring system memory power consumption [82], the estimated memory-related energy metrics were excluded from this evaluation to maintain the integrity of reported results.

To provide consistency, each experiment was repeated 10 times under identical conditions, with no other computationally intensive processes active during the experiments. The processor was consistently operated at Intel's default optimised power limit and the GPU was maintained at its nominal TDP. The experimental setup, described in detail in section 3.8.1, provided a controlled and standardised computational environment to ensure comparability and reproducibility. The average values for the power usage and time were calculated over the 10 runs. Such metrics provide practical insights into sustainability and operational costs associated with different modelling approaches.

3.7.3 Evaluation procedure

All models were evaluated using a consistent experimental procedure. Initially, the dataset was partitioned using an 80/20 randomised train-test split, ensuring representative distributions of benign and malicious instances across both subsets. Data preprocessing procedures, including median imputation for missing numerical values, embedding of categorical variables using the FastAI heuristic, and standardisation of numeric features based solely on the training set, were uniformly applied to maintain consistency across models.

Model training procedures varied depending on the nature of the respective machine learning methods. For the deep learning approach, specifically the convolutional neural network, the binary cross-entropy loss function was utilised, optimised via the Adam algorithm with a learning rate of 0.001. To reduce the risk of overfitting, an early stopping criterion was implemented, monitoring improvements in the validation F1-score, with training halted if no improvement occurred over three consecutive epochs. The CNN model training used a batch size of 1024, which provided a balance between computational efficiency and convergence stability.

In contrast, the traditional machine learning method, specifically the Random Forest classifier, followed a different training procedure. Hyperparameters including the number of trees (estimators) and tree depth, were tuned to the values found in the literature. Unlike the CNN, this training process did not involve iterative optimisation or early stopping criteria. Instead, the Random Forest model internally constructs an ensemble of decision trees, combining their predictions automatically to produce the final classification. All hyperparameters were selected to optimise classification performance while ensuring computational feasibility and avoiding overfitting.

To account for inherent variability in stochastic training processes, a total of 10 independent training runs were conducted, each initialised with distinct random seeds. This approach enabled the calculation of average performance metrics and reduced variance attributable to random initialisation conditions.

3.8 Deployment and reproducibility

Although the models evaluated in this thesis are not intended for immediate deployment into a production environment, the CRISP-DM framework highlights the necessity of carefully documenting deployment considerations and ensuring the reproducibility of conducted experiments. Consequently, this section outlines the experimental environment in detail, describes hardware and software configurations employed, discusses reproducibility aspects, and explores considerations regarding the scalability and potential deployment of the evaluated models.

3.8.1 Experimental setup

As detailed previously, experiments conducted throughout this thesis utilised a hardware configuration selected to represent realistic computational constraints common in cybersecurity contexts. Specifically, the experimental hardware included an Intel Core i5-13500 CPU with 14 cores, operated at Intel's default optimised power limit of 65 watts [84], paired with an Nvidia RTX 3090 GPU maintained at its nominal TDP of 350 watts [85]. Additionally, the system was equipped with 64 GB of DDR4 RAM at 3200 MHz. These hardware specifications, chosen to closely mirror typical operational environments, supported accurate power and resource usage measurements using the codecarbon utility. This approach facilitated practical assessments of model feasibility, including evaluations of power consumption, computational efficiency, and potential limitations in realistic cybersecurity scenarios.

Software configurations involved Python version 3.12.3 within an Ubuntu 24.02 environment, operating under Windows Subsystem for Linux (WSL) 2.0. The use of WSL 2.0 was specifically motivated by compatibility requirements related to the cuML library, which provides GPU-accelerated machine learning algorithms but does not offer native Windows support. The underlying host system was version 24H2 of Windows 11 Education, chosen for its enhanced support of virtualisation technologies and improved integration with WSL 2.0. This configuration enabled seamless interoperability between Linux-based GPU-accelerated computing environments and the Windows host system, thereby providing a unified and efficient enough computational setup for model training and evaluation.

To enable GPU acceleration uniformly across the chosen software stack, the CUDA toolkit version 12.8 was installed. This specific version was selected because it enabled compatibility with both PyTorch (version 2.8.0 nightly preview) and cuML (version 25.4) in the same environment, thereby streamlining the development and minimising software conflicts. In addition to cuML and PyTorch, the scikit-learn library, version 1.6.1, was employed primarily for implementing and evaluating the machine learning models, while other supporting libraries such as pandas, numpy, and codecarbon were utilised extensively throughout data processing and evaluation procedures.

3.8.2 Scalability and deployment considerations

Given that intrusion detection systems often need to handle large volumes of network data in near-real-time conditions, the scalability and efficiency of evaluated models must be carefully considered. Traditional machine learning algorithms such as Random Forests have been shown to scale effectively using parallel processing techniques inherent to modern CPUs, as they naturally lend themselves to distributed training and inference due to their ensemble structure. Rane et al. [56] highlight that Random Forest models exhibit favourable scalability and can be effectively parallelised across multiple processor cores without substantial algorithmic complexity. However, the efficiency of these models tends to plateau as data volume significantly increases, potentially necessitating more complex infrastructure.

In contrast, deep learning models, particularly convolutional neural networks, have demonstrated considerable scalability through GPU acceleration. Frameworks like PyTorch and TensorFlow leverage CUDA-based parallel computations, enabling efficient training and inference on large-scale data, at the cost of increased memory usage [56]. Moreover, CNNs inherently support batch processing, further enhancing their suitability for high-throughput, real-time network monitoring scenarios. Nonetheless, the computational requirements for GPU-based deep learning methods introduce higher power and infrastructure demands compared to traditional models.

Regarding distributed computing possibilities, both traditional and deep learning models support varying degrees of distributed implementation. Traditional ML methods can utilise parallel processing environments easily accessible in distributed computing frameworks such as Apache Spark [86]. Conversely, deep learning methods primarily benefit from GPU-centric distributed architectures or specialised inference servers such

as NVIDIA Triton Server [87], which allows efficient scaling of CNN inference across multiple GPUs and server nodes.

3.8.3 Explainability

Explainability in machine learning, particularly within intrusion detection systems, is a critical factor for trust, transparency, and regulatory compliance. Burkart and Huber [88] outline several approaches to achieve explainability, broadly categorising them into interpretable models, surrogate model fitting, and direct explanation generation methods. Interpretable models, such as decision trees and linear regression, provide transparency by design, allowing direct insight into the decision-making process.

Random Forest models, though more complex than single decision trees, still offer global explainability through techniques like feature importance metrics and partial dependence plots. Feature importance metrics quantify the contribution of each feature towards the predictive accuracy of the model, thereby identifying influential factors within the dataset. Partial dependence plots visually represent the dependency of the model predictions on selected features, helping stakeholders interpret model behaviour in specific conditions and enhance trust in the predictive outcomes. [89] Although Random Forest models excel at providing global explainability, their inherent complexity makes them limited in offering detailed local explanations for individual predictions. As highlighted by Plumb et al, specialised interpretability methods such as MAPLE, which employ local linear approximations derived from Random Forest neighbourhoods, are necessary to effectively generate accurate local explanations and thus address this limitation [90].

Neural networks, despite their high complexity, offer several interpretability techniques such as feature visualisation, saliency maps, and Grad-CAM (Gradient-weighted Class Activation Mapping). Burkart and Huber highlight these methods as critical in explaining CNN behaviour by visualising activations and heatmaps, effectively illustrating the specific areas within input data that contribute significantly to model decisions [88]. These visualisation methods are instrumental in understanding CNN predictions, thus enhancing transparency and facilitating debugging, trust-building, and compliance with regulatory requirements, such as the EU AI act [91].

3.8.4 Deployment plan

While not directly within the scope of this thesis, a high-level deployment strategy could involve initially integrating the best-performing intrusion detection models within existing cybersecurity infrastructure as auxiliary threat-detection tools, complementing established signature-based systems. Treating ML-based intrusion detection as an auxiliary tool at first mitigates potential risks associated with limited interpretability and vulnerability to adversarial evasion, which could undermine trust and operational reliability if relied upon exclusively. Once these ML models have demonstrated sustained accuracy and robustness under real-world conditions, their role could be progressively expanded, potentially evolving into primary detection mechanism. This evolution towards primary reliance on ML-based detection systems is supported by recent research; for example, Dijk et al. [3] demonstrated that machine learning-based approaches can significantly outperform traditional signature-based solutions such as Suricata in accurately identifying network threats, underscoring the potential benefits of fully integrating these advanced models into operational cybersecurity frameworks.

Deployment would ideally leverage containerisation technologies such as Docker to simplify environment management and facilitate seamless scaling across distributed network segments. Continuous monitoring and logging infrastructure, potentially employing frameworks like Prometheus and Grafana, could enable ongoing performance and resource-usage tracking. Periodic retraining and model updates, facilitated by automation pipelines, would help maintain long-term effectiveness against evolving cyber threats.

4 Results

This chapter presents a detailed comparative evaluation of traditional machine learning (Random Forest) and deep learning (1D Convolutional Neural Network) approaches applied to network intrusion detection using the LSPR23 and LSPR24 datasets. The analysis is structured into three primary sections: firstly, a performance and computational efficiency comparison between GPU-accelerated (cuML) and CPU-based (scikit-learn) implementations of the Random Forest model; secondly, a thorough comparison of the Random Forest and CNN models trained and evaluated on the LSPR23 dataset; and thirdly, a cross-dataset validation assessing the robustness and generalisation capabilities of both models on the subsequent year's LSPR24 dataset. Each section includes explicit metrics, confusion matrices, computational performance comparisons, and discussions of key findings to comprehensively highlight the practical trade-offs, strengths, and limitations associated with each modelling approach.

4.1 Comparison of CPU-based and GPU-based random forest implementations

This section presents a comparative analysis of the computational efficiency and performance of two implementations of the Random Forest algorithm: a GPU-accelerated version utilising cuML, and a conventional CPU-based implementation utilising scikit-learn. The comparison is based on identical default hyperparameters (100 estimators) trained and evaluated on the LSPR23 dataset with a reduced feature set. Although this comparison utilised a preliminary feature selection resulting in fewer features than the final optimised set, the relative computational differences observed between the GPU and CPU implementations remain relevant. This is because the primary aim of this comparison is to illustrate the efficiency and resource advantages inherent in GPU

acceleration, rather than the absolute predictive performance which is addressed separately in subsequent sections using the fully optimised feature set.

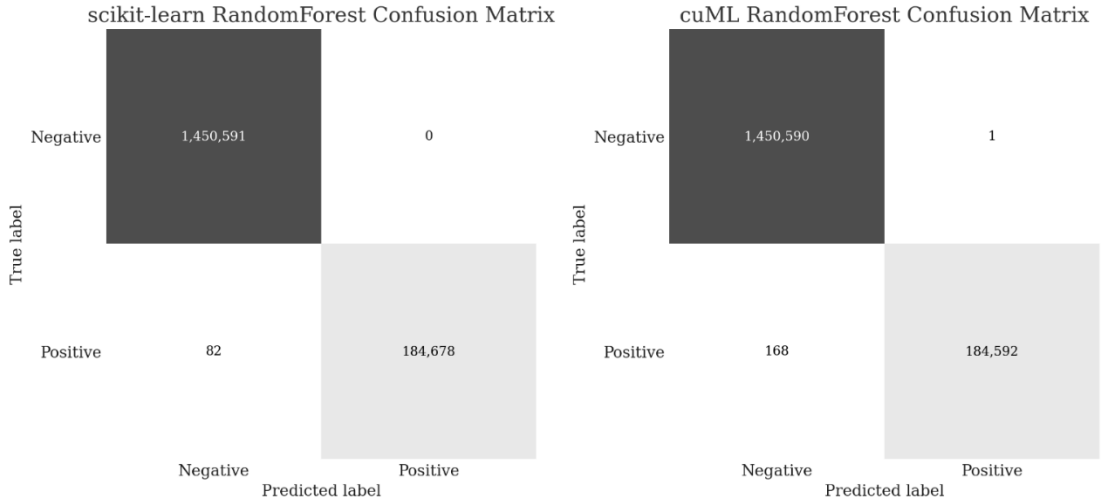

Figure 11. Confusion matrices for GPU-based and CPU-based RF models

The classification performance of both implementations is summarised in Figure 11, which shows the confusion matrices obtained from each model. Both the GPU-based and CPU-based models demonstrated excellent predictive performance, but subtle differences were observed, with the cuML model having marginally more false prediction. The difference, although small, indicates that the CPU-based implementation had marginally fewer misclassifications overall, correctly identifying 86 additional malicious instances compared to the GPU-based implementation, and reporting no false positives. Nevertheless, both models showed very high classification reliability suitable for operational cybersecurity contexts, and the differences can be attributed to runtime variance.

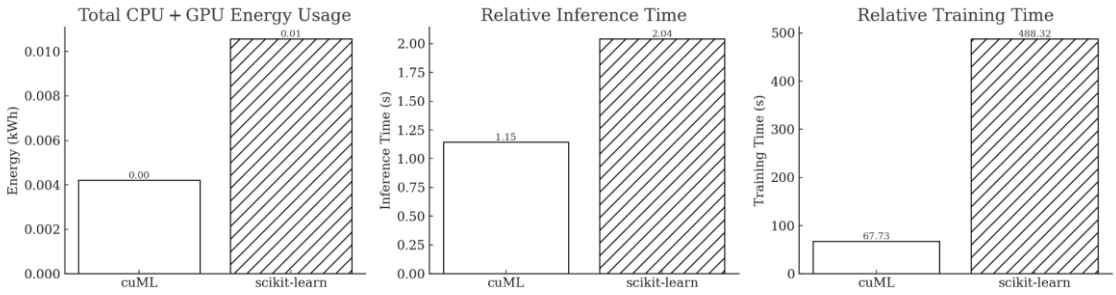

Figure 12. cuML vs scikit-learn RF: training time, inference time & energy consumption.

Figure 12 provides a comprehensive comparison of computational performance metrics, including training time, inference time, and energy consumption based on CPU and GPU

power measurement readings. The energy consumption analysis revealed significant distinctions between the GPU-based and CPU-based implementations in terms of hardware utilisation and efficiency. The GPU-based cuML implementation consumed substantially less CPU energy, approximately 0.83 Wh, highlighting efficient CPU resource usage due to extensive offloading of computational tasks onto the GPU. In contrast, the CPU-based scikit-learn implementation exhibited notably higher CPU energy consumption, totalling 8.96 Wh. This elevated CPU energy usage directly reflects the intensive computational load placed exclusively on the CPU cores when GPU acceleration is not leveraged. Conversely, the GPU energy consumption analysis naturally indicated higher GPU power draw for the cuML implementation (3.38 Wh), consistent with its full utilisation of GPU hardware resources to accelerate Random Forest training and inference. The scikit-learn model, limited by its minimal reliance on GPU computation, utilised significantly less GPU energy, 1.62 Wh, reflecting only baseline idling GPU usage associated with general system operations rather than dedicated computational acceleration.

The GPU-based Random Forest implementation exhibited a substantial reduction in computational time requirements compared to the traditional CPU-based implementation, underscoring its suitability for time-sensitive cybersecurity applications. The GPU-accelerated cuML Random Forest completed the training phase in 67.7 seconds, demonstrating an improvement over the CPU-based scikit-learn model, which required 488.3 seconds; over eight minutes. This decrease in training duration represents a considerable advantage, particularly in real-world scenarios demanding frequent model retraining to adapt rapidly to evolving cyber threats.

Regarding inference performance, the GPU-accelerated model similarly displayed notable speed advantages, completing the inference task in approximately 1.15 seconds, whereas the CPU-based implementation required 2.04 seconds. Although the absolute reduction in inference time was less pronounced compared to the training phase, nearly halving inference latency still constitutes a meaningful improvement. This inference speed-up is particularly relevant in operational environments like the Locked Shields exercise, where rapid detection and timely response to network intrusions are critical to minimising potential impacts. Consequently, GPU-based acceleration demonstrates substantial value, not only in enhancing training efficiency but also in significantly improving real-time predictive responsiveness within cybersecurity operations.

In summary, despite a minor performance advantage of the CPU-based Random Forest in absolute classification accuracy with fewer misclassifications, the GPU-based implementation demonstrated significantly greater efficiency in terms of training speed, inference speed, and overall energy consumption. Such computational efficiency advantages have direct practical implications in operational cybersecurity environments, where timely threat detection and frequent model retraining are often crucial. Consequently, the GPU-accelerated cuML Random Forest emerges as a compelling choice, offering an optimal balance between high classification performance and resource efficiency suitable for real-world deployment scenarios.

4.2 Comparison of ML and DL models on LSPR23 dataset

This section presents a detailed comparative analysis of the classification performance, computational efficiency, and resource consumption between the traditional machine learning approach (Random Forest, RF) and the deep learning approach (1D Convolutional Neural Network, CNN), utilising the LSPR23 dataset. Multiple aspects of each model's performance are evaluated in depth, including detailed metrics such as accuracy, precision, recall, F1-score, confusion matrices, computational times for training and inference, GPU energy consumption, and GPU memory usage. The analysis also includes graphical representations and comprehensive tables for a thorough overview of the practical trade-offs involved in choosing between these two modelling approaches.

4.2.1 Classification performance comparison

The classification results for both models demonstrate excellent overall performance, reflecting their strong ability to distinguish malicious network flows from benign traffic. Table 8 presents a detailed comparison of CNN and RF classifiers based on accuracy, precision, recall, and F1-score, summarising their respective mean (marked as avg), minimum, maximum, and standard deviation values calculated over 10 independent runs. The best result per metric is marked in bold.

Table 8. Performance comparison between CNN and RF

Metric		Convolutional Neural Network	Random Forest
Accuracy	Avg	**99.974%**	99.951%
	Min	99.969%	99.946%
	Max	99.978%	99.956%
	Std	0.0028%	0.0041%
Precision	Avg	99.913%	**99.996%**
	Min	99.874%	99.996%
	Max	99.952%	99.997%
	Std	0.0249%	0.0003%
Recall	Avg	**99.824%**	99.520%
	Min	99.749%	99.471%
	Max	99.880%	99.570%
	Std	0.0392%	0.0406%
F1-Score	Avg	**99.868%**	99.758%
	Min	99.847%	99.733%
	Max	99.888%	99.783%
	Std	0.0140%	0.0203%

Both models achieved exceptionally high accuracy, surpassing 99.9%. As can be seen from the above table, the CNN slightly outperformed the RF model in terms of accuracy, recall and F1-score. In particular, the higher recall of the CNN model indicates superior sensitivity, effectively minimising false negatives. However, the RF model demonstrated close to perfect precision, 99.996%, which implies almost zero false positives in this evaluation scenario, significantly reducing the risk of generating unnecessary operational alerts. Such precision is highly advantageous in practical cybersecurity deployments where alert fatigue can negatively impact the efficiency and effectiveness of security teams.

The confusion matrices depicted in figure 13 give closer insight into each model's predictive behaviours. The CNN model accurately identified more malicious flows compared to RF, demonstrating fewer false negatives. Specifically, CNN correctly identified 999 fewer false negatives compared to RF, significantly reducing the likelihood of missing critical threats. The standard deviation for false negatives was 129 for the CNN and 133 for the RF. This means that even the worst performing CNN out of the 10 had fewer false negatives than the best performing RF.

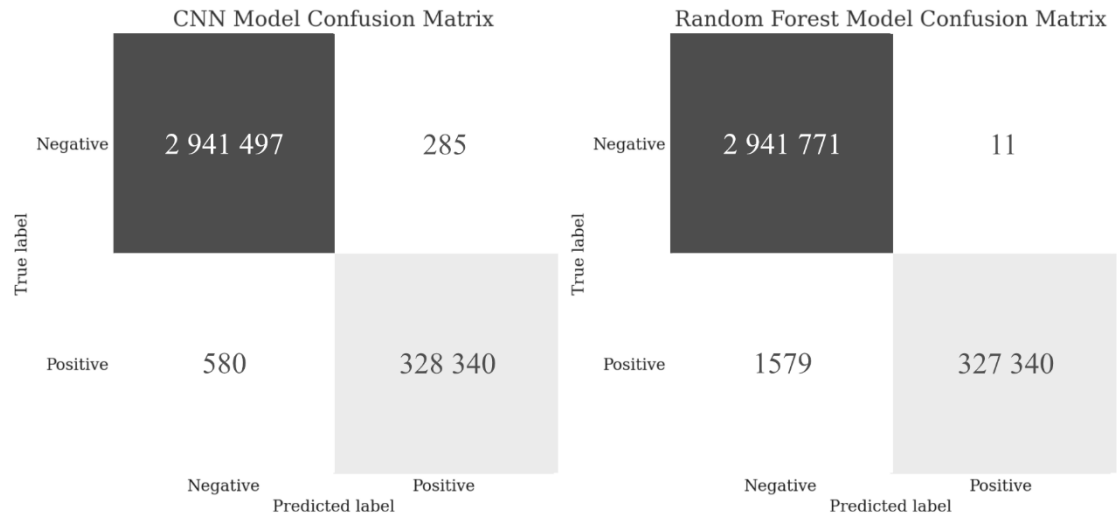

Figure 13. Average confusion matrices for the CNN and RF models

Conversely, the RF model demonstrated an exceptionally low false-positive rate, 11 false positives compared to 285 of the CNN, greatly decreasing the generation of unnecessary alarms. The best performing CNN had 159 false negatives, while the worst RF had 13. Ultimately the confusion matrix results clearly guide decisions based on the priorities of the cybersecurity environment; either selecting the CNN for reducing missed threats or minimising false alarms by choosing RF.

4.2.2 Computational efficiency comparison

In cybersecurity operational scenarios, computational efficiency directly influences the practicality, scalability, and real-world applicability of an intrusion detection system. A comprehensive comparison of computational performance metrics for the CNN and RF models is summarised in Table 9:

Table 9. Computational efficiency of CNN and RF models

Metric		CNN	Random Forest
Training time	Avg	1313.48 s	146.83 s
	Min	801.00 s	145.30 s
	Max	1607.14 s	149.33 s
GPU Energy (Training)	Avg	0.076 kWh	0.0069 kWh
Peak GPU memory usage	Avg	7024.62 Mb	1024.3 Mb
	Min	6813.65 Mb	1024.3 Mb
	Max	7245.66 Mb	1024.3 Mb

Metric		CNN	Random Forest
Inference Time	Avg	7.41 s	0.33 s
GPU Energy (Inference)	Avg	0.0007 kWh	0.0000021 kWh
Eval CO_2 Equivalent	Avg	≈0.3g CO_2-eq	≈0.0033g CO_2-eq

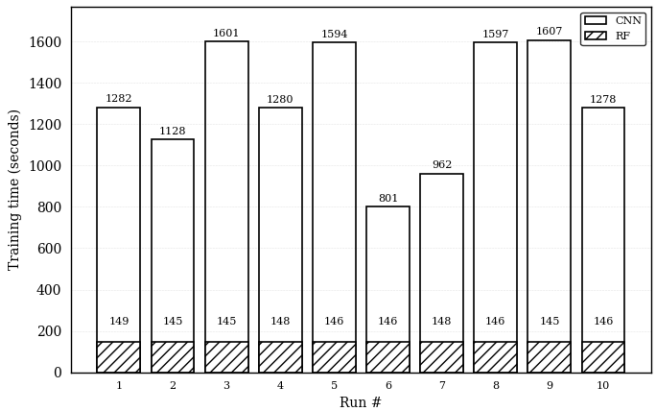

Figure 14. CNN vs RF training times over 10 runs

The computational timings presented in Table 9 highlight substantial efficiency advantages for the Random Forest model. In terms of training speed, the RF completed training on average in 147 seconds, whereas the CNN required 1313 seconds. This significant difference means that the RF model trained roughly 10 times faster than the CNN model, as can be seen from figure 14. The reduced training time provided by the RF approach offers critical advantages in dynamic cybersecurity environments, where rapid retraining is essential to adapt promptly to continuously evolving threats.

Inference time, critical in operational contexts for real-time decision-making, similarly favoured the RF approach. The RF model completed inference on average in just 0.33 seconds, significantly faster than the CNN, which required 7.41 seconds. While a difference of several seconds may appear nonconsequential, in practical cybersecurity applications demanding rapid response and minimal detection latency, this improvement of more than 20 times in inference speed substantially enhances the RF model's suitability for operational deployment, enabling faster identification and response to threats. The pronounced disparity between CNN and RF inference times can be observed when comparing Table 9 and Figure 15, where the RF bar is noticeably shorter than the CNN bar on the inference time chart.

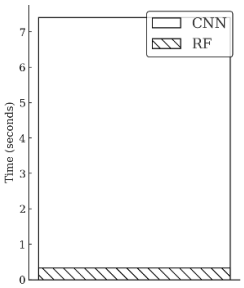

Figure 15. CNN vs RF average inference time

Resource efficiency is another key operational consideration, encompassing GPU power consumption, GPU memory usage, and environmental impact. GPU energy consumption

during training was 10 times higher for the CNN model compared to the RF model. Similarly, during inference, the CNN consumed significantly more GPU energy: 0.0007 kWh compared to the RF with 0.0000021 kWh, reflecting a difference exceeding 300 times. This contrast illustrates the difference of computational demands and power usage inherent to deep learning architectures, which can pose constraints in resource-limited environments or scenarios aiming for more sustainable, lower-impact computing operations.

Additionally, GPU memory utilisation revealed substantial disparities. The CNN model required approximately 7 Gb of GPU memory, whereas the RF model operated with only 1 Gb. This difference highlights the considerable resource efficiency advantage of the RF model, making it particularly suitable for scenarios where hardware limitations or resource constraints might restrict deployment feasibility.

Taken together, these computational and resource efficiency metrics illustrate clear practical trade-offs: the CNN offers marginally superior detection accuracy but at a substantial computational and resource cost. In contrast, the RF provides excellent predictive performance combined with significantly superior computational efficiency, lower resource consumption, and reduced environmental impact, making it highly suitable for operational cybersecurity environments requiring rapid response, frequent retraining, and resource-conscious implementation.

4.2.3 Answer to research question RQ2

RQ2: How do the traditional learning techniques compare to the deep learning techniques in classification performance in classifying malicious network traffic?

The comparative analysis conducted using the LSPR23 dataset clearly demonstrates strong performance from both traditional machine learning (Random Forest) and deep learning (1D Convolutional Neural Network) techniques in identifying malicious network traffic, with each exhibiting distinct strengths and trade-offs. Specifically, both models achieved exceptional accuracy exceeding 99.9%, highlighting their suitability for cybersecurity applications. However, nuanced differences emerged in the performance.

The CNN marginally surpassed RF in overall accuracy and demonstrated a superior balance between precision and recall, achieving an F1-score of 99.87% compared to RF's

99.76%. CNN exhibited notably higher sensitivity, recall of 99.82% versus RF's 99.52%, indicating greater effectiveness in identifying malicious traffic and decreasing missed threats. This advantage positions CNN favourably in scenarios where detecting the maximum number of threats is paramount. Conversely, RF demonstrated near-perfect precision and substantially outperforming CNN, thus significantly reducing false positives. This precision advantage makes RF highly useful in operational cybersecurity contexts, where minimising false alarms is critical to maintaining security team responsiveness and effectiveness.

In terms of computational efficiency, RF significantly outperformed CNN, training roughly 10 times faster and achieving inference speeds more than 20 times quicker. Furthermore, RF required substantially fewer computational resources and lower environmental impact, making it particularly suitable for rapid, resource-constrained deployments.

In conclusion, traditional machine learning techniques, represented by Random Forest, offer outstanding predictive performance combined with significantly greater computational efficiency, resource economy, and superior precision. Deep learning techniques, represented by CNN, though marginally superior in overall accuracy and recall, impose considerably higher computational costs and complexity. The choice between these approaches should thus be guided by operational priorities: maximum threat detection sensitivity (CNN) versus operational efficiency and minimal false positives (RF).

4.3 Cross-dataset validation results on LSPR24

This section explores the robustness and generalisation capability of the previously trained CNN and RF models by conducting a cross-dataset validation using the Locked Shields Partners Run 2024 (LSPR24) dataset. Cross-year validation provides insight into how effectively models trained on past datasets (LSPR23) can handle previously unseen traffic patterns and novel cyber threats present in subsequent years. The LSPR24 dataset is larger and more imbalanced, comprising 20,227,356 flows, with benign flows constituting approximately 97.43% of the total (19,707,365 flows), and malicious flows accounting for only 2.57% (519,991 flows). This notable shift in class distribution compared to the LSPR23 dataset poses additional challenges for both classification

models, along with the changed network environment. For this evaluation, different preprocessing was used to deal with non-finite values. Rows containing NaN values were removed prior to evaluation and consequently, therefore the total counts presented in the confusion matrices below differ slightly from the original dataset counts.

4.3.1 Importance of inter-arrival time features

As noted earlier in section 3.4.1.4, Inter-arrival Time (IAT) features offer valuable timing-based information, capturing temporal patterns within network flows. However, these features may become misleading if significant changes in the network infrastructure occur across different datasets or years, as Gehri et al. highlighted previously [58]. Therefore, this validation explicitly compares the performance of the CNN and RF models trained with and without the inclusion of IAT features to assess their impact on cross-year generalisation capability.

4.3.2 Performance comparison without IAT features

Table 10 summarises the classification metrics for both CNN and RF models when IAT features were excluded from training and evaluation:

Table 10. Cross-dataset performance on LSPR24 without IAT Features

Metric	CNN	Random Forest
Accuracy	84.66%	97.30%
Precision	8.05%	15.65%
Recall	47.69%	1.18%
F1-score	13.78%	2.20%

In the absence of IAT features, both models exhibited significant performance degradation, indicating substantial difficulty in generalising to the 2024 network scenario. Despite achieving a higher accuracy compared to the CNN, the RF model's recall was notably poor, suggesting severe difficulty in correctly identifying malicious flows. As can be seen from the confusion matrices below, this high accuracy is predominantly due to the large number of benign flows (15,906,251 true negatives) compared to malicious flows, thereby inflating the accuracy metric and rendering it misleading. The RF confusion matrix clearly illustrates the severity of this issue, with the model identifying only 4,980 out of 420,419 malicious flows correctly and thus missing the vast majority

of the malicious flows, posing critical operational risks. The CNN, conversely, exhibited a substantially higher recall with 47.69% and successfully identified 200,484 malicious flows out of the total 420,419. However, the CNN's improved sensitivity came with a significantly higher false-positive count, 2,288,557 false alarms, as clearly depicted in its confusion matrix.

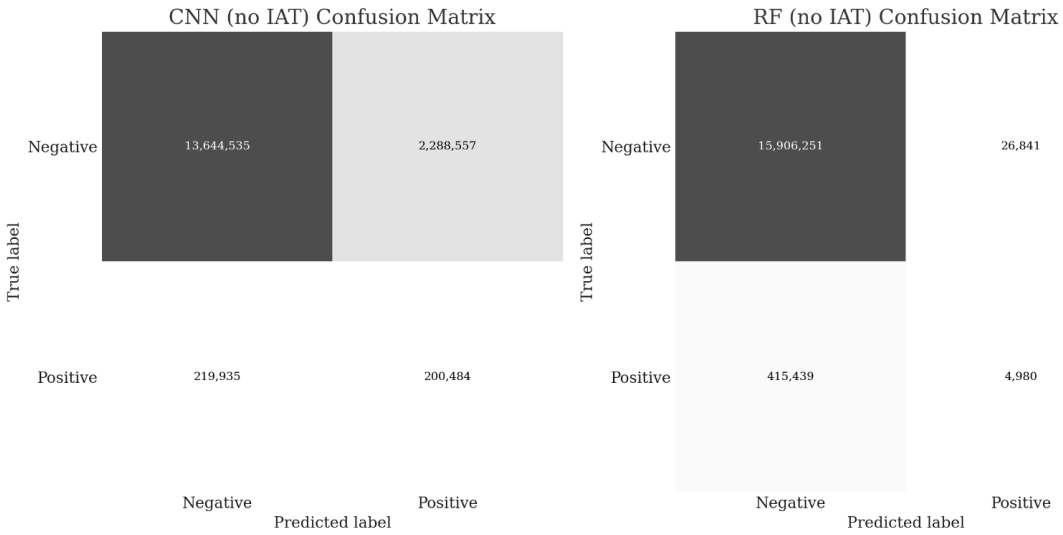

Figure 16. Confusion matrices comparing performance of CNN and RF models without IAT features

4.3.3 Performance comparison with IAT features

Table 11. Cross-dataset performance on LSPR24 with IAT Features

Metric	CNN	Random Forest
Accuracy	91.98%	97.25%
Precision	12.47%	38.86%
Recall	35.23%	12.06%
F1-score	18.42%	18.41%

The inclusion of Inter-arrival Time features markedly influenced the predictive performance of both CNN and RF models in the cross-dataset evaluation, as clearly illustrated in Table 11 and Figure 17. Specifically, the Random Forest model exhibited substantial improvements when these temporal features were included, particularly in precision and recall. Precision increased significantly, rising from 15.65% to 38.86%, demonstrating that the RF model became considerably more accurate in correctly identifying malicious flows while simultaneously reducing false-positive detections.

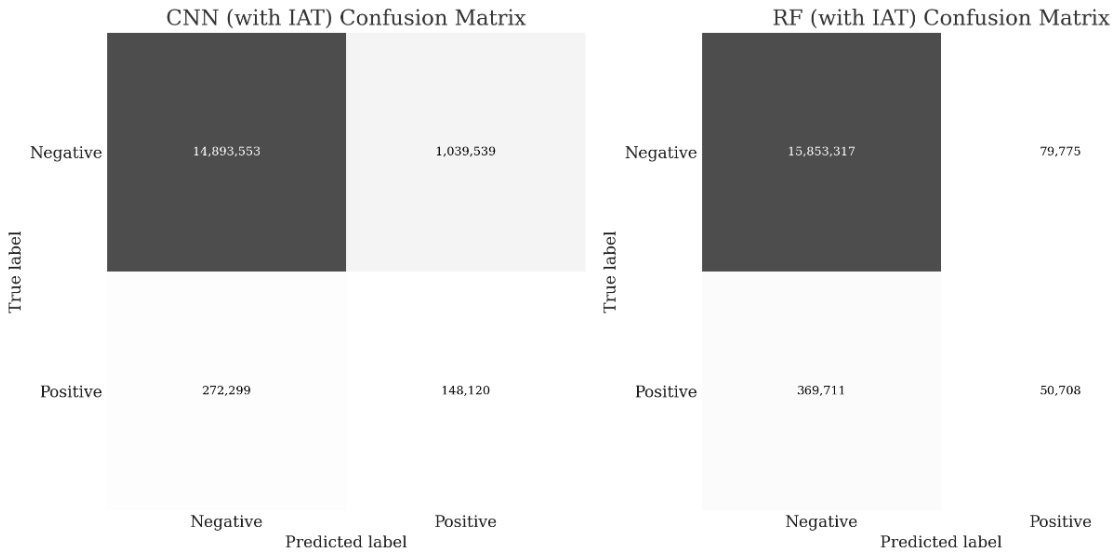

Figure 17. Confusion matrices comparing performance of CNN and RF models with IAT features

Additionally, the recall improved notably from a very low 1.18% without IAT features to a moderate 12.06% with these features. The confusion matrices in Figure 17 explicitly illustrate this improvement, showing a noticeable increase in correctly identified malicious flows, with true positives rising from 4,980 to 50,708 when using IAT features. While the RF model still exhibited limited overall sensitivity, this tenfold improvement highlights the considerable informational value of IAT features. For the CNN, incorporating IAT features increased accuracy substantially, from 84.66% to 91.98%, and precision improved from 8.05% to 12.47%. However, the recall decreased somewhat from 47.69% to 35.23%, reflecting fewer true positive detections. Still, the overall balance between precision and recall improved, as indicated by the F1-score increase from 13.78% to 18.42%, confirming that IAT features positively influenced CNN's generalisation capability.

Despite concerns raised by Gehri et al. regarding the potential instability and reduced effectiveness of temporal features like IAT across changing network infrastructures [58], the inclusion of these features has clearly enhanced the generalisation capability of both RF and CNN models in this cross-dataset validation. The improvements observed in precision, recall, and overall classification stability confirm that IAT features contribute to capturing distinctive traffic patterns and temporal behaviours, thus enabling models to better adapt to evolving threats and novel network conditions. Consequently, these results emphasise the practical importance of carefully managing and incorporating temporal

features into intrusion detection models, provided network conditions remain sufficiently stable.

4.3.4 Answer to research question RQ3

RQ3: How effective are traditional machine learning models compared to deep learning models in transfer learning across different databases?

The cross-dataset validation using the LSPR24 dataset highlights significant challenges faced by both traditional machine learning and deep learning approaches when transferring learned patterns from the LSPR23 dataset to novel traffic and threat conditions encountered in the following year. This evaluation clearly reveals distinct differences in robustness and generalisation capabilities between the two modelling strategies.

Without temporal inter-arrival time features, the CNN exhibited substantially higher recall than RF, 47.69% vs 1.18%, thus indicating superior generalisation in recognising malicious patterns despite changes in network conditions. However, this came at the cost of extremely high false positives, significantly reducing practical operational effectiveness. RF, despite high overall accuracy of 97.30% severely struggled to generalise, identifying few malicious flows, emphasising critical limitations in transferability without additional temporal context.

Including temporal IAT features significantly improved the generalisation capabilities of both models. The RF model particularly benefited, achieving a substantial improvement in recall, increasing from 1.18% to 12.06%, and precision, rising from 15.65% to 38.86%. This indicates an enhanced ability to recognise malicious patterns across datasets when temporal context is maintained. The CNN also experienced improvements, with accuracy rising from 84.66% to 91.98% and precision increasing from 8.05% to 12.47%. However, this improvement occurred at some expense to recall, underscoring the sensitivity of deep learning models to changes in feature representation.

Although CNN consistently demonstrated higher recall, indicating better capability in capturing new threats, its pronounced tendency toward false positives poses substantial operational challenges. In contrast, the RF model, though initially poor in recall without

temporal features, significantly improved when temporal context was provided, offering balanced and practical operational performance.

To summarise, random forest demonstrated notable robustness and significantly improved generalisation across databases when supported by meaningful temporal context. Convolutional neural network inherently captured more generalisable features, resulting in better initial transfer capability, yet faced substantial trade-offs in precision and operational practicality. Consequently, for effective transfer learning across cybersecurity datasets, traditional models appear advantageous when carefully curated features are consistently available, while deep learning models offer inherent adaptability at higher computational and operational costs.

5 Discussion

This thesis compared a representative traditional machine learning model (Random Forest, RF) and a deep learning model (1D Convolutional Neural Network, CNN) for network intrusion detection, using the LSPR23 dataset derived from the Locked Shields 2023 cyber exercise. Both models achieved excellent predictive performance, with overall accuracies exceeding 99.9%. CNN demonstrated slightly better sensitivity, with higher recall compared to RF, indicating fewer missed threats. Conversely, the RF showed near-perfect precision and thus substantially reduced the number of false alarms (see Table 8). This trade-off, clearly visible in the confusion matrices (Figure 13), implies operational decisions between minimising missed attacks (CNN) versus reducing alert fatigue (RF). These findings align closely with prior literature: CNN-based models are recognised for superior recall and accuracy [34], whereas ensemble tree-based methods like RF consistently demonstrate robust, precise performance [25], [41]. Thus, this work reinforces the literature's conclusion that well-optimised traditional methods remain highly competitive for intrusion detection tasks.

5.1 Practical implications

The practical implications of these findings suggest distinct operational roles for each model. The RF's exceptionally low false-positive rate makes it highly suitable as a primary IDS, reducing analyst workload from false alerts. Conversely, CNN's higher recall positions it favourably for environments where missing threats is particularly costly, such as critical infrastructure or military exercises. Thus, a hybrid or tiered IDS approach combining the strengths of both models could be optimal. Using RF for routine threat detection and CNN for advanced analysis of potential false negatives. Additionally, the interpretability advantage of RF makes it attractive in compliance-driven environments, while CNN offers the potential to automatically capture subtle attack patterns beyond traditional feature engineering.

In this thesis the RF model required significantly less computational effort than CNN, achieving roughly a tenfold improvement in training speed and more than twentyfold improvement in inference speed along with dramatically lower GPU memory and energy demands (see table 9). These differences, visually confirmed in figures 14 and 15,

highlight RF's practical advantages, particularly in real-time, resource-constrained environments requiring rapid threat detection and frequent model retraining. By contrast, CNN's higher computational load may impose operational constraints and limit practical deployment to scenarios with sufficient computational infrastructure.

Ultimately, operational considerations, such as training and inference speed, resource availability, and environmental impact, are as critical as raw accuracy when evaluating model suitability for real-world cybersecurity deployment. Selecting the optimal IDS approach therefore requires a careful balance between technical performance metrics and the practical realities and constraints related to deployment, specific to each operational environment.

5.1.1 Necessity of machine- and deep learning in intrusion detection

The near-perfect detection results on the LSPR23 dataset for both the RF and CNN approaches raises the question on the practical necessity of employing advanced machine learning models for intrusion detection. Indeed, several studies have demonstrated that simpler ML and statistical methods can achieve strong results in certain intrusion detection scenarios. Yu et al. conducted an empirical evaluation revealing that classical algorithms such as k-Nearest Neighbours (kNN) outperformed DL models in log anomaly detection, achieving superior accuracy and significantly reduced computational complexity [19]. Specifically, kNN provided better detection performance compared to complex neural network methods, highlighting the potential for simpler models to effectively solve specific, well-defined intrusion detection problems. Supporting these findings, Leon et al. conducted an extensive comparative analysis demonstrating that on simpler datasets such as KDD99 and NSL-KDD, Linear Discriminant Analysis (LDA) and Support Vector Machine (SVM) achieved performance comparable to artificial neural networks but with significantly reduced computational complexity [41]. Moreover, a recent comprehensive survey by Natarajan et al. reinforced that simpler supervised algorithms, e.g., CART, Logistic Regression, frequently attained accuracy in the range of 98–99% on common datasets, underscoring their continued relevance in intrusion detection tasks [37].

However, despite the strong performance of simpler methods reported in the literature, the necessity for advanced ML and DL approaches becomes apparent when addressing more realistic and complex intrusion detection scenarios. For instance, in the previously

cited study by Leon et al., the LDA model exhibited significantly lower performance on the more recent UNSW-NB15 dataset, whereas advanced models such as RF and ANN experienced only minor performance degradation [41]. The LSPR23 dataset used in this thesis exemplifies such a realistic and challenging scenario. It features highly sophisticated, multi-stage attacks embedded within legitimate background traffic from a large-scale cybersecurity exercise, thereby representing conditions significantly more complex than typical benchmark datasets. This complexity was particularly evident in the performance evaluation conducted by Dijk et al. who assessed Suricata, a widely recognised open-source IDS employing traditional signature-based detection methods. Suricata demonstrated notably poor performance on the LSPR23 dataset, achieving only 53.1% F1-score with its standard rulesets, resulting in many missed detections and false positives on the same time [3]. This significant performance gap clearly illustrates that traditional signature-based methods, while simpler and computationally less intensive, fail to capture the nuanced and novel attack patterns inherent in realistic cybersecurity scenarios.

The excellent results achieved by the ML and DL models presented in this thesis are therefore not indicative of overperforming on a trivially simple detection task. Rather, they highlight the capacity of these models to effectively learn intricate and subtle features distinguishing malicious from benign traffic. The comparative analysis of RF and CNN approaches presented in the experimental chapter of this thesis reinforces this interpretation. While the RF model achieved impressive results, the CNN model demonstrated marginally superior accuracy, suggesting that the deep learning approach was able to capture additional complexities and nuances in the data that simpler classifiers could not. Although the computational costs associated with the CNN were notably higher, its ability to generalise from complex patterns provides significant advantages when faced with evolving or novel attack scenarios.

Nevertheless, the suitability of ML/DL solutions must be assessed on a case-by-case basis. Where simpler statistical methods suffice, their advantages in interpretability and computational efficiency are indeed valuable. The findings of this thesis align with existing literature, demonstrating that simpler methods like Random Forest can achieve remarkable results given well-structured data and informative features. However, the security domain often necessitates minimising false negatives to the greatest possible extent, as the costs associated with missed intrusions can be substantial. Thus, even

marginal improvements in detection performance afforded by complex ML/DL models can significantly enhance operational security. An optimal strategy might therefore involve employing a combination of simpler methods for initial rapid detection and advanced ML/DL methods for in-depth analysis and detection of sophisticated attacks.

5.1.2 Binary vs multiclass classification considerations

When interpreting the results of this thesis within the broader context of NIDS, it should be noted that both the evaluation presented in Sections 3 and 4 and the majority of related studies presented in the literature review, focus predominantly on binary classification tasks: distinguishing malicious traffic from benign traffic. This binary approach considerably simplifies the problem space and achieving accuracy levels exceeding 90%, as frequently reported in the literature (see table 1 and table 2), is inherently less challenging in a binary scenario compared to more granular classification tasks.

Such impressive figures, however, come with important caveats: models can attain high overall accuracy by potentially exploiting class imbalance by learning to always predict the majority class or by glossing over distinctions between different attack types. For example, Chindove and Brown observed in their study that several algorithms achieved almost 100% accuracy on the CICIDS 2017 dataset simply because benign flows dominated the traffic, whereas the results for the minority classes was much lower [40]. This serves as a reminder that an aggregate “malicious vs. benign” metric can mask poor detection of specific attack categories.

This thesis, like much of the prior literature, adopts a binary classification paradigm primarily because it aligns with how ground truth labels were provided in the dataset and facilitates a clear evaluation of benign vs. malicious classification performance. The LSPR23 dataset used for training and testing was labelled at the flow level with a simple binary indicator for each network flow [3] and this made binary classification a natural and pragmatic choice for measuring overall intrusion detection efficacy. It is worth noting that the malicious portion of LSPR23 traffic is about 10% of flows (see section 3.3.5), which is an unusually high attack prevalence compared to typical enterprise networks. This imbalance enabled both the RF and CNN models to achieve over 99% accuracy on the LSPR23 data by mainly learning to distinguish normal vs. attack behaviour, consistent with other studies on imbalanced datasets [31], [40], [47]. In a multiclass setting, by contrast, each specific attack type would constitute an even smaller minority of the traffic,

exacerbating the class imbalance problem. Moreover, any misclassification between malicious classes, e.g. confusing a SQL injection for a brute-force attack, counts against a multiclass classifier's accuracy [31], whereas a binary classifier is oblivious to such errors as long as the traffic is flagged as malicious.

Focusing solely on binary classification, while useful for benchmarking detection capability, limits the operational usefulness of an IDS. Security operators often need to know the nature of an intrusion to respond appropriately. A model that simply raises an alert for "malicious activity" without further context offers limited guidance for incident response [14]. Nonetheless, the high detection performance demonstrated by binary models in section 4.2 serves as a strong foundation upon which more detailed multiclass classification methods can be built, enabling incremental improvements in practical intrusion response capabilities. Both the CNN and Random Forest models developed and evaluated in this thesis inherently support multiclass classification with minimal structural adjustments, making the transition towards more detailed threat categorisation practically achievable.

5.1.3 Model transferability and environment specifics

The cross-year validation using the LSPR24 dataset highlighted significant generalisation challenges for both models. RF in particular exhibited a severe reduction in recall, identifying only a small fraction of new malicious flows and the CNN, though more resilient with a higher recall still experienced a sharp rise in false positives, particularly without temporal features (see table 10).

The comparatively low performance of both the Random Forest and CNN models on the newer LSPR24 dataset is an expected outcome given the shift in data between these two evaluations. In the domain of network intrusion detection, it is well-recognised that a model trained in one environment often struggles when applied to a different environment or a later dataset [40], [58], [92]. This is because the underlying network conditions and threat landscape can change significantly: normal background traffic profiles vary across organisations and time, and attackers continually adapt their tactics [75]. For instance, Locked Shields exercise introduces up-to-date attack techniques and employs different network infrastructures annually, meaning that the 2024 iteration inevitably differs from 2023 in both benign traffic and attack vectors [2]. A classifier built on LSPR23's traffic patterns and threat profile will therefore encounter previously unseen patterns in LSPR24,

both in terms of legitimate traffic behaviour and new malicious activities. This results in degraded detection performance. This phenomenon is essentially an instance of concept drift, where the statistical properties of the input data change over time. As Jordaney et al. explain, models trained on older data will show signs of aging and begin to miss new or evolved threats [92]. In other words, what the model learned as malicious or benign in 2023 may no longer hold true in 2024, leading to misclassifications.

Recent research underlines how pronounced this effect can be when evaluating intrusion detectors across different time periods or environments. Vaarandi and Guerra-Manzanares [74] observe that in a realistic chronological evaluation, meaning training on earlier network alert data and testing on later data, many new alert types emerged in the later period that the model had never seen before. In their study, over half of the attack signatures present in the test set were completely absent in the training set. They caution that the common practice of randomly splitting data for training and testing ignores such temporal evolution, yielding overly optimistic performance estimates [74]. In summary, the LSPR23 to LSPR24 performance drop aligns with expectations for a cross-environment deployment; it highlights the challenge of model transferability in the presence of changing network conditions, new attacker strategies, and concept drift.

Gehri et al. emphasise that it is possible to train more generalised models by carefully selecting features that are less dependent on any one network's specifics, but even then such models "generally fail to achieve the same performance" as models optimised and evaluated in a single, static environment [58]. Känzig et al. similarly note that benign and malicious traffic profiles can vary considerably between organisations or exercise iterations, so a detector might work well on the network it was trained on yet underperform on a new network without adaptation [63]. These findings justify the drop in models' efficacy on LSPR24 in section 4.3: without recalibration, the models trained on 2023 data are not fully equipped for the 2024 scenario.

To address this practical challenge of balancing model accuracy and resource constraints associated with frequent retraining, a hybrid approach combining generalisation and adaptive mechanisms is recommended. Purely environment-specific retraining is often impractical due to the high resource demands of continuously labelling data and updating models, resulting in either diminishing returns or unreliable detection over time if updates are infrequent [92]. Instead, leveraging time-independent features and historical diversity

in training datasets, such as aggregating multiple years of Locked Shields exercises, can enhance baseline model generalisation across environments by establishing robust decision boundaries capable of detecting a broader spectrum of malicious behaviours [58], [63]. Additionally, incorporating active learning frameworks, where models selectively solicit human expertise on uncertain alerts, can incrementally tailor detection capabilities to specific environments without exhaustive retraining cycles [74]. Coupled with proactive drift detection tools such as the Transcend framework [92], which identify shifts in data distribution and prompt targeted interventions, the combined strategy can maintain detection effectiveness while significantly reducing the manual overhead typically associated with model maintenance.

5.2 Limitations

This thesis acknowledges several limitations which should be considered when interpreting the findings. Recognising these limitations provides valuable insights and directions for future research and development, discussed in section 5.3.

Firstly, the comparison deliberately focused on one representative algorithm from each category. Random Forest for traditional machine learning (section 3.5), and a one-dimensional CNN for deep learning (section 3.6). Although this pragmatic approach provided clear insights, it excludes other potentially strong alternatives, such as gradient-boosted trees, support vector machines, and alternative neural network architectures like Long Short-Term Memory networks and Transformers.

Secondly, the evaluation in this thesis was limited exclusively to binary classification, distinguishing only between malicious and benign traffic. This binary approach does not account for the diversity, specificity, and varying characteristics of distinct attack categories, such as denial-of-service attacks, data exfiltration attempts, port scans, or privilege escalation efforts. High accuracy achieved within a simplified binary scenario may not directly generalise or translate effectively to more nuanced multiclass classification, where the system must correctly identify and differentiate multiple distinct attack types, or anomaly detection scenarios, in which previously unseen or novel attacks must be recognised and categorised.

Thirdly, the dataset used, although large-scale and realistic, originated from a single cyber exercise environment, Locked Shields 2023. This environment possessed distinct characteristics such as a fixed set of services, a relatively high malicious traffic ratio of about 10% and particular attack patterns explicitly designed for the exercise scenario. Consequently, the absolute performance metrics achieved by the evaluated models may not directly generalise to networks that differ significantly in their operational profiles, threat landscapes, and user behaviours, such as typical corporate or military networks. The considerable reduction in model performance observed when transitioning from the 2023 Locked Shields dataset to the closely related yet distinct LSPR24 dataset underscores these generalisation challenges. This performance drop highlights how even minor variations in network architecture, threat distribution, and attack techniques can substantially affect the effectiveness of trained models.

Another limitation relates to model hyperparameter tuning. While RF hyperparameters were adopted from prior research [63] and CNN architecture was inspired by previous works [77], [78], exhaustive hyperparameter optimisation tailored explicitly to this dataset was not performed due to computational constraints and practical considerations. Thus, performance could potentially be improved through more comprehensive model-specific tuning and optimisation strategies. Moreover, feature engineering choices, such as dropping certain fields or encoding categorical variables, were carefully considered but not exhaustively evaluated. Although the impact of specific features (IAT timings) was tested, subtle feature interactions that could further improve model robustness or predictive performance might remain unaddressed.

Finally, the generalisation assessment conducted in this thesis utilised a static model trained exclusively on data from one specific year (2023) and subsequently evaluated on data collected the following year (2024), without employing incremental updates, continuous learning, or periodic retraining strategies. Although this methodological choice was intentionally designed to rigorously test the models' generalisation capabilities under challenging, worst-case conditions, it may not fully reflect practical deployment scenarios in real-world cybersecurity operations, where intrusion detection models typically benefit from ongoing adaptation to evolving threat landscapes.

5.3 Future research

Future research could extend the ML vs DL comparison by systematically evaluating a broader array of algorithms, including those prominently discussed in the comprehensive literature review presented in this thesis enriching the understanding of their relative strengths and limitations within intrusion detection contexts. These currently untested models could improve detection by incorporating temporal sequences and contextual relationships among network flows. Techniques such as recurrent neural networks, 2-dimensional convolutional neural networks, attention-based models, or graph neural networks can effectively capture patterns and dependencies over time or across hosts, potentially improving identification of coordinated or multi-stage attacks. Employing these methods could reveal attack behaviours that single-flow analysis fails to identify, thereby reducing false negatives and increasing model precision.

Similarly, future work could move beyond binary classification tasks on the Locked Shields datasets by incorporating multiclass classification scenarios and addressing the detection of novel or emerging threats. Specifically, efforts should focus on identifying and differentiating specific attack types, such as port scans, brute-force attempts, and DDoS, which provide more actionable insights by distinguishing among threat categories and severity levels. This would align intrusion detection research more closely with realistic operational cybersecurity requirements, ultimately offering a deeper understanding of model capabilities, practical limitations, and suitability for detailed threat characterisation. Given that the Locked Shields 2023 dataset prepared by Dijk et al. already includes additional attack narratives in a separate file suitable for such analyses, future studies could leverage this resource to develop more sophisticated multi-class or hierarchical classification frameworks.

The deep learning model developed in this thesis can be easily adapted to support multi-class predictions by modifying the final prediction layer to include multiple neurons instead of one. Similarly, the Random Forest model evaluated here inherently supports multi-class classification, requiring only appropriately labelled data without structural adjustments.

The significant performance drop observed between the LSPR23 and LSPR24 datasets emphasises the importance of addressing concept drift [92]. Future studies could

investigate continuous or incremental learning strategies, such as periodic model retraining, online algorithms, or domain adaptation techniques to better represent realistic operational conditions. Employing drift detection mechanisms to trigger timely model updates could further enhance generalisation and resilience against evolving threats. Additionally, integrating anomaly detection methods, such as the N-outlier approach suggested by Vaarandi et al. [74], alongside supervised models could substantially improve model robustness. Investigating these approaches could potentially reduce or mitigate the performance degradation observed when models encounter previously unseen network behaviours or emerging threat types, thereby enhancing their practical applicability, resilience, and long-term effectiveness in dynamic and continually evolving cybersecurity.

Evaluating models in live operational environments, such as future Locked Shields exercises or controlled enterprise settings, would provide practical insights beyond lab-based testing. Research could assess real-time detection performance, system stability under load, latency, alert manageability, and integration with automated response mechanisms. Such experiments would validate not only the detection accuracy and the efficacy of continuous retraining, but also the models' suitability for operational cybersecurity environments, informing future model refinement, real-time adaptation strategies, and deployment considerations.

Expanding evaluations beyond the LSPR23 and LSPR24 datasets to include additional Locked Shields datasets from earlier years (e.g., LS17, LS18, LS19, LS21) presents a promising direction to explore the generalisability of findings across multiple iterations of this prominent cyber exercise. As highlighted by Gehri et al. [58], machine learning models trained on traffic data from one year's Locked Shields exercise often fail to generalise effectively to other editions due to differences in network conditions and attack patterns. Thus, systematically evaluating models across these diverse yet related datasets could provide deeper insights into model robustness, reveal factors influencing performance stability, and clarify the conditions under which specific modelling approaches succeed or fail. Additionally, integrating multiple data modalities, such as host logs, IDS/IPS alerts, and endpoint data, into unified models represents another compelling avenue, potentially uncovering cross-domain correlations and further enhancing detection performance across varying network scenarios.

Given the demonstrated importance of feature selection, as exemplified by the impact of Inter-Arrival Time features, further research should explore automated feature engineering techniques and unsupervised representation learning approaches, such as autoencoders. These strategies could help identify invariant or robust features that consistently generalise across changing environments and over time, thereby improving model resilience and reducing sensitivity to dataset shifts.

These future research avenues collectively aim to enhance the granularity, adaptability, temporal awareness, operational realism, and generalisability of intrusion detection systems. Pursuing these directions will significantly contribute to developing robust detection mechanisms capable of maintaining high accuracy and operational effectiveness amidst the continuously evolving cyber threat landscape.

6 Conclusion

This thesis compared traditional machine learning and deep learning methods for network intrusion detection using realistic datasets from the Locked Shields 2023 and 2024 cybersecurity exercises. The literature review identified Random Forest and a one-dimensional Convolutional Neural Network as prominent representatives of their respective methodologies, supported by their widespread use and demonstrated efficacy in prior research.

Utilising the CRISP-DM framework ensured methodological rigour throughout the research process, from dataset preparation and feature engineering to model evaluation. The models were systematically trained, validated, and compared using accuracy, precision, recall, and F1-score metrics. Both the Random Forest and 1D Convolutional Neural Network achieved F1-scores exceeding 99% on the LSPR23 dataset. CNN demonstrated slightly higher recall, indicating greater sensitivity in identifying malicious network activities while conversely RF offered near-perfect precision with minimal false alarms, highlighting key trade-offs for operational decisions. Significant differences emerged in computational efficiency: RF trained approximately ten times faster and performed inference twenty times faster than CNN, also consuming substantially less GPU memory and power. Thus, RF is highly efficient for real-time or resource-constrained environments, whereas CNN provides marginal but potentially important improvements at increased computational cost.

Cross-dataset validation using the LSPR24 dataset exposed substantial generalisation challenges due to the evolving threat patterns and network environment. Both models showed performance degradation, especially RF with a significantly lowered recall. CNN being more resilient maintained higher sensitivity but produced more false alarms. These findings strongly emphasise the necessity of continual retraining or adaptive learning mechanisms to maintain intrusion detection effectiveness over time.

Future research building on top of this thesis should explore multi-class attack classification, continuous learning methods to handle evolving network configurations, and model evaluation in live, real-time operational contexts. Validation across additional Locked Shields datasets and integrating temporal, contextual, and multimodal data could further enhance practical applicability for the ever-evolving cyberthreat landscape.

References

[1] ‘What is Intrusion Detection Systems (IDS)? How does it Work?’, Fortinet. Accessed: Nov. 16, 2024. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/intrusion-detection-system

[2] Maj. E. Halisdemir, H. Karacan, M. Pihelgas, T. Lepik, and S. Cho, ‘Data Quality Problem in AI-Based Network Intrusion Detection Systems Studies and a Solution Proposal’, in *2022 14th International Conference on Cyber Conflict: Keep Moving! (CyCon)*, May 2022, pp. 367–383. doi: 10.23919/CyCon55549.2022.9811014.

[3] A. Dijk, E. Halisdemir, C. Melella, A. Schu, M. Pihelgas, and R. Meier, ‘LSPR23: A novel IDS dataset from the largest live-fire cybersecurity exercise’, *J. Inf. Secur. Appl.*, vol. 85, p. 103847, Sep. 2024, doi: 10.1016/j.jisa.2024.103847.

[4] S. Thapa and A. M. Dissanayaka, ‘The Role of Intrusion Detection/Prevention Systems in Modern Computer Networks: A Review’, 2020. Accessed: Mar. 30, 2025. [Online]. Available: https://www.semanticscholar.org/paper/The-Role-of-Intrusion-Detection-Prevention-Systems-Thapa-Dissanayaka/0f572624baf9404887ba55b4a2baebbf5a03d015

[5] K. Scarfone, P. Mell, P. Stavroulakis, and M. Stamp, ‘Intrusion Detection and Prevention Systems’, in *Handbook of Information and Communication Security*, Springer, 2010, pp. 177–192. doi: 10.1007/978-3-642-04117-4_9.

[6] K. A. Scarfone and P. M. Mell, ‘Guide to Intrusion Detection and Prevention Systems (IDPS)’, National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007. doi: 10.6028/NIST.SP.800-94.

[7] GuardRails, ‘False Positives and False Negatives in Information Security’, GuardRails. Accessed: Mar. 30, 2025. [Online]. Available: https://www.guardrails.io/blog/false-positives-and-false-negatives-in-information-security/

[8] ‘Understanding False Negatives in Cybersecurity’, Check Point Software. Accessed: Mar. 29, 2025. [Online]. Available: https://www.checkpoint.com/cyber-hub/cyber-security/understanding-false-negatives-in-cybersecurity/

[9] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, ‘Network intrusion detection system: A systematic study of machine learning and deep learning approaches’, *Trans Emerg Telecommun Technol*, vol. 32, no. 1, Jan. 2021, doi: 10.1002/ett.4150.

[10] M. Lanvin, P.-F. Gimenez, Y. Han, F. Majorczyk, L. Mé, and E. Totel, ‘Towards Understanding Alerts raised by Unsupervised Network Intrusion Detection Systems’, in *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, in RAID ’23. New York, NY, USA: Association for Computing Machinery, Oct. 2023, pp. 135–150. doi: 10.1145/3607199.3607247.

[11] H. Liu and B. Lang, ‘Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey’, *Appl. Sci.*, vol. 9, no. 20, Art. no. 20, Jan. 2019, doi: 10.3390/app9204396.

[12] CCDCOE, ‘Locked Shields’. Accessed: Mar. 29, 2025. [Online]. Available: https://ccdcoe.org/exercises/locked-shields/

[13] CCDCOE, ‘World’s largest cyber defense exercise Locked Shields kicks off in Tallinn’. Accessed: Mar. 29, 2025. [Online]. Available: https://ccdcoe.org/news/2023/worlds-largest-cyber-defense-exercise-locked-shields-kicks-off-in-tallinn/

[14] R. Meier, A. Lavrenovs, K. Heinaaro, L. Gambazzi, and V. Lenders, ‘Towards an AI-powered Player in Cyber Defence Exercises’, in *2021 13th International Conference on Cyber Conflict (CyCon)*, Tallinn, Estonia: IEEE, May 2021, pp. 309–326. doi: 10.23919/CyCon51939.2021.9467801.

[15] S. Sharma, N. Becker, B. Tepera, and D. G. Dessavre, ‘NVIDIA cuML Brings Zero Code Change Acceleration to scikit-learn’, NVIDIA Technical Blog. Accessed: May 17, 2025. [Online]. Available: https://developer.nvidia.com/blog/nvidia-cuml-brings-zero-code-change-acceleration-to-scikit-learn/

[16] V. Phillips and E. Barker, ‘Systematic reviews: Structure, form and content’, *J. Perioper. Pract.*, vol. 31, no. 9, pp. 349–353, Sep. 2021, doi: 10.1177/1750458921994693.

[17] M. J. Page *et al.*, ‘PRISMA 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews’, *BMJ*, vol. 372, p. n160, Mar. 2021, doi: 10.1136/bmj.n160.

[18] M. Gusenbauer and N. R. Haddaway, ‘Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources’, *Res. Synth. Methods*, vol. 11, no. 2, pp. 181–217, 2020, doi: 10.1002/jrsm.1378.

[19] B. Yu *et al.*, ‘Deep Learning or Classical Machine Learning? An Empirical Study on Log-Based Anomaly Detection’, in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, in ICSE ’24. New York, NY, USA: Association for Computing Machinery, Feb. 2024, pp. 1–13. doi: 10.1145/3597503.3623308.

[20] S. Li, Y. Lu, and J. Li, ‘Can We Half the Work with Double Results: Rethinking Machine Learning Algorithms for Network Intrusion Detection System’, in *2021 Ninth International Conference on Advanced Cloud and Big Data (CBD)*, Mar. 2022, pp. 242–247. doi: 10.1109/CBD54617.2021.00049.

[21] J. Hu, X. Wang, and Y. Liu, ‘Enhancing Network Intrusion Detection: A Comparative Analysis of Machine Learning Models on Complex Network Traffic Data’, in *Proceedings of the 2024 2nd International Conference on Internet of Things and Cloud Computing Technology*, Paris France: ACM, Sep. 2024, pp. 364–368. doi: 10.1145/3702879.3702942.

[22] T. Liu *et al.*, ‘A Hybrid Supervised Learning Approach for Intrusion Detection Systems’, in *Knowledge and Systems Sciences*, J. Chen, V.-N. Huynh, X. Tang, and J. Wu, Eds., Singapore: Springer Nature, 2023, pp. 3–17. doi: 10.1007/978-981-99-8318-6_1.

[23] S. Saif, Ansari ,Aqeef Alim, Biswas ,Suparna, and D. and Giri, ‘A comprehensive analysis of machine learning-based intrusion detection systems: evaluating datasets and algorithms for internet of things’, *J. Cyber Secur. Technol.*, vol. 0, no. 0, pp. 1–27, Oct. 2023, doi: 10.1080/23742917.2024.2447124.

[24] S. Chauhan, L. Mahmoud, S. Gangopadhyay, and A. K. Gangopadhyay, ‘A Comparative Study of LAD, CNN and DNN for Detecting Intrusions’, in *Intelligent Data Engineering and Automated Learning – IDEAL 2022*, H. Yin, D. Camacho, and P. Tino, Eds., Cham: Springer International Publishing, 2022, pp. 443–455. doi: 10.1007/978-3-031-21753-1_43.

[25] R. Garg and S. Mukherjee, ‘A comparative study using supervised learning for anomaly detection in network traffic’, *J. Phys. Conf. Ser.*, vol. 2161, no. 1, p. 012030, Jan. 2022, doi: 10.1088/1742-6596/2161/1/012030.
[26] S. Gnanasivam, D. Tveter, and N. Dinh, ‘Performance Evaluation of Network Intrusion Detection Using Machine Learning’, in *2024 IEEE International Conference on Consumer Electronics (ICCE)*, Jan. 2024, pp. 1–6. doi: 10.1109/ICCE59016.2024.10444363.
[27] A. Attia, M. Faezipour, and A. Abuzneid, ‘Network Intrusion Detection with XGBoost and Deep Learning Algorithms: An Evaluation Study’, in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2020, pp. 138–143. doi: 10.1109/CSCI51800.2020.00031.
[28] R. A. Bridges *et al.*, ‘Beyond the Hype: An Evaluation of Commercially Available Machine Learning–based Malware Detectors’, *Digit. Threats Res. Pract.*, vol. 4, no. 2, pp. 1–22, Jun. 2023, doi: 10.1145/3567432.
[29] M. Leon, T. Markovic, and S. Punnekkat, ‘Feature encoding with autoencoder and differential evolution for network intrusion detection using machine learning’, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Boston Massachusetts: ACM, Jul. 2022, pp. 2152–2159. doi: 10.1145/3520304.3534009.
[30] O. Faraj, D. Megias, and J. Garcia-Alfaro, ‘ZW-IDS: Zero-Watermarking-based network Intrusion Detection System using data provenance’, in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, in ARES ’24. New York, NY, USA: Association for Computing Machinery, Jul. 2024, pp. 1–11. doi: 10.1145/3664476.3670933.
[31] M. Pawlicki, R. Kozik, and M. Choraś, ‘Comparison of neural network paradigms for their usefulness in a real-world network intrusion detection deployment and a proposed optimised approach’, in *EICC 2022: Proccedings of the European Interdisciplinary Cybersecurity Conference*, Barcelona Spain: ACM, Jun. 2022, pp. 53–56. doi: 10.1145/3528580.3532840.
[32] F. Alotaibi and S. Maffeis, ‘Mateen: Adaptive Ensemble Learning for Network Anomaly Detection’, in *The 27th International Symposium on Research in Attacks, Intrusions and Defenses*, Padua Italy: ACM, Sep. 2024, pp. 215–234. doi: 10.1145/3678890.3678901.
[33] F. Alshuaibi, F. Alshamsi, A. Saeed, and S. Kaddoura, ‘Machine Learning-Based Classification Approach for Network Intrusion Detection System’, in *2024 15th Annual Undergraduate Research Conference on Applied Computing (URC)*, Apr. 2024, pp. 1–6. doi: 10.1109/URC62276.2024.10604566.
[34] S. M. Nour and S. A. Said, ‘Deep Learning Performance Evaluation Model for Enhancing Network Intrusion Detection Systems’, in *2024 6th Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, Oct. 2024, pp. 61–65. doi: 10.1109/NILES63360.2024.10753184.
[35] M. Saied, S. Guirguis, and M. Madbouly, ‘A Comparative Study of Using Boosting-Based Machine Learning Algorithms for IoT Network Intrusion Detection’, *Int. J. Comput. Intell. Syst.*, vol. 16, no. 1, p. 177, Nov. 2023, doi: 10.1007/s44196-023-00355-x.
[36] S. Layeghy and M. Portmann, ‘Explainable Cross-domain Evaluation of ML-based Network Intrusion Detection Systems’, *Comput. Electr. Eng.*, vol. 108, p. 108692, May 2023, doi: 10.1016/j.compeleceng.2023.108692.
[37] B. Natarajan, S. Bose, N. Maheswaran, G. Logeswari, and T. Anitha, ‘A Survey: An Effective Utilization of Machine Learning Algorithms in IoT Based Intrusion

Detection System', in *2023 12th International Conference on Advanced Computing (ICoAC)*, Aug. 2023, pp. 1–7. doi: 10.1109/ICoAC59537.2023.10249672.
[38] T. Hariguna and A. R. Hananto, 'Improved Intrusion Detection System (IDS) performance using Machine Learning: A Comparative Study of Single Classifier and Ensemble Learning', in *2022 IEEE Creative Communication and Innovative Technology (ICCIT)*, Nov. 2022, pp. 1–7. doi: 10.1109/ICCIT55355.2022.10118993.
[39] I. A. Najm *et al.*, 'Enhanced Network Traffic Classification with Machine Learning Algorithms', in *Proceedings of the Cognitive Models and Artificial Intelligence Conference*, İstanbul Turkiye: ACM, May 2024, pp. 322–327. doi: 10.1145/3660853.3660935.
[40] H. Chindove and D. Brown, 'Adaptive Machine Learning Based Network Intrusion Detection', in *Proceedings of the International Conference on Artificial Intelligence and its Applications*, in icARTi '21. New York, NY, USA: Association for Computing Machinery, Dec. 2021, pp. 1–6. doi: 10.1145/3487923.3487938.
[41] M. Leon, T. Markovic, and S. Punnekkat, 'Comparative Evaluation of Machine Learning Algorithms for Network Intrusion Detection and Attack Classification', in *2022 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2022, pp. 01–08. doi: 10.1109/IJCNN55064.2022.9892293.
[42] R. A. Disha and S. Waheed, 'A Comparative study of machine learning models for Network Intrusion Detection System using UNSW-NB 15 dataset', in *2021 International Conference on Electronics, Communications and Information Technology (ICECIT)*, Sep. 2021, pp. 1–5. doi: 10.1109/ICECIT54077.2021.9641471.
[43] J. Jeyasoundari, S. Sridevi, and S. C. Raja, 'Comparison of Supervised Machine Learning Algorithms with Bagged-Ensemble Method for Intrusion Detection', in *2024 IEEE Students Conference on Engineering and Systems (SCES)*, Jun. 2024, pp. 1–6. doi: 10.1109/SCES61914.2024.10652494.
[44] S. Altamimi and Q. Abu Al-Haija, 'Maximizing intrusion detection efficiency for IoT networks using extreme learning machine', *Discov. Internet Things*, vol. 4, no. 1, p. 5, Jul. 2024, doi: 10.1007/s43926-024-00060-x.
[45] V. Joshi and J. Korah, 'Formulating Parallel Supervised Machine Learning Designs For Anomaly-Based Network Intrusion Detection in Resource Constrained Use Cases', in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, Oct. 2022, pp. 748–753. doi: 10.1109/MASS56207.2022.00117.
[46] S. Gamage and J. Samarabandu, 'Deep learning methods in network intrusion detection: A survey and an objective comparison', *J. Netw. Comput. Appl.*, vol. 169, p. 102767, Nov. 2020, doi: 10.1016/j.jnca.2020.102767.
[47] A. Meliboev, J. Alikhanov, and W. Kim, 'Performance Evaluation of Deep Learning Based Network Intrusion Detection System across Multiple Balanced and Imbalanced Datasets', *Electronics*, vol. 11, no. 4, Art. no. 4, Jan. 2022, doi: 10.3390/electronics11040515.
[48] Abubucker. S. Shaffi, J. V. Chacko, G. Eliyan, and S. Balaji, 'A study on Anomaly-based Intrusion Detection Systems Employing Supervised Deep Learning Techniques', in *2024 8th International Conference on Inventive Systems and Control (ICISC)*, Jul. 2024, pp. 366–370. doi: 10.1109/ICISC62624.2024.00069.
[49] B. N. Shaker, B. Q. Al-Musawi, and M. F. Hassan, 'A Comparative Study of IDS-Based Deep Learning Models for IoT Network', in *Proceedings of the 2023*

International Conference on Advances in Artificial Intelligence and Applications, Wuhan China: ACM, Nov. 2023, pp. 15–21. doi: 10.1145/3603273.3635058.
[50] N. Chaibi, B. Atmani, and M. Mokaddem, 'Deep Learning Approaches to Intrusion Detection: A new Performance of ANN and RNN on NSL-KDD', in *Proceedings of the 1st International Conference on Intelligent Systems and Pattern Recognition*, Virtual Event Tunisia: ACM, Oct. 2020, pp. 45–49. doi: 10.1145/3432867.3432889.
[51] Y. Li, 'Network Anomaly Detection Algorithm Based on Deep Learning and Data Mining', in *Proceedings of the 2024 3rd International Conference on Cryptography, Network Security and Communication Technology*, Harbin China: ACM, Jan. 2024, pp. 220–225. doi: 10.1145/3673277.3673316.
[52] T. Talaei Khoei and N. Kaabouch, 'A Comparative Analysis of Supervised and Unsupervised Models for Detecting Attacks on the Intrusion Detection Systems', *Information*, vol. 14, no. 2, Art. no. 2, Feb. 2023, doi: 10.3390/info14020103.
[53] Md. A. Uddin, S. Aryal, M. R. Bouadjenek, M. Al-Hawawreh, and Md. A. Talukder, 'A dual-tier adaptive one-class classification IDS for emerging cyberthreats', *Comput. Commun.*, vol. 229, p. 108006, Jan. 2025, doi: 10.1016/j.comcom.2024.108006.
[54] A. R. Tapsoba and T. Frédéric OUEDRAOGO, 'Evaluation of supervised learning algorithms in binary and multi-class network anomalies detection', in *2021 IEEE AFRICON*, Sep. 2021, pp. 1–6. doi: 10.1109/AFRICON51333.2021.9570886.
[55] D. Han *et al.*, 'Rules Refine the Riddle: Global Explanation for Deep Learning-Based Anomaly Detection in Security Applications', in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, Salt Lake City UT USA: ACM, Dec. 2024, pp. 4509–4523. doi: 10.1145/3658644.3670375.
[56] N. L. Rane, J. Rane, S. K. Mallick, and Ö. Kaya, *Scalable and adaptive deep learning algorithms for large-scale machine learning systems*. Deep Science Publishing, 2024. doi: 10.70593/978-81-981271-0-5.
[57] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, 'SMOTE: synthetic minority over-sampling technique', *J Artif Int Res*, vol. 16, no. 1, pp. 321–357, Jun. 2002.
[58] L. Gehri, R. Meier, D. Hulliger, and V. Lenders, 'Towards Generalizing Machine Learning Models to Detect Command and Control Attack Traffic', in *2023 15th International Conference on Cyber Conflict: Meeting Reality (CyCon)*, Tallinn, Estonia: IEEE, May 2023, pp. 253–271. doi: 10.23919/CyCon58705.2023.10182001.
[59] 'CRISP-DM Help Overview'. Accessed: Mar. 16, 2025. [Online]. Available: https://www.ibm.com/docs/en/spss-modeler/saas?topic=dm-crisp-help-overview
[60] T. W. Edgar and D. O. Manz, 'Chapter 11 - Applied Experimentation', in *Research Methods for Cyber Security*, T. W. Edgar and D. O. Manz, Eds., Syngress, 2017, pp. 271–297. doi: 10.1016/B978-0-12-805349-2.00011-X.
[61] F. Martínez-Plumed *et al.*, 'CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories', *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 8, pp. 3048–3061, Aug. 2021, doi: 10.1109/TKDE.2019.2962680.
[62] E. Sügis, A. Tampuu, A. Aljanaki, M. Fišel, and M. Kull, *Praktiline andmeteadus Kõrgkooliõpik*, 2nd ed. Tartu, Estonia: Tartu Ülikooli arvutiteaduse instituut, 2025. Accessed: Mar. 16, 2025. [Online]. Available: https://courses.cs.ut.ee/t/andmeteadus/
[63] N. Känzig, R. Meier, L. Gambazzi, V. Lenders, and L. Vanbever, 'Machine Learning-based Detection of C&C Channels with a Focus on the Locked Shields

Cyber Defense Exercise', in *2019 11th International Conference on Cyber Conflict (CyCon)*, May 2019, pp. 1–19. doi: 10.23919/CYCON.2019.8756814.
[64] 'NVIDIA GeForce 3090/Ti For AI Software - Still Worth It? - Tech Tactician'. Accessed: Mar. 29, 2025. [Online]. Available: https://techtactician.com/nvidia-geforce-3090-ti-for-local-ai-software-use/
[65] B. Sangster *et al.*, 'Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets', presented at the 2nd Workshop on Cyber Security Experimentation and Test (CSET 09), 2009. Accessed: Mar. 29, 2025. [Online]. Available: https://www.usenix.org/conference/cset-09/toward-instrumenting-network-warfare-competitions-generate-labeled-datasets
[66] M. Ahmed, A. Naser Mahmood, and J. Hu, 'A survey of network anomaly detection techniques', *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, Jan. 2016, doi: 10.1016/j.jnca.2015.11.016.
[67] A. Dijk, E. Halisdemir, C. Melella, A. Schu, M. Pihelgas, and R. Meier, 'Locked Shields Partners Run 23 (LSPR23): A novel IDS dataset from the largest live-fire cybersecurity exercise'. Zenodo, Aug. 06, 2024. doi: 10.5281/zenodo.8042347.
[68] A. Dijk, R. Meier, C. Melella, and M. Pihelgas, 'Locked Shields Partners Run 24 (LSPR24): A Next-Generation Cybersecurity Dataset for Blue Team Automation', doi: 10.5281/zenodo.14900873.
[69] B. Leo, 'OSI Model Layers and Protocols in Computer Network', GURU99. Accessed: Apr. 09, 2025. [Online]. Available: https://www.guru99.com/layers-of-osi-model.html
[70] R. Dube, 'Faulty use of the CIC-IDS 2017 dataset in information security research', *J. Comput. Virol. Hacking Tech.*, vol. 20, no. 1, pp. 203–211, 2024, doi: 10.1007/s11416-023-00509-7.
[71] Y. Merkli, R. Meier, M. Strohmeier, and V. Lenders, 'Defeating and Improving Network Flow Classifiers Through Adversarial Machine Learning', in *2024 16th International Conference on Cyber Conflict: Over the Horizon (CyCon)*, May 2024, pp. 103–121. doi: 10.23919/CyCon62501.2024.10685592.
[72] L. Bhuva, 'Handling Missing Values in Categorical Data: Techniques and Best Practices', Medium. Accessed: Apr. 20, 2025. [Online]. Available: https://medium.com/@lomashbhuva/handling-missing-values-in-categorical-data-techniques-and-best-practices-0a3ddd523824
[73] 'Tabular model', fastai. Accessed: Apr. 20, 2025. [Online]. Available: https://docs.fast.ai/tabular.model.html
[74] R. Vaarandi and A. Guerra-Manzanares, 'Network IDS alert classification with active learning techniques', *J. Inf. Secur. Appl.*, vol. 81, p. 103687, Mar. 2024, doi: 10.1016/j.jisa.2023.103687.
[75] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, 'A survey of network-based intrusion detection data sets', *Comput. Secur.*, vol. 86, pp. 147–167, Sep. 2019, doi: 10.1016/j.cose.2019.06.005.
[76] D. Choudhary, 'Bootstrapping and OOB samples in Random Forests', Analytics Vidhya. Accessed: Apr. 20, 2025. [Online]. Available: https://medium.com/analytics-vidhya/bootstrapping-and-oob-samples-in-random-forests-6e083b6bc341
[77] D. Kilichev and W. Kim, 'Hyperparameter Optimization for 1D-CNN-Based Network Intrusion Detection Using GA and PSO', *Mathematics*, vol. 11, no. 17, Art. no. 17, Jan. 2023, doi: 10.3390/math11173724.
[78] K. Singh, A. Mahajan, and V. Mansotra, '1D-CNN based Model for Classification and Analysis of Network Attacks', *Int. J. Adv. Comput. Sci. Appl.*

IJACSA, vol. 12, no. 11, Art. no. 11, Jan. 2021, doi: 10.14569/IJACSA.2021.0121169.
[79] A. Dijk, *Code Example for LSPR23*. (Oct. 14, 2024). Python. Accessed: Nov. 17, 2024. [Online]. Available: https://github.com/scilicet64/LSPR23
[80] R. Thomas, 'Rachel Thomas, PhD - An Introduction to Deep Learning for Tabular Data'. Accessed: Apr. 20, 2025. [Online]. Available: https://rachel.fast.ai/posts/2018-04-29-categorical-embeddings/
[81] *nvidia-ml-py: Python Bindings for the NVIDIA Management Library*. Python. Accessed: May 04, 2025. [Microsoft :: Windows, POSIX :: Linux]. Available: https://forums.developer.nvidia.com
[82] S. Friedler and J. Wilson, 'Methodology', Codecarbon. [Online]. Available: https://mlco2.github.io/codecarbon/methodology.html
[83] S.-W. Kim, J. Jin-Sung Lee, V. Dugar, and J. De Vega, 'Using the Intel® Power Gadget 3.0 API on Windows*', Intel. Accessed: May 04, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/training/using-the-intel-power-gadget-30-api-on-windows.html
[84] 'Intel® Core™ i5-13500 Processor (24M Cache, up to 4.80 GHz) - Product Specifications', Intel. Accessed: May 17, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/sku/230580/intel-core-i513500-processor-24m-cache-up-to-4-80-ghz/specifications.html
[85] '3090 & 3090 Ti Graphics Cards'. Accessed: May 17, 2025. [Online]. Available: https://www.nvidia.com/en-eu/geforce/graphics-cards/30-series/rtx-3090-3090ti/
[86] G. P. Gupta and M. Kulariya, 'A Framework for Fast and Efficient Cyber Security Network Intrusion Detection Using Apache Spark', *Procedia Comput. Sci.*, vol. 93, pp. 824–831, Jan. 2016, doi: 10.1016/j.procs.2016.07.238.
[87] 'NVIDIA Triton Inference Server — NVIDIA Triton Inference Server'. Accessed: Apr. 20, 2025. [Online]. Available: https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/index.html
[88] N. Burkart and M. F. Huber, 'A Survey on the Explainability of Supervised Machine Learning', *J. Artif. Intell. Res.*, vol. 70, pp. 245–317, Jan. 2021, doi: 10.1613/jair.1.12228.
[89] C. Molnar, *Interpretable Machine Learning*. 2020. [Online]. Available: https://christophm.github.io/interpretable-ml-book/
[90] G. Plumb, D. Molitor, and A. Talwalkar, 'Model Agnostic Supervised Local Explanations', Jan. 05, 2019, *arXiv*: arXiv:1807.02910. doi: 10.48550/arXiv.1807.02910.
[91] 'EU AI Act: first regulation on artificial intelligence', Topics | European Parliament. Accessed: Apr. 20, 2025. [Online]. Available: https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence
[92] R. Jordaney *et al.*, 'Transcend: Detecting Concept Drift in Malware Classification Models', presented at the 26th USENIX Security Symposium (USENIX Security 17), 2017, pp. 625–642. Accessed: Apr. 20, 2025. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/jordaney

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Johan Valdemar Leoste

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Comparative Analysis of Deep Learning and Machine Learning for Network Intrusion Detection Using Data from the Largest Live-Fire Cyber Defence Exercise”, supervised by Risto Vaarandi and Allard Dijk.
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

18.05.2025

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Literature review table

Ref.	Authors	Date	ML / DL	Datasets Used	Models Used	Best Model	Model Acc	Libraries/ Frameworks	Code Repo
[26]	S. Gnasivam et al.	2024	ML	UNSW-NB15	ANN, RF, DT, KNN, SVM, LR	DT	85+%	Python, Jupyter Lab, Visual Studio Code, Anaconda	x
[28]	R. A. Bridges et al.	2023	ML	VirusShare, Custom Polyglot, Zero-day, APT, Hyperion	RF, proprietary models	Net Dynamic	x	x	link
[32]	F. Alotaibi, S. Maffeis	2024	DL	CICIDS2017, CSE-CICIDS2018, Kitsune, mKitsune, rKitsune	DAE-based ensemble (“Mateen”)	Mateen	95+%	PyTorch, Python, CUDA	link
[37]	B. Natarajan et al.	2023	ML	KDD Cup ’99	CART (DT), LDA, RF	RF	99+%	Apache Spark, Spark MLlib	x
[33]	F. Alshuaibi et al.	2024	ML	USAF LAN simulation	LR, SVM, KNN, RF, XGBoost	RF	99+%	Google Colab, Python	x
[20]	S. Li, Y. Lu, J. Li	2022	ML	CIC-IDS2017, NSL-KDD	SVM, RF, KNN, GNB, DT, Bagging, GraTree, MLP, LR, SGD, ridge, ridgeCV, MNB, CNB, BNB	DT	99+%	x	x
[38]	T. Hariguna, A. R. Hananto	2022	ML	NSL-KDD	NB, SVM, RF, Ensemble	Ensemble (SVM + RF)	90+%	x	x

[39]	I. A. Najm et al.	2024	ML	Computer Network Traffic Data (IOT)	DT, RF, SVM, RNN	RF	95+%	x	x
[29]	M. Leon et al.	2022	ML	UNSW-NB15	RF, SVM, LDA, ANN, KNN, K-means	RF	95+%	MATLAB	x
[21]	J. Hu, X. Wang, Y. Liu	2024	ML	Kaggle HydraS Network Traffic	LSTM, RF, Isolation Forest, GBM, XGBoost	GBM / XGBoost	99+%	Python, Keras, TensorFlow, scikit-learn	x
[40]	H. Chindove, D. Brown	2021	Both	CIC-IDS2017; CIC-IDS2018	RF, DT, KNN, SVM, MLP, RNN	ML: RF	~99+%	Python	x
						DL: RNN	99+%		
[41]	M. Leon et al.	2022	ML	KDD99, NSL-KDD, UNSW-NB15, CIC-IDS-2017	ANN, SVM, RF, LDA, K-NN, K-means, Mean-shift, DBSCAN	RF	99+%	MATLAB	x
[42]	R. A. Disha, S. Waheed	2021	ML	UNSW-NB15	DT, RF, GBT, MLP	DT	90+%	Python, NumPy, scikit-learn	x
[43]	J. Jeyasoundari et al.	2024	ML	KDDCUP 99	LR, KNN, NB, SVM, RF	RF	99+%	Python, Google Colab	x
[22]	T. Liu et al.	2023	ML	Edge-IIoTset	DT, RF, ET, XGBoost, Stacking	XGBoost	95+%	Python, scikit-learn, XGBoost, Optuna	x
[25]	R. Garg, S. Mukherjee	2022	ML	NSL-KDD, CSE-CIC-IDS2018	DT, RF, K-NN, SVM, LR, XGBoost, NB	RF	80+%	Python	x
[44]	S. Altamimi,Q. Abu Al-Haija	2024	ML	NSL-KDD, Distilled-Kitsune	ELM, KNN, DT, RF	RF	99+%	Python, Google Colab, StandardScaler, SMOTE, scikit-learn	x
[23]	S. Saif et al.	2024	Both	15 IoT/Network datasets (ToN_IoT, CIC-IDS, KDD99, UNSW-NB15, IoT-Botnet, …)	kNN, RF, DT, NB, ANN, AE	ML: kNN	95+%	Python, scikit-learn	x
						DL: ANN	99+%		

[36]	S. Layeghy, M. Portmann	2023	Both	NFv2-BoT-IoT, NFv2-CIC-2018, NFv2-ToN-IoT, NFv2-UNSW-NB15	ET, RF, Feed Forward NN, LSTM, IsolationForest, oSVM, SGD-oSVM, AE	ML: RF	x	VMware ESXi, Python, scikit-learn, TensorFlow, SHAP	x
						DL: LSTM	x		
[19]	B. Yu et al.	2024	Both	HDFS, BGL, Spirit, Thunderbird, Liberty	KNN, DT, SLFN; CNN, LogRobust (Bi-LSTM), NeuralLog (Transformer)	ML: KNN	x	sklearn, Drain, Loglizer, BERT	link
						DL: Transformer	x		
[30]	O. Faraj, D. Megías, J. Garcia-Alfaro	2024	ML	CIC-IDS2017	SVM (Linear, Polynomial, RBF, Sigmoid), ExtraTrees	SVM (RBF)	99+%	Python, sklearn	x
[35]	M. Saied, S. Guirguis, M. Madbouly	2023	ML	N-BaIoT (Mirai IoT botnet)	ADB, GDB, XGB, CAB, HGB, LGB	HGB	99+%	Python, sklearn, XGBoost, CatBoost, LightGBM	link
[46]	S. Gamage, J. Samarabandu	2020	DL	KDD 99, NSL-KDD, CIC-IDS2017, CIC-IDS2018	ANN, AE, DBN, LSTM, RF	ANN (Deep NN)	99+%	x	x
[27]	A. Attia, M. Faezipour, A. Abuzneid	2020	DL	Kitsune (IoT network)	XGBoost, ANN	XGBoost	99+%	x	x
[34]	S. M. Nour, S. A. Said	2024	DL	CICIDS2017	CNN, DNN, RNN (LSTM), Attention	CNN	95+%	Google Colab, Python	x
[47]	A. Meliboev, J. Alikhanov, W. Kim	2022	DL	UNSW_NB15, KDD_cup99, NSL-KDD	CNN, LSTM, RNN, GRU, CNN+LSTM	CNN	95+%	Keras, TensorFlow, NumPy, scikit-learn, Pandas	x
[31]	M. Pawlicki, R. Kozik, M. Choraś	2022	DL	CICIDS2017	DNN, CNN, LSTM (RNN)	LSTM (RNN)	95+%	Python, TensorFlow	x

[48]	A. S. Shaffi et al.	2024	DL	NSL-KDD, KDDcup99	CNN, RNN	RNN	85+%	TensorFlow, Python	x
[49]	B. Shaker et al.	2023	DL	NF-BoT-IoT, NF-ToN-IoT, NF-UNSW-NB15	RNN, CNN, DNN	DNN	95+%	x	x
[50]	N. Chaibi et al.	2020	DL	NSL-KDD	ANN, RNN (LSTM)	RNN (LSTM)	99+%	Python, Keras, TensorFlow, scikit-learn	x
[51]	Y. Li	2024	DL	KDD Cup 99	CNN, RNN, DBSCAN (combined)	CNN+RNN+DBSCAN (hybrid)	80+%	x	x
[24]	S- Chauhan et al.	2022	DL	UNSW-NB15, CSE-CIC-IDS2018	LAD, CNN, DNN	LAD (UNSW) / CNN & DNN (CIC)	95+%	Keras, TensorFlow	x
[52]	T. Talaei Khoei, N. Kaabouch	2023	DL	CIC-DDoS2019	GNB, DT, LR, C-SVM, LightGBM, AlexNet + PCA, K-means, VAE	AlexNet	95+%	ADAM optimiser	x
[53]	M. A. Uddin et al.	2025	DL	NSL-KDD, UNSW-NB15, CIC-DoS2017, CIC-DDoS2019, Darknet2020, MalMem2022, X-IIoTID, ToN-IoT (Net/Linux), ISCX-URL2016	AE; (LOF, IOF, OCSVM, usfAD)	usfAD	95+%	Python, Pandas, NumPy, scikit-learn	x
[54]	A. R. Tapsoba et al.	2021	DL	NSL-KDD	MLP, SVM, KNN, RF, DT, LR	MLP	80+%	scikit-learn	x