TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Frank Christopher Kirch 221925IVCM

Federated Learning-based Intrusion Detection for Cyberattack Detection in Healthcare IoMT Networks

Master's Thesis

Supervisor: Hayretdin Bahsi
PhD

Co-Supervisor: Rajesh Kalakoti
PhD

Tallinn 2026

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Frank Christopher Kirch 221925IVCM

Liitõppel põhinev sissetungi tuvastamine tervishoiu IoMT võrkude küberrünnakute avastamiseks

Magistritöö

Juhendaja: Hayretdin Bahsi

PhD

Kaasjuhendaja: Rajesh Kalakoti

PhD

Tallinn 2026

Author’s declaration of originality

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

Author: Frank Christopher Kirch

04.01.2026

Abstract

The widespread adoption of Internet of Medical Things (IoMT) devices in healthcare environments introduces significant cybersecurity challenges, as sensitive network traffic is generated under strict data-protection regulations that restrict data sharing across institutions. These constraints limit the applicability of centralized intrusion detection systems and motivate the use of privacy-preserving learning approaches such as Federated Learning (FL).

This thesis evaluates Federated Learning–based intrusion detection systems (FL-IDS) for healthcare IoMT networks, with a focus on the impact of server-side optimization methods under heterogeneous data distributions. A Gated Recurrent Unit (GRU)–based deep learning model was trained in a cross-silo horizontal federated learning setting and evaluated on the IoMT-TrafficData and CICIoMT2024 datasets for both binary and multiclass intrusion detection tasks. Non-identically distributed client data was simulated using Dirichlet-based partitioning.

Experimental results show that adaptive server-side optimization methods outperform Federated Averaging in Non-IID healthcare settings. In binary intrusion detection, the Federated Optimization algorithm (FedOpt) achieved the highest performance, with F1-scores of 99.67

Overall, the findings indicate that adaptive federated optimization is well suited for intrusion detection in healthcare IoMT networks, where strict privacy requirements and heterogeneous data distributions necessitate decentralized learning approaches.

The thesis is in English and contains 84 pages of text, 9 chapters, 43 figures, 5 tables.

Lühikokkuvõte

Liitõppel põhinev sissetungi tuvastamine tervishoiu IoMT võrkude küberrünnakute avastamiseks

Tervishoiu asjade interneti (IoMT) seadmete laialdane kasutuselevõtt on toonud kaasa märkimisväärseid küberturvalisuse väljakutseid. Tervishoiu IoMT-võrkudes kehtivad ranged andmekaitsenõuded takistavad tundliku võrguliikluse tsentraliseeritud kogumist ja jagamist tervishoiuasutuste vahel. Sellest tulenevalt on traditsiooniliste tsentraliseeritud sissetungituvastussüsteemide kasutamine piiratud, mistõttu suureneb vajadus privaatsust säilitavate liitõppel (*Federated Learning*, FL) põhinevate lahenduste järele

Käesolev magistritöö hindab liitõppel põhinevaid sissetungituvastussüsteeme (FL-IDS) tervishoiu IoMT võrkudes, keskendudes serveripoolsete optimeerimismeetodite mõjule heterogeensete andmejaotuste korral. *Gated Recurrent Unit* (GRU) tüüpi süvaõppemudel treeniti rist-silo horisontaalse liitõppe raamistikus ning hinnati IoMT-TrafficData ja CICIoMT2024 andmestikel nii binaarse kui ka mitmeklassilise sissetungituvastuse ülesannetes. Klientidevahelise mitteühtlase andmejaotuse modelleerimiseks kasutati Dirichlet' jaotusel põhinevat andmete jaotamist.

Tulemused näitavad, et adaptiivsed serveripoolsed optimeerimismeetodid ületavad *Federated Averaging* algoritmi mitteidentsete (Non-IID) andmete korral tervishoiukeskkondades. Binaarse sissetungituvastuse puhul saavutas *Federated Optimization* algoritm (FedOpt) parima tulemuse, F1-skooriga 99,67% IoMT-TrafficData ja 97,27% CICIoMT2024 andmestikul. Mitmeklassilise sissetungituvastuse korral pakkus *Federated Adaptive Gradient* algoritm (FedAdagrad) tasakaalustatumat sooritust, saavutades F1-skoorid 85,44% IoMT-TrafficData ja ligikaudu 71% CICIoMT2024 andmestikul.

Kokkuvõttes näitavad saadud tulemused, et adaptiivne serveripoolne optimeerimine sobib hästi sissetungituvastuseks tervishoiu IoMT võrkudes, kus ranged privaatsusnõuded ja heterogeensed andmejaotused eeldavad detsentraliseeritud õppemeetodite kasutamist.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 84 leheküljel, 9 peatükki, 43 joonist, 5 tabelit.

List of abbreviations and terms

ANN	Artificial Neural Network
API	Application Programming Interface
ANN	Artificial Neural Network
CPU	Central Processing Unit
DL	Deep Learning
ERM	Empirical Risk Minimization
FL	Federated Learning
FedAvg	Federated Averaging Algorithm
FedAvgM	Federated Averaging Algorithm with Server Momentum
FedOpt	Federated Server-Optimization
FedAdagrad	Federated Adaptive Gradient Server-Optimization
FedAdam	Federated Adaptive Momentum Server-Optimization
FedYogi	Federated Yogi Server-Optimization
FL-IDS	Federated Learning based Intrusion Detection System
FTL	Federated Transfer Learning
GRU	Gated Recurrent Unit
HFL	Horizontal Federated Learning
IDE	Integrated Development Environment
IID data	Independent and Identically Distributed Data
IoMT	Internet Of Medical Thing
IoT	Internet Of Things
LSTM	Long Short-Term Memory
MitM	Man-in-the-Middle Attack
ML	Machine Learning
NN	Neural Networks
Non-IID data	Non-Independent and Identically Distributed Data
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SMC	Secure Multiparty Computation
VFL	Vertical Federated Learning

Table of contents

List of figures

List of tables

1 Introduction

The Internet of Things (IoT) creates a system of devices that operate and communicate without requiring human-to-human or human-to-computer interaction. When this interlinked system is introduced into healthcare, it forms Internet of Medical Things (IoMT). This infrastructure offers real-time access to patient data, enables immediate medical response, maintains continuous monitoring, and supports early diagnosis, thereby avoiding harmful consequences and delivering dependable, high-quality care.

There are several examples of healthcare services supported by IoMT devices. Among these, IoMT devices enable continuous health monitoring by rapidly deploying physiological sensors that relay real-time data to medical personnel, improving treatment outcomes and ease-of use in hospitals [1, 2]. Beyond monitoring, they also automate medication delivery through smart infusion pumps, ensuring precise dosing [3]. Equally important, remote medicine leverages these sensors and actuators to monitor patients and administer care from afar, enhancing comfort and continuous oversight [4].

Healthcare providers increasingly rely on Internet of Medical Things (IoMT) to streamline (deliver) continuous distant patient care and ensure uninterrupted service [5]. Among these services, remote medicine and alert systems employ IoMT sensors and actuators for teleconsultations, remote diagnosis and therapy, ensuring continuous care [6][7]. Critical monitoring alerts staff to falls or emergencies when patients require immediate medical attention [8] [9]. Wearable IoMT devices enable wellness tracking - heart rate, blood pressure oxygen saturation[10], and ECG [11][12] - thereby supporting preventive (cardiac) care and chronic disease management [13]. They also monitor disorder-specific metrics such as blood glucose [14] for diabetes [15] and blood pressure for hypertension [16]. Finally, IoMT-enabled infusion pumps automate medication delivery and transmit dosage data for remote oversight [17].

1.1 Research Problem

The rapid development of Internet of Things (IoT) devices has resulted in a substantial growth of cyberattacks on IoT infrastructure. To combat the challenges posed by such attacks, intrusion detection systems (IDS) are concealed in networks to detect malicious activities. Intrusion Detection Systems (IDS) can be categorized into signature-based and anomaly-based. Signature-based detection is limited to identifying known attacks by matching them to a superior collection of attack signatures [18]. This limitation is resolved with anomaly-based systems that detect unknown attacks by discovering deviations from normal system behavior, especially prominent in network attacks [19]. Anomaly-based IDS presumes large volumes of multi-dimensional data to correlate intricate patterns and relationships, a formidable challenge to complete for cybersecurity analysts and other security personnel. Machine learning can operate on such large-scale complex data to uncover recurring intrusion patterns to infer crucial information about anomalous network behavior. Intrusion detection benefits from machine learning that can reduce false positives and increase detectability [20] [21]. Therefore, ML-enhanced anomaly-based IDS reduces the burden of security personnel.

Malicious attacks are prominent against the healthcare industry due to heightened requirements for medical devices to function properly, and less tolerance to withstand malfunctions and system downtime. Disruptions in life-critical medical devices can cause deliberate harm to a patient's health and irrevocable consequences. The importance of intrusion detection of state-of-the-art malicious attacks is coupled with the intrinsic purpose of medical devices (IoMT) - to preserve human life despite patients' rare medical conditions. These critical IoMT devices are vulnerable to network attacks such as denial-of-service attacks, reconnaissance, spoofing attacks, and many other [22]. Network attacks can occur regardless of geographic location, device idle or non-idle status, date, and time. Thus, intrusion detection capabilities are relevant to all network-connected IoMT devices to detect anomalies and malicious attacks.

Anomaly-based IDS requires large-scale data to detect intricate anomalous patterns of behavior, and machine learning inherently thrives on large-scale multi-dimensional data. Commonly, a traditional machine learning model would transmit raw data from data-owners to a central device to conduct machine learning tasks [23]. Inherently, any such traditional

machine learning necessitates the transfer of raw data from IoMT devices to a central device for model learning tasks. In the healthcare industry, data pertaining to medical information or patient data must remain confidential [24]. This is enforced by international data-protection regulation and laws [24] such as GDPR in the European Union (EU) and the Cybersecurity Law of the People's Republic of China. Therefore, network intrusion detection systems cannot be enhanced with traditional centralized machine learning models due to data-protection laws, which prohibit the transfer of sensitive raw data [25]. As such, an alternative machine learning approach must the explored for intrusion detection.

To overcome this challenge, a collaborative architecture, known as federated learning (FL), was proposed in 2016 by Google [26]. In horizontal federated learning, connected devices are exempt from the transfer of raw data. Specifically, in an FL scenario, IoMT devices are participants, data-owners that participate in data collection to train a local ML model and emit model parameters (weights and biases). Various IoMT devices participate in on-site collection and transmit their local model parameters to a coordinating central device. This central server aggregates received local model parameters to the global machine learning model, thus facilitating model convergence. Subsequently, the central server converges a global machine learning model without any communication of raw data [27]. The inherently decentralized and collaboratively federated learning provides privacy-preserving capabilities to train an ML model for anomaly-based intrusion detection systems whilst complying with privacy regulations and data protection laws.

The geographically unconfined nature of IoMT devices introduces dissimilar data, including distinct feature dimensionality, different class size or proportions of benign and anomalous data, as well as imbalanced frequency of data collection. The subsequent traits of data are commonly characterized as non-independent and identically distributed (non-IID) data [28]. Prior works suggest that non-IID data is reflective of real-world IoMT scenarios and introduce a new set of challenges for federated learning. Global models receive updates from data-rich participants, and these inherently hold greater weight than model updates originated from data-scarce participants with less extensive datasets. Imbalanced parameter updates can lead the global model to oscillate and stagnate during training, impacting the convergence of the global optimal model [28]. In federated learning, system heterogeneity is characteristic due to limitations in computational resources, storage, and communication

capabilities [27]. Some IoT devices may be limited in their computational and storage capabilities and preventing them from timely updates to model parameters. As such, client devices or participants can limit the scalability of federated learning algorithms. To overcome these challenges of system heterogeneity, and more specifically, global model convergence, various optimization algorithms can be employed. Therefore, further investigation is necessary to examine and comparatively evaluate various state-of-the-art optimization techniques to build a machine learning model on non-IID data. This thesis study provided an empirical evaluation of FL optimization techniques on common metrics such as accuracy, precision, recall, and F1-score.

The outcome of the research provided decision-making capabilities to enhance intrusion detection in IoMT systems with state-of-the-art machine learning, whilst preserving sensitive data integrity and confidentiality.

A conclusive and most crucial research question can be construed as follows:

[RQ3] Which optimization techniques can improve the performance of federated learning in decentralized Internet of Medical Things (IoMT) networks for cyberattack detection?

In-depth approach to answer these research methods is examined in Chapter 4.

1.2 Scope and Goal

The prominent **goal** of this research is to enhance intrusion detection systems with machine learning to identify network attacks directed towards the Internet of Medical Things. Based on the motivation of the study, new cyber-attacks can be difficult to detect in signature-based intrusion detection systems (IDS). To combat less-known network intrusions, anomaly-based IDS can detect anomalous behaviors and identify malicious attacks. Machine learning can recognize intricate patterns and relationships in large-scale multi-dimensional data and increase anomaly detection rates. Therefore, the goal of the study is to provide empirical evaluation and comparable results of state-of-the-art optimization techniques in Federated Learning to detect network intrusions against decentralized healthcare devices in IoMT networks. To achieve the proposed goal, a literature review was conducted to explore various optimization techniques within federated learning (FL). Additionally, these various FL optimization methods were trained on two unique peer-reviewed datasets [29] [30] from

reputable journal publications collected from IoMT networks. The research outcomes were validated with validation data subsets that were excluded from model training tasks.

The research was confined to the **scope** of network intrusion attacks on Internet of Medical Things (IoMT) decentralized architecture. This thesis study did not examine host-based intrusion detection systems or privacy-preserving capabilities in decentralized, that is, horizontal federated learning scenarios. The development of health-enabling and life-prolonging IoMT devices has subsequently increased the rapid growth of network intrusions directed at these IoMT devices. To train a sophisticated FL model, existing datasets were used to find intricate patterns and relationships, and any attacks outside the scope of network intrusions could introduce underfitting or overfitting of the model. More crucially, in a decentralized architecture, the attacks outside the scope of network intrusions could hinder the convergence to a global machine learning model. Thus, to facilitate model convergence, the research was scoped to network-based cyberattacks in IoMT networks.

Furthermore, an evident **limitation** pertained to training a predictive ML-model based on one specific machine learning algorithm with different state-of-the-art optimization algorithms. Regardless, a single machine learning algorithm served a crucial role – it provided the comparative results of various optimization algorithms in model performance, including accuracy, precision, recall, and F1-score, which are all domain-specific metrics that validate the machine learning model.

This thesis makes clear **key assumptions** about the quality and validity of existing datasets. The referenced datasets were entered into the index and assumed relevance. Similarly, this research made key assumptions about the quality of machine learning and optimization algorithms, and the underlying mathematical correctness and validity. Thus, this study presumed that existing literature is peer-reviewed and valid in its results.

This thesis study was not subject to **ethical concerns** or implications. The research was conducted on publicly available peer-reviewed datasets from reputable journals about cyber-attack classification. Thus, the datasets did not disclose sensitive data about healthcare providers, patients, or other medical information. This research evaluated the performance of a machine-learning model with various optimization algorithms on those datasets. Inherently, no new sensitive datasets were collected or inferred throughout the thesis

study. Additionally, the thesis is not funded by any external organizations, and research was supervised by TalTech University professor Hayredthin Bahsi (PhD) and Early-Stage Researcher Rajesh Kalakoti (PhD).

The outcome of the research exhibited empirical evaluation of optimization algorithms with quantitative results about their performance in common machine learning metrics, specifically precision, recall, F1-score, and specificity. These performance benchmarks provided clear comparisons and identified well-performing optimization technique for model convergence to be applied in a horizontal federated learning architecture. This outcome enabled educated decision-making to select optimization algorithms in federated learning. As a result, FL-enhanced intrusion detection (FL-IDS) can be used to more reliably identify state-of-the-art new cyber-attacks directed towards IoMT devices. The serves vital importance to react to detected traits of malicious actions on life-critical health devices before patients' lives are harmed.

Further work can be explored about possible future research involving sophisticated privacy-preserving techniques and their implications on communication speed, computing resources, and latency. These privacy-preserving techniques in federated learning architectures are commonly homomorphic encryption, secure sharing, and differential privacy. Similarly, this opens an avenue for future research to examine attacks or conduct case studies of cryptoanalysis to dismantle weak implementations of cryptography.

1.3 Novelty and Contributions

This unexamined **research gap is crucial** to further progress ML-enhanced intrusion detection systems in medical devices. To fill the research gap, this thesis examined and implemented optimization algorithms to benchmark their performance in a horizontal federated learning setting to detect network intrusion attacks in IoMT devices. This cross-domain study provided performance metrics benchmarking of state-of-the-art optimization algorithms in a federated learning architecture bound in decentralized IoMT networks. The novelty of the study was also validated by the absence of prior work operating on multiple datasets to evaluate optimization techniques in federated learning.

The **main contributions** of this research were numerical measurements of the optimization

techniques in federated machine learning to improve the decision-making process for industry experts, both machine learning engineers and security engineers. This includes comparative results about multiple state-of-the-art optimization techniques for industry application within a decentralized network of IoMT devices for ML-enhanced intrusion detection systems. Additionally, this thesis contributed to the state-of-the-art of federated learning by validating the performance of optimization techniques with multiple peer-reviewed datasets published in reputable journals - the CIC-IoMT2024 dataset [29] and the IoMT-TrafficData dataset [30].

Lastly, the research contributed by providing code implementations in the Python programming language with libraries such as PyTorch, NumPy, and scikit-learn. The implementation can be used as a starting point for future work in the federated learning domain.

1.4 Thesis Structure

This thesis is organized as follows. Chapter 2 provides background information to contextualize the research. The chapter describes Intrusion Detection Systems, their categorization, and potential application. Furthermore, the chapter describes Internet of Medical Things (IoMT) and its applications in healthcare.

Chapter 3 reviews existing work, focusing on various applications of Machine-Learning (ML) and Deep Learning (DL) in FL-based intrusion detection in IoT and IoMT networks. The chapter focuses on prior research that examined model training with non-independently and identically distributed (non-IID) data, and conversely in IID data.

Chapter 5 presents the dataset preparation to conduct experiments in Federated Learning environments. These include mathematical formula to simulate data heterogeneity from the static datasets. Furthermore, systematic approach is presented to split the clients' private datasets into training, validation and testing sets across each cyberattack type.

Chapter 4 examines the methodology used in this our research. This includes common Federated Learning Algorithms, adaptive server-side optimization approaches. This includes the problem definition, assumptions, mathematical equations, and formulated pseudo-code of each server-side optimizer.

Chapter 6 unveils the results of the experiments to conclude an empirical evaluation of model performance. The chapter describes relevant findings from both binary detection of malicious activities and multiclass classification of cyberattacks. Various FL optimization techniques, including FedAvg, FedAvgM, FedAdagrad, FedAdam, FedYogi, and FedOpt algorithms evaluated and compared in terms of model performance.

Chapter 7 presents the discussion of findings, interpreting the results in the context of the research questions. It describes the challenges and limitations, as well as proposed potential improvements and directions of future work.

Lastly, **??** concludes the thesis, summarizing the key insights and contributions. It also reflects on the broader implications of the research and its potential impact on the field of intrusion detection, federated learning and Internet of Medical Things devices.

2 Background information

According to Gartner research and advisory company, 20.4 billion things will be connected to the internet [31] [**el2018towards**] (SAME) by 2020. Also, the worldwide market of IoT (including IoMT) will reach 1.7 trillion USB by 2020 from 655.8 billion USB in 2014 with an annual rate of 16.9 percent [32][33]. An IoMT platform is a smart system mainly comprises of (1) sensors and electronic circuits to acquire biomedical signals, (2) a network device to transmit the biomedical data over a network, (3) a temporary or permanent storage unit, (4) a visual platform with artificial intelligence schemes to take decisions according to the convenience of physician [34] [35]. In a smart healthcare environment, vital medical services are actualized with Internet of Things (IoT) technologies that are connected through various types of networks through the internet. The recent growth in IoT technologies has sprung healthcare into a digital era [33]

2.1 Federated Learning

Federated Learning (FL) is transformative across numerous fields within the Internet of Things (IoT), advancing retail, manufacturing, transportation and healthcare [36]. Smart homes and transportation systems are improved with collaborative energy management and security protocols [36]. Energy sectors and manufacturing benefit from operational efficiency and sustainability [36]. Adoption of FL in drones provide greater reach into environmental monitoring, and smart cities foster personalized experiences with efficient data handling [36]. While in healthcare, federated learning safeguards patient privacy and advances diagnostic accuracy.

Federated Learning (FL) is a machine learning paradigm in which multiple clients cooperate to learn a model under the orchestration of a central server [37]. In federated learning, raw data from clients is never shared with other clients or the central server. This distinguishes FL from other traditional distributed paradigms, whilst introducing a new set of challenges that stem from contending with heterogeneous data.

FL has two primary settings, cross-silo (eg. FL between large institutions) and cross-device (eg. FL across edge devices) [25] - 2019. In cross-silo FL, most clients participate in every round and maintain state between rounds. Our work focuses on cross-silo circumstances, as healthcare institutions are clients in Federated Learning, and there exists a Federated-Learning-based- IDS service-provider as a central server

In more challenging cross-device FL, our primary focus, only a small fraction of clients pariticipate in each round, and clients cannot maintain state across rounds - Kairouz et. al (2019) [38] and Li et. al (2019a) [39]. Federated Learning (FL) presents innovative privacy-preserving collaboration across healthcare institutions without sharing sensitive patient information.

Standard optimization methods such as distributed SGD, are often unsuitable in FL an can incur high communication costs. To solve this, typically federated optimization methods employ *local client updates*, in which clients update their model multiple times before communicating with the server. This immensely reduces the amount of communication required to train a model. One such method is FedAvg [40], wherein clients conduct multiple epochs of SGD on their local datasets. The clients communicate their models to the server, which averages them to form a new global model. While FedAvg has seen great success, recent works have highlighted its convergence issues in certain settings [41] [42]. This stem from a variety of factors including (1) *client drift* [42], where local client models move away from globally optimal models, and (2) lack of *adaptivity*. FedAvg is similar in spirit to SGD, and may be suitable for settings with heavy-tail stochastic gradient noide distributions, which often arise when training language models [43]. Such settings benefit from adaptive learning rates, which incorporate knowledge of past iterations to perform more informed optimization.

Main contributions In light of the above, we highlight the main contributions of this paper.

- We study a general framework for federation optimization using server and client optimizers. This framework generalizes many existing federated optimization methods, including FedAvg
- We use this framework to compare novel cross-device compatible federated optimization methods and provide a convergence analysis in general non-convex settings. To

the best of our knowledge, these convergence analysis on more than one dataset are the first for FL. We show an important interplay between the number of local steps and the heterogeneity among clients.

- We demonstrate a strong empirical performance of various server optimizers, improving upon commonly used baselines. Our results indicate that these optimizers are robust and highlight their utility in cross-device settings.

Related work FedAvg was first introduced by McMahan [40], who demonstrated it can significantly reduce communication costs. Many variant have since been proposed to tackle issues such as convergence and client drift. Examples include adding a **regularization term** in the client objectives towards the broadcast model [39], and server momentum [41]. When clients are homogeneous, FedAvg reduces local SGD [44], which has been analyzed by many works [45] [46] [47] [48] [49]. In order to analyze FedAvg in heterogeneous settings, many works derive conveRgence rates depending on amount of heterogeinity [50] [51] [52] [53]. Typically, the convergence rate of FedAvg gets worse with client heterogeneity. By using control variates to reduce client drift, the SCAFFOLD method [42] achieves convergence rates that are independent of the amount of heterogeneity. While effective in cross-silo FL, the method is incompatible with cross-device FL as it requires clients to maintain state across rounds (more detailed comparisons, defer to [25]

The term **federated learning** (FL) was introduced in 2016 in a paper published at Google [37]. In their work the authors developed a decentralized approach, namely federated learning, with the objective of ensuring data privacy of participating clients.

In FL, a server or aggregator takes the role in coordinating several clients into analyzing data using a shared model. The data owned by the clients remains with the data owner and it is never transferred between devices. The model is shared amongst clients and only parameter updates are exchanged.

As explained by [37], when implementing FL, several key constraints need taking into consideration in order to optimize a solution to the problem. These constraints exist as FL must be able to train data with the following characteristics [54, 37]:

- **Non-IID** — The data that reside on an individual client device are not representative of the global population distribution.

- **Unbalanced** — The amount of local data varies considerably among devices; consequently, some clients train on substantially larger datasets than others.
- **Massively Distributed** — The learning process involves a very large number of participating clients.
- **Limited Communication** — Reliable communication cannot be assumed for all clients (some may be offline). Hence, training may need to continue with a reduced set of devices or in an asynchronous manner.

Typical federated learning data is not identically distributed. For instance, in IoT environments devices acquiring data for analysis capture data in different formats and of different types to each other. Therefore, the local data cannot be used as an example of the entire data distribution. Similarly, data size can vary drastically between devices, depending on a given scenario. The large number of devices involved and issues with communication between clients and server must also be taken into consideration when developing FL applications.

2.2 Optimization Methods

To address heterogeneity and manage high communication costs, optimization methods are presented in federated learning to allow local updating and low participation rates [40]. The most common and standard is FedAvg [40] that performs stochastic gradient descent (SGD) on K devices and repeats iteratively of E epochs - here, E is small constant and K is a small fraction of total devices in the network. These devices transmit model updates to a central server where total sum is averaged. This optimizer established success in heterogeneous conditions to train federated model [40]. Albeit this success, the standard FedAvg is unable to consider system constraints when client devices conduct local work. Specifically, local devices that fail to complete work (E epochs) within a predefined time-frame, are dropped despite any network interruptions or computation errors [55]. In that case, the hosts' local updates are excluded from final aggregation. Although, McMahan [40] demonstrated that FedAvg diverges empirically when data is non-identically distributed across devices, more recent works conclude otherwise and provide alternatives.

2.3 Federated Learning Methods

Federated Learning (FL) [40] is a distributed learning approach to enable machine learning across multiple remote client devices with the intent to avoid any raw data exposure [40] [39] [56]. FL facilitates training machine learning models on remote edge devices by sharing (local) model updates instead of data samples, addressing data ownership and privacy issues [57]. Multiple remote clients train models on their local data and send model updates to a central server, which updates the global model and distributes newly updated global model to clients. These characteristics set FL apart from other distributed optimizations [58], introducing heterogeneous data due to localized data samples. Standard optimization methods, such as Stochastic Gradient Descent (SGD) are often inadequate for FL and can create high communication costs [40]. Recent advancements propose that client devices repeatedly update their local models before communicating with the server in order to reduce communication costs. Federated Averaging (FEDAVG) optimization method implements this by requiring clients to perform multiple epochs of SGD on local data before sending model updates to the server for model averaging [40]. While FEDAVG has achieved considerable success, recent studies have revealed that convergence rates depend on the amount of client heterogeneity [50] [53] [51] [52]. Key factors are *client drift* [42], where local models diverge from the globally optimal model, and *adaptivity* [58].

Client drift occurs when local data in federated learning does not reflect the global distribution, causing inconsistencies between each client's local objectives and the overall global optimum [59]. As clients fine-tune their local models with local data, they diverge from the initial global model and from each other, moving towards their individual optima. This drift intensifies with more local updates, whether due to a higher number of epochs or smaller batch sizes, leading to greater differences among client models. Conversely, too few local iterations can increase communication costs, requiring more rounds to achieve global model convergence. Prior studies have highlighted these observations about client drift [42, 50, 60].

Reddi et al. [58] introduce an approach to decouple the learning rates of the server and clients, aiming to address client drift through adaptive learning rates. Clients use a local optimizer to minimize loss on their datasets, while the server employs a gradient-based optimizer for loss reduction across participants. Their framework, Federated Optimization

(FedOpt), supports per-coordinate adaptive server optimizers alongside Stochastic Gradient Descent (Sgd) for clients. They implement three adaptive server optimizers—FedAdagrad, FedAdam, and FedYogi —which are federated versions of Adagrad, Yogi, and Adam. Compared to FedAvg [40] and FedAvgM [61], their methods often significantly mitigate client drift and reduce the total communication rounds needed for model convergence. They also provide a theoretical analysis, noting the importance of a decaying learning rate on the client side. Adaptivity can be incorporated in either the server-side or client-side, while joint adaptivity - comprising of both global and local adaptive updates - has shown to be vital in accelerating convergence and increasing accuracy [62] [63]. More thorough description of adaptive federated optimization is described upcoming section, formulated in Algorithm 7.

3 Literature review

To conduct a literature review, relevant keywords are selected to conduct a thorough search for peer-reviewed literature. Therefore, all studies reviewed in this thesis pertain to Intrusion Detection Systems (IDS) in Federated Learning (FL) and involve keywords or content **"IDS in IoT networks using Federated Learning"**. This investigation examined papers indexed in Scopus, Google Scholar, or Web of Science. Additionally, this thesis included papers published by reputable publishers such as Springer, Elsevier, Institute of Electrical and Electronics Engineers (IEEE), Association of Computing Machinery (ACM), and Multidisciplinary Digital Publishing Institute (MDPI). The preliminary search for relevant peer-reviewed papers was completed with a search string with the following keywords:

("intrusion detection system" OR "IDS" OR "anomaly detection" AND "internet of things" OR "IoT" OR "IoMT" or "edge device" or "smart device" and "federated learning" OR "FL" or "collaborative learning")

The **selection criteria** included manual examination of the title, abstract, and results. Furthermore, the search keywords provided an initial inclusion and exclusion of existing literature. The publication date was relevant to identifying the state-of-the-art contributions in the literature, and thus any research prior to the year 2018 was excluded for the purpose of this thesis. An inherent exclusion criterion was identified, namely accessibility from the university network - any papers that were inaccessible were excluded.

A more in-depth **inclusion/exclusion** involved optimization techniques, more specifically, thereof. As such, any prior research without a federated learning optimization algorithm or a strictly baseline arithmetic mean algorithm (FedAvg) was excluded. Moreover, literature outside the domain of federated learning was excluded. Conversely, the literature review examined the state-of-the-art optimization algorithms and their respective local metrics in a federated learning scenario to enhance intrusion detection capabilities with machine learning. Therefore, taxonomies, surveys, and whitepapers were excluded from the

selection. Throughout the literature review, an analysis of identified studies was conducted. To reiterate, the existing research was thoroughly identified with the search keywords, inclusion, and exclusion criteria.

A preliminary analysis involved the identification of research questions, data sources, data analysis methods, and key findings. The analysis comparative of traits among the literature focused on crucial information pertaining to (1) optimization technique and communication rounds, (2) federated architecture (cross-device or cross-silo), (3) multiclass or binary inference/classification, (4) number of attack size, and (5) communication metrics. The relevant information will be represented in a comparative table. [table will be added later]

Conversely, prior examinations of optimization techniques in an IoMT network were conducted in previous studies, but the lack of benchmarking and comparison of federated learning optimization techniques illustrates the research gap in the existing domain. Due to the overall lack of comprehensive research with datasets pertaining to IoMT devices in a federated architecture, further research must be conducted.

Therefore, the literature review exhibits an existing gap. To elaborate, a predominant focus in prior research has been conducted to implementing and benchmarking the performance of optimization algorithms in collaborative IoT settings. The research illustrated the validity and novelty of these innovative optimization algorithms in federated architectures within decentralized IoT networks. Prior works have not examined and compared the model performance of machine learning algorithms with multiple different optimization techniques in healthcare settings, that is, decentralized Internet of Medical Things (IoMT) networks to detect intrusions. This serves as a vital research topic to detect anomalies from large-scale multidimensional network data and thus combat network intrusion attacks against life-critical healthcare devices.

4 Methodology

This Chapter provides in-depth description of Federated Learning and server-side optimization methods. To conduct our experiments, we must first describe the various implementations and practical considerations.

4.1 Problem Formulation in FL

In the basic conception of federated learning, the goal is to minimize the objective function,

$$F(x) = \mathbb{E}_{i\sim\mathcal{P}}[F_i(x)], \quad \text{where} \quad F_i(x) = \mathbb{E}_{\xi\sim\mathcal{D}_i}[f_i(x,\xi)], \tag{4.1}$$

where $x \in \mathbb{R}^d$ represents the parameter for the global model, $F_i : \mathbb{R}^d \to \mathbb{R}$ denotes the local objective function at client i, $\mathcal{P}$ denotes a distribution on the population of clients $\mathcal{I}$. If $\xi \in \mathcal{D}_i$ represents a randomly sampled mini-batch for client i, then the update rule is as follows: $x_k + 1 = x_k - \eta g(x_k;\xi_k)$, where η is the server learning rate, and stochastic gradient is defined as: $g(x;\xi) = \frac{1}{|\xi|}\sum_{b_i\in\xi}\nabla f(x;b_i)$.

In standard implementation of mini-batch stochastic gradient descent (SGD) [64], the local loss functions $f_i(x,\xi)$ are generally consistent across all clients. However, the local data distributions $\mathcal{D}_i$ often differ, representing clear the data heterogeneity among clients. Therefore, for simplicity we omit gradient step k for the rest of the paper, and instead we will denote $g(x_k;\xi_k)$ as $g_i^{(t)}$ at client i for given during round t of Federated Learning.

Algorithms designed for the cross-device setting cannot directly compute $F_i(x)$ or $\nabla F_i(x)$ because it is assumed that they only have access to a random sample $\mathcal{S}$ of clients during each communication round. However, the objective function $F_i(x)$ can still serve as a mathematical construct in analyzing such algorithms or can be computed numerically in simulations as part of empirical evaluation procedures. When modeling cross-device FL with a fixed dataset, to align with the population risk in 4.1, we typically utilize a held-out

set of *clients*[1] rather than a held-out set of examples for each client from a fixed dataset. This represents a "stylized" scenario because although such a simplified client selection model could be useful for analyzing or comparing certain optimization algorithms, other methods that incorporate a client selection strategy will necessitate a more complex model of device availability and participation.

The cross-silo setting can generally be modeled with a finite number of clients, expressed as $F^{\text{silo}}(x) = \sum_{i=1}^{M} p_i F_i(x)$. A train/test split can typically be achieved by dividing the per-client datasets into local training and testing sets, with generalization assessed against the held-out per-client data.

In both the cross-device and cross-silo settings, the objective function in 4.1 can take the form of an empirical risk minimization (ERM) objective function with finite clients, each possessing finite local data:

$$F_i^{\text{ERM}}(x) = \sum_{i=1}^{M} p_i F_i^{\text{ERM}}(x), \quad \text{where } F_i^{\text{ERM}}(x) = \frac{1}{|\mathcal{D}_i|} \sum_{\xi \in \mathcal{D}_i} f_i(x, \xi) \text{ and } \sum_{i=1}^{M} p_i = 1. \tag{4.2}$$

Notably, $M = |\mathcal{I}|$ represents the total number of clients and p_i is the relative weight of client i. Crucially, $p_i = |\mathcal{D}_i| / \sum_{i=1}^{M} |\mathcal{D}_i|$ makes the objective function $F_i^{\text{ERM}}(x)$ becomes equivalent to the empirical risk minimization objective function for the combined local datasets. The objective in (4.1) equals (in expectation) the ERM objective that would be optimized centrally if a random selection of clients were made and a central training dataset constructed from the union of their local datasets.

Compared to the centralized training, we want to highlight several key properties [40, 54] of Equations (4.1) and (4.2):

- **Heterogeneous and imbalanced data**: The local datasets $\mathcal{D}_i$'s may have non-uniform distributions and differ in size. As a result, the local objectives $F_i(x)$'s can be different. For example, they may have arbitrarily different local minima.
- **Data privacy limitations**: The local datasets D_i's cannot be shared with the server or shuffled across clients.

[1]While in the cross-device setting the server cannot access client IDs to partition clients into disjoint training and test sets, devices can locally flip a coin to determine whether to belong to the training or test population.

- **Limited client availability (more typical of cross-device FL)**: In cross-device FL, the number of clients M in 4.2 can be extremely large and not precisely defined. At any given time, only a small subset of clients are available to connect with the server and participate in the training. The client distribution $\mathcal{P}$, total number of clients M or the total number of data samples $\sum_{i=1}^{M} |\mathcal{D}_i|$ are not known *a priori* before the training starts. Similarly, cross-silo environments may be subjected to the same problem, due to computational and communication-specific challenges, or system failures.

4.1.1 Gradient Descent Basics

Problem 4.1 can potentially be solved by *gradient descent* (GD), which performs iterations of the form $x_{t+1} = x_t - \eta \nabla F(x_t)$, $t = 0, 1, 2, \ldots,$, where η is an appropriately chosen learning rate. Under appropriate regularity conditions, we can swap differentiation and expectation, which gives the following formula for the gradient: $\nabla F(x) = \nabla \mathbb{E}_{i \sim \mathcal{P}}[F_i(x)] = \mathbb{E}_{i \sim \mathcal{P}}[\nabla F_i(x)]$. Note that the gradient of the global loss function F is equal to the expectation (or "average") of the gradients of the local functions F_i [65]. In many federated learning settings, the clients can't communicate among themselves directly, but can communicate indirectly via an orchestrating server.

When applying to the ERM formulation 4.2, server has to calculate $\nabla F^{\mathrm{ERM}}(x) = \sum_{i=1}^{M} p_i \nabla F_i^{\mathrm{ERM}}(x)$ by weighted average of *all* the local gradients from the clients.

While can be conceptually applied in the context of FL, it is not used in practice for various constraints and considerations discussed in Chapter 1. A number of techniques can be used to enhance to make it theoretically or practically efficient as a method for solving federated optimization problems. Moreover, many of these techniques are (orthogonal) enhancements that can be combined for a more dramatic effect. Having said that, many of the possible combinations are not well understood and are still subject of active research.

Partial participation is a requirement for cross-device FL and some cross-silo settings. In communication round t, only a (finite) subset $\mathcal{S}^{(t)}$ of clients can connect to the server, and the update rule becomes $x_{t+1} = x_t - \eta \frac{1}{|\mathcal{S}^{(t)}|} \sum_{i \in \mathcal{S}^{(t)}} \nabla F_i(x_t)$. In practice, the sequence of active clients $\mathcal{S}^{(t)}$ is typically dictated by complicated circumstances beyond the control of the orchestrating server (for example mobile devices might only participate when idle, connected to particular unmetered networks, and charging). In theory, assumptions on

client sampling are necessary to guarantee convergence (see **??** for more discussion), and partial participation can be expected to lead to an increase in the number of communication rounds.

Independently of whether partial participation is necessary, clients can use the *stochastic approximation (SA)*, replacing the exact gradient of their local loss function with an unbiased stochastic gradient $g_i(x_t)$ such that $\mathbb{E}_{\xi\sim\mathcal{D}_i}[g_i(x_t)] = \nabla F_i(x_t)$. SA is preferred when the size of $\mathcal{D}_i$ is large and the calculation of the exact gradient is inefficient.

Local steps is a popular technique to reduce communication costs. Each active client updates their local model for τ_i steps before the server aggregates the model deltas $\Delta_i^{(t)} = x_i^{(t,\tau_i)} - x_t$. Combining partial participation, stochastic approximation and local steps leads to federated averaging, a popular practical algorithm for federated optimization, which is further discussed in Section 4.1.2. We defer the discussion of additional techniques like compression, momentum and acceleration, adaptive method, and control variates to **??** and the theoretical analysis to **??**.

4.1.2 Federated Averaging Technique

Alternative formulation of Federated Averaging (FedAvg) is presented in Algorithm 3.

An alternative algorithm to solve 4.1 is *federated averaging* (FedAvg), proposed by [40]. The algorithm divides the training tasks into (communication) rounds. At the start, we must input (or randomly select) an initial model x_0. , During each round t the server broadcasts the current global model x_t to a random set of clients $S^{(t)}$ clients (often uniformly sampled) for that given round t. Each sampled client performs τ_i Sgd (stochastic gradient descent) updates on its own local dataset and sends the local model changes $\Delta_i^{(t)} = x_i^{(t,\tau_i)} - x_t$ to the server. After receiving these aggregates local model changes$\Delta_i^{(t)}$, the server aggregates them to update the global model as follows:

$$x_{t+1} = x_t + \frac{\sum_{i\in\mathcal{S}^{(t)}} p_i \Delta_i^{(t)}}{\sum_{i\in\mathcal{S}^{(t)}} p_i}, \tag{4.3}$$

where p_i denotes the relative weight of client i. This process repeats until the algorithm converges. In a cross-silo FL scenario, where all clients participate in every round, the set

$S^{(t)}$ includes the entire population: $S^{(t)} = \{1, 2, ..., M\}$.

Algorithm 1 Generalized FedAvg (also known as FedOpt [58])

Data: Input: x_0, ClientOpt, ServerOpt
for r **do**ound $t = 0$ to $T - 1$
  Sample a subset $\mathcal{S}^{(t)}$ of client
  for client $i \in \mathcal{S}^{(t)}$ in parallel **do**
    Initialize local model $x_i^{(t,0)} = x_t$
    for $k = 0, \ldots, \tau_i - 1$ **do**
      Compute an unbiased estimate $g_i^{(t,k)}$ of $\nabla F_i(x_i^{(t,k)})$
      Perform local update $x_i^{(t,k+1)} = \text{ClientOpt}\left(x_i^{(t,k)}, g_i^{(t,k)}, \eta_l, t\right)$
    Compute local model changes $\Delta_i^{(t)} = x_i^{(t,\tau_i)} - x_t$
  Aggregate local changes $\Delta_t = \frac{1}{|S|} \sum_{i \in \mathcal{S}^{(t)}} \Delta_i^{(t)}$
  Update global model $x_{t+1} = \text{ServerOpt}\left(x_t, -\Delta_t, \eta, t\right)$

Note that, this thesis observes cross-silo Federated Learning, therefore p_i relative weight is not applicable any implemented Algorithms, all clients are participating, $C = 1.0$ (100% participation) among $K = 20$ clients.

FedAvg can be easily adapted into a flexible framework that allows the algorithm designer to change the client update rule [60, 66, 62], the update rule of the global model [58, 41, 67], or the aggregation method applied to updates [68, 69].

In particular, [58] proposed a generalized version of FedAvg, the pseudo-code of which is presented in 3. The algorithm is parameterized by two gradient-based optimizers: ClientOpt and ServerOpt with client learning rate η_l and server learning rate η, respectively. While ClientOpt is used to update the local models, ServerOpt treats the negative of aggregated local changes $-\Delta^{(t)}$ as pseudo-gradient and applies it to the global model. The original FedAvg algorithm implicitly set ServerOpt and ClientOpt to be Sgd, with a fixed server learning rate $\eta = 1.0$

The FedAvg algorithm can be viewed as a generalization of Local SGD (also called local-update SGD or periodic averaging SGD), which is studied for reducing communication cost in classic distributed settings [70, 45, 46, 47, 71]. Some distinguishing properties of FedAvg are that unlike classic distributed settings, only a subset of clients participate

in each training round. In terms of analysis, convergence analyses of Local SGD often assume that the local data is homogeneous and each client performs the same number of local updates, which may not hold in the federated setting.

4.1.3 Problem Formulation Assumptions

In Federated Learning, we solve an optimization problem of the form [61] [58]:

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{m}\sum_{i=1}^{m} F_i(x) \text{ where } F_i(x) = \mathbb{E}_{\xi_i\sim\mathcal{D}_i}[f_i(x,\xi_i)]. \tag{4.4}$$

Here, the variable ξ_i denotes a random local data point available at client i. Function $F_i(x)$ represents the non-convex local function associated with client i, while $\xi \in \mathcal{Z}$, and $\mathcal{D}_i$ is the data distribution for the i^{th} client. More precisely, for any pair of clients $i \neq j$, their local distributions $\mathcal{D}_i$ and $\mathcal{D}_j$ may differ substantially, resulting in the inequality of local loss functions $F_i(x) \neq F_j(x)$. This phenomena of unequal distributions is often termed as *data heterogeneity*. Inversely, homogeneous (sampled) datasets refer to local data samples equating to the same distribution $\mathcal{D}$, we would have $F_i(x) = F_j(x)$ for any i and j. Additionally, for i and x, we assume access to an *unbiased* stochastic gradient $g_i(x)$ of the client's true gradient $\nabla F_i(x)$ [58]. The optimization problem may require explicit assumptions, and related works [50] [58] formulate these assumptions as follows:

Assumption 1 [58] (Lipschitz Gradient). The function F_i is L-smooth for all $i \in [m]$ i.e., $\|\nabla F_i(x) - \nabla F_i(y)\| \leq L\|x - y\|$, for all $x, y \in \mathbb{R}^d$.

Assumption 2 [58] (Bounded Variance). The function F_i have σ_l-bounded (local) variance i.e., $\mathbb{E}[\|\nabla[f_i(x,z)]_j - [\nabla F_i(x)]_j\|^2] = \sigma_{l,j}^2$ for all $x \in \mathbb{R}^d$, $j \in [d]$ and $i \in [m]$. Furthermore, we assume the (global) variance is bounded, $(1/m)\sum_{i=1}^{m}\|\nabla[F_i(x)]_j - [\nabla f(x)]_j\|^2 \leq \sigma_{g,j}^2$ for all $x \in \mathbb{R}^d$ and $j \in [d]$.

Assumption 3 [58] (Bounded Gradients). The function $f_i(x,z)$ have G-bounded gradients i.e., for any $i \in [m]$, $x \in \mathbb{R}^d$ and $z \in \mathcal{Z}$ we have $|[\nabla f_i(x,z)]_j| \leq G$ for all $j \in [d]$.

Assumption 4 [58] (Bounded Gradient Dissimilarity, σ_g^2-BGD):

$\forall x \in \mathbb{R}^d, \quad \forall_i \in \{1,\ldots,n\}, \quad E[||\Delta f_i(x) - \Delta f(x)||^2] \leq \sigma_g^2.$

Assumption 5 [50] (*Assume $S^{(t)}$ to be a set of active clients in the t-th round, such that $S^{(t)} \subset [N]$. Let $S^{(t)}$ contain a subset of $|S^{(t)}| = S$ nodes randomly selected with*

replacement according to the sampling probability $p_i = \frac{n_i}{\sum_{i=1}^{N}}$*, where* n_i *is the number of samples located on client* i*. Assume that the clients' capabilities are unbalances, i.e.,* p_i*'s can be distinct for different* i*).*

Reddi et al. [58] uses σ_l^2 and σ_g^2 to denote $\sum_{j=1}^{d} \sigma_{l,j}^2$ and $\sum_{j=1}^{d} \sigma_{g,j}^2$. Assumptions 1 and 3 are relatively standard in nonconvex optimization literature [72, 73]. Assumption 2 is a form of bounded variance, but between the client objective functions and the overall objective function. This assumption has been used in various works on federated optimization [39, 51]. Intuitively, the parameter σ_g quantifies similarity of client objective functions and $\sigma_g = 0$ corresponds to the IID setting [58]. Under Assumption 5 [74], the exact average step is calculated as $x_{t+1} = \frac{1}{S^{(t)}} \sum_{i \in S^{(t)}} x_i^{(t,K)}$. In their analysis [74], $g_i^{(t,k)}$ is defined as the accumulated gradient direction from all participating devices at the k-th iteration of the t-round, represented by $g^{(t,k)} = \frac{1}{S^{(t)}} \sum_{i \in S^{(t)}} g_i^{(t,k)}$. Server aggregates local updates by averaging $x_{t+1} = \frac{1}{S} \sum_{(i \in S^t)} x_i^{(t,K)}$. Then on, the sum of all local gradients on the server in Algorithm 1 is given by $\Delta^{(t)} = (x_t - x_{t+1}) = \sum_{k=0}^{K-1} \eta_l g^{(t,k)} = \frac{1}{S} \sum_{(i \in S^t)} (x_t - x_i^{(t,K)}) = \frac{1}{S} \sum_{(i \in S^t)} \sum_{k=0}^{K-1} \eta_l g^{(t,k)}$.

4.1.4 Federated Averaging Algorithm

A widely used method to solving Equation (4.4) in federated environments is Federated Averaging Algorithm (FedAvg) [40]. During each round t of FedAvg, a randomly selected subset of clients receives a global model from the server. Simultaneously, these clients perform Sgd on their individual loss functions and transmit their updated models back to the server. The server refines its global model by averaging these local models. For additional details, refer to Algorithm 2:

Assume that round t, the server posses model x_t and selects a set S of clients. Let $x_i^{(t)}$ represent the model of client i where $i \in S$ after completing local training. We can formulate the update rule of FedAvg as follows:

$$x_{t+1} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} x_i^{(t)} = x_t - \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (x_t - x_i^{(t)}) \tag{4.5}$$

To simplify the concept, we present a streamlined version of FedAvg. In Algorithm 2, we outline a simplified version of the FedAvg algorithm [40]. We use the notation to $\text{Sgd}_K(x_t, \eta_l, f_i)$ to indicate K steps of Sgd using gradients $\nabla f_i(x, \xi)$ for $\xi \sim \mathcal{D}_i$ with

(local) learning rate η_l, starting from model point x_t. This formulation is a special case of Algorithm 3 where ClientOpt is Sgd, and ServerOpt is Sgd with learning rate 1. The thesis does not leverage the streamlined Algorithm 2 in any experiments.

Algorithm 2 Simplified FedAvg

1: Input: Initial model x_0
2: **for** $t = 0, \ldots, T-1$ **do**
3: Sample a subset $\mathcal{S}$ of clients
4: $x_i^t = x_t$
5: **for** each client $i \in \mathcal{S}$ **in parallel do**
6: $x_i^t = \text{Sgd}_K(x_t, \eta_l, f_i)$
7: $x_{t+1} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} x_i^t$

Algorithm 2 includes the only the rudimentary concepts of Federated Averaging. Alternative formulation Federated Averaging (FedAvg) is presented in Algorithm 3. At the start, we must input (or randomly select) an initial model x_0. During each round t the server broadcasts the current global model x_t to a random set of clients $S^{(t)}$ clients (often uniformly sampled) for that given round t. Each sampled client performs τ_i Sgd (stochastic gradient descent) updates on its own local dataset and sends the local model changes $\Delta_i^{(t)} = x_i^{(t,\tau_i)} - x_t$ to the server. After receiving these aggregates local model changes$\Delta_i^{(t)}$, the server aggregates them to update the global model as follows:

$$x_{t+1} = x_t + \frac{\sum_{i \in \mathcal{S}^{(t)}} p_i \Delta_i^{(t)}}{\sum_{i \in \mathcal{S}^{(t)}} p_i}, \tag{4.6}$$

where p_i denotes the relative weight of client i. This process repeats until the algorithm converges. In a cross-silo FL scenario, where all clients participate in every round, the set $S^{(t)}$ includes the entire population: $S^{(t)} = \{1, 2, ..., M\}$.

Generalized FedAvg (also known as FedOpt) algorithm was developed to tackle the challenges associated with federated learning optimization, where data resides on local devices and remains unavailable to the central server. This algorithm overcomes these difficulties by employing a distributed strategy for optimizing the global model, all while preserving the confidentiality of client data. Specifically, it aims to minimize the

communication overhead between clients and servers, ensuring convergence of the global model throughout the process.

Algorithm 3 Generalized FedAvg (also known as FedOpt [58])

Data: Input: x_0, ClientOpt, ServerOpt
for r **do**ound $t = 0$ to $T - 1$
  Sample a subset $\mathcal{S}^{(t)}$ of client
  for client $i \in \mathcal{S}^{(t)}$ in parallel **do**
    Initialize local model $x_i^{(t,0)} = x_t$
    for $k = 0, \ldots, \tau_i - 1$ **do**
      Compute an unbiased estimate $g_i^{(t,k)}$ of $\nabla F_i(x_i^{(t,k)})$
      Perform local update $x_i^{(t,k+1)} = \text{ClientOpt}\left(x_i^{(t,k)}, g_i^{(t,k)}, \eta_l, t\right)$
    Compute local model changes $\Delta_i^{(t)} = x_i^{(t,\tau_i)} - x_t$
  Aggregate local changes $\Delta_t = \frac{1}{|S|} \sum_{i \in \mathcal{S}^{(t)}} \Delta_i^{(t)}$
  Update global model $x_{t+1} = \text{ServerOpt}\left(x_t, -\Delta_t, \eta, t\right)$

FedAvg can be easily adapted into a flexible framework that allows the algorithm designer to change the client update rule [60, 66, 62], the update rule of the global model [58, 41, 67], or the aggregation method applied to updates [68, 69].

In particular, [58] proposed a generalized version of FedAvg, the pseudo-code of which is presented in 3. The algorithm is parameterized by two gradient-based optimizers: ClientOpt and ServerOpt with client learning rate η_l and server learning rate η, respectively. While ClientOpt is used to update the local models, ServerOpt treats the negative of aggregated local changes $-\Delta^{(t)}$ as pseudo-gradient and applies it to the global model. The original FedAvg algorithm implicitly set ServerOpt and ClientOpt to be Sgd, with a fixed server learning rate $\eta = 1.0$

The FedAvg algorithm can be viewed as a generalization of Local SGD (also called local-update SGD or periodic averaging SGD), which is studied for reducing communication cost in classic distributed settings [70, 45, 46, 47, 71]. Some distinguishing properties of FedAvg are that unlike classic distributed settings, only a subset of clients participate in each training round. In terms of analysis, convergence analyses of Local SGD often assume that the local data is homogeneous and each client performs the same number of local

updates, assumptions that may not necessarily hold in a federated learning context. While Algorithms 2 and 3 are useful for understanding relations between federated optimization methods, the objective of this thesis is to observe Federated Learning under Non-IID data, requiring more sophisticated algorithmic structure, such as training on data mini-batches. The next sections expand on these streamlined version to Non-IID setting.

4.1.5 Practical Implementations

When training during experiments, instead of uniformly using K gradient steps, as in Algorithm 3, we will instead perform E epochs of training over each client's dataset. Additionally, we will take a weighted average of the client updates, where we weight according to the number of examples n_i in each client's dataset. This leads to a batched data version of FedOpt in Algorithm 6, and batched data version of FedAdagrad, FedAdam, and FedYogi given in Algorithm 7. Below we can see our practical implementation of Federated Averaging algorithm (labeled *FedAvg Algorithm with batched data* Algorithm 4).

Federated Learning is closely related to data heterogeneity. Both heterogeneous data and inversely homogeneous data are observed upon this thesis in FL tasks. This infers implicit preconditions to any dataset; it must be divided across clients in precise logic with emphasis on distributions. The most intuitive approach is to alter degree of data heterogeneity; any increase in Non-IID settings, while minimization of heterogeneity produces IID scenarios. Therefore, the thesis includes relevant steps taken to recreate IID situations and Non-IID scenarios in Federated Learning context. Careful consideration must be taken to prepare distributions for machine learning tasks, and the next section describes that in detail.

4.1.6 Batched Data

This thesis draws upon two datasets, CICIoMT-2024 [29] and IoMT-TrafficData [30], both subjected to structure dataset construction, data pre-processing and arrangement into cohesive tabular form. Beyond those action, federated learning requires clients to hold private data without disclosing, sharing or broadcasting any data streams to the central server. Alongside that, traditional machine learning presumes certain data-specific procedures, crucial for conducting reliable experiments aimed infer applicable results and avoiding common ML-related mishaps. For instance, these mishaps include *overfitting*,

underfitting or evaluating trained model on already trained data. Therefore, these two datasets were subjected to splitting and data partitioning to meet the requirements of Federated Learning. Here is a streamlined list that this section more thoroughly:

- **Data Ownership:** provide local data to each client, separate from server
- **IID:** enforce uniform distribution of local data across clients for IID scenarios
- **Non-IID:** produce data heterogeneity with non-uniform distributions
- **Data Splits:** divide local dataset into sets of training, testing and validation
- **Federated Evaluation:** assess global model on untrained local data
- **Centralized Evaluation:** provide unseen data for server assess global model

In FL, the most detrimental issue is the data ownership; each client has their own private data that is not shared with the central server or any other clients. Therefore, a rigorous and systematic procedure must be followed to accomplish valid simulation of Federated Learning, presented in Section 5.2. Firstly, to avoid any data exposure, the raw dataset D_{total} is split into: (1) $D_{local} = \{D_1, D_2, \ldots, D_K\}$ to conduct local training and on-client testing (term known as federated evaluation), and (2) D_{server} to test global model per FL communication round (also known as central evaluation). Note that, the entire raw dataset D_{total} is divided to 80% D_{local} and 20% D_{server}, common procedure in ML tasks. The described systematic approach provides suitable preconditions to experiment Federated Learning tasks without server accessing client owned data samples.

Once client possess their own private data, subsequent data splitting must be done to ensure any model testing must occur on previously unseen (untrained) data. Importantly, the thesis observes both Non-IID and IID settings which require data partitioning logic exclusive to each other (see Section 5.2). Firstly, IID scenario aimed to simulate uniform data distribution across clients. Secondly, Non-IID scenario is replicable with **Dirichlet distributions** utilized in Equation (5.5). The formula leverages concentration power α to control the degree data heterogeneity, with smaller parameter α value producing high class-imbalance. In both IID and Non-IID scenarios, each client possess its own local dataset D_i. More specifically, $D = \{D_1, D_2, \ldots, D_K\}$ where K is total number of clients, and $D_i = \{z_1, z_2, \ldots, z_n\}$ with z is a random data sample and n total number of data samples.

In machine learning, a unified approach is to divide a dataset into distinct exclusive sets: *training set* utilized during model training, *validation set* to mitigate model overfitting, and *testing set* to empirically measure the results of the final model. To follow this crucial rule, and mitigate *overfitting* and *underfitting*, the thesis experiments systematically separate each client's local dataset D_i into 60% training $\mathcal{P}_i^{(\text{training})}$, 20% validation $\mathcal{P}_i^{(\text{validation})}$ and 20% testing $\mathcal{P}_i^{(\text{testing})}$ data-split. This applies to both Non-IID and IID related experiments. In IID-scenario, all split partitions have uniform distribution, depicting homogeneous data across of same type, for instance $\mathcal{D}_{h_i}$ and $\mathcal{P}_i^{(\text{training})}$ where $h_i = \mathcal{P}_i^{(training)}$, $h_j = \mathcal{P}_j^{(\text{training})}$ and $i \neq j$. More precisely, in IID setting the partitioned data-splits retains the distribution of the original raw dataset, such that $i \neq j$ then D_i In experiments related to Non-IID, each data-split is partitioned using Dirichlet distributions in Equation (5.5) with a value to control degree of data heterogeneity.

Lastly, these two published datasets are relatively large in size and data points. An appropriate measure to assist training, client's partition of training data is divide into batches of equal data points, specifically $\xi_i \leftarrow$(split $\mathcal{P}_i^{(\text{training})}$ into batched of size B). Henceforth, we represent client's partition of training data as $\mathcal{P}_i$, instead of $\mathcal{P}_i^{(\text{training})}$, because validation and testing do not require batched data, strictly the training tasks in FL. This concludes the rigorous data splitting and partitioning procedure.

During training, D_i is partitioned into $\mathcal{P}_i$ a collection of batches ξ_i, each of size B. Local model of client i is represented as $x_i^{(t)}$ at round t. For $b \in \xi_i^{(t)}$, we let $f_i(x_i^{(t)}; b)$ denote the average loss on this batch at $x_i^{(t)}$ with corresponding gradient $\nabla f_i(x_i^{(t)}; b)$. Thus, if b is sampled uniformly at random from ξ_i, then $\nabla f_i(x_i^{(t)}; b)$ is an unbiased estimate of $\nabla F_i(x_i^{(t)})$.

4.1.7 Federated Averaging

Federated Learning assumes use of distributed data across clients, and in practical scenarios often access to finite data samples, the number of which may significantly vary between clients, $\mathcal{P}_i = \{\xi_1, \xi_2, \ldots, \xi_n\}$, where $\mathcal{P}_i$ is participating client's partitioned dataset (in the training loop), consisting of batches of size B. Crucially, Algorithms 2 and 3 require to be adjusted such practical scenarios. Reddi et al. [58] acknowledge this limitations in their work, and relate these algorithms behaving as 'gradient oracles', where we compute

unbiased estimates of the client's gradient.

To implement Federated Learning into real-world scenarios, incoming data streams may exhibit large class imbalance, especially in the domain of intrusion detection in networks. Therefore, we present FedAvg Algorithm with batch data Algorithm 4.

Algorithm 4 FedAvg Algorithm with batched data. K clients are indexed by i; B is local mini-batch size, E is number of local epochs and η_l is the client-side learning rate.

Server executes:

initialize x_0;

for $t = 0; t < T - 1; t = t + 1$ **do**

  $m \leftarrow \max\{C \cdot K, 1\}$;

  $S^{(t)} \leftarrow$ random set of m clients;

  for each $i \in S^{(t)}$ **in parallel do**

    Initialize $x_i^{(t,0)} = x_t$;

    $x_i^{(t+1)} \leftarrow \text{ClientUpdate}(i, x_i^{(t,0)}, \eta_l)$;

  $x_{t+1} \leftarrow \sum_{i \in S^{(t)}} \frac{n_i}{n_\sigma} x_i^{(t+1)}$, where $n_\sigma = \sum_{i \in S^{(t)}} n_i$;

ClientUpdate$(i, x_i^{(t)}, \eta_l)$**:** *//conducted by client i*

$\xi_i^{(t)} \leftarrow$ (split $\mathcal{P}_i$into batches of size B)

for each $e = 1, \ldots, E$ epochs **do**

  for batch $b \in \xi_i^{(t)}$ **do**

    $x_i^{(t)} \leftarrow x_i^{(t)} - \eta_l \nabla F(x_i^{(t)}; b)$, where $\nabla F(x_i^{(t)}; b)$ is avg. gradient on b for $x_i^{(t)}$;

Return $x_i^{(t)}$ to Server;

When training during experiments, instead of uniformly using K gradient steps, as in Algorithm 3, we will instead perform E epochs of training over each client's dataset. Additionally, we will take a weighted average of the client updates, where we weight according to the number of examples n_i in each client's dataset. This leads to a batched data version of FedOpt in **??**, and a batched data version of FedAdagrad, FedAdam, and FedYogi given in **??**. Below we can see our practical implementation of Federated Averaging algorithm, defined in Algorithm 4.

Our experiments include supplementary experiments contending with homogeneous distributions (illustrated in Figure 11). Each client distribution $\mathcal{D}_i$ is the uniform distribution

over some finite set D_i of size n_i data samples, referring to IID setting ($\xi_i \sim^{(iid)} \mathcal{D}_i$). Therefore, we assume that in (4.4), each client distribution $\mathcal{D}_i$ is the uniform distribution over some finite set D_i of size n_i. The n_i may vary significantly between clients, common in Dirichlet distributions with high class imbalance (more in Section 5.2).

4.1.8 Federated Averaging with Server Momentum

Momentum was introduced in federated learning to improve the estimation of the stochastic gradient, resulting algorithm FedAvgM [61], depicted as Algorithm 5. Momentum is based on a moving average of previous gradients, and it is commonly viewed as a method to decrease the variance of model updates [75]. Prior research states momentum is crucial for training deep learning networks [56], and has explicitly demonstrated to improve client-optimization [76]. In server-side optimization, momentum parameter modifies the equation as follows [61]:

$$g_i^{(t,k)} = \beta \nabla F(x_i^{(t,k)}; \xi_i^{(t,k)}) + (1 - \beta) g^{(t)} \tag{4.7}$$

where $\beta \in [0, 1]$ is the momentum coefficient, and $g^{(t)}$ denotes a global estimate updated in the outer loop t of FL rounds. By order, subscript t represents rounds in the most outer loop, subscript i reflects the client index, while k indicates the inner loop, local update index. In this thesis, experiments are conducted with batched data, $\forall b \in \xi_i^{(t)}$ as inner-loop, instead of iterated $\forall k = \{0, \ldots, \tau - 1\}$ gradient steps. Thus, henceforth k gradient steps are omitted throughout this thesis, also in Algorithm 5.

Federated Averaging with Server-Momentum (FedAvgM) is introducing momentum parameter to improve server-optimization [61]. FedAvgM extends the original FedAvg with momentum parameter in gradient computation [61] (see highlight in Algorithm 5).

Authors remark [61] that FedAvgM retains the same algorithmic structure of FedAvg [40] whilst incurring no additional up-link communication overhead compared to the latter. Apparently, no extra down-link communication cost is required if clients store the last iterate model x_t so that momentum g_{t+1} can be recovered through $(x_{t+1} - x_t)/\eta$ [61]. A recent study has focused on utilizing adaptive optimizers with momentum to support federated learning [58]. The next section expands upon some common adaptive optimizers, that are implemented in this thesis.

Algorithm 5 FEDAVGM. K clients are indexed by i; B is local mini-batch size, E is number of local epochs and η_l is the client-side learning rate, η the server-side learning rate.

Input: local learning rate η_l, global learning rate η, momentum $\beta = 0.9$.

Server executes:

Initialize local model x_0

Initialize gradient estimate $g^{(0)}$

for $t = 0; t < T - 1; t = t + 1$ **do**

$m \leftarrow \max\{C \cdot K, 1\}$;

$S^{(t)} \leftarrow$ random set of m clients;

for each $i \in S^{(t)}$ **in parallel do**

Initialize local model $x_i^{(t,0)} = x_t$;

Perform local update $x_i^{(t+1)} \leftarrow \text{ClientUpdate}(i, g^{(t,0)}, x_i^{(t,0)}, \eta_l)$;

Aggregate local updates $x_{t+1} \leftarrow \sum_{k \in S^{(t)}} \frac{n_k}{n_\sigma} x_i^{(t+1)}$, where $n_\sigma = \sum_{k \in S^{(t)}} n_k$;

Update global model $x_{t+1} = x_t - \eta g^{(t+1)}$;

ClientUpdate$(i, x_i^{(t)}, \eta_l)$**:** *//conducted by client i*

$\xi_i^{(t)} \leftarrow$ (split $\mathcal{P}_i$ into batches of size B)

for each $e = 1, \ldots, E$ epochs **do**

for batch $b \in \xi_i^{(t)}$ **do**

$g_i^{(t)} = \beta \nabla F(x_i^{(t)}; \xi_i^{(t)}) + (1 - \beta) g^{(t)}$

$x_i^{(t)} \leftarrow x_i^{(t)} - \eta_l g_i^{(t)}$

Return $x_i^{(t)}$ to Server

4.1.9 Federated Optimization Algorithm

The FedOpt algorithm was developed to tackle the challenges associated with federated learning optimization, where data resides on local devices and remains unavailable to the central server. This algorithm overcomes these difficulties by employing a distributed strategy for optimizing the global model, all while preserving the confidentiality of client data. Specifically, FedOpt aims to minimize the communication overhead between clients and servers, ensuring convergence of the global model throughout the process.

Algorithm 6 outlines the FedOpt update procedure, which utilizes a parallel update function that iteratively modifies client states (lines 5-11) and subsequently aggregates these updates to compute the global model update (lines 12-13). Additionally, the FedOpt algorithm can implement adaptive optimizers, such as ADAM, YOGI, and ADAGRAD, to enhance both the performance and convergence of model training. By incorporating these optimizers, FedOpt effectively handles uneven data distributions and fluctuations in connection quality. Algorithm 6 describes FedOpt, also known as generalized FedAvg [56].

Algorithm 6 FedOpt Algorithm (also known as Generalized FedAvg) [58]).

1: **Input:** Initial model x_0; ClientOpt, ServerOpt with learning rates η_l, η
2: **for** $t \in \{0, 1, \ldots, T-1\}$ **do**
3: $m \leftarrow \max\{C \cdot K, 1\}$;
4: $\mathcal{S}^{(t)} \leftarrow$ random set of m clients;
5: **for client** $i \in \mathcal{S}^{(t)}$ **in parallel do**
6: Initialize local model $x_i^{(t,0)} = x_t$
7: **for local epoch** $e = 1, \ldots, E$ **do**
8: **for batch** $b \in \xi_i^{(t)}$ **do**
9: Compute local stochastic gradient $g_i^{(t)} = \nabla F_i(x_i^{(t)}; b)$
10: Perform local update $x_i^{(t)} = \text{ClientOpt}\ (x_i^{(t)}, g_i^{(t)}, \eta_l, t)$
11: Compute local model changes $\Delta_i^{(t)} = x_i^{(t)} - x_t$
12: Aggregate local changes $n = \sum_{i \in \mathcal{S}^{(t)}} n_i,\ \Delta_t = \sum_{i \in \mathcal{S}^{(t)}} \frac{n_i}{n} \Delta_i^t$
13: Update global model $x_{t+1} = \text{ServerOpt}\ (x_t, -\Delta^{(t)}, \eta, t)$
14: **end**

Alternative optimization methods may be implemented in both client-side optimization and

server-side optimization, instead of the Stochastic Gradient Descent. Adaptive methods have been researched in previous studies in convex [77] [78] and non-convex settings [72] [79]. Adaptive optimization preconditions the gradients to enhance optimization efficacy, dynamically adjusting the learning rate for each model parameter [77] [78] [80]. More recent advancements in federated learning have implemented adaptive methods for server and client model parameter updates.

4.1.10 Adaptive Federated Optimization Methods

In previous studies, some researchers have shown that using an adaptive coordination update can help enhance the convergence of federated learning (FL) [81]. Additionally, it has been demonstrated that adaptive methods offer distinct advantages in environments with a heavy-tailed random gradient noise distribution, which is often seen in non-IID data [43]. Consequently, Reddi et al. [58] were the first to explore the adaptive server optimizer that incorporates client learning rate attenuation with three distinct algorithms, including FedAdam, FedAdagrad and FedYogi. These algorithms modify FedAvg during global update step by using server-side optimizers Adam, Adagrad and Yogi respectively, as shown below:

$$
\begin{aligned}
m &= \beta_1 m + (1-\beta_1)\Delta \\
v &= v + \Delta^2 \quad \text{(FedAdagrad)} \\
v &= v - (1-\beta_2)\Delta^2 \,\text{sign}(v - \Delta^2) \quad \text{(FedYogi)} \\
v &= \beta_2 v - (1-\beta_2)\Delta^2 \quad \text{(FedAdam)} \\
w &= \beta_1 w + \frac{\eta_g}{\sqrt{v}+\varepsilon} m
\end{aligned}
\tag{4.8}
$$

where m is the global momentum (also known as first-order momentum) and accumulator (second-order momentum) v, with two parameters β_1 and β_2. Setting τ is employed to control the adaptive size, also known as *degree of adaptivity*; smaller values of τ lead to greater self-adaptation [58]. The variable η_l represents the client learning rate. By adjusting η_l, one can derive the average difference Δ_t and the accumulator x_t for the model across different optimization methods. The server utilizes the aggregated local update Δ as pseudo-gradient to compute the global momentum m. This approach updates the server model based on the accumulated update history rather than just current average local update (which may vary greatly across rounds) [82]. Results in [58] indicate that using an adaptive server optimizer offer significant improvements over Federated Averaging

(FedAvg) Algorithm, especially when dealing with heterogeneous data. In this work, server and client optimizers are utilized to leverage optimization schemes such as Adagrad, Adam, and Yogi, all based on FedAvg. Algorithm 7 presents the pseudo-codes for Adagrad, Adam, and Yogi in relation to FedAvg.

Algorithm 7 **FedAdagrad**, **FedYogi**, and **FedAdam**. K clients are indexed by i; B is local mini-batch size, E is number of local epochs and η_l is the client-side learning rate.

Initialization: $x_0, v_{-1} \geq \tau^2$, optional $\beta_1, \beta_2 \in [0, 1)$ for FedAdam and FedYogi
for $t = 0, \ldots, T-1$ **do**
  $m \leftarrow \max\{C \cdot K, 1\}$;
  $\mathcal{S}^{(t)} \leftarrow$ random set of m clients;
  Initialize local model $x_k^{(t,0)} = x_t$
  for each client $i \in \mathcal{S}^{(t)}$ **in parallel do**
    for $e = 1, \ldots, E$ **do**
      for batch $b \in \xi_i^{(t)}$ **do**
        Perform local update $x_i^{(t)} = x_i^{(t)} - \eta_l \nabla F_i(x_i^{(t)}; b)$
    Compute local model changes $\Delta_i^{(t)} = x_i^{(t)} - x_t$
  Aggregate local changes $\Delta_t = \sum_{i \in \mathcal{S}^{(t)}} \frac{n_i}{n} \Delta_i^{(t)}$, where $n = \sum_{i \in \mathcal{S}^{(t)}} n_i$
  Compute global momentum $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\Delta_t$
  Find 2^{nd}-order momentum (accumulator) $v_t = v_{t-1} + \Delta_t^2$ **(FedAdagrad)**
  Accumulate 2^{nd}-order momentum $v_t = v_{t-1} - (1 - \beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FedYogi)**
  Compute 2^{nd}-order momentum $v_t = \beta_2 v_{t-1} + (1 - \beta_2)\Delta_t^2$ **(FedAdam)**
  Update global model $x_{t+1} = x_t + \eta \frac{m_t}{\sqrt{v_t} + \tau}$
end

Throughout the experiments, we use Algorithm 7 for all implementations FedAdagrad, FedAdam, and FedYogi in Chapter 6 set of parameters have been chosen by relying on similar experiments by other authors. For FedAdagrad, parameter selection was heavily reliant on a previous work [73], which states that typical versions of Adagrad do not use momentum, consequently $\beta_1 = \beta_2 = 0$. For FedAdam and FedYogi, similarly, predefined values were extracted from that past work [73, 58], and we set $\beta_1 = 0.9$, $\beta_2 = 0.99$.

4.1.11 Flower

The overall FL architecture can be described in several steps. Before any training can commence, the clients must establish a connection with the orchestrating server. The training initiates once the server shares the initial parameters to the client. Upon receiving these parameters, each client performs training on its local dataset by updating weights locally. After completing training, clients send their updates back to the server. Utilizing FedAvg, outlined in Algorithm **??**, the server aggregates these updates into a global model updates, which is communicated back to the clients for next training round. This cycle continues until all rounds are finished. In summary, the steps for each round are as follows [83]:

- The server accepts connections from a set number of clients.
- The server transmits the initial parameters of the global model to the clients.
- Each client conducts training on its local data, computes local weights or gradients, and sends an update to the server.
- The server updates the parameters for the global model and aggregates the results.

Flower library leverages the concept of *strategies* to configure several options, include the type of averaging or optimization algorithm which is used to aggregate parameters during training. A *strategy* in the Flower library offers other directly configurable settings. Some of which are - the number of total clients participating in FL, the minimum number of clients required to be available for training, and the minimum number of clients required for validation. The hyperparameters, displayed in Table 1 were configured on the server side using a method classified by a strategy, and were communicated to the clients for training. These are the learning rate η set to 0.01, number of rounds T set to 50, and local epochs E set to 100.

Table 1. Hyperparameter Settings.

Hyperparameter	**Value**
Learning Rate (η)	0.01
Local Epochs (E)	100
FL Rounds (T)	50

Algorithm 8 Federated Learning algorithm in high-level pseudocode. S is the server, C_n are the clients, D_c is the client's data samples, and R is the aggregated results.

Server S **starts:**
Initialize parameters: p
Clients: C_n
Client's Data: D_c
Total number of rounds: T
for each round $t = 1 \rightarrow T$ **do**
    $C_n \leftarrow p$
    for each Client C_n **in parallel do**
        Classify D_c
        $S \leftarrow$ ClientUpdate(p)
    $S \rightarrow$ ParameterUpdate(p)
    $S \rightarrow$ AggregateResults(R)
Return R

The displayed Algorithm 8 depicts a simplified implementation of Federated Learning. More detailed descriptions of specific averaging algorithms and optimization are provided in upcoming sections of this Chapter.

To evaluate the proposed models, we utilized two distinct datasets: CICIoMT2024 [29] and IoMT-TrafficData [30]. These datasets are advantageous as they are recent and specifically tailored to address attacks in Internet of Medical Things (IoMT) environments. A comprehensive description of these datasets can be found in the Chapter 5.

Within the federated learning (FL) framework, it is essential to allocate the training data among the various clients participating in the model training process. In this case, we implemented a horizontal partitioning approach, referred to as Horizontal FL (HFL). This method allows clients to share the same feature space (that is, columns in tabular data) while having different sample sets (meaning, rows in tabular data) [84]. For comparative results, we provide two separate degrees of data heterogeneity, by utilizing Dirichlet Partitioning. Data partitioning is described in **??**. HFL is especially useful when datasets comprise records from various organizations or devices that provide similar types of information, in our case, these are separate healthcare organization's network infrastructures.

The federated learning (FL) process outlined in this model includes several steps, as described in Algorithm 8. We consider K clients $(C_1, C_2, \ldots, C_K)$, each possessing local data $D_i \subset D$ where K is indexed by i. Here, D represents a dataset comprised of n samples and m features, denoted as $D = (x_i, y_i)$ where $x_i \in \mathbb{R}_m$ and $y_i \in \mathbb{R}_m$. A coordinating server, S_{server} oversees the entire learning process. The FL procedure involves multiple rounds, R, with $R \geq 1$, to ensure effective convergence of the model. In the initial round, server S_{server} initializes global model x_0 and broadcasts to each clients C_i. The clients then creates an local model trained on its local data, D_i, and transmits it to the server. This transmitted data consists solely of model parameters. The server subsequently aggregates all received local model updates from clients to construct a new global model. The number of iterations in federated learning significantly impacts performance. Each iteration allows the server to combine updates from the clients' local models to refine the global model. Typically, a single round is insufficient, making it vital to repeat this process multiple times for optimal results. In the subsequent rounds, the server sends the global model to all K clients. Each client loads the global model and updates it using their local data. The newly generated local model updates are then sent back to the server. The server continues to improve the global model by aggregating the ensemble of trees from the models sent by the clients. The outcome of the FL algorithm is a global model that aggregates the findings of the clients' local models while safeguarding data privacy. This approach allows clients to more accurately detect malicious activities in Internet of Medical Things (IoMT) environments without compromising their own data. In the following section, we assess the global models — created with various existing FL optimizations — using new test data and compare its effectiveness to that of a FL environment with IID data.

5 Experimental setup

The thesis is an empirical evaluation to train various federated learning models to enhance cyber-attack detection in a decentralized Internet of Medical Things (IoMT) device architecture. Thomas W. Edgar and David O. Manz authored a comprehensive book titled *Rearch Methods for Cyber Security* [85] to describe various scientific approaches for conducting rigorous research in this respective domain.

Thomas W. Edgar and David O. Manz authored a detailed book titled *Research Methods for Cyber Security* [85] to outline various scientific methods for conducting research in this field. This section draws on information relevant to rigorous research and experimental design. In this thesis, we investigated how existing systems including machine learning and deep learning interact within federated learning and with peer-reviewed public datasets to enhance our understanding of intrusion detection capabilities. Therefore, our research applies experimentation, focusing on federated learning concepts and existing datasets, thereby contributing to scientific advancement.

In our experimental design it is vital the dependent variables. Prediction must be divided into one or more measurable observables, each serving as a dependent variables. A dependent variable is an aspect we measure or observe as a result of an intervention in our experiment [85]. To identify a dependent variable, we must specify what we will observe, how we will measure it, the possible range of values, and the expected results. A clear understanding of the dependent variables is crucial for setting up an experiment that will provide the necessary evidence to conclusively answer our research questions. In our case, machine-learning tasks observer data points, and measure the rate of true and false predictions to provide efficacy of machine-learning model performance. Therefore, we continue to define the performance metrics, and observable dependent variables.

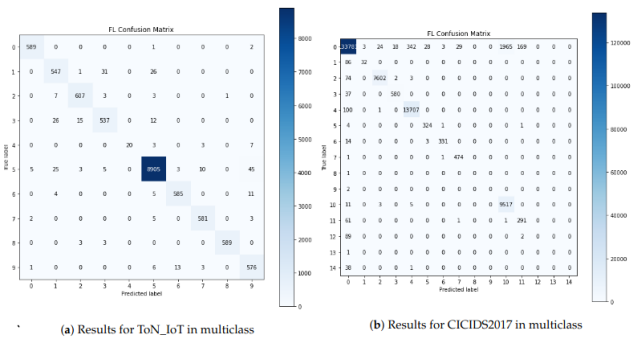
5.0.1 Performance Metrics

Evaluating the performance of machine-learning or deep-learning models for classification problems - such as the one presented in this work - is mainly based on metrics obtained from a confusion matrix (CM). A confusion matrix is a table that matches each predicted label with the real, true label of a data sample. This provides information about how many times the model was able to predict the correct class and how often the model made a mistake. Therefore, class labels are the dependent variables we observer to infer predictions.

CM counts both correct and wrong classification and those counts are used to compute the usual performance metrics. In anomaly-based intrusion detection, a confusion matrix (CM) can be employed to assess how often the model manages to:

- Detect anomalies or attacks correctly — i.e., **True Positives** (TP);
- Detect normal traffic correctly — i.e., **True Negatives** (TN);
- Mistake normal traffic as anomalous — i.e., **False Positives** (FP);
- Mistake anomalous traffic as normal — i.e., **False Negatives** (FN).

Typically, a CM is presented in table similar to Figure 1. The values on the right-hand side follow the same color coding (for example, dark blue may represent numbers around 80 K, but this figure varies with the total number of classified samples.



(a) Results for ToN_IoT in multiclass (b) Results for CICIDS2017 in multiclass

Figure 1. Confusion matrix example

A perfect model would label every malicious sample as true positive (TP) and every benign sample as true negative (TN), without mixing the two categories. An ideal model is nearly impossible due to limited training examples. However, the objective is to minimize the number of false positives (FP) and false negatives (FN), since high FP rates generate unnecessary alerts while high FN rates let attacks slip through unnoticed. By filling out a

confusion matrix (CM), we obtain the raw count needed to compute the essential evaluation metrics - accuracy, precision, recall, F1-score. These quantify how close the model comes to that ideal behavior. Therefore, we present the precise formulas to calculate these metrics:

Accuracy

The accuracy of the model is the percentage of times that a model is correct. More formally, accuracy is a ratio between the number of correctly predicted data points and the total number of data points [86], as shown in Equation (5.1).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.1}$$

To capture the impact of the misclassifications - false positives and false negatives - we introduce error-sensitive metrics that provide a more comprehensive evaluation of model performance.

Precision

Precision provides the rate of elements that have been classified as positive and that are actually positive. It is obtained by dividing correctly classified anomalies (TP) by the total number of positive instances (TP+FP), as shown in Equation (5.2).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.2}$$

Recall

Recall describes the proportion of correct predictions among data points with a positive label [86], that is, the number of true positives (TP) divided by the total number of positives (TP+FN).

Also defined as sensitivity, recall is obtained from the correctly classified attacks (TP) divided by the total number of attacks (TP+FN) and measures the model's ability to identify all positive instances (i.e., attacks) in the data. Recall is calculated by Equation (5.3).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

F1-Score

The F1-score combines precision and recall by computing their harmonic mean, as shown in Equation (5.4). The higher the score, the better the model.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.4}$$

In multiclass classification, an averaging technique is used to obtain an overall score for each metric. Several exist, as explained by [87]. In this case a weighted averaging score is used where class imbalance is considered according to the number of samples of each class in the data.

Throughout the experimentation, the models were trained, tested and validated in two unique data sets - the CIC-IoMT2024 dataset [29] and the IoMT-TrafficData dataset [30]. The author preprocessed the dataset, extracted relevant features, and distributed the data for machine learning tasks. More specifically, these datasets are non-independently and identically distributed (non-IID) across the various cyber-attack types, a common trait for collected data from many devices. Horizontal federated learning can overcome the challenges posed by non-IID data, and this empirical evaluation was used to benchmark the performance of a model with variances in optimization techniques.

This empirical study constituted an inherent assumption – a dedicated machine learning model must be able to host various optimization techniques to compare the model performance results between different optimization techniques. Thus, this assumption concludes that ML models confined to a single optimization technique were excluded, for instance, Random Forrest (RF), which strictly can host one optimization algorithm (that is, bootstrap aggregating or bagging) based on existing literature. Conversely, any such model would inherently hinder any comparative empirical results. Thus, a conscious assumption confined the empirical study to find one, and Gated Recurrent Unit (GRU) was chosen for that purpose, as it has not been implemented for IoMT-related intrusion detection. Recent study [88] investigated XGBoost in Federated Learning-based intrusion detection.

A preliminary measurement was conducted to determine a performant machine learning model with a common optimization technique for a baseline. The baseline optimization technique was a federated averaging (FedAvg) algorithm that used the arithmetic mean.

The comparative results were conducted using traditional model performance metrics – accuracy, precision, recall, and F1-score. This provided confidence in model accuracy as well as rates of error with false negatives and false positives. The numerical results facilitated comparative evidence to determine the most effective optimization algorithm for model convergence for the purpose of enabling ML-based network intrusion systems.

Throughout this research, varied optimization techniques provided results of model performance. These empirical results on two separate datasets - the CIC-IoMT2024 dataset [29] and IoMT-TrafficData [30] - provided confidence in the research results and validated the thesis study. This empirical evaluation **contributed to the domain** by addressing an existing gap in the literature and comparative results of machine learning on two datasets.

The validation of research outcomes is coupled with experimental setup and dataset separation. As such, to validate the research outcome, we must first examine the experimental setup, the separation of the data set into training, testing, and validation data, as well as the separation of the data corpus across binary and multiclass labels. Thus, prerequisite data partitioning serves a vital process to ensure validation of the research outcomes, alongside the use of two distinct datasets.

In Chapter 4 we presented an Federated Optimization Methods. These are defined to be applicable Non-IID situations, suitable for our intended work to measure intrusion detection capabilities in these circumstances. To assess a formulated prediction inferred by a machine-learning model, we must have followed rigorous procedures to minimise bias. Therefore, a systematic approach is introduced in this Chapter to create unseen data for model testing purposes on the federated client's local dataset, and the central server's dataset.

Our work is an empirical evaluation of various Federated Learning server-side optimization methods to compare results. As such, we follow the principles outlined in *Research Methods for Cyber Security* by Thomas W. Edgar and David O. Manz [85]. The authors describe that a good experiment must exhibit the following traits and actions:

- **Clear**: Explicitly describe the experimental design, underlying assumptions, procedural steps, analytical methods, and the rationale for each choice so any reader can understand what was done and why.

- **Precise**: Define unambiguous steps that every researcher can follow, thereby eliminating confounding due to inaccurate execution of the experiment.
- **Repeatable**: Record every stage - from design through execution and analysis - in detail for other researchers to replicate the experiment under same conditions.
- **Reproducible**: Provide the software developed to execute and analyze the experiment with the generated data for peers to compare and validate results.

These four characteristics serve as key pillars of rigorous experimentation in this thesis. Henceforth, the author emphasizes throughout this chapter the procedural aspects that uphold clear, precise, repeatable and reproducible experimentation.

5.1 Datasets

Intrusion Detection is aimed to infer possible attacks from abundant data streams. To produce efficacy, sophisticated relationships and patterns must be recognized from incoming sources of potential varied types of data. The thesis aims to utilized existing peer-reviewed datasets - CIC-IoMT-2024 [29] and IoMT-TrafficData [30] in Federated Learning. Both datasets meticulously constructed based on network logs consisting of plethora of attack instances related IoMT devices. As such, the thesis draws upon these two published datasets [29] [30] to build ML-based Intrusion Detection capabilities for healthcare institutions to bolster detection rates against potential IoMT-related cyberattacks.

To construct reliable intrusion detection based upon any machine learning approach, an existing dataset must be utilized to infer relations and intricate patterns across data points. The task is to build a classifier that infers a prediction on new unseen data. Binary classifier in intrusion detection is capable of identifying whether an attack occurred or not. More sophisticated form of detection capability is the ability infer the type of attack, a reasonable expectation for a ML-based Intrusion Detection System. Thus, this chapter divides these accordingly into two distinct tasks: **binary classification** and **multiclassification**.

An adequate classifier is trained on a large quantity of data, and the final testing must be conducted on previously unseen data which the model has not been trained upon. Therefore, a preliminary task was to subject the dataset to splitting across the target label the expected model is able to infer. In federated learning, a systematic approach is required to partition

the data into distributions, either uniform distribution to depict homogeneous data or non-uniform to create data heterogeneity (discussed in Section 5.2).

5.2 Dataset Preparation

This experimental setup acted on existing datasets, namely two separate datasets: CIC-IoMT2024 dataset [29] and IoMT-TrafficData dataset [30]. These provided different distributions by geographic, time-spatial, and overall purpose from various IoMT devices. Thus, no explicit data collection was carried out. Both datasets necessitated in-depth pre-processing, including correlation heatmaps, normalization, or standardization by authors [29], [30] respectively. In our work, we partitioned these peer-reviewed datasets into smaller subsets (1) by class labels, to distinguish by target label, (2) into training, testing, and validation purposes across these class labels, (3) in order to simulate heterogeneous (and homogeneous) scenarios and distribute clients for Federated Learning.

Firstly, the partitioning was a detrimental step to gather equal distributions of class labels, and then deliberately distributed labels unevenly to simulate the traits of non-independently and identically distributed (Non-IID) data between client devices. The study focused on FL-enhanced intrusion detection in HFL architectures within IoMT networks, and these decentralized IoMT networks collect non-independently and identically distributed (non-IID data) that is not shared with a central server. Thus, the experiment must disparately distribute data to replicate imbalanced data distributions among IoMT devices, including class imbalances, varied data sizes, and other non-uniform data characteristics.

An effective method for simulating real-world data distributions is the Dirichlet distribution. This partitioning technique, introduced in [89] and utilized in subsequent studies [60], [90], [68], allocates samples of each label to parties based on the Dirichlet distribution. Specifically, we sample $p_m \sim Dir_N(\alpha)$ and distribute $p_{m,j}$ proportion of instances of class m to FL client j with α serving as concentration parameter $(\alpha > 0)$. A key advantage of this approach is its flexibility in adjusting imbalance levels by varying α. Smaller values of β lead to more unbalanced partitions. Henceforth, we use $Dir_\alpha(\cdot)$ to use of indicate the Dirichlet distribution Equation (5.5), where α acts as a degree of heterogeneity.

The Dirichlet distribution effectively models data distributions as a multivariate probability distribution that describes the distribution of probability vectors. Its probability distribution function is represented as follows [91]:

$$P(p \mid \alpha) = \frac{1}{\mathcal{B}(\alpha)} \prod_{m=1}^{M} p_m^{\alpha_m - 1} \tag{5.5}$$

In this equation, p is an M-dimensional vector that represents the probability of each object's tag m (i.e., class label of each data sample) for each client, and α_m is a positive parameter that influences the concentration of the generated distribution. Therefore, a smaller value of α leads to a more uneven distribution of different class labels across clients, reflecting greater variability in the objects assigned to each client. Particularly when α is significantly small, it becomes increasingly likely that not all class labels will be represented across each client. Therefore, adjusting parameter α to lower values leads to more concentrated and skewed data distributions, introducing Non-IID data in our Federated Learning experiments, illustrated in upcoming Figure 5.

Secondly, this partitioning allowed the author to separate the data into training data, test data, and validation data by target label. Therefore, in this experimental setup, each client's dataset was partitioned into 60% training data, 20% testing data, and 20% validation data. Training data was used throughout the model building process, testing data and validation data were strictly excluded from any model training steps. In essence, validation data concluded whether the trained model exhibited traits reminiscent of overfitting or underfitting, thus providing crucial feedback on whether the training tasks requires additional attention. Lastly, testing data was strictly used to validate the results and performance of the model in accuracy, precision, recall, and F1-score. Figure 2 depicts a rudimentary overview.

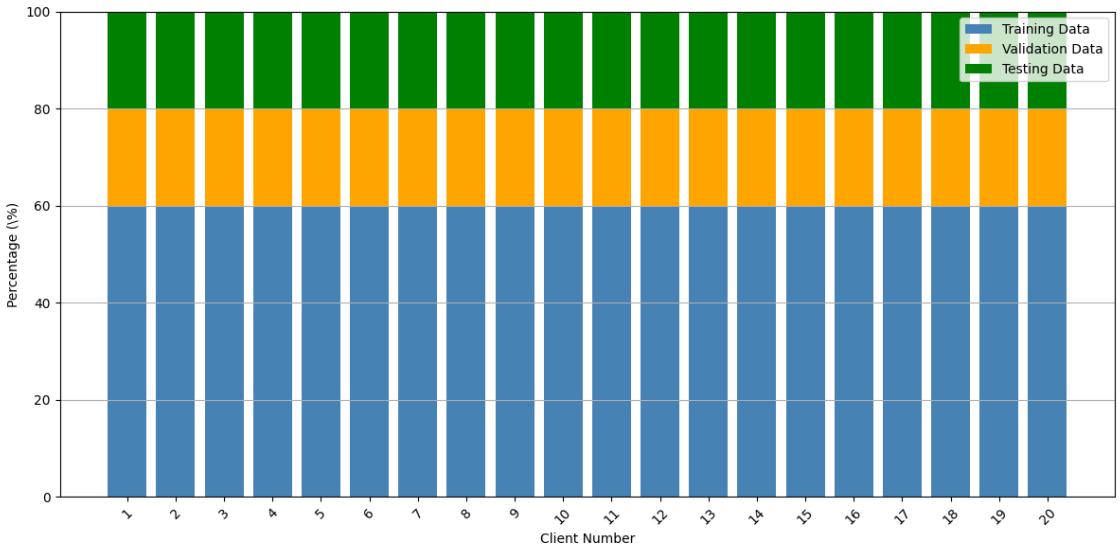


Figure 2. Each FL client splits their local data into training, validation and testing sets

In our work, we investigate both binary and multiclass classification, and these distinctive

tasks implicitly require that each class label is split systematically. Similarly, we opted to split each class label into, 60% training, 20% validation, and 20% testing subsets. Therefore, we continue presenting more the

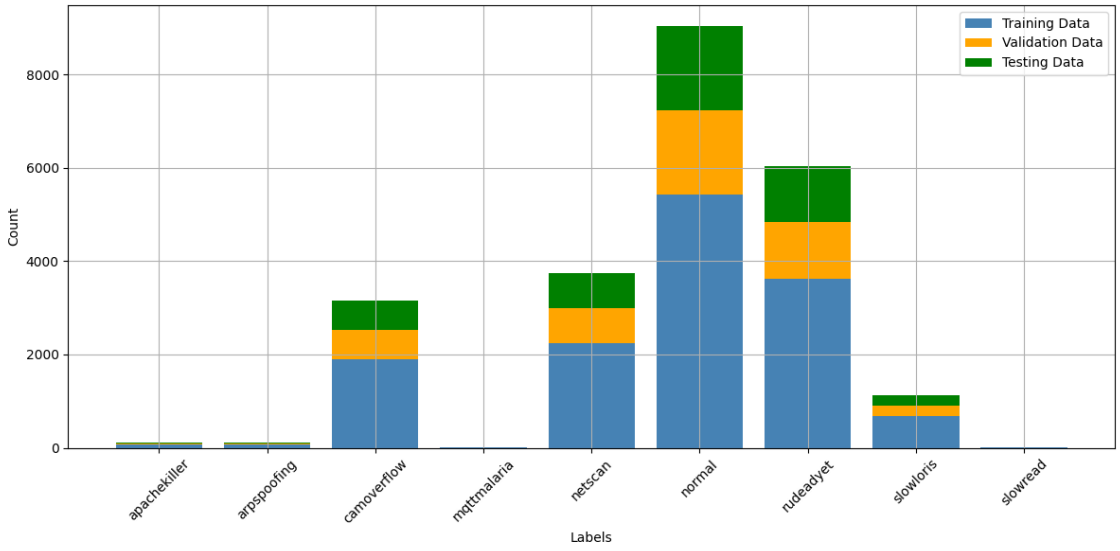


Figure 3. IoMT-TrafficData Client 4: Each multiclass label adheres to data splitting - 60% training, 20% validation and 20% testing sets (Non-IID, $\alpha = 0.5$)

Previously defined Equation (5.5) outputs the required class imbalances for our experiments. Subsequently, these local distributions were subjected to data splitting prodecures to provide each client with their local splits of 60% training, 20% validation and 20% testing data cross each cyberattack category, as depicted in Figure 3 and Figure 4.

An important nuance must be acknowledged - distributions differ significantly across multiple clients, some equipped with an abundant size of a single class, while only a small amount of other classes.

In our work, we present results of experiments related to two separate datasets. As such, data heterogeneity is also extended to dataset CICIoMT2024 for multiclass classification tasks, depicted in Figure 7. Conversely, CICIoMT2024 dataset has inherently skewed distribution towards two more prevalent attack (labeled as "DDoS" and "DoS"), as seen in Figure 8.

The class imbalance inherent in the original dataset influences the partitioned Dirichlet distributions. In the case of this dataset, client possess local dataset with an imbalance

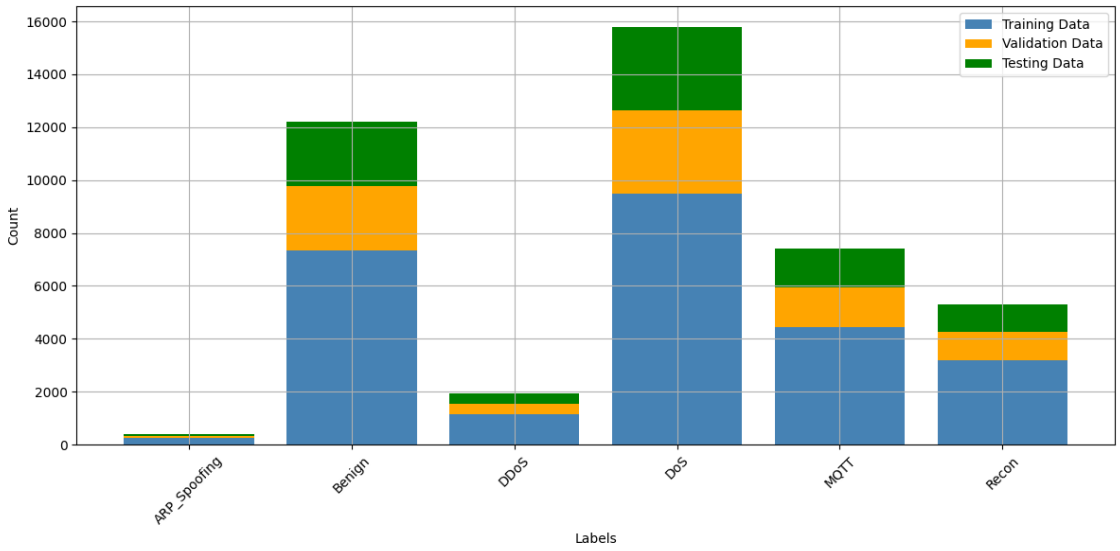


Figure 4. CICIoMT2024, Client 15: Each multiclass label adheres to data splitting - 60% training, 20% validation and 20% testing sets (Non-IID, $\alpha = 0.5$)

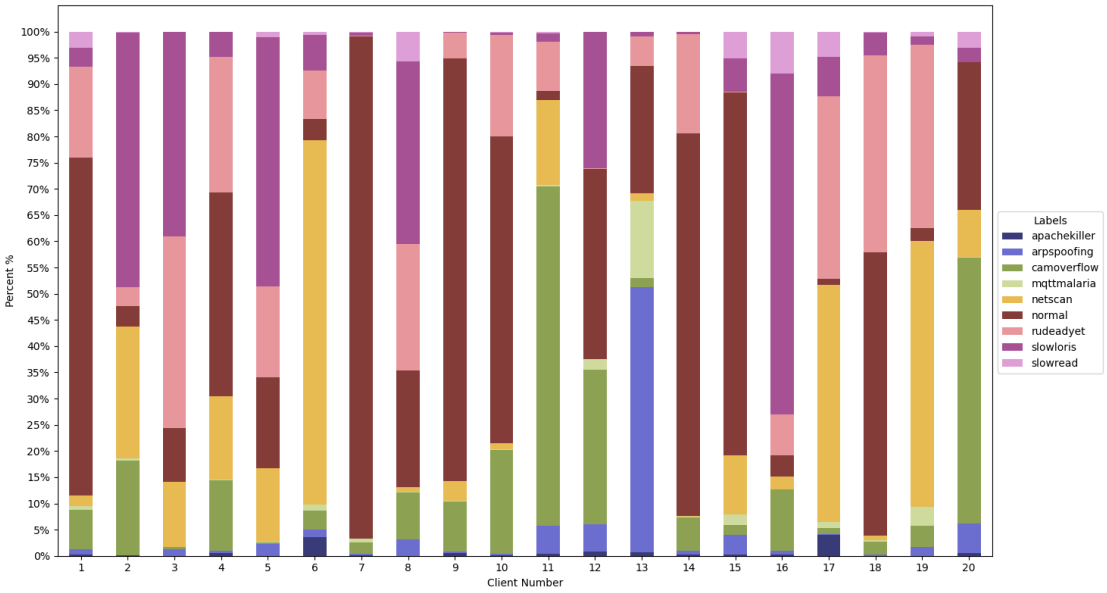


Figure 5. IoMT-TrafficData: Class Distribution for Multiclass Classification (Non-IID, $\alpha = 0.5$).

towards most represented class types, namely network intrusions categorized as Distributed Denial of Service Attacks (DDoS) or Denial of Service Attacks (DoS).

Binary classification with Non-IID implicitly presumes similarly principle to create heterogeneous data environments. In Figure 9, dataset IoMT-TrafficData is split among the binary labels ("Attack", and "Normal"). In addition, the procedure is applied provide the second dataset as well, to simulate heterogeneous data from CICIoMT2024 dataset, for binary classification tasks, illustrated in Figure 10.

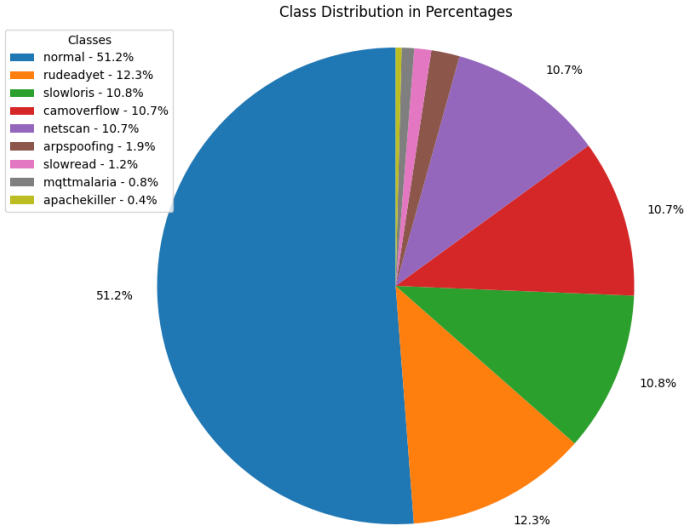


Figure 6. IoMT-TrafficData: Distribution of the original [30] published dataset.

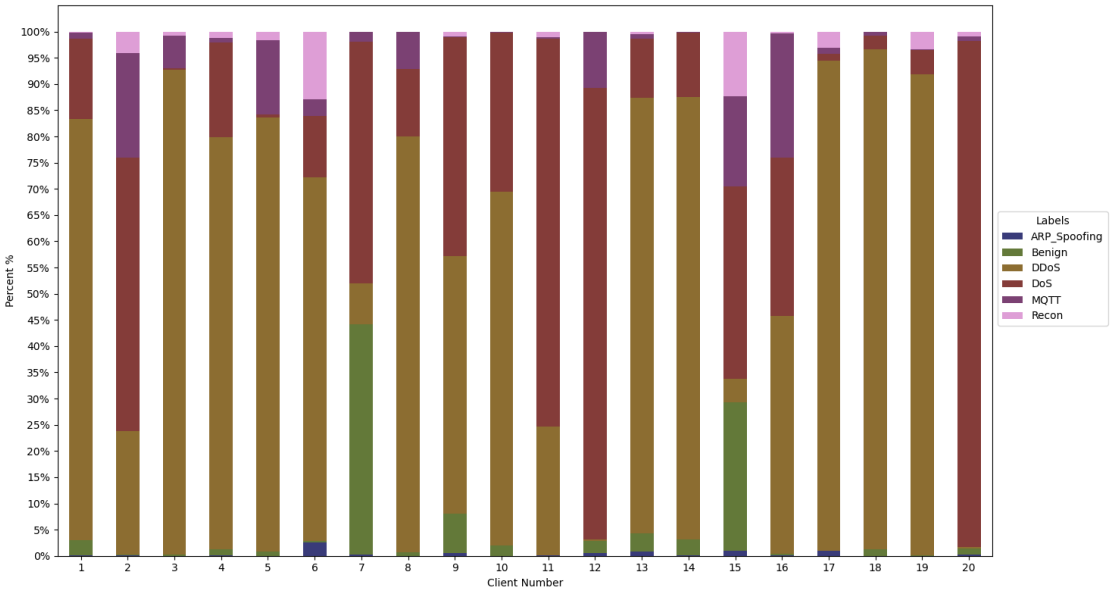


Figure 7. CICIoMT2024: Class Distribution for Multiclass Classification (Non-IID, $\alpha = 0.5$).

Our work also presents supplementary results in Federated Learning with IID data, particularly experiments contending with more homogeneous distributions for both binary and multiclass classification tasks.

In Figure 11 we can see similar distributions across clients in FL for multiclass classification tasks, whilst acknowledging the relatively predominant class labeled "normal" in all clients.

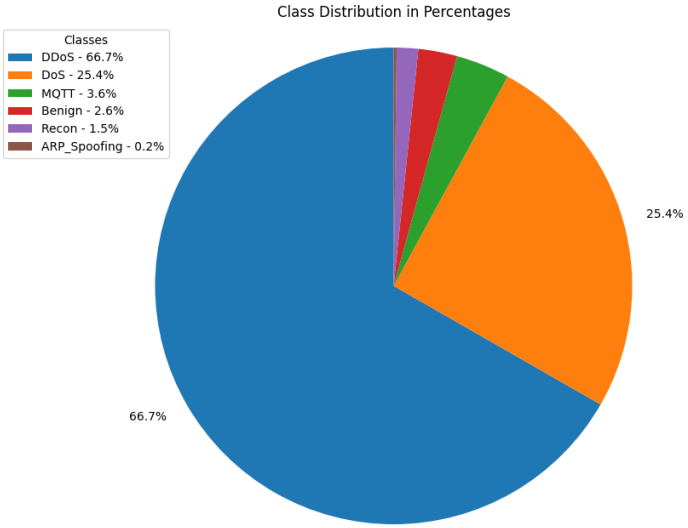


Figure 8. CICIoMT2024: Distribution of the original [29] published dataset.

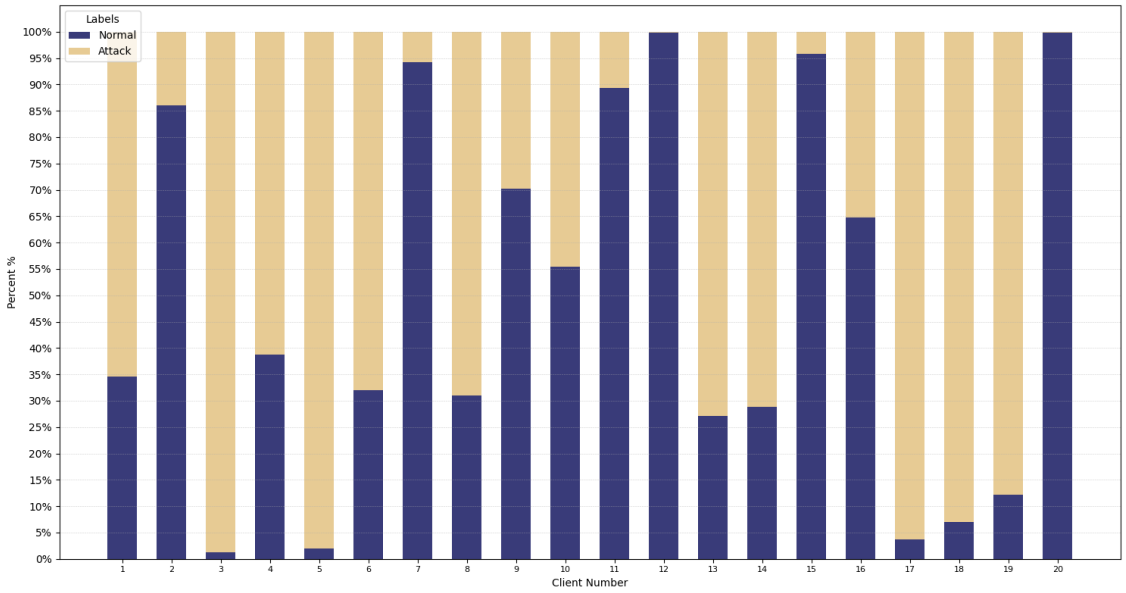


Figure 9. IoMT-TrafficData: Class Distribution for Binary Classification (Non-IID, $\alpha = 0.5$).

In both Figure 11 and Figure 12 we can see more uniform distributions across clients, an suitable precondition for experiments to be conducted in Federated Learning with IID data.

It is important to distinguish between federated Evaluation and centralized Evaluation in relation to testing data. The previously mentioned testing data subsets refer to locally owned data, which are used for federated evaluation. Federated evaluation involves assessing the performance of each client's local model using their distributed testing data (shown

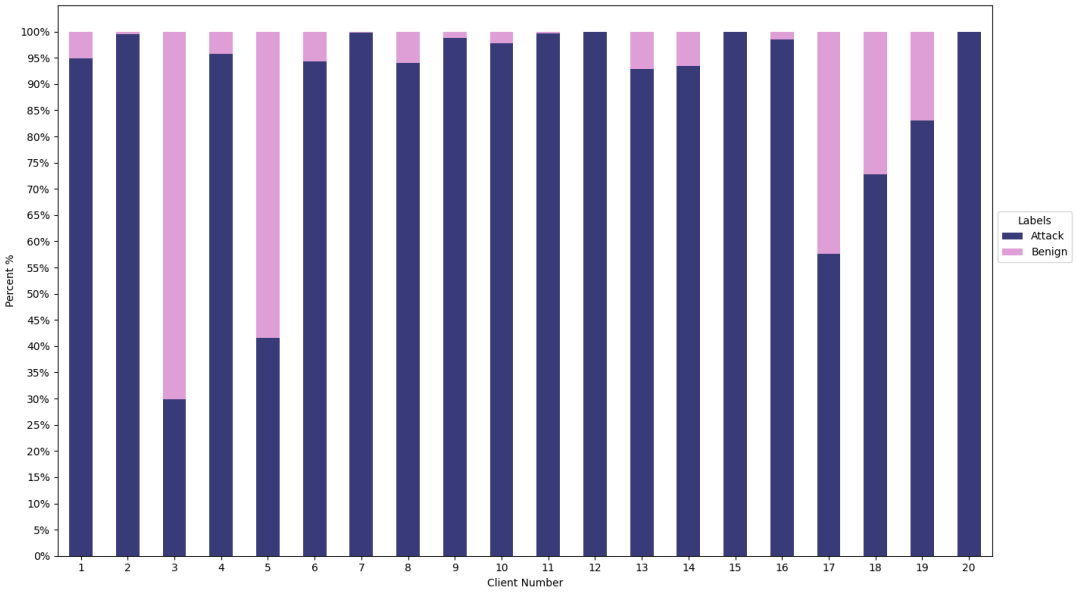


Figure 10. CICIoMT2024: Class Distribution for Binary Classification (Non-IID, $\alpha = 0.5$).

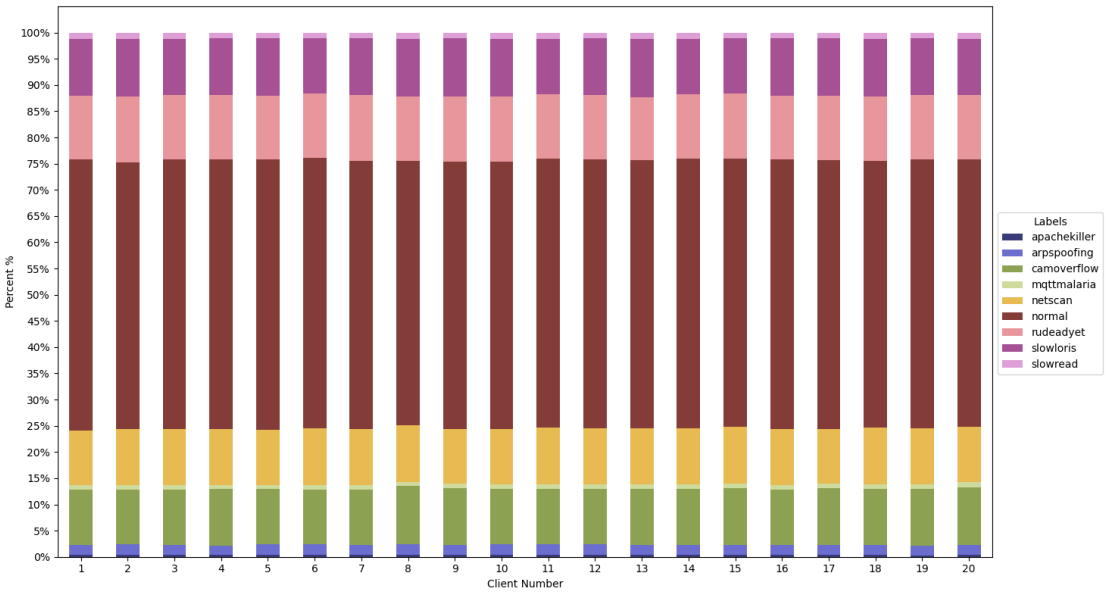


Figure 11. IoMT-TrafficData: Class Distribution for Multiclass Classification (IID).

in Figure 3 and Figure 4). In contrast, Centralized evaluation is conducted on the global model, which the server creates by aggregating local updates sent by the clients.

In our work, global model testing is done with isolated testing data, that resides strictly on the server and not included in any training tasks. For simplicity, we refer to is as server testing data, and label it has such in Figure 13.

The process of analyzing resulting data is crucial step to determine the whether research

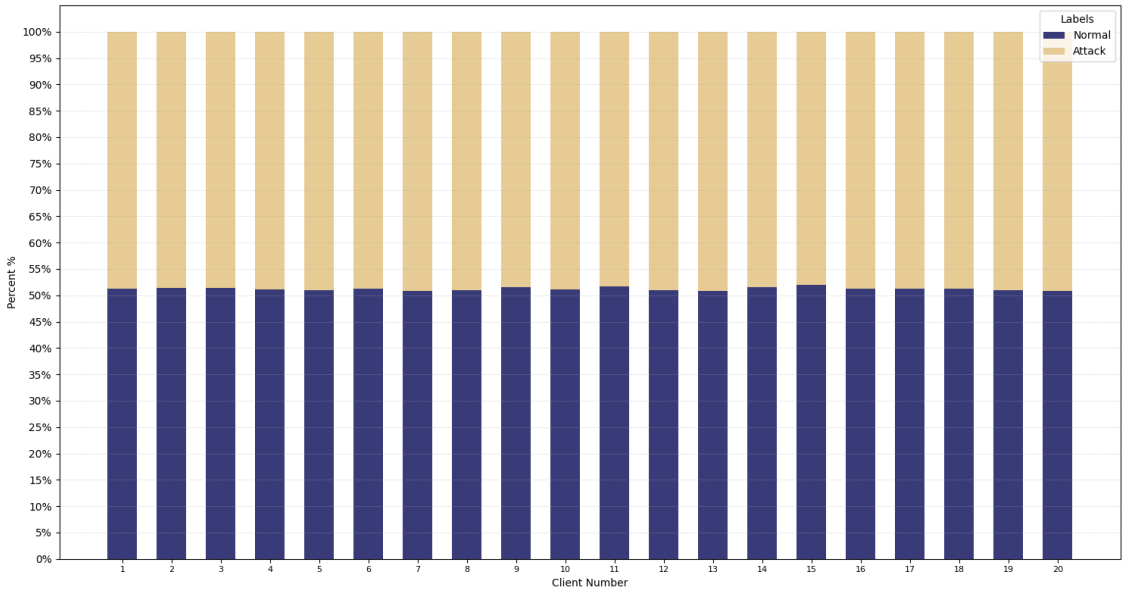


Figure 12. IoMT-TrafficData: Class Distribution for Binary Classification (IID).

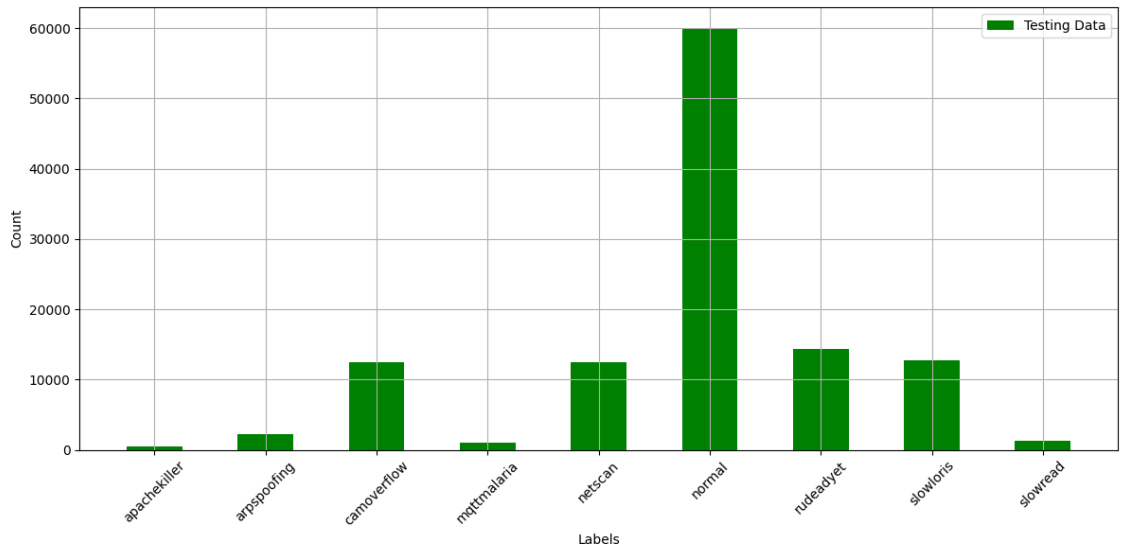


Figure 13. IoMT-TrafficData: Testing data in Centralized evaluation for Multiclass classification

questions were answered, and to provide solid conclusions. The proposed work provides results in Chapter 6 and corresponding discussion in Chapter 7.

6 Results

In this section, we present the findings from experiments conducted with the IoMT-TrafficData [30] and CIC-IoMT2024 [29] datasets. We evaluated the performance of the federated learning (FL) model in both binary and multiclass classification, organizing the results into distinct subsections for clarity. Performance metrics such as accuracy, precision, recall, and F1-score were used to compare the FL optimizers to other contenders.

The experiments were categorized into three scenarios based on the degree of class imbalance in the datasets. Firstly, we assumed non-IID data to better reflect real-world network traffic. Using Dirichlet Distributions with =0.5, we created high label imbalance across clients, referred to as $Dir_{\alpha=0.3}(\cdot)$. Secondly, we explored slightly more imbalanced distributions with =0.3 to assess how different levels of imbalance might affect the performance of FL optimization methods, noted as $Dir_{\alpha=0.3}(\cdot)$. Lastly, we gathered supplementary results in an IID scenario to evaluate whether traditional server-side optimizers outperform the previously mentioned FL optimization approaches.

Results were obtained by sending server-side testing data to the global model for prediction inference. Server-side testing data is not to be confused with client's local testing data, which is used for federated evaluation — assessing a client's ability to make correct predictions using untrained local data on their local model.

Moving forward, we will concentrate on empirical results gathered from centralized evaluation using server-side testing data, which remains unaffected by specific imbalances other than those inherent in the raw data. It's important to note that the global model on the server is built by aggregating local updates generated from machine learning tasks on individual devices. Thus, this centralized evaluation of the global model is crucial for minimizing biases that clients may encounter.

Moving forward, we focus on acquiring empirical results on basis of server-side testing data, that is not subjected to any particular type of imbalances, excluding the inherent imbalance

that exists in the raw ingested data. It is crucial to comprehend that the global model (on the server) was aggregated upon local model updates, and these updates originated from machine-learning tasks on the local device, on their training data. Therefore, the server-side testing data is vital to cast aside distributions that the client is vaguely familiar.

6.1 Binary Classification

In our work, binary classification is used to detect occurrences of network intrusions. The thesis provides results for both homogeneous and heterogeneous data circumstances in separate experiments for the two different datasets.

6.1.1 Binary Classification: Dataset IoMT-TrafficData

Table 2 summarizes the performance of different FL server-side optimizers (optimization methods) for binary intrusion detection on IoMT-TrafficData in the Non-IID setting ($Dir_{\alpha=0.5}(\cdot)$). This table reports the centralized evaluation metrics at the last communication round (i.e., the final global model after training), which is vital because it reflects the model that would be deployed in a healthcare IoMT intrusion detection system. However, it is not sufficient to only look at the final (50^{th}) round results, because the FL process can show instability, slow convergence, or temporary degradation across communication rounds.

Table 2. IoMT-TrafficData: Results for FL Optimizers in Binary Classification ($Dir_{\alpha=0.5}$, Non-IID)

Optimizer	**Accuracy**	**Precision**	**Recall**	**F1-score**
FedAvg	0.99532	0.99549	0.99532	0.99540
FedAvgM	0.99418	0.99425	0.99418	0.99422
FedOpt	**0.99670**	**0.99682**	**0.99670**	**0.99675**
FedYogi	0.99662	0.99674	0.99662	0.99668
FedAdam	0.98959	0.98942	0.98959	0.98949
FedAdagrad	0.99667	0.99678	0.99667	0.99672

Vital information is presented in Figure 14, as it illustrates the performance of different Federated Learning optimizers on the F1-score metric across communication rounds. In particular, the figure shows that the Federated Optimization algorithm (FedOpt) achieves both high performance and stable behavior throughout the training process, with limited

fluctuations. This stability is important in binary intrusion detection, where consistent detection capability is required for healthcare IoMT networks.

Figure 14 further supports this observation by showing the F1-score across communication rounds. It can be observed that FedOpt is resistant to fluctuations and performance degradation. A similar trend is empirically identified for the CICIoMT2024 dataset under the same Non-IID configuration ($Dir_{\alpha=0.5}(\cdot)$), as shown in Section 6.1.2 and Figure 20.

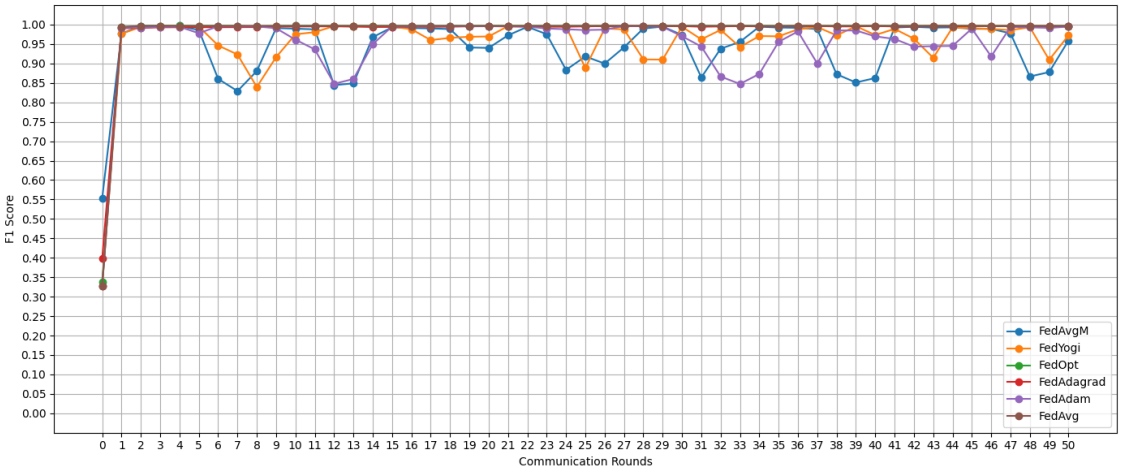


Figure 14. IoMT-TrafficData: F1-score of FL Methods in Binary Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

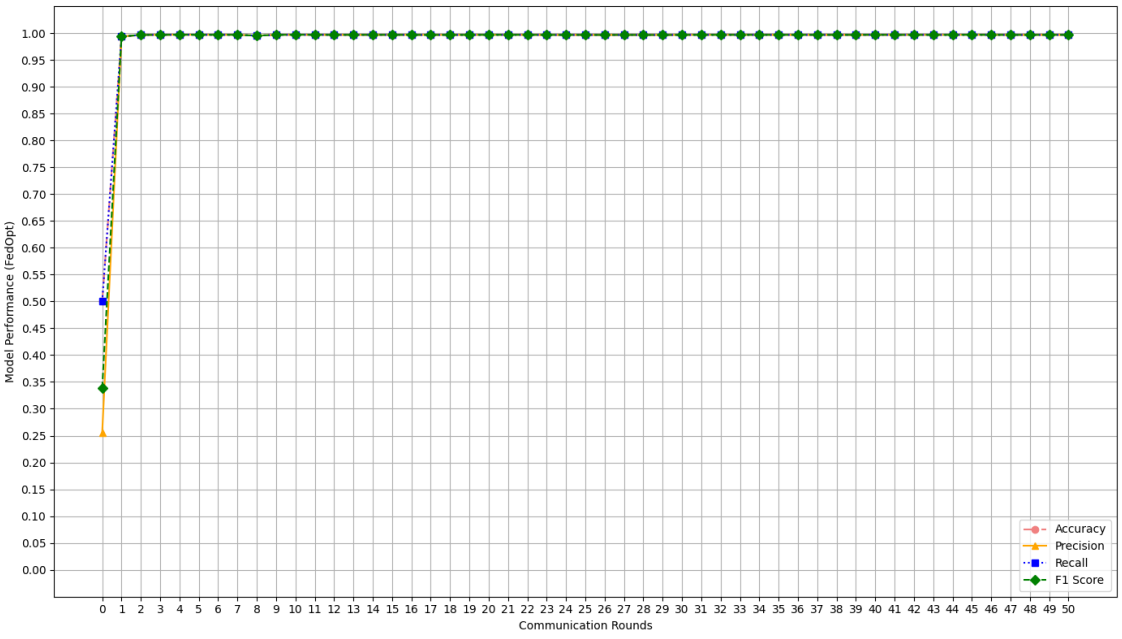


Figure 15. IoMT-TrafficData: Metrics of FedOpt Method in Binary Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

Overall, these results indicate that FedOpt is the most reliable server-side optimizer for binary intrusion detection in federated healthcare IoMT networks under Non-IID data

distributions.

Federated Optimization Algorithm (FedOpt) merits outperforms in F1-score. As such, more elaborate results can be extracted from this particular optimizer in the context of these two datasets in Non-IID situation ($Dir_{\alpha=0.5}(\cdot)$ configuration) in Equation (5.5)).

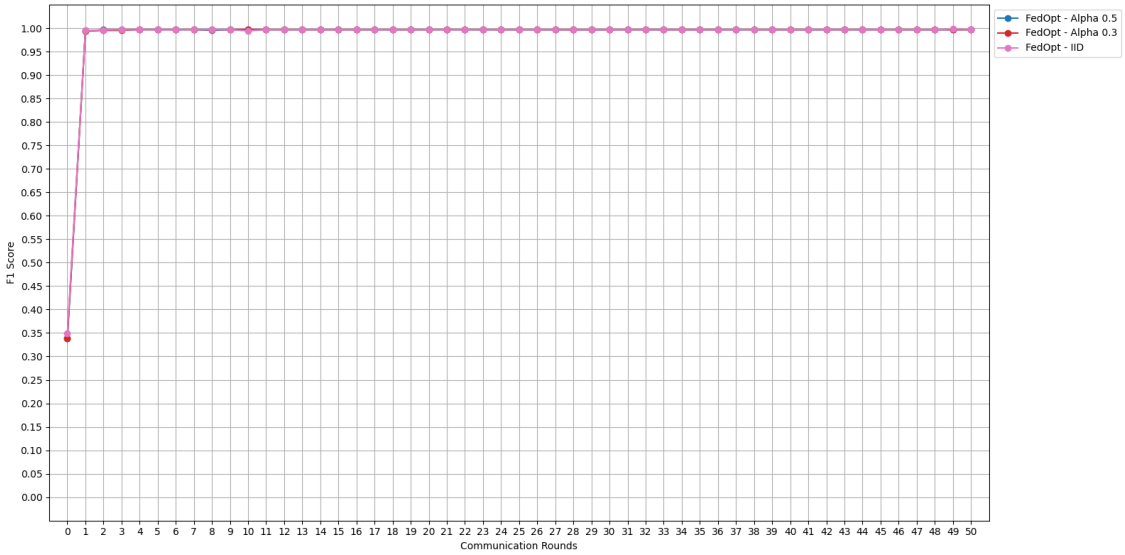


Figure 16. IoMT-TrafficData: F1-score of FedOpt Method in Binary Classification ($Dir_{\alpha=0.3}(\cdot)$, $Dir_{\alpha=0.5}(\cdot)$ and IID).

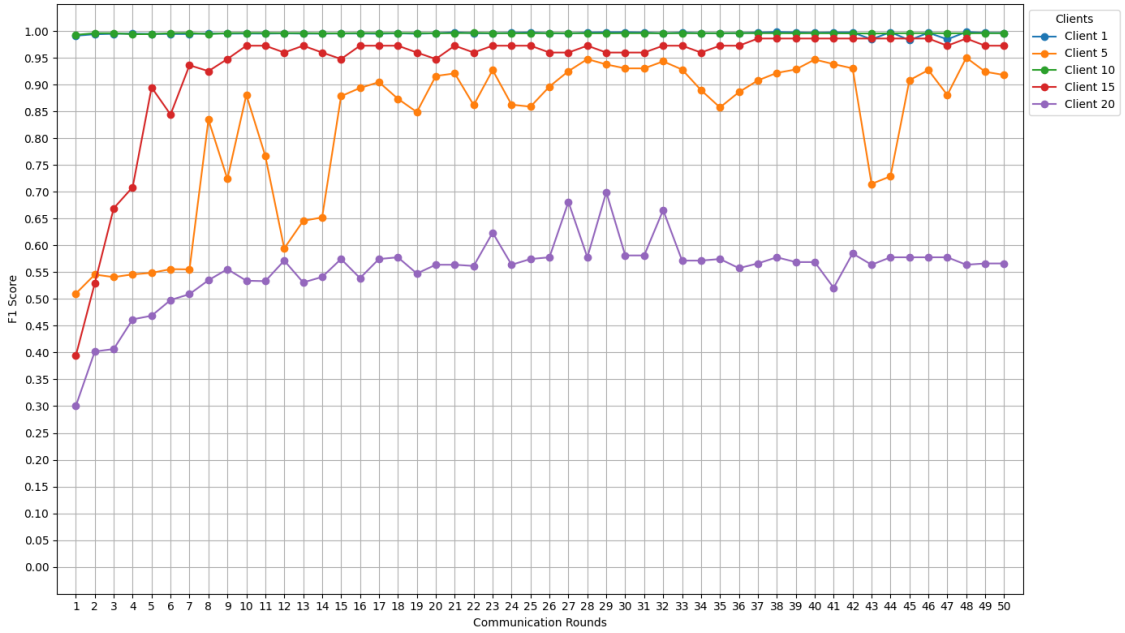


Figure 17. IoMT-TrafficData: Clients using FedOpt Method in Binary Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

As mentioned before, the experiments can be divided based on the degree of imbalance, data heterogeneity. Given the results of FedOpt, we can see potential difference in model

performance in Non-IID with $\alpha = 0.3$, Non-IID with $\alpha = 0.5$ and IID scenario. This can be seen for both datasets, depicted in Figure 16 and Figure 22.

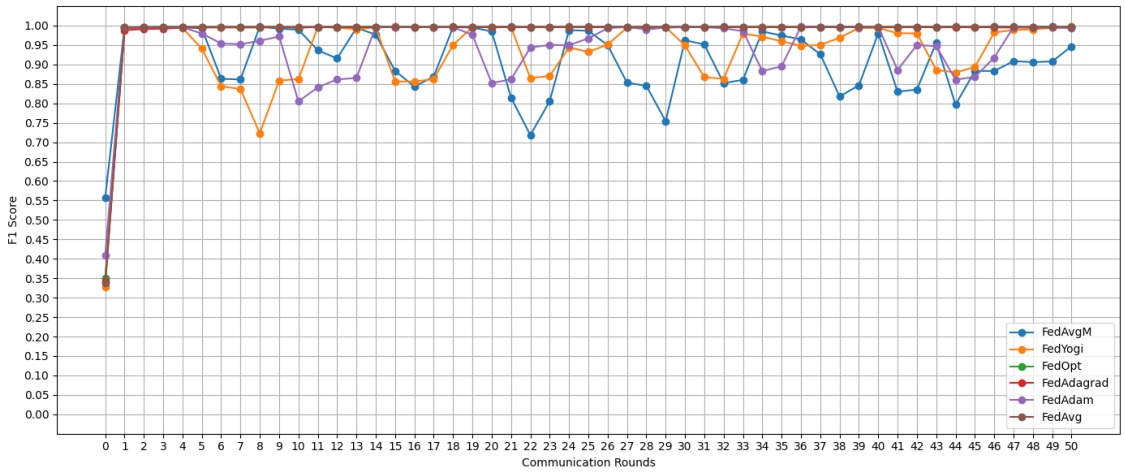


Figure 18. IoMT-TrafficData: F1-score of FL Methods in Binary Classification (IID).

Federated evaluation was discussed to measure model performance on the client, with their dedicated testing data. Importantly, the testing data must have remained isolated from any training or validation tasks to ensure it remained unseen by the trained model. Meeting this requirement, the client-side local testing data can be used to measure local model performance. As seen in the Figure 17, on each communication round clients' local model adjusts to the global model.

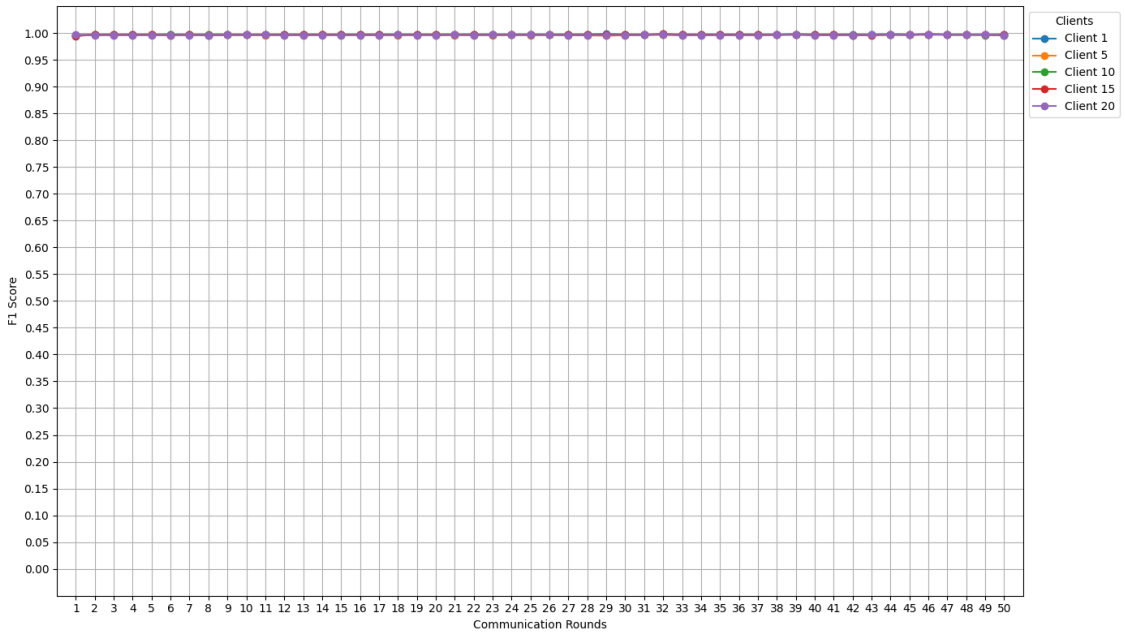


Figure 19. IoMT-TrafficData: Clients using FedAvg Method in Multiclass Classification (IID).

It is imperative to identify the model performance in IID situation, regardless its inapplicability in real-world scenarios of Intrusion Detection in networks. We can notice FedAvg Algorithm has outstanding results in both centralized evaluation Figure 18 and client-side federated evaluation Figure 19.

6.1.2 Binary Classification: Dataset CICIoMT2024

Table 3 presents the centralized evaluation results of binary intrusion detection on CICIoMT2024 for multiple FL server-side optimizers under Non-IID ($Dir_{\alpha=0.5}(\cdot)$). This table depicts the metrics at the last communication round, which is vital because it represents the final global model performance that is relevant for deployment of IDS in healthcare IoMT networks. Nevertheless, only reporting the last (50th) round does not fully describe the learning behaviour of each optimizer, since some methods may converge faster, fluctuate more, or degrade before stabilizing. In Table 3, FedAdagrad seemingly exhibits higher results, but across the communication rounds FedOpt seems to be more reliable. Section 6.1.1 IoMT-TrafficData also exhibited FedOpt the most reliable and effective, common trend among the datasets (illustrated in Figure 14 and Table 2.

but closer inspection of model performance across FL communication rounds depicts FedOpt experiments

For this reason, the detailed behavior across the full federated learning process is shown in Figure 20, where the communication-round evolution enables comparison of all FL optimizers (server-side optimization methods), and not strictly the final round.

Table 3. CICIoMT: Results for FL Optimizers in Binary Classification ($Dir_{\alpha=0.5}$, Non-IID)

Optimizer	**Accuracy**	**Precision**	**Recall**	**F1-score**
FedAvg	0.9228	0.9767	0.9228	0.9481
FedAvgM	0.7646	0.8497	0.7646	0.8009
FedOpt	0.8815	0.9874	0.8815	0.9276
FedYogi	0.8648	0.9843	0.8648	0.9157
FedAdam	0.5359	0.9740	0.5360	0.5608
FedAdagrad	**0.9200**	**0.9794**	**0.9200**	**0.9476**

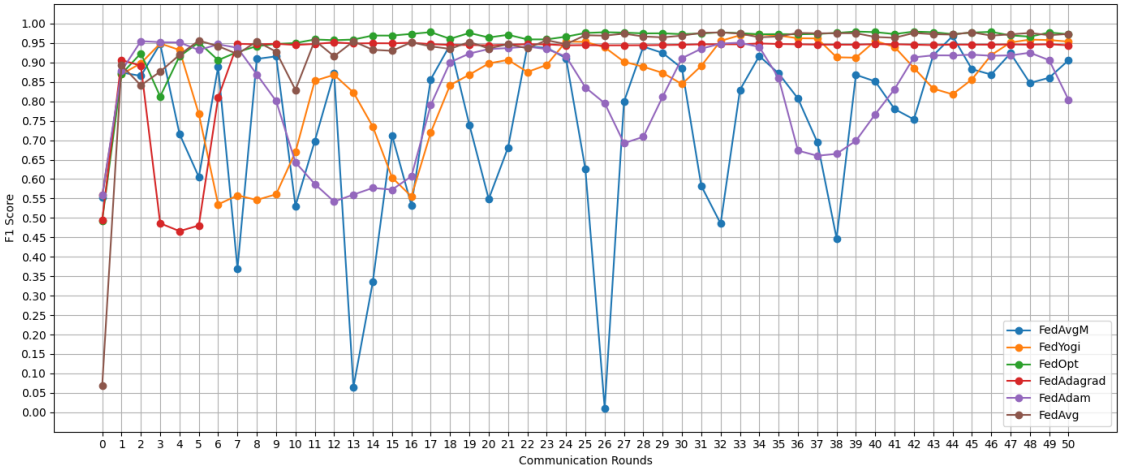


Figure 20. CICIoMT2024: F1-score Metrics for FL Methods in Binary Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

The Federated Optimization Algorithm (FedOpt) provides relatively high performance across accuracy, precision, recall and F1-score. Results indicate that this federated server-

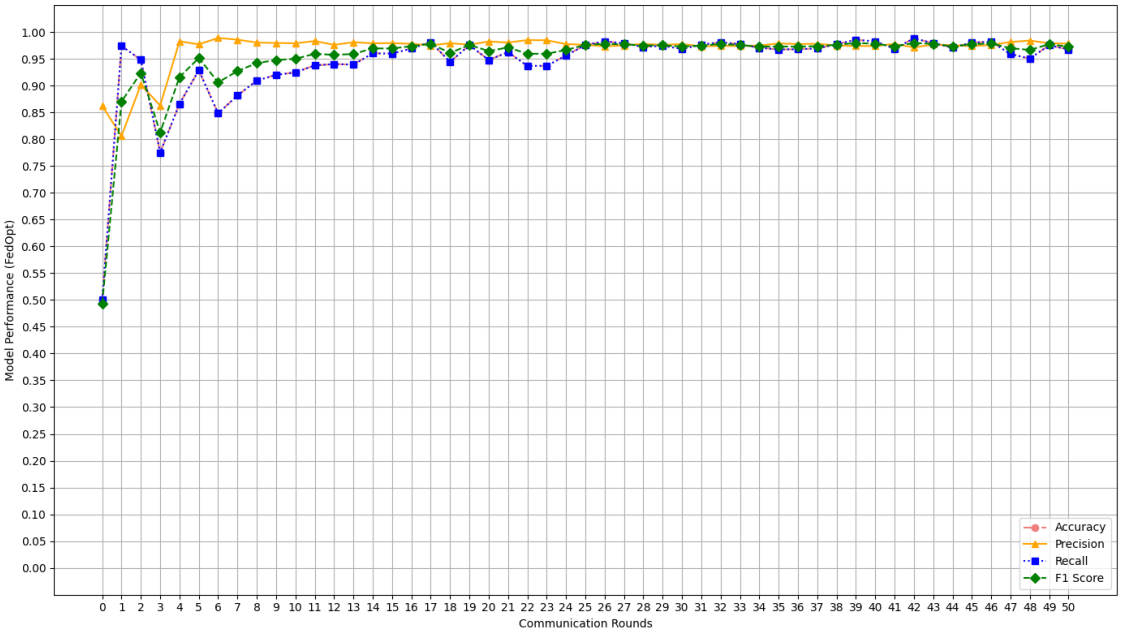


Figure 21. CICIoMT2024: Metrics of FedOpt Method in Binary Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

side optimization method is performing well in various scenarios of data heterogeneity. This common pattern in FedOpt Algorithm is noticeable also in Non-IID scenarios, where $Dir_{\alpha=0.3}(\cdot)$, and in IID settings (depicted in Figure 22).

Model performance rates can be measured on the client-side with their locally owned testing data. Federated evaluation is depicted in Figure 23. The same principle was applied to dataset IoMT-TrafficData (in Figure 17) to observe client-side model performance per

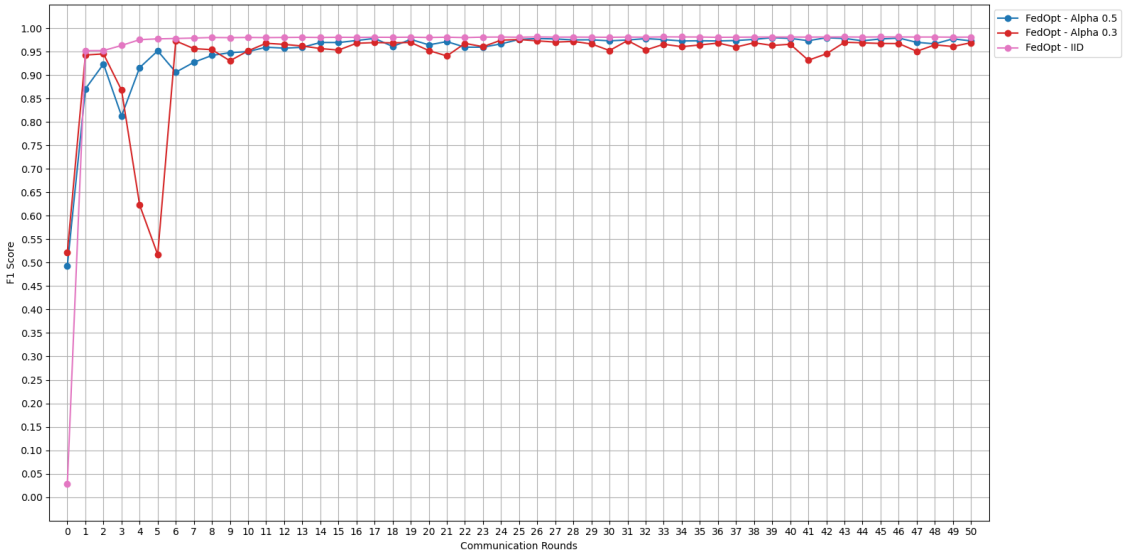


Figure 22. CICIoMT2024: F1-score of FedOpt Method in Binary Classification ($Dir_{\alpha=0.3}(\cdot)$, $Dir_{\alpha=0.5}(\cdot)$ and IID).

communication round. Results describe poor performance for some clients, and some experiencing failures to participate entirely due to limited allocated resources, common in real-world Federated Learning architecture with resource-strained clients.

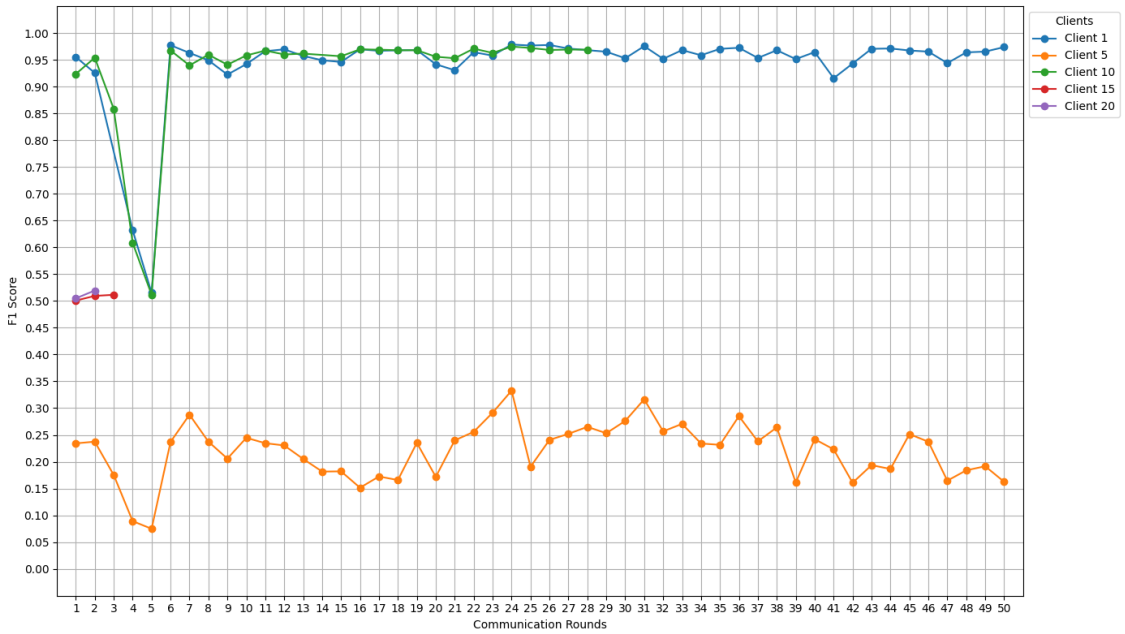


Figure 23. CICIoMT2024: Clients using FedOpt Method in Binary Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

Experiments conducted on IID data, FedAvg Algorithm and provide consistent performance across accuracy, recall, precision and F1-score Figure 25. Similar trend can be noticed that federated evaluation across clients is much more uniform in IID setting for both

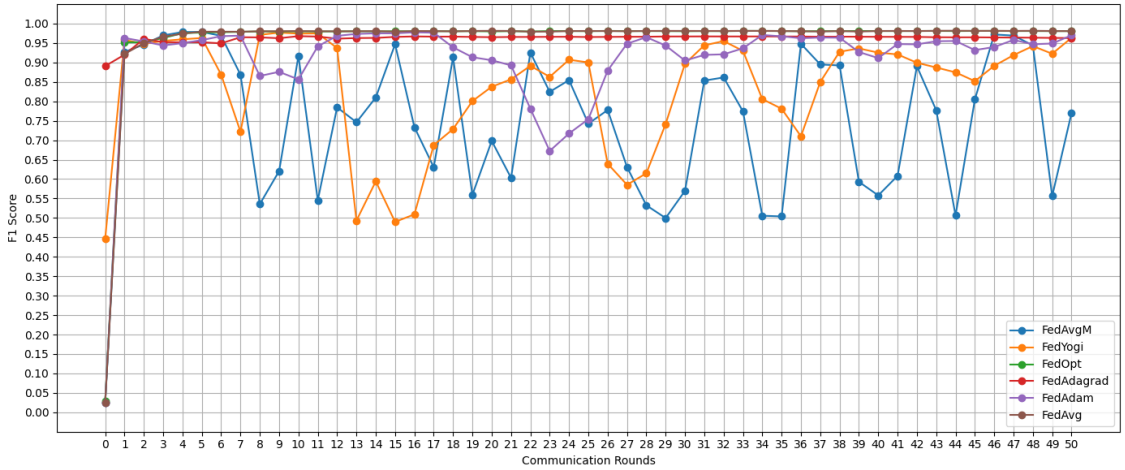


Figure 24. CICIoMT2024: F1-score of FL Methods in Binary Classification (IID).

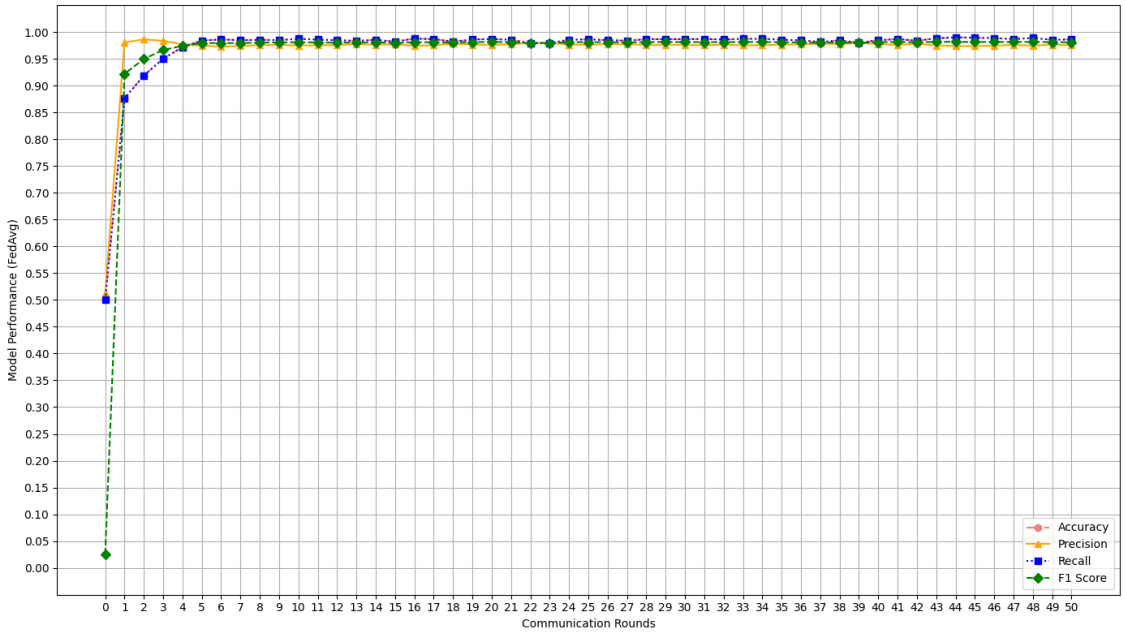


Figure 25. CICIoMT2024: Metrics of FedAvg Algorithm in Binary Classification (IID).

datasets CICIoMT2024 Figure 26, and IoMT-TrafficData (previously depicted in Figure 19) once FedAvg is applied without adaptive server-side optimization. Server-side optimizers seem to exhibit volatility on F1-score metric across communication rounds in the case of homogeneous datasets (IID). Note that, adaptive server-side optimizers fluctuation substantially more in the case for IID dataset CICIoMT2024 (in Figure 24) compared to IID dataset IoMT-TrafficData (previously illustrated in Figure 18).

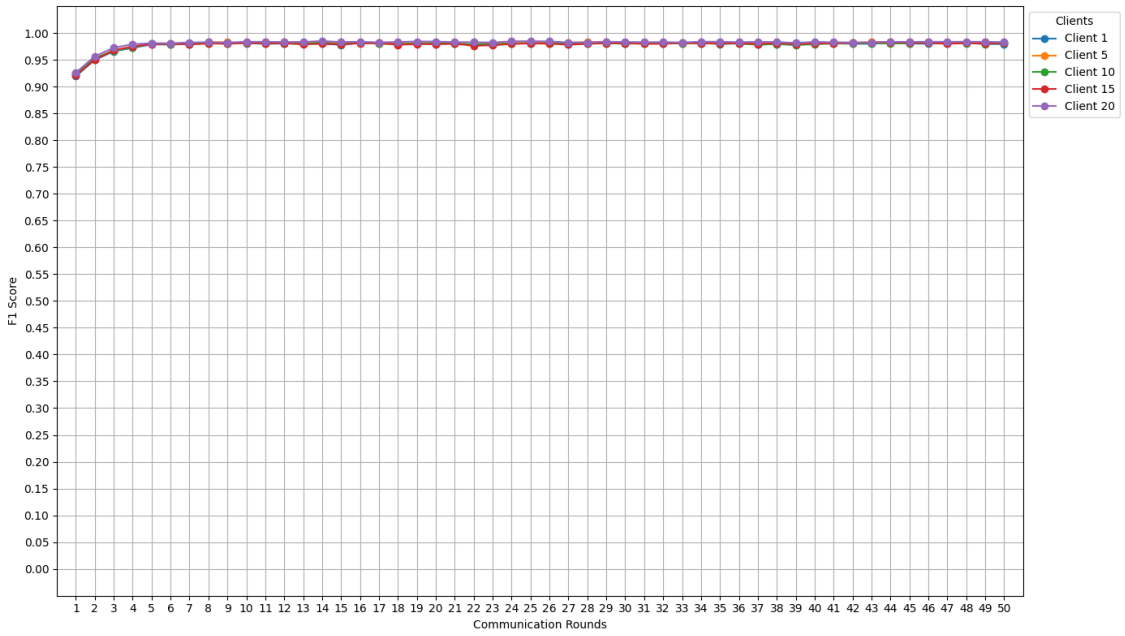


Figure 26. CICIoMT2024: Clients using FedAvg Method in Binary Classification (IID).

6.2 Multiclass Classification

Multiclass classification in federated learning requires innovative approaches to model training, as it involves multiple class labels for prediction. Federated environments add a layer of complexity due to the decentralized nature of the data and potential variations in class distributions among clients. Implementing multiclass classification can significantly enhance Intrusion Detection Systems by allowing them to differentiate various attack vectors, thus improving overall network security.

Since data heterogeneity more accurately reflects real network traffic, we present results based on a specified degree of class imbalance, represented as $Dir_{\alpha=0.5}(\cdot)$ in the Dirichlet distributions Equation (5.5). Furthermore, additional results are derived from more uniform local datasets to evaluate the server-side optimizers in IID conditions, and determine efficacy in such scenarios.

6.2.1 Multiclass Classification: Dataset CICIoMT2024

Table 4 reports the centralized evaluation performance of the global model for multiclass intrusion detection on CICIoMT2024 in the Non-IID setting ($Dir_{\alpha=0.5}(\cdot)$). This table corresponds to the last communication round, which is important because it describes the final capability of the deployed global IDS model to classify different attack types in

healthcare IoMT networks. However, in multiclass classification, it is especially important to not only consider the last (50th) round, since optimizers can show different convergence behaviours and class-level instability during training.

Table 4. CICIoMT2024: Accuracy, Precision, Recall and F1-score Metrics for FL Optimizers in Multiclass Classification ($Dir_{\alpha=0.5}$, Non-IID)

Optimizer	**Accuracy**	**Precision**	**Recall**	**F1-score**
FedAvg	0.6390	0.7928	0.6390	0.6298
FedAvgM	0.7069	0.7049	0.7069	0.6910
FedOpt	0.7573	0.7422	0.7573	0.6570
FedYogi	0.7326	0.7008	0.7326	0.7045
FedAdam	0.7353	0.7585	0.7353	0.6378
FedAdagrad	**0.7643**	**0.8227**	**0.7643**	**0.7682**

Gathered results show that FedAdagrad is performing reliably across communication rounds (refer to Figure 27), outperforming FedOpt Algorithm that exhibited efficacy in binary classification tasks for this dataset (illustrated in Figure 14) IoMT-TrafficData (depicted in Figure 20). Multiple federated server-side methods demonstrate stable

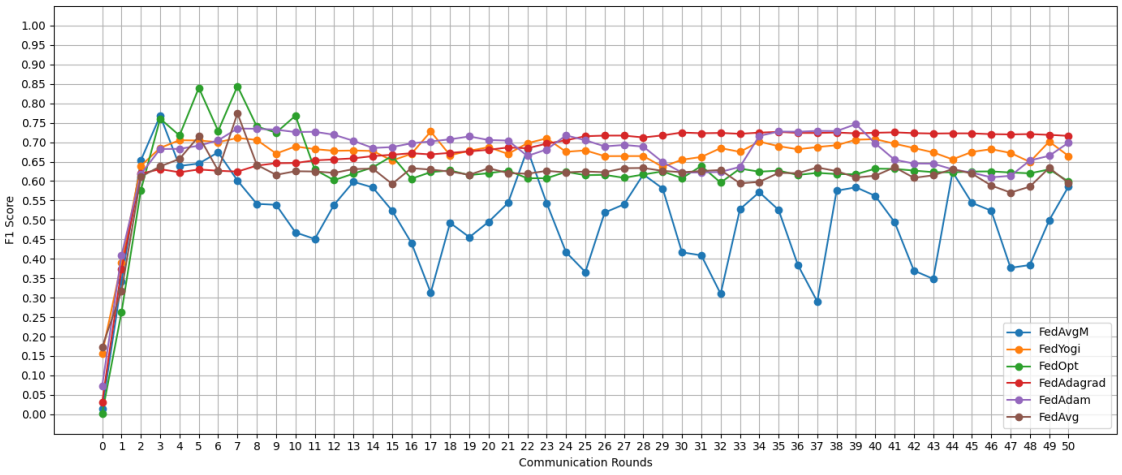


Figure 27. CICIoMT2024: F1-score for FL Methods in Multiclass Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

rates on F1-score metric. Results indicate that Federated Adaptive Gradient Algorithm (FedAdagrad) outperforms FedOpt which excelled in binary classification. Similar performance of FedAdagrad Algorithm is identifiable also under IID, while higher imbalance ($Dir_{\alpha=0.3}(\cdot)$) induces a model with lowered performance, as seen in Figure 29.

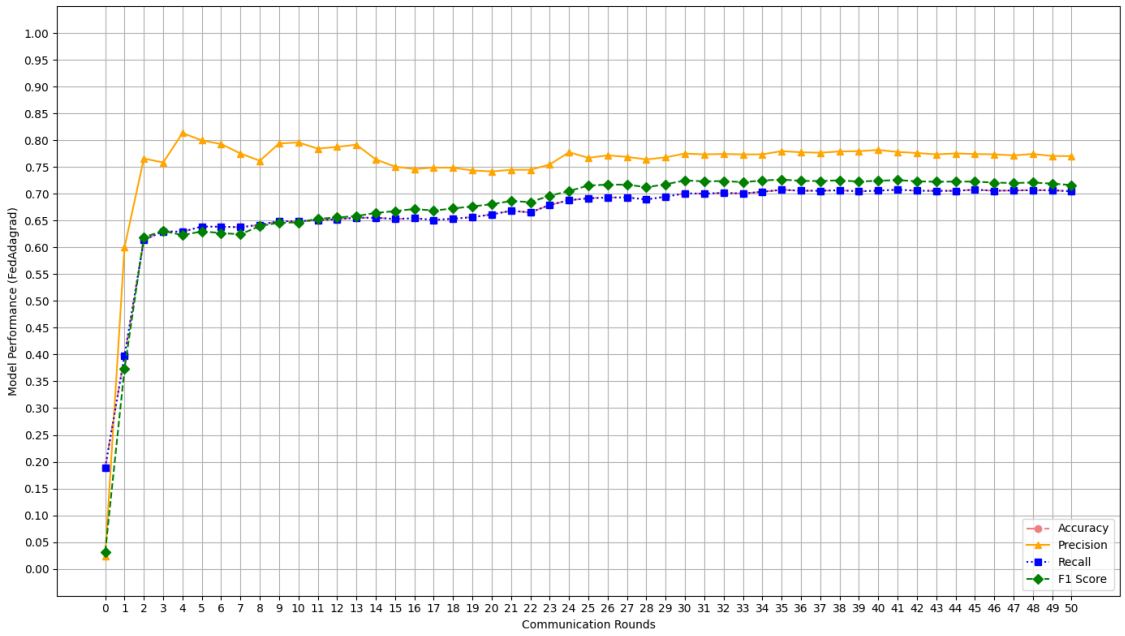


Figure 28. IoMT-TrafficData: Metrics of FEDADAGRAD Method in Multiclass Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

The Federated Adaptive Gradient Algorithm (FEDADAGRAD) provides stable model performance in metrics such as accuracy, precision, recall and F1-score (in Figure 28).

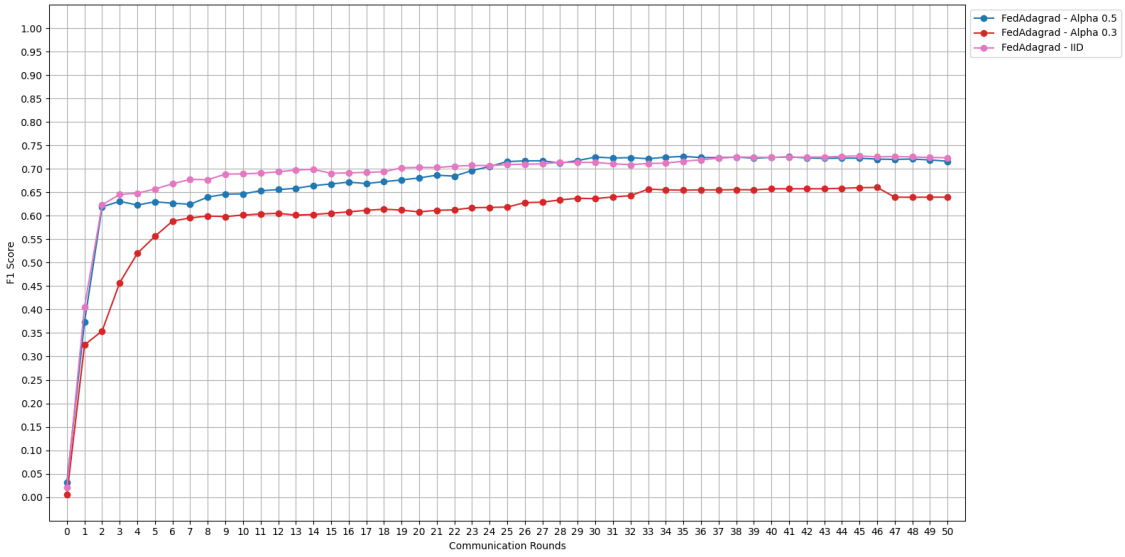


Figure 29. CICIoMT2024: F1-score of FEDADAGRAD Method in Multiclass Classification ($Dir_{\alpha=0.3}(\cdot)$, $Dir_{\alpha=0.5}(\cdot)$ and IID).

Federated evaluation was conducted to observe client-side model behavior. Seemingly, some clients exhibit poor performance (depicted in Figure 30), potentially due to discrepant

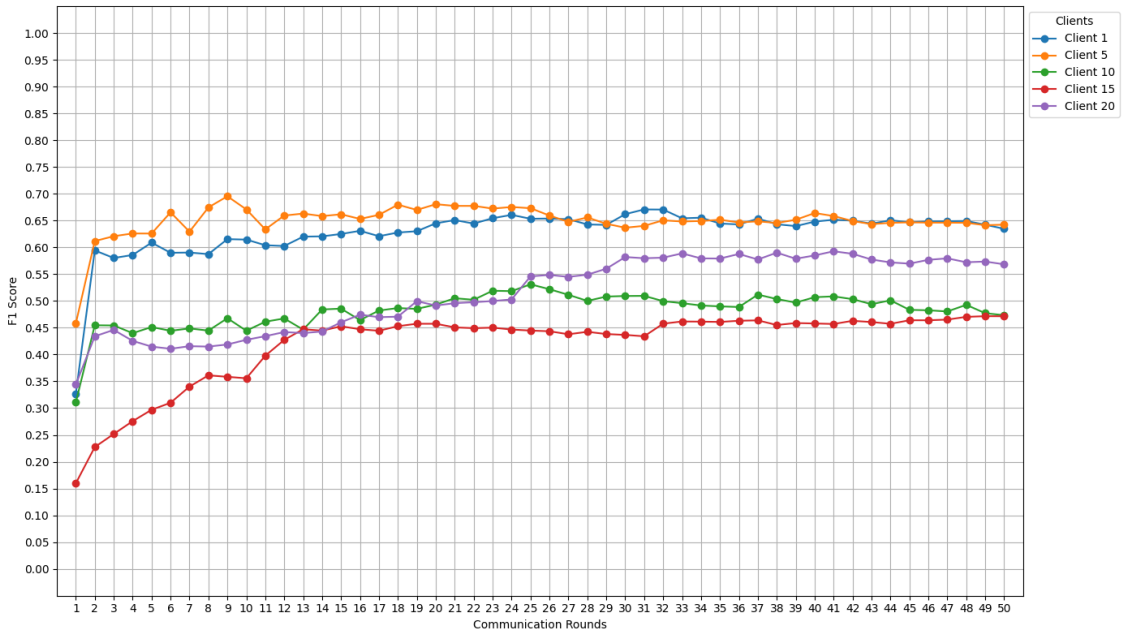


Figure 30. CICIoMT2024: Clients using FedAdagrad Method in Multiclass Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

representation of class labels (depicted in Figure 7). Intricate results are shown in Figure 31 which depict global model performance in F1-score by each class. Clear discrepancies are shown, as model struggles to correctly infer between DoS and DDoS type of attacks. Furthermore, ARP Spoofing attacks are difficult to infer, due to their under representation in all local datasets (illustrated Experimental Setup, Figure 7).

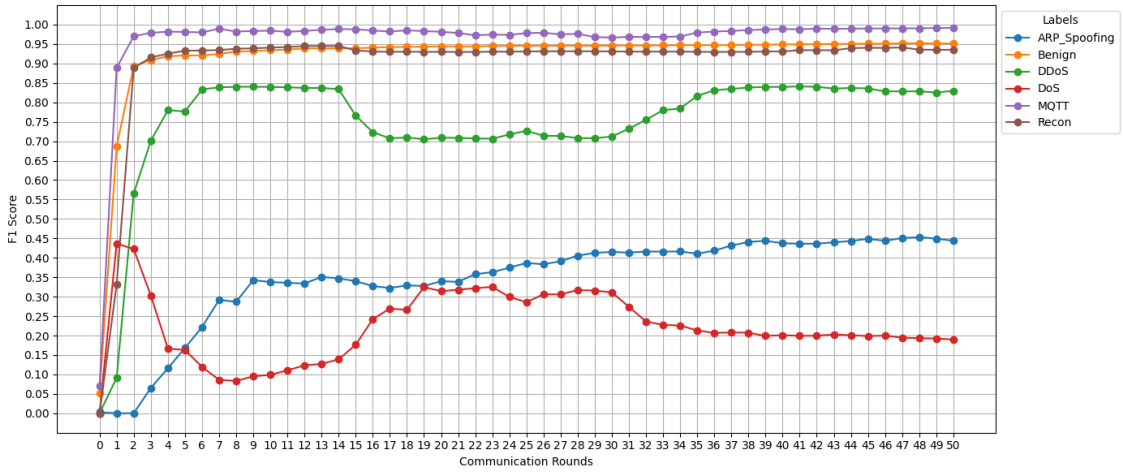


Figure 31. CICIoMT2024: F1-score in class predictions using FedAdagrad Method in Multiclass Classification ($Dir_{\alpha=0.5}(\cdot)$, Non-IID).

Adaptive federated optimizers fluctuate significantly more in the case for IID dataset

CICIoMT2024 (in Figure 32) compared to IID dataset IoMT-TrafficData (illustrated later in Figure 41). Regardless, FedAvg outpeforms adaptive optimizers in IID scenario.

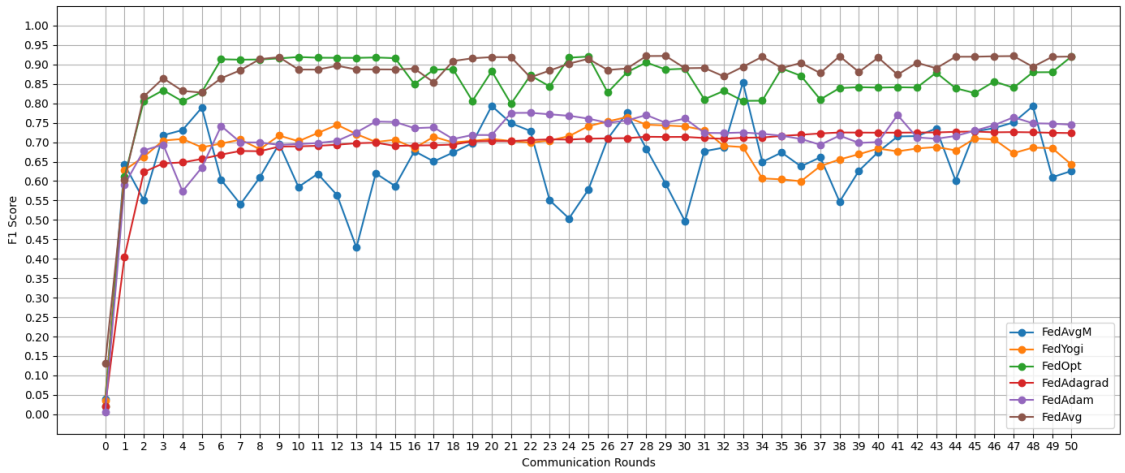


Figure 32. CICIoMT2024: F1-score for FL Methods in Multiclass Classification (IID).

Equally important is to provide results conducted in homogeneously distributed local datasets, reflecting IID data environments. In Figure 33, centralized evaluation is depicted in accuracy, recall, precision and F1-score with recurring insignificant fluctuations.

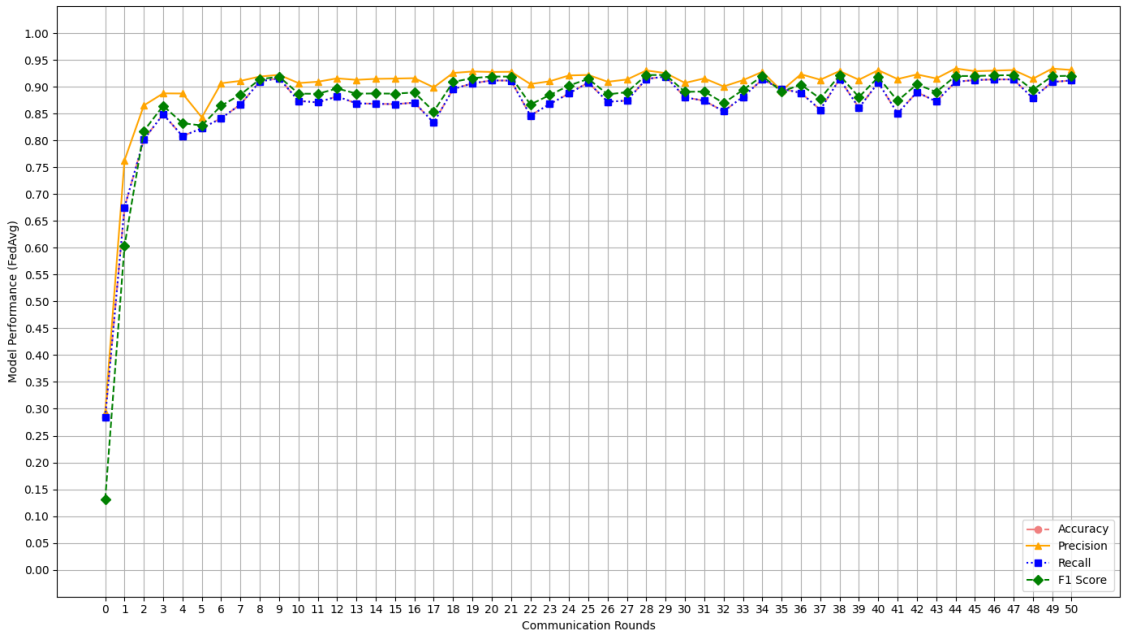


Figure 33. CICIoMT2024: Metrics of FedAvg Algorithm in Multiclass Classification (IID).

Federated Evaluation offers a detailed description of local model performance for clients. Surprisingly, Federated Learning with FedAvg Algorithm lacks in performance across

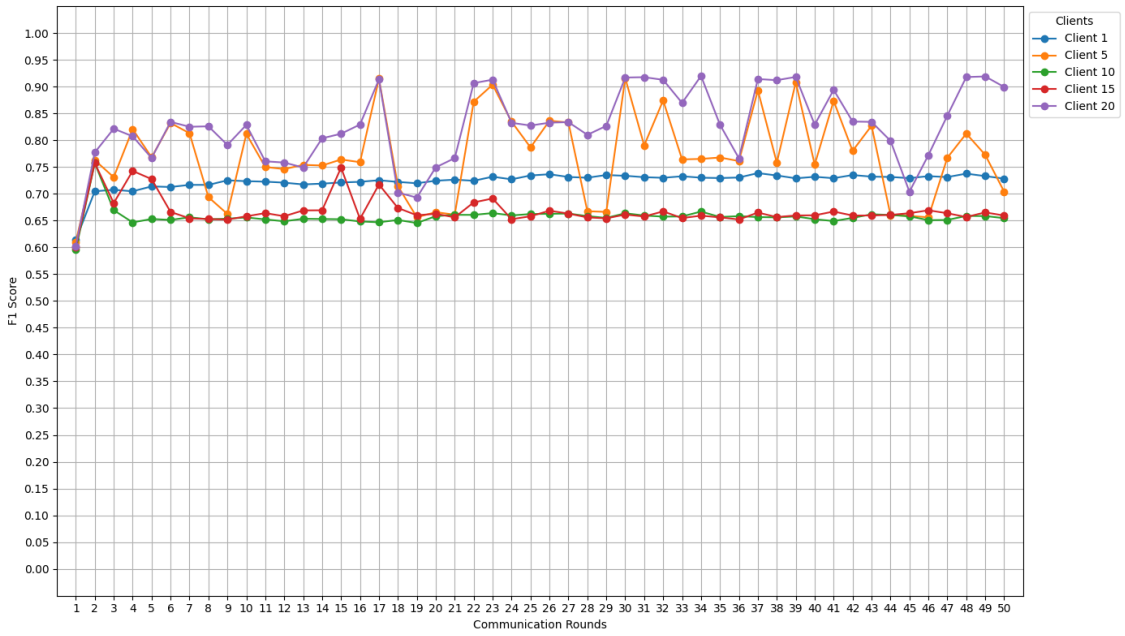


Figure 34. CICIoMT2024: Clients using FedAvg Method in Multiclass Classification (IID).

clients, dropping down to 65% to 66% on F1-score (shown in Figure 34). Moreover, the model performance can also be measured by specific class label, and illustrated in Figure 35. Evidence suggests inability to reliably infer predictions about type of attacks categorized

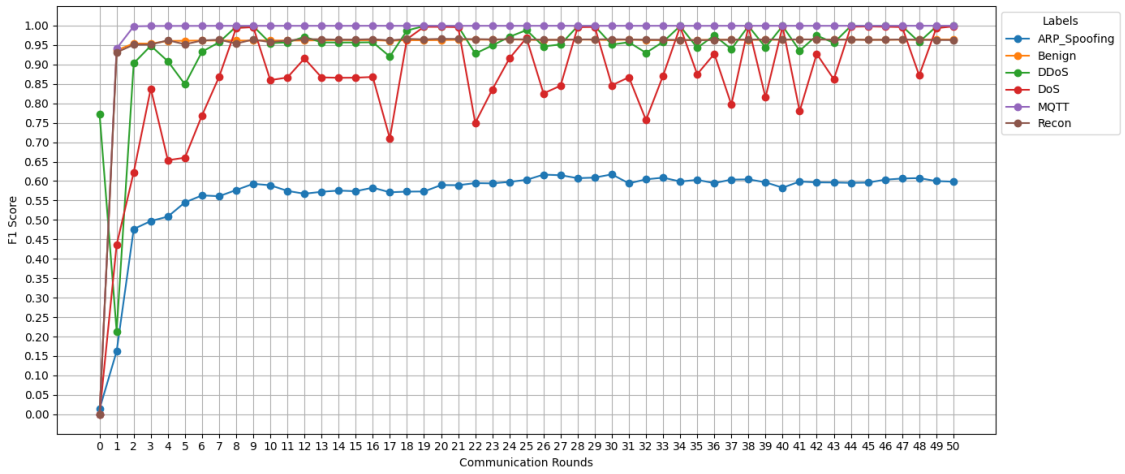


Figure 35. CICIoMT2024: F1-score in class predictions using FedAvg Method in Multiclass Classification (IID).

as "ARP Spoofing". The label is greatly under represented across all clients, as in the original CICIoMT2024 datasets original distribution [29]. This can be seen as a potential limitation of our experimental setup, as we too heavily relied upon the default distribution of the ingested raw datasets, when performing Dirichlet distribution Equation (5.5).

6.2.2 Multiclass Classification: Dataset IoMT-TrafficData

To identify the best-performing federated server-side optimizer, we first look at the results from IoMT-TrafficData dataset. Here, both FedOpt and FedAdagrad achieved high F1-scores in multiclass classification tasks, as shown in Figure 36. However, this conclusion does apply to the CICIoMT2024 dataset.

Table 5 reports the centralized evaluation performance for multiclass intrusion detection on the IoMT-TrafficData dataset, comparing several FL server-side optimizers in a Non-IID setting ($Dir_{\alpha=0.5}(\cdot)$).). The metrics are taken from the last communication round, which is important since it represents the final global model that could be used for deployment in a healthcare IoMT intrusion detection system. However, it is not enough to only consider the final (50^{th}) round, because different optimizers can show fluctuations or slower convergence during training, and these behaviours are also important when choosing an optimizer for practical use.

Therefore, the communication-round based comparison is illustrated in Figure 36, which allows observing the full FL process and comparing all federated optimization methods across rounds, not only the final results.

Table 5. IoMT-TrafficData: Accuracy, Precision, Recall and F1-score Metrics for FL Optimizers in Multiclass Classification ($Dir_{\alpha=0.5}$, Non-IID)

Optimizer	Accuracy	Precision	Recall	F1-score
FedAvg	0.7969	0.8932	0.7969	0.8316
FedAvgM	0.5970	0.6475	0.5970	0.5942
FedOpt†	0.8722	0.9121	0.8722	0.8875
FedYogi	0.7309	0.7586	0.7309	0.6719
FedAdam	0.7202	0.7222	0.7202	0.6780
FedAdagrad	**0.8405**	**0.8841**	**0.8405**	**0.8544**

† FedOpt achieves the highest absolute accuracy and F1-score; however, FedAdagrad is highlighted due to its more balanced and stable performance under Non-IID multiclass settings.

In the CICIoMT2024 experiments, FedAdagrad significantly outperformed other server-side optimizer (in Figure 27), under a similar degree of imbalance represented by $Dir_{\alpha=0.5}(\cdot)$.

Therefore, we will investigate results about FedAdagrad in detail as an effective federated learning method in multiclass classification tasks.

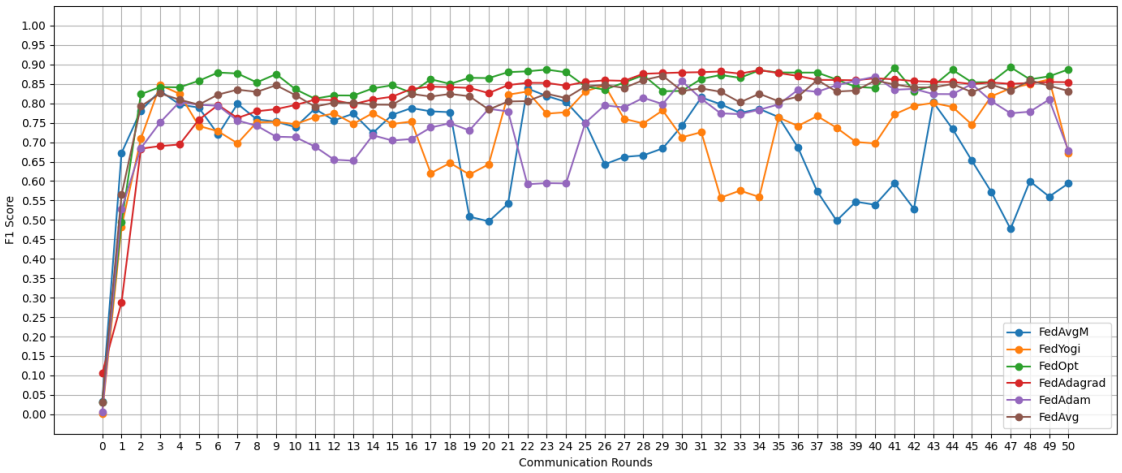


Figure 36. IoMT-TrafficData: F1-score Metrics for FL Methods in Multiclass Classification ($Dir_{\alpha=0.5}(\cdot)$), Non-IID).

The Federated Adaptive Gradient Algorithm (FedAdagrad) demonstrates stable performance across various metrics, including accuracy, precision, recall, and F1-score throughout different communication rounds, as illustrated in Figure 37. These consistent results suggest that FedAdagrad maintains its effectiveness in the context of multiclass classification, particularly in scenarios with a degree of class imbalance, such as $Dir_{\alpha=0.5}(\cdot)$. The ability to sustain performance across rounds is essential for ensuring reliable predictions in federated learning environments.

Federated Adaptive Gradient Algorithm (FedAdagrad) provides stable performance in accuracy, precision, recall and F1-score across communications rounds (shown in Figure 37). FedAdagrad experiments with varying degrees of imbalance impact model performance, as shown in Figure 38. We applied $Dir_{a=0.3}(\cdot)$, $Dir_{a=0.3}(\cdot)$ and IID distributions to identify significant decreases in performance across these scenarios.

Evaluating model performance through a class-by-class approach helps to identify discrepancies in predictions. Attacks that are less represented in the dataset are particularly difficult to detect, as clients have smaller distributions of these classes, leading to less effective model training on them. The distributions of attack types are shown in Figure 7. Figure 39 illustrates this evidently.

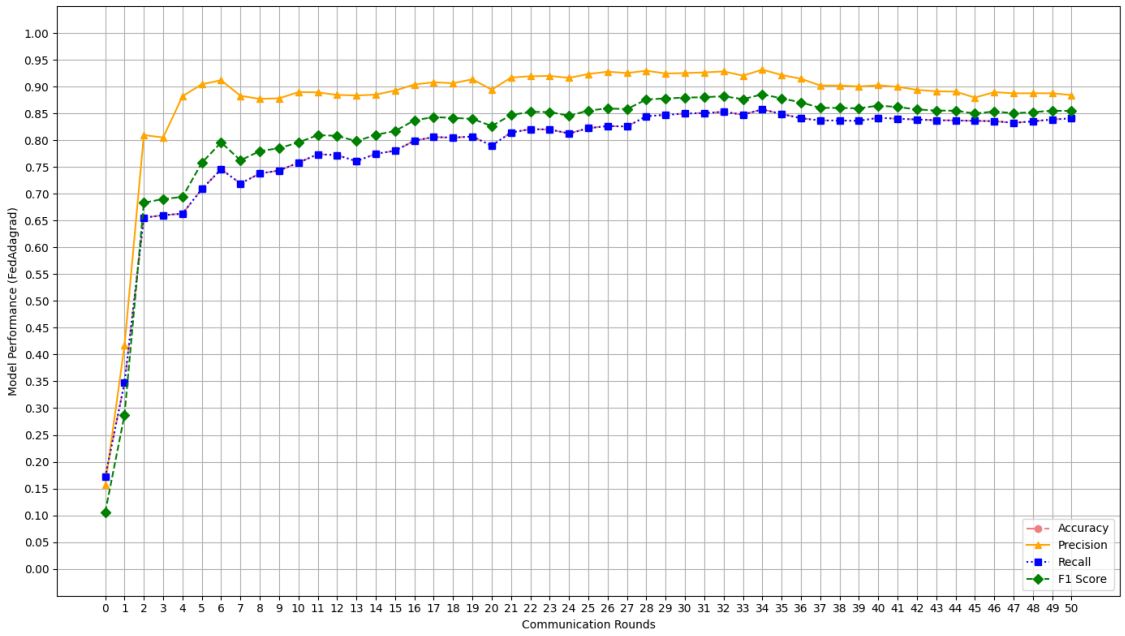


Figure 37. IoMT-TrafficData: Metrics of FedAdagrad Method in Multiclass Classification ($Dir_{\alpha=0.5}(\cdot)$), Non-IID).

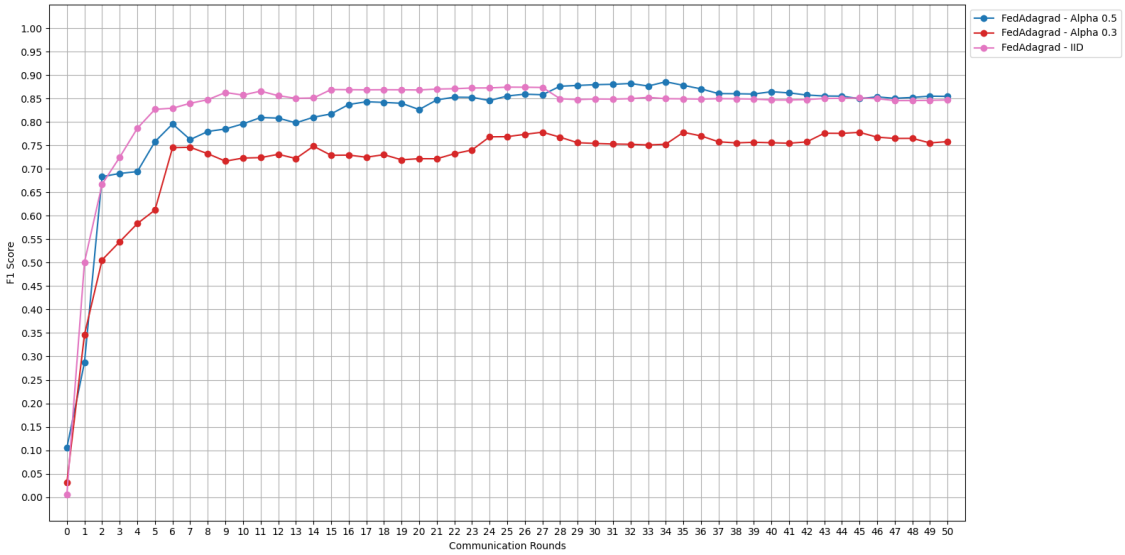


Figure 38. IoMT-TrafficData: F1-score of FedAdagrad Method in Multiclass Classification ($Dir_{\alpha=0.3}(\cdot)$, $Dir_{\alpha=0.5}(\cdot)$ and IID).

Federated evaluation reveals poor performance in local models ability to infer correct predictions, as shown in Figure 40, likely due to class imbalance (see Figure 5).

Inversely, more balanced datasets present less volatile results all around. The homogeneous local datasets influence results, such that FedAvg retains stability while adaptive optimizers are prone to decrease in performance in IID settings (denoted in Figure 41).

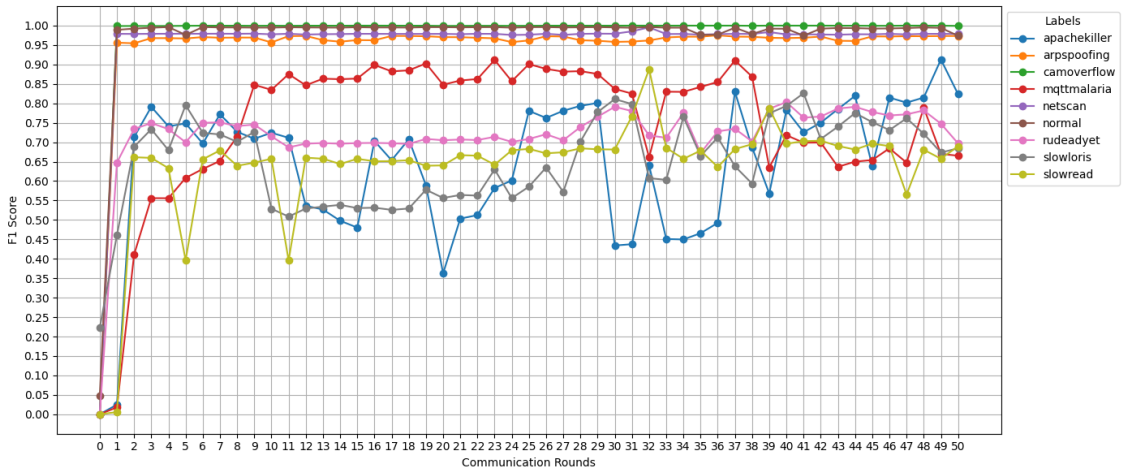


Figure 39. IoMT-TrafficData: F1-score in class predictions using FedAvg in Multiclass Classification (IID).

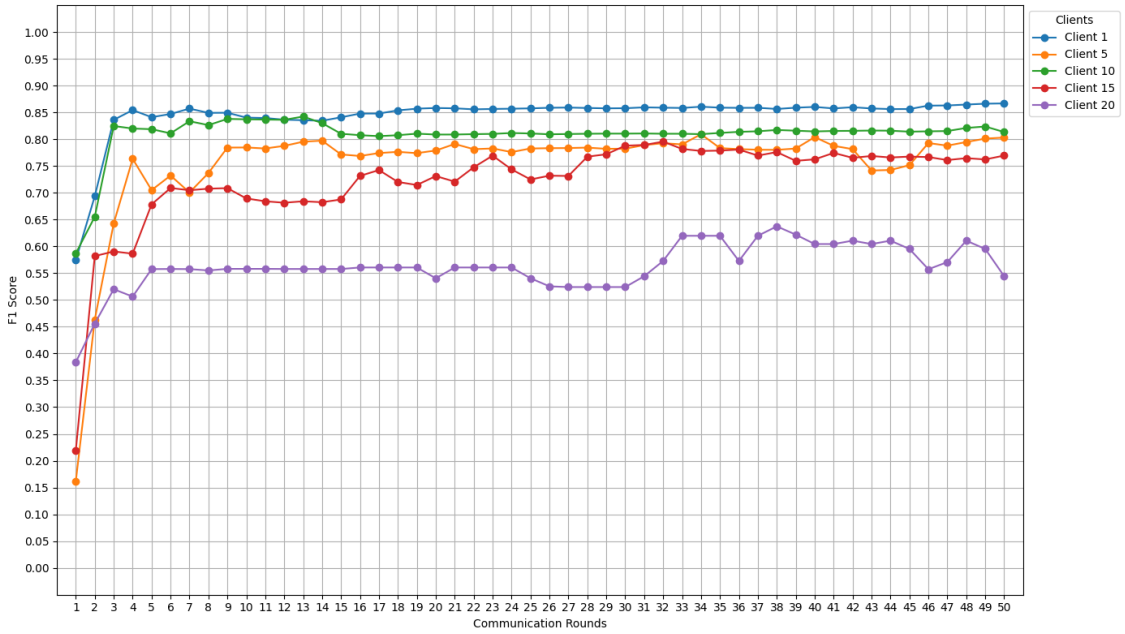


Figure 40. IoMT-TrafficData: Clients using FedAdagrad Method in Multiclass Classification ($Dir_{\alpha=0.5}(\cdot)$), Non-IID).

Evidently, experiments related to homogeneous local datasets provide consistent federated evaluation results across clients, shown in Figure 42, due to approximately similar local datasets. Regardless, intrusion detection capabilities cannot rely on the premise that network traffic is similar across different network architecture.

Regardless the above mentioned cohesive federated evaluation results in IID scenario, FedAvg Algorithm exhibits limited ability to infer some classes, as show in Figure 43.

In contrast, FedAvg excels in federated evaluation, client converging upon nearly identical

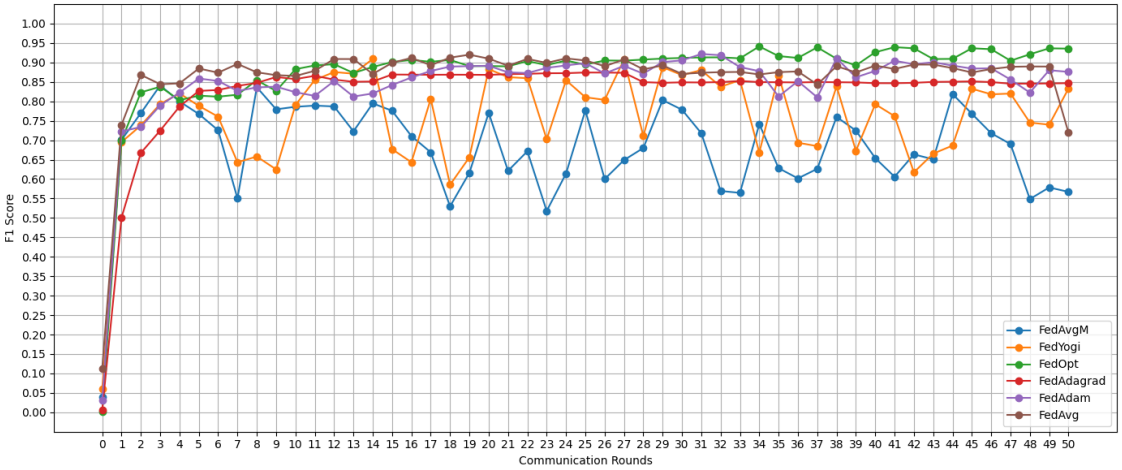


Figure 41. IoMT-TrafficData: F1-score for FL Methods in Multiclass Classification (IID).

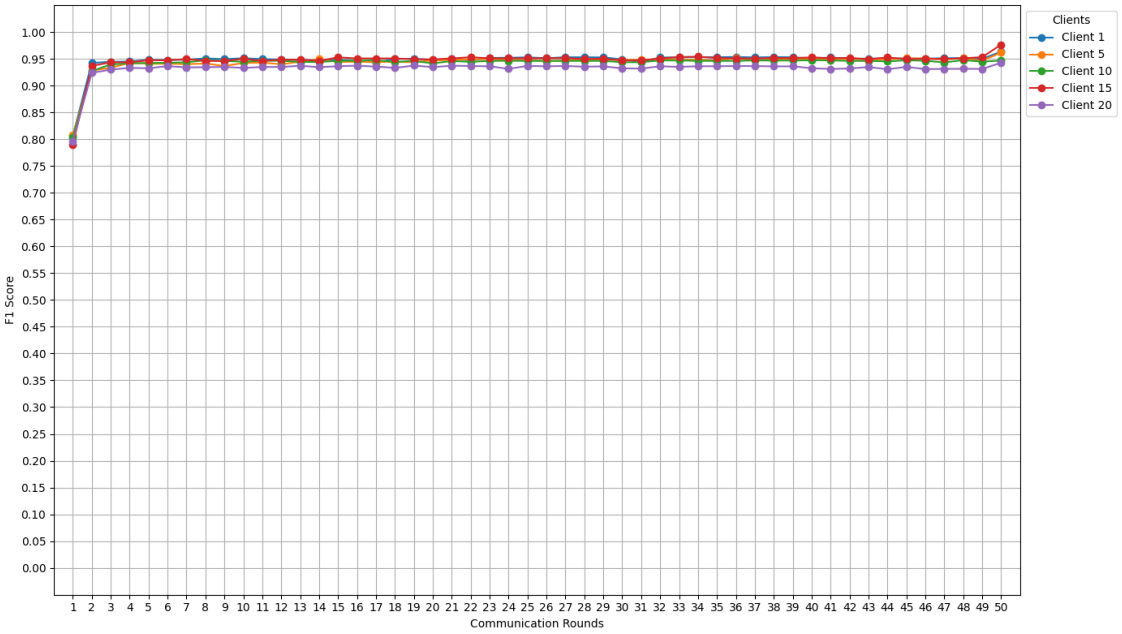


Figure 42. IoMT-TrafficData: Clients using FedAvg Method in Multiclass Classification (IID).

results in F1-score (between 0.95 and 0.96). Our work refrains emphasize experiments relating to IID situations. Specifically, network data relayed through Intrusion Detection Systems will significantly vary across different network infrastructures.

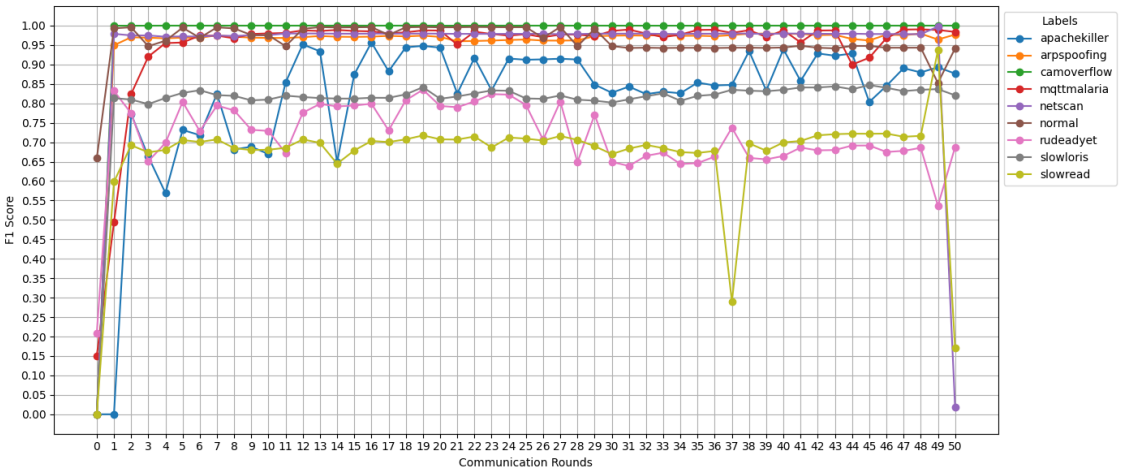


Figure 43. IoMT-TrafficData: F1-score in class predictions using FedAvg Method in Multiclass Classification (IID).

7 Discussion

This chapter discusses the experimental findings presented in Chapter 6 and interprets them in direct relation to the research questions defined in the Introduction. The discussion focuses on Federated Learning–based Intrusion Detection System (FL-IDS) capabilities for healthcare Internet of Medical Things (IoMT) networks, with particular emphasis on the role of server-side optimization methods under heterogeneous data distributions. The findings are examined using both centralized evaluation and federated evaluation, allowing for a comprehensive understanding of global model behavior as well as client-level disparities. In addition, this chapter highlights methodological limitations that emerged from the experimental setup and aligns the observations with existing federated learning literature.

The following sections address the research question based on empirical evidence from the results in Chapter 6, while also discussing the practical implications for federated intrusion detection systems. The research question as defined in Chapter 1 as follows:

RQ1: Which optimization techniques can improve the performance of federated learning in decentralized Internet of Medical Things (IoMT) networks for cyber-attack detection?

The experimental results clearly demonstrate that adaptive server-side optimization techniques outperform the traditional Federated Averaging algorithm in Non-IID settings. In particular, the Federated Optimization algorithm (FedOpt) consistently surpassed FedAvg in binary intrusion detection tasks across both evaluated datasets. Under $Dir_{\alpha=0.5}(\cdot)$ configurations, FedOpt achieved F1-scores of 0.9967 on IoMT-TrafficData and 0.9727 on CICIoMT2024, while also maintaining stable convergence behavior across communication rounds.

These findings empirically validate the theoretical claims introduced by Reddi et al. [58], who showed that adaptive server-side optimization mitigates the adverse effects of gradient heterogeneity caused by Non-IID client data. In contrast, FedAvg exhibited

increased volatility and reduced robustness under the same conditions, particularly in later communication rounds. This behavior reinforces known limitations of FedAvg when the assumption of homogeneous data distributions does not hold, which is typically the case in real-world IoMT deployments.

In multiclass intrusion detection scenarios, adaptive behavior remained critical. While FedAvg performed adequately in early rounds, it failed to maintain competitive performance as training progressed. The Federated Adaptive Gradient algorithm (FedAdagrad) consistently outperformed FedAvg, particularly on the CICIoMT2024 dataset, indicating that adaptive learning rates at the server level are beneficial when dealing with complex class boundaries and uneven data distributions. These observations further support the conclusion that FedAvg is insufficient for realistic Non-IID healthcare traffic.

When comparing federated optimization techniques across evaluation metrics—accuracy, precision, recall, and F1-score — it becomes evident that optimizer effectiveness is strongly task-dependent. In binary classification, FedOpt achieved the highest scores across all reported metrics. For instance, on IoMT-TrafficData, FedOpt reached an accuracy of 0.9967, precision of 0.9968, recall of 0.9967, and an F1-score of 0.9968 under centralized evaluation. Comparable trends were observed on CICIoMT2024, indicating that FedOpt effectively aggregates heterogeneous client updates without sacrificing detection reliability.

In contrast, multiclass classification tasks benefited more from gradient-adaptive methods. Here, FedAdagrad consistently delivered superior performance, achieving an F1-score of approximately 0.7116 on CICIoMT2024 and 0.8544 on IoMT-TrafficData. These results suggest that adaptive gradient scaling enables the global model to better adjust to uneven class contributions and overlapping attack patterns. Importantly, these gains were not limited to a single metric but were reflected consistently across accuracy, precision, recall, and F1-score, highlighting FedAdagrad Algorithm's balanced optimization behavior.

Taken together, these findings indicate that no single optimization technique dominates across all intrusion detection tasks. Instead, the choice of server-side optimizer should be guided by the specific detection objective, a conclusion that aligns with prior work on adaptive federated optimization [50, 51].

The experimental results provide strong evidence that adaptive federated optimization

techniques significantly enhance federated intrusion detection capabilities in decentralized IoMT environments. All experiments employed a Gated Recurrent Unit (GRU)–based deep learning model, which proved effective in classifying network traffic in cross-silo horizontal federated learning setting. Although each communication round instructed full client participation, federated evaluation revealed that some clients failed to complete local training due to resource constraints or extended execution times, leading to timeouts. Such behavior reflects realistic deployment conditions in healthcare IoMT infrastructures, where devices differ in computational capacity and availability. Despite these challenges, adaptive optimizers—most notably FedOpt in binary tasks and FedAdagrad in multiclass tasks—maintained stable global model performance, demonstrating robustness to partial participation and uneven client contributions. The distinction between centralized evaluation and federated evaluation was essential for capturing these dynamics. Centralized evaluation provided an unbiased assessment of global model performance, while federated evaluation exposed client-level disparities and training instability that would otherwise remain hidden. This combined evaluation strategy enabled a more realistic assessment of federated intrusion detection performance than relying solely on final-round centralized metrics.

A key limitation identified through the Chapter 6 is the reliance on raw datasets without explicit class rebalancing. Both IoMT-TrafficData and CICIoMT2024 exhibit inherent class imbalance, which directly influenced the Dirichlet-based data partitioning strategy defined in Equation (5.5). As a result, the imposed Non-IID distributions were shaped not only by the Dirichlet concentration parameter but also by the original skew in the datasets. This effect manifested in federated clients predominantly possessing samples from already dominant classes, while minority attack types were sparsely distributed or entirely absent in some local datasets. In multiclass experiments, this imbalance contributed to reduced class-wise F1-scores and confusion between attack types such as (DoS) and (DDoS). These discrepancies indicate that data imbalance, rather than optimizer choice alone, can significantly constrain the achievable performance of FL-IDS models.

The experiments further confirmed that Federated Averaging performs well under IID conditions, achieving stable convergence and competitive results across all evaluation metrics. This observation is consistent with the original findings by McMahan et al. [40]. However, Chapter 6 clearly demonstrates that IID data distributions are highly unlikely in

real-world healthcare IoMT networks, where devices differ substantially in function, traffic characteristics, and exposure to cyber threats.

Consequently, while FedAvg remains a useful baseline and reference point, it is not well suited for practical federated intrusion detection deployments. Adaptive server-side optimization methods provide a more reliable alternative for handling the heterogeneity inherent in IoMT environments.

In summary, this thesis demonstrates that adaptive server-side optimization methods substantially improve Federated Learning–based intrusion detection systems in heterogeneous IoMT environments. The experimental results answer the research question posed in the Introduction, showing that FedOpt is best suited for binary intrusion detection, FedAdagrad excels in multiclass classification, and that centralized evaluation alone is insufficient to fully characterize federated learning behavior. At the same time, the study exposes important limitations related to raw dataset imbalance, heterogeneity, and partial participation, which should be addressed in future research to further strengthen federated intrusion detection capabilities.

8 Limitations and Future Work

In our exploration of Federated Learning with Non-IID data, we presented empirical results on various FL optimization methods. To handle data heterogeneity, algorithms must adapt to unequal data distributions.

Our collaborative distributed approach is resilient against different types of intrusion attacks, demonstrated in both homogeneous and heterogeneous environments, without disclosing raw data to central server or other local clients. Privacy-protective measures should also protect model updates from adversarial actors to reduce the risk of model poisoning attacks. While the primary purpose of intrusion detection is to identify attacks, the system itself can also become a target. This highlights the need for cryptographic solutions to ensure the security of model update transfers. Furthermore, Intrusion Detection System may encounter complex and sophisticated attacks from Advanced Persistent Threats (APTs). As network intrusions evolve, we may face new types of cyberattacks that are not represented in existing datasets. In such cases, ML-based or FL-based IDS should be adaptable to continuously improve their detection capabilities. These considerations are outside the scope of this thesis and are recognized as limitations.

8.1 Limitations

Privacy and security may significantly restrict the development of Federated Learning-based Intrusion Detection Systems. While existing solutions aim to enhance privacy protection and system security, they often come with their own drawbacks. Existing surveys suggest secure multiparty computation (SMC) [92] and differential privacy [93] to provide privacy protection, but there's a still a gap in understanding the levels of encryption in secure multiparty computation and the amount of noise added in differential privacy [93]. While higher encryption levels and more noise do improve privacy, they can also negatively impact the detection performance of models [94]. Additionally, verifying model updates and employing techniques like homomorphic encryption [95] increase the computational

costs for both servers and clients. These extra measures may also prolong communication rounds, slowing down the convergence speed of global models [96]. Understanding this trade-off, privacy-protecting techniques in FL — differential privacy [97], secure multiparty computation [98], and homomorphic encryption [99] — will help ensure safe transfers of model updates between the central server and the distributed clients. Using these privacy-preserving methods will be greatly improve adaptation of our presented work.

Continuous data streams impose a crucial need for improvement to any ML-based Intrusion Detection System, an adaptability to learn from previously unknown intrusion types. One approach to accomplish that is with **incremental learning** [100]. A clear limitation to the thesis surfaces once the trained FL-IDS is implemented into a practical network infrastructure, evidently facing continuous incoming data flows. When trained models encounter new data distributions, they tend to forget previous information, leading to a sharp decline in detection performance. This issue, known as catastrophic forgetting and concept drift [101] [102], highlights a critical gap in current approaches. To adapt to these changing data distributions and retain existing knowledge, FL-IDS should develop incremental learning capabilities. Jin et al. [103] proposed a Federated Learning-based Incremental Intrusion Detection System that addresses catastrophic forgetting. Their system uses a loss function that combines recall with a regularization approach, as well as incorporates relay clients and sample reconstruction techniques to enhance its adaptability. Their proposed work suggests potential future developments aimed at continuously enhancing the capabilities of FL-IDS, including our presented solution.

Advanced Persistent Threats (APTs) can be challenging to catch with any machine learning-based Intrusion Detection Systems. Their complex network attacks are often executed by organized groups that target large organizations and facilities to steal sensitive data. APTs are persistent, with attackers continuously adapting their methods to achieve their objectives [104]. Commonly, the military, political, financial, and technological networks are favored targets, posing serious risks [105]. Currently, FL-IDS depend on horizontal federated learning, which makes it challenging to identify APTs affecting multiple devices. In such distributed communication systems, intrusion detection mechanisms are decentralized, complicating the detection of related malicious activities and limiting APT detection efforts [106]. Since APTs can involve multiple devices, the limitations of most

FL-IDS based on Horizontal Federated Learning become evident. For instance, Hu et al. [107] proposed a detection scheme that assumes an APT attack involves only one device and uses horizontal federated learning for model training. This approach overlooks scenarios where APTs impact multiple devices, missing the opportunity to leverage data generated by the same attack across different locations. Continuing our proposed work in this area would greatly merit from integrating incremental learning to improve the FL-IDS capabilities in the long-term.

8.2 Future Work

Using federated knowledge distillation can help address the issue of limited communication resources [108]. In edge networks, for example, there are constraints on how much data can be sent, so improving the efficiency of federated learning transmission is an important area of research. Knowledge distillation methods send logits instead of full model parameters, making it easier to transmit smaller amounts of data. Many studies have explored federated learning with knowledge distillation, some of which integrate semi-supervised and unsupervised learning techniques. This approach allows for better use of large amounts of unlabeled data, which is important for real-world applications [109] [110]. For instance, Zhao et al. [111] reduced the communication overhead needed to send parameters by employing federated knowledge distillation, while also leveraging unlabeled data through a method called soft-hard label voting.

Utilizing eXplainable Artificial Intelligence (XAI) with Federated Learning-based Intrusion Detection Systems (FL-IDS) may increase the credibility of detection rates. Intrusion Detection capabilities may be solved with deep learning, but in turn obfuscate inner mechanism and withhold any actionable analysis detection behavior. Consequently, this can lead to lack of confidence in decision made by administrators and security experts [112]. The main purpose of XAI is to explain how black-box models function, showing how the inner workings of these machine learning models affect their predictions. Some studies have investigated how different features of the samples impact detection results [113] [114]. By using XAI, we can not only boost the credibility of these detection outcomes but also help researchers better understand why false positives occur in FL-IDS. For example, Amiri-Zarandi and their team worked [115] on a model to spot insider threats using

federated learning. They discovered that the Shapley Additive Explanation algorithm could show which features were causing more errors in the model's predictions. This direction merits more thorough implementation in future studies, as it may substantially improve the comprehension of the FL-IDS models, and improves confidence in its decisions.

Adversarial attacks are a concern in Federated Learning, aiming to manipulate model updates in order to skew predictions or extract sensitive information. These threats can be categorized based on their sources [116]. Causative attacks involve a malicious participant—often a client device—submitting poisoned data or altering model updates during training. This leads to skewed predictions [117] [118] [119]. Evasion attacks, on the other hand, change the input test data during inference to misguide the model's predictions [120] [121]. Both of these attack types pose significant risks to the accuracy and privacy of Federated Learning models. In addition, communication channel attacks represent another threat. These attacks happen when someone intercepts sensitive data or interferes with the communication between clients and the server [116]. A common example is the Man-in-the-Middle (MitM) [122] attack, where an attacker taps into the communication channel and manipulates the data being exchanged during training. This severely compromises the privacy and security of the model. Moving forward, future research can be explored to investigate the impact of these adversarial attacks in FL-based Intrusion Detection Systems in healthcare environments, and identify corresponding measures to address these concerns. In healthcare, accurate intrusion detection is vital to ensure continuous and untampered system functions for heart-rate monitors, pacemakers, remote medication systems, and prevent irrevocable consequences.

9 Conclusion

This thesis investigated the effectiveness of Federated Learning–based intrusion detection systems (FL-IDS) for healthcare Internet of Medical Things (IoMT) networks, with a particular focus on the role of server-side optimization techniques under heterogeneous data distributions. Motivated by the limitations of centralized intrusion detection and the privacy constraints inherent in healthcare environments, the work explored whether adaptive federated optimization methods can improve detection performance when data is decentralized, imbalanced, and non-identically distributed.

The study was grounded in the observation that traditional signature-based intrusion detection systems struggle to detect novel or evolving cyber threats, especially in dynamic IoMT infrastructures. Building on prior research in anomaly-based detection and federated learning, this thesis adopted a cross-silo horizontal federated learning framework to enable collaborative model training without sharing raw network traffic data. A Gated Recurrent Unit (GRU)–based deep learning model was selected due to its ability to capture temporal dependencies in sequential network traffic, while remaining computationally feasible for distributed training across heterogeneous clients.

The experimental methodology was carefully designed to reflect realistic IoMT deployment conditions. Two publicly available datasets, IoMT-TrafficData and CICIoMT2024, were used to evaluate both binary and multiclass intrusion detection tasks. Client datasets were partitioned using Dirichlet distributions to systematically control the degree of data heterogeneity, with a primary focus on Non-IID scenarios defined by $Dir_{\alpha=0.5}(\cdot)$. Local training was performed at each client using fixed batch sizes and learning rates, while global aggregation was conducted at the server using different optimization strategies. Each communication round instructed full client participation; however, the system design allowed for partial participation to capture realistic client failures due to resource constraints or execution timeouts.

A key methodological contribution of this work is the explicit separation between centralized evaluation and federated evaluation. Centralized evaluation, conducted on server-side testing data, provided an unbiased measurement of global model performance across accuracy, precision, recall, and F1-score. Federated evaluation, on the other hand, revealed client-level disparities, including uneven convergence behavior and occasional client dropout. This dual evaluation approach enabled a more comprehensive assessment of federated intrusion detection performance than relying solely on final-round global metrics.

The experimental results directly addressed the research questions posed in the Introduction. First, adaptive server-side optimization techniques were shown to outperform the commonly used Federated Averaging algorithm under Non-IID conditions. In binary intrusion detection tasks, the Federated Optimization algorithm (FedOpt) consistently achieved the highest performance across all evaluation metrics and demonstrated stable convergence across communication rounds. These findings empirically support prior theoretical work by Reddi et al [58], confirming that adaptive optimization mitigates the negative effects of data heterogeneity in federated learning.

Second, optimizer effectiveness was found to be task-dependent. While FedOpt proved most effective for binary intrusion detection, multiclass classification benefited more from the Federated Adaptive Gradient algorithm (FedAdagrad). Across both datasets, FedAdagrad achieved the most balanced performance in terms of accuracy, precision, recall, and F1-score, particularly in scenarios with complex class boundaries and uneven class representation. This observation aligns with existing literature showing that gradient-adaptive methods better handle heterogeneous and imbalanced learning signals.

Third, the results demonstrated that adaptive federated optimization techniques significantly enhance federated intrusion detection capabilities in decentralized IoMT networks. Despite partial client participation and varying local data distributions, adaptive optimizers maintained stable global model performance, highlighting their robustness in practical deployment scenarios. In contrast, Federated Averaging performed well only under IID assumptions, which were noted to be unrealistic for healthcare IoMT environments where devices differ in function, traffic characteristics, and exposure to cyberattacks.

At the same time, this thesis identified important limitations that constrain the achievable

performance of FL-IDS models. A major limitation arises from the use of raw datasets without explicit class re-balancing. Both IoMT-TrafficData and CICIoMT2024 exhibit inherent class imbalance, which influenced the Dirichlet-based partitioning process and resulted in federated clients predominantly possessing samples from already dominant classes. In multiclass experiments, this imbalance contributed to reduced class-wise performance and confusion between structurally similar attack types, such as Denial of Service Attacks and Distributed Denial of Service Attacks. These findings indicate that data imbalance, rather than optimization strategy alone, plays a critical role in shaping federated learning outcomes.

In summary, this thesis demonstrates that adaptive server-side optimization is essential for reliable Federated Learning–based intrusion detection in heterogeneous IoMT environments. By combining privacy-preserving federated learning, GRU-based deep learning models, systematic data partitioning, and complementary evaluation strategies, this work provides empirical evidence that adaptive optimizers such as FedOpt and FedAdagrad substantially improve detection performance under realistic conditions. At the same time, the study highlights the need for future research on data balancing, client heterogeneity mitigation, and resource-aware federated training to further strengthen federated intrusion detection systems for healthcare IoMT networks.

References

[1] Kefeng Wei et al. „Health monitoring based on internet of medical things: architecture, enabling technologies, and applications“. In: *IEEE Access* 8 (2020), pp. 27468–27478.

[2] Qinqin Wu, Panyu Tang, and Maolin Yang. „Data processing platform design and algorithm research of wearable sports physiological parameters detection based on medical internet of things“. In: *Measurement* 165 (2020), p. 108172.

[3] Naveen Kumar, Rajesh Kumar Kaushal, and Surya Narayan Panda. „IoT based smart and portable system for remote patient monitoring and drug delivery“. In: *Journal of Physics: Conference Series*. Vol. 1950. 1. IOP Publishing. 2021, p. 012017.

[4] Hoe Tung Yew et al. „Iot based real-time remote patient monitoring system“. In: *2020 16th IEEE international colloquium on signal processing & its applications (CSPA)*. IEEE. 2020, pp. 176–179.

[5] Ruby Dwivedi, Divya Mehrotra, and Shaleen Chandra. „Potential of Internet of Medical Things (IoMT) applications in building a smart healthcare system: A systematic review“. In: *Journal of oral biology and craniofacial research* 12.2 (2022), pp. 302–318.

[6] Serigne MK Mbengue et al. „Internet of Medical Things: Remote diagnosis and monitoring application for diabetics“. In: *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE. 2020, pp. 583–588.

[7] Surabhi Joshi and Sunil Joshi. „A sensor based secured health monitoring and alert technique using iomt“. In: *2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT)*. IEEE. 2019, pp. 152–156.

[8] Laavanya Rachakonda, Saraju P Mohanty, and Elias Kougianos. „cStick: a calm stick for fall prediction, detection and control in the IoMT framework“. In: *IFIP International Internet of Things Conference*. Springer. 2021, pp. 129–145.

[9] Akash Gupta et al. „IoT based fall detection monitoring and alarm system for elderly“. In: *2020 IEEE 7th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. IEEE. 2020, pp. 1–5.

[10] F John Dian, Reza Vahidnia, and Alireza Rahmati. „Wearables and the Internet of Things (IoT), applications, opportunities, and challenges: A Survey“. In: *IEEE access* 8 (2020), pp. 69200–69211.

[11] Mohamed Adel Serhani et al. „ECG monitoring systems: Review, architecture, processes, and key challenges“. In: *Sensors* 20.6 (2020), p. 1796.

[12] Katya Arquilla, Andrea K Webb, and Allison P Anderson. „Textile electrocardiogram (ECG) electrodes for wearable health monitoring“. In: *Sensors* 20.4 (2020), p. 1013.

[13] Ranganathan Chandrasekaran, Vipanchi Katthula, and Evangelos Moustakas. „Patterns of use and key predictors for the use of wearable health care devices by US adults: insights from a national survey“. In: *Journal of medical Internet research* 22.10 (2020), e22443.

[14] Antonio Alarcón-Paredes et al. „An IoT-Based Non-Invasive Glucose Level Monitoring System Using Raspberry Pi“. In: *Applied Sciences* 9.15 (2019). ISSN: 2076-3417. DOI: 10.3390/app9153046. URL: https://www.mdpi.com/2076-3417/9/15/3046.

[15] Michael Fang et al. „Trends in Diabetes Treatment and Control in U.S. Adults, 1999–2018“. In: *New England Journal of Medicine* 384.23 (2021), pp. 2219–2228. DOI: 10.1056/NEJMsa2032271. eprint: https://www.nejm.org/doi/pdf/10.1056/NEJMsa2032271. URL: https://www.nejm.org/doi/full/10.1056/NEJMsa2032271.

[16] Francesco Lamonaca et al. „A new measurement system to boost the IoMT for the blood pressure monitoring“. In: *2019 IEEE International Symposium on Measurements & Networking (M&N)*. IEEE. 2019, pp. 1–6.

[17] Richa Sharma et al. „Intelligent automated drug administration and therapy: future of healthcare“. In: *Drug Delivery and Translational Research* 11.5 (2021), pp. 1878–1902.

[18] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. „A Survey of Network Anomaly Detection Techniques“. In: *Journal of Network and Computer Applications* 60 (2016), pp. 19–31. DOI: 10.1016/j.jnca.2015.11.016.

[19] Gilberto Jr. Fernandes et al. „A comprehensive survey on network anomaly detection“. In: *Telecommunication Systems* 70.3 (2019), pp. 447–489. DOI: 10.1007/s11235-018-0475-8.

[20] Richa Singh, Nidhi Srivastava, and Ashwani Kumar. „Machine Learning Techniques for Anomaly Detection in Network Traffic“. In: *2021 Sixth International Conference on Image Information Processing (ICIIP)*. 2021. DOI: 10.1109/ICIIP53038.2021.9702647.

[21] Shijoe Jose et al. „A Survey on Anomaly Based Host Intrusion Detection System“. In: *Journal of Physics: Conference Series* 1000 (Apr. 2018), p. 012049. DOI: 10.1088/1742-6596/1000/1/012049.

[22] Zhen Yang et al. „A systematic literature review of methods and datasets for anomaly-based network intrusion detection“. In: *Computers & Security* 116 (Mar. 2022), p. 102675. DOI: 10.1016/j.cose.2022.102675.

[23] Zeeshan Ahmad et al. „Network intrusion detection system: A systematic study of machine learning and deep learning approaches“. In: *Transactions on Emerging Telecommunications Technologies* 32 (Jan. 2021). DOI: 10.1002/ett.4150.

[24] Kai Chen and Qiang Yang. *Privacy-preserving Computing: for Big Data Analytics and AI*. Cambridge University Press, Oct. 2023. ISBN: 9781009299510. DOI: 10.1017/9781009299534.

[25] Peter Kairouz et al. *Advances and Open Problems in Federated Learning*. Dec. 2019. DOI: 10.48550/arXiv.1912.04977.

[26] Jakub Konečný et al. „Federated Learning: Strategies for Improving Communication Efficiency“. In: *CoRR* abs/1610.05492 (2016). arXiv: 1610.05492. URL: http://arxiv.org/abs/1610.05492.

[27] Qiang Yang et al. „Federated Machine Learning: Concept and Applications“. In: *ACM Transactions on Intelligent Systems and Technology* 10 (Jan. 2019), pp. 1–19. DOI: 10.1145/3298981.

[28] Zili Lu et al. „Federated Learning With Non-IID Data: A Survey“. In: *IEEE Internet of Things Journal* 11.11 (2024), pp. 19188–19209. DOI: 10.1109/JIOT.2024.3376548.

[29] Sajjad Dadkhah et al. „CICIoMT2024: A benchmark dataset for multi-protocol security assessment in IoMT“. In: *Internet of Things* 28 (Aug. 2024), p. 101351. DOI: 10.1016/j.iot.2024.101351.

[30] José Areia et al. „IoMT-TrafficData: Dataset and Tools for Benchmarking Intrusion Detection in Internet of Medical Things“. In: *IEEE Access* PP (Jan. 2024), pp. 1–1. DOI: 10.1109/ACCESS.2024.3437214.

[31] Gartner. *Gartner Says 8.4 Billion Connected 'Things' Will Be in Use in 2017, Up 31 Percent From 2016*. Online. Gartner Press Release, Stamford, CT, USA. Accessed: September 9, 2025. Feb. 2017. URL: http://www.gartner.com/newsroom/id/3598917 (visited on 09/09/2025).

[32] Telecompaper. *Global IoT Market to Reach USD 1.7 Tln in 2020*. Online. Accessed: Accessed: September 9, 2025. 2016. URL: https://www.telecompaper.com/news/global-iot-market-to-reach-usd-17-tln-in-2020-idc--1085269.

[33] Yasir Mehmood et al. „Internet-of-things-based smart cities: Recent advances and challenges“. In: *IEEE Communications Magazine* 55.9 (2017), pp. 16–24.

[34] Jerrin T John and SR Jino Ramson. „Energy-aware duty cycle scheduling for efficient data collection in wireless sensor networks“. In: *IJARCET Volume* 2 (2013).

[35] SR Jino Ramson and D Jackuline Moni. „A case study on different wireless networking technologies for remote health care“. In: *Intelligent Decision Technologies* 10.4 (2016), pp. 353–364.

[36] Konstantinos Lazaros et al. „Federated learning: Navigating the landscape of collaborative intelligence“. In: *Electronics* 13.23 (2024), p. 4744.

[37] H. Brendan McMahan et al. „Federated Learning of Deep Networks using Model Averaging“. In: *CoRR* abs/1602.05629 (2016). arXiv: 1602.05629. URL: http://arxiv.org/abs/1602.05629.

[38] Peter Kairouz et al. „Advances and open problems in federated learning“. In: *Foundations and trends® in machine learning* 14.1–2 (2021), pp. 1–210.

[39] Tian Li et al. „Federated learning: Challenges, methods, and future directions“. In: *IEEE signal processing magazine* 37.3 (2020), pp. 50–60.

[40] Brendan McMahan et al. „Communication-efficient learning of deep networks from decentralized data“. In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[41] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. „Measuring the effects of non-identical data distribution for federated visual classification“. In: *arXiv preprint arXiv:1909.06335* (2019).

[42] Sai Praneeth Karimireddy et al. „Scaffold: Stochastic controlled averaging for on-device federated learning“. In: *arXiv preprint arXiv:1910.06378* 2.6 (2019).

[43] Jingzhao Zhang et al. „Why ADAM beats SGD for attention models“. In: (2019).

[44] Martin Zinkevich et al. „Parallelized stochastic gradient descent“. In: *Advances in neural information processing systems* 23 (2010).

[45] Sebastian U Stich. „Local SGD converges fast and communicates little“. In: *arXiv preprint arXiv:1805.09767* (2018).

[46] Hao Yu, Sen Yang, and Shenghuo Zhu. „Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning“. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 5693–5700.

[47] Jianyu Wang and Gauri Joshi. „Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms“. In: *arXiv preprint arXiv:1808.07576* (2018).

[48] Sebastian U Stich and Sai Praneeth Karimireddy. „The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication“. In: *arXiv preprint arXiv:1909.05350* (2019).

[49] Debraj Basu et al. „Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations“. In: *Advances in Neural Information Processing Systems* 32 (2019).

[50] Tian Li et al. „Federated optimization in heterogeneous networks“. In: *Proceedings of Machine learning and systems* 2 (2020), pp. 429–450.

[51] Shiqiang Wang et al. „Adaptive federated learning in resource constrained edge computing systems“. In: *IEEE journal on selected areas in communications* 37.6 (2019), pp. 1205–1221.

[52] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. „First analysis of local GD on heterogeneous data“. In: *arXiv preprint arXiv:1909.04715* (2019).

[53] Xiang Li et al. „On the convergence of fedavg on non-iid data“. In: *arXiv preprint arXiv:1907.02189* (2019).

[54] Jakub Konečný, Brendan McMahan, and Daniel Ramage. „Federated Optimization: Distributed Optimization Beyond the Datacenter“. In: *CoRR* abs/1511.03575 (2015). arXiv: 1511.03575. URL: http://arxiv.org/abs/1511.03575.

[55] Keith Bonawitz et al. „Towards federated learning at scale: System design“. In: *Proceedings of machine learning and systems* 1 (2019), pp. 374–388.

[56] Jianyu Wang et al. „A field guide to federated optimization“. In: *arXiv preprint arXiv:2107.06917* (2021).

[57] Bouziane Brik, Adlen Ksentini, and Maha Bouaziz. „Federated learning for UAVs-enabled wireless networks: Use cases, challenges, and open problems“. In: *IEEE Access* 8 (2020), pp. 53841–53849.

[58] Sashank Reddi et al. „Adaptive federated optimization“. In: *arXiv preprint arXiv:2003.00295* (2020).

[59] Qinbin Li et al. „Federated learning on non-iid data silos: An experimental study“. In: *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE. 2022, pp. 965–978.

[60] Jianyu Wang et al. „Tackling the objective inconsistency problem in heterogeneous federated optimization“. In: *Advances in neural information processing systems* 33 (2020), pp. 7611–7623.

[61] Ziheng Cheng et al. „Momentum benefits non-iid federated learning simply and provably“. In: *arXiv preprint arXiv:2306.16504* (2023).

[62] Jianyu Wang et al. „Local adaptivity in federated learning: Convergence and consistency“. In: *arXiv preprint arXiv:2106.02305* (2021).

[63] Yan Sun et al. „Efficient federated learning via local adaptive amended optimizer with linear speedup“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.12 (2023), pp. 14453–14464.

[64] Ofer Dekel et al. „Optimal distributed online prediction using mini-batches“. In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 165–202.

[65] Jakub Konečnỳ et al. „Federated optimization: Distributed machine learning for on-device intelligence“. In: *arXiv preprint arXiv:1610.02527* (2016).

[66] Honglin Yuan and Tengyu Ma. „Federated accelerated stochastic gradient descent“. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 5332–5344.

[67] Jianyu Wang et al. „Slowmo: Improving communication-efficient distributed sgd with slow momentum“. In: *arXiv preprint arXiv:1910.00643* (2019).

[68] Tao Lin et al. „Ensemble distillation for robust model fusion in federated learning“. In: *Advances in neural information processing systems* 33 (2020), pp. 2351–2363.

[69] Chaoyang He, Murali Annavaram, and Salman Avestimehr. „Group knowledge transfer: Federated learning of large cnns at the edge“. In: *Advances in neural information processing systems* 33 (2020), pp. 14068–14080.

[70] Ryan McDonald, Keith Hall, and Gideon Mann. „Distributed training strategies for the structured perceptron“. In: *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*. 2010, pp. 456–464.

[71] Eduard Gorbunov, Filip Hanzely, and Peter Richtárik. „Local SGD: Unified theory and new efficient methods“. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 3556–3564.

[72] Rachel Ward, Xiaoxia Wu, and Leon Bottou. „Adagrad stepsizes: Sharp convergence over nonconvex landscapes“. In: *Journal of Machine Learning Research* 21.219 (2020), pp. 1–30.

[73] Manzil Zaheer et al. „Adaptive methods for nonconvex optimization“. In: *Advances in neural information processing systems* 31 (2018).

[74] Qianqian Tong, Guannan Liang, and Jinbo Bi. „Effective federated adaptive gradient methods with non-iid decentralized data“. In: *arXiv preprint arXiv:2009.06557* (2020).

[75] Riccardo Zaccone, Sai Praneeth Karimireddy, and Carlo Masone. „On the Limits of Momentum in Decentralized and Federated Optimization“. In: *arXiv preprint arXiv:2511.20168* (2025).

[76] Ilya Sutskever et al. „On the importance of initialization and momentum in deep learning“. In: *International conference on machine learning*. pmlr. 2013, pp. 1139–1147.

[77] John Duchi, Elad Hazan, and Yoram Singer. „Adaptive subgradient methods for online learning and stochastic optimization.“ In: *Journal of machine learning research* 12.7 (2011).

[78] Diederik P Kingma. „Adam: A method for stochastic optimization“. In: *arXiv preprint arXiv:1412.6980* (2014).

[79] Xiaoxia Wu, Simon S Du, and Rachel Ward. „Global convergence of adaptive gradient methods for an over-parameterized neural network“. In: *arXiv preprint arXiv:1902.07111* (2019).

[80] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. „On the convergence of adam and beyond“. In: *arXiv preprint arXiv:1904.09237* (2019).

[81] David Leroy et al. „Federated learning for keyword spotting“. In: *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2019, pp. 6341–6345.

[82] Hiep Nguyen et al. „Federated learning for non-iid data via client variance reduction and adaptive server update“. In: *arXiv preprint arXiv:2207.08391* (2022).

[83] Riccardo Lazzarini, Huaglory Tianfield, and Vassilis Charissis. „Federated learning for IoT intrusion detection“. In: *Ai* 4.3 (2023), pp. 509–530.

[84] Dinh C Nguyen et al. „Federated learning for internet of things: A comprehensive survey“. In: *IEEE communications surveys & tutorials* 23.3 (2021), pp. 1622–1658.

[85] Thomas W Edgar and David O Manz. *Research methods for cyber security*. Syngress, 2017.

[86] Luis Serrano. *Grokking machine learning*. Simon and Schuster, 2021.

[87] Margherita Grandini, Enrico Bagli, and Giorgio Visani. „Metrics for Multi-Class Classification: an Overview“. In: *arXiv preprint* (Aug. 2020). DOI: 10.48550/arXiv.2008.05756. arXiv: 2008.05756 [stat.ML]. URL: https://arxiv.org/abs/2008.05756.

[88] Abdallah Ghourabi and Adel Alkhalil. „A federated learning model for detecting cyberattacks in internet of medical things networks“. In: *IEEE Access* (2025).

[89] Mikhail Yurochkin et al. „Bayesian nonparametric federated learning of neural networks“. In: *International conference on machine learning*. PMLR. 2019, pp. 7252–7261.

[90] Hongyi Wang et al. „Federated learning with matched averaging“. In: *arXiv preprint arXiv:2002.06440* (2020).

[91] Boyuan Zhang and Mohammad Reza Shikh-Bahaei. „Federated Learning Enhancement Through Transfer and Continual Learning Integration: Analyzing Effects of Different Levels of Dirichlet Distribution“. In: *Wireless World Research and Trends Magazine* (2024), pp. 65–70.

[92] Ian Zhou et al. „Secure multi-party computation for machine learning: A survey“. In: *IEEE Access* 12 (2024), pp. 53881–53899.

[93] Cynthia Dwork. „Differential privacy: A survey of results“. In: *International conference on theory and applications of models of computation*. Springer. 2008, pp. 1–19.

[94] Viraaji Mothukuri et al. „A survey on security and privacy of federated learning“. In: *Future Generation Computer Systems* 115 (2021), pp. 619–640.

[95] Qipeng Xie et al. „Efficiency optimization techniques in privacy-preserving federated learning with homomorphic encryption: A brief survey“. In: *IEEE Internet of Things Journal* 11.14 (2024), pp. 24569–24580.

[96] Alberto Blanco-Justicia et al. „Achieving security and privacy in federated learning systems: Survey, research challenges and future directions“. In: *Engineering Applications of Artificial Intelligence* 106 (2021), p. 104468.

[97] Kang Wei et al. „Federated learning with differential privacy: Algorithms and performance analysis“. In: *IEEE transactions on information forensics and security* 15 (2020), pp. 3454–3469.

[98] Zhouyong Tan et al. „Secure and accurate personalized federated learning with similarity-based model aggregation“. In: *IEEE Transactions on Sustainable Computing* 10.1 (2024), pp. 132–145.

[99] Neveen Mohammad Hijazi et al. „Secure federated learning with fully homomorphic encryption for iot communications“. In: *IEEE Internet of Things Journal* 11.3 (2023), pp. 4289–4300.

[100] Mahendra Data and Masayoshi Aritsugi. „An incremental learning algorithm on imbalanced data for network intrusion detection systems“. In: *Proceedings of the 10th International Conference on Computer and Communications Management*. 2022, pp. 191–199.

[101] Jiangpeng He et al. „Incremental learning in online scenario“. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 13926–13935.

[102] Yong Luo et al. „An appraisal of incremental learning methods“. In: *Entropy* 22.11 (2020), p. 1190.

[103] Zhigang Jin et al. „FL-IIDS: A novel federated learning-based incremental intrusion detection system“. In: *Future Generation Computer Systems* 151 (2024), pp. 57–70.

[104] Adel Alshamrani et al. „A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities“. In: *IEEE Communications Surveys & Tutorials* 21.2 (2019), pp. 1851–1877.

[105] Branka Stojanović, Katharina Hofer-Schmitz, and Ulrike Kleb. „APT datasets and attack modeling for automated detection methods: A review“. In: *Computers & Security* 92 (2020), p. 101734.

[106] Andrew Vance. „Flow based analysis of Advanced Persistent Threats detecting targeted attacks in cloud computing“. In: *2014 first international scientific-practical conference problems of infocommunications science and technology*. IEEE. 2014, pp. 173–176.

[107] Yilun Hu et al. „Privacy-preserving few-shot traffic detection against advanced persistent threats via federated meta learning“. In: *IEEE Transactions on Network Science and Engineering* 11.3 (2023), pp. 2549–2560.

[108] Hao Zhang et al. „Survey of federated learning in intrusion detection“. In: *Journal of Parallel and Distributed Computing* 195 (2025), p. 104976.

[109] Xiaochen Zhou, Yuchuan Tian, and Xudong Wang. „Source-target unified knowledge distillation for memory-efficient federated domain adaptation on edge devices“. In: (2022).

[110] Sohei Itahara et al. „Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data“. In: *IEEE Transactions on Mobile Computing* 22.1 (2021), pp. 191–205.

[111] Ruijie Zhao et al. „Semisupervised federated-learning-based intrusion detection method for internet of things“. In: *IEEE Internet of Things Journal* 10.10 (2022), pp. 8645–8657.

[112] Tahmina Zebin, Shahadate Rezvy, and Yuan Luo. „An explainable AI-based intrusion detection system for DNS over HTTPS (DoH) attacks“. In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 2339–2349.

[113] Izhar Ahmed Khan et al. „XSRU-IoMT: Explainable simple recurrent units for threat detection in Internet of Medical Things networks“. In: *Future generation computer systems* 127 (2022), pp. 181–193.

[114] Tim Miller. „Explanation in artificial intelligence: Insights from the social sciences“. In: *Artificial intelligence* 267 (2019), pp. 1–38.

[115] Mohammad Amiri-Zarandi, Hadis Karimipour, and Rozita A Dara. „A federated and explainable approach for insider threat detection in IoT“. In: *Internet of Things* 24 (2023), p. 100965.

[116] Kummari Naveen Kumar, Chalavadi Krishna Mohan, and Linga Reddy Cenkeramaddi. „The impact of adversarial attacks on federated learning: A survey“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.5 (2023), pp. 2672–2691.

[117] Minghong Fang et al. „Local model poisoning attacks to {Byzantine-Robust} federated learning“. In: *29th USENIX security symposium (USENIX Security 20)*. 2020, pp. 1605–1622.

[118] Gilad Baruch, Moran Baruch, and Yoav Goldberg. „A little is enough: Circumventing defenses for distributed learning“. In: *Advances in Neural Information Processing Systems* 32 (2019).

[119] Gan Sun et al. „Data poisoning attacks on federated machine learning“. In: *IEEE Internet of Things Journal* 9.13 (2021), pp. 11365–11375.

[120] Su Wang, Rajeev Sahay, and Christopher G Brinton. „How potent are evasion attacks for poisoning federated learning-based signal classifiers?“ In: *ICC 2023-IEEE International Conference on Communications*. IEEE. 2023, pp. 2376–2381.

[121] Taejin Kim et al. „pFedDef: Characterizing evasion attack transferability in federated learning“. In: *Software Impacts* 15 (2023), p. 100469.

[122] Danish Javeed et al. „Man in the middle attacks: Analysis, motivation and prevention“. In: *International Journal of Computer Networks and Communications Security* 8.7 (2020), pp. 52–58.

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

I Frank Christopher Kirch

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Federated Learning-based Intrusion Detection for Cyberattack Detection in Healthcare IoMT Networks", supervised by Hayretdin Bahsi and Rajesh Kalakoti
    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 nonexclusive 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.

04.01.2026

[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 licence shall not be valid for the period.