UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

HELIO DO NASCIMENTO CUNHA NETO

Providing a Collaborative and Private Solution for Intrusion Detection System with Federated Learning

NITERÓI

2024

UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE TELECOMUNICAÇÕES

HELIO DO NASCIMENTO CUNHA NETO

Providing a Collaborative and Private Solution for Intrusion Detection System with Federated Learning

Tese de Doutorado apresentado ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Doutor em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sistemas de Telecomunicações.

Orientadores:

Natalia Castro Fernandes

Diogo Menezes Ferrazani Mattos

Ivana Dusparic

NITERÓI

2024

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

C972p Cunha Neto, Helio do Nascimento
Providing a Collaborative and Private Solution for Intrusion Detection System with Federated Learning / Helio do Nascimento Cunha Neto, Natalia Castro Fernandes. - 2024.
129 f.

Orientador: Natalia Castro Fernandes.
Coorientador: Diogo Menezes Ferrazani Mattos; Ivana Dusparic.
Tese (doutorado)-Universidade Federal Fluminense, Escola de Engenharia, Niterói, 2024.

1. Aprendizado Federado. 2. Aprendizado de Máquina. 3. Segurança Cibernética. 4. IDS. 5. Produção intelectual. I. Fernandes, Natalia Castro II. Fernandes, Natalia Castro, orientadora. III. Mattos, Diogo Menezes Ferrazani, coorientador. IV. Dusparic, Ivana, coorientadora. V. Universidade Federal Fluminense. Escola de Engenharia. VI. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

HELIO DO NASCIMENTO CUNHA NETO

Providing a Collaborative and Private
Solution for Intrusion Detection System with Federated Learning

Tese de Doutorado apresentado ao Programa de Pós-Graduação em Engenharia Elétrica e de Telecomunicações da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Doutor em Engenharia Elétrica e de Telecomunicações. Área de concentração: Sistemas de Telecomunicações.

Aprovada em 29 de Janeiro de 2024.

BANCA EXAMINADORA

Profª. Natalia Castro Fernandes, D.Sc. – Orientadora, UFF

Prof. Diogo Menezes Ferrazani Mattos, D.Sc. – Orientador, UFF

Profª. Ivana Dusparic, Ph.D. – Orientadora, Trinity College Dublin

---

Profª. Dianne Scherly Varela de Medeiros, D.Sc. – UFF

Prof. Célio Vinicius Neves de Albuquerque, Ph.D. – UFF

Prof. Diego Gimenez Passos, D.Sc. – ISEL

Profª. Thi-Mai-Trang Nguyen, Ph.D. – Université Sorbonne Paris Nord

Niterói

2024

To my family

Acknowledgments

I would like to thank my partner, Yona, for her kindness, understanding, and support, particularly for blessing me with the gift of our daughter, Alice. I thank my daughter Ana Júlia for her cheerful way of being. She always gave me reasons to be joyful, even on the worst days. I thank my parents and sister for their support; despite our difficulties, they were by my side for whatever I needed. I would never have come so far without you.

I extend my gratitude to my advisors, Natalia, Diogo, and Ivana whose unwavering presence, support, and guidance have been instrumental throughout my journey. Reflecting on my progress, I recognize the substantial evolution I have undergone, and I attribute this growth to your invaluable contributions. Beyond mentors, you have become cherished friends, and for this, I am truly grateful.

I thank the Universidade Federal Fluminense that opened its doors and made me proud to be part of this University. I also want to thank CAPES for the scholarships. I thank the friends I have made at MídiaCom Laboratory, who always maintained a fun and joyful atmosphere. I also want to express my gratitude for the enriching and unique experience at Trinity College Dublin and the invaluable collaboration with the lab members.

I express my gratitude for the support of my friends from the lab, both at UFF and Trinity, who consistently motivated me during challenging moments. They gave me excellent advice and never let me forget my initial motivation and dreams.

Resumo

As técnicas de aprendizado de máquina têm apresentado soluções eficazes para sistemas de detecção de intrusões e a maioria dos problemas de cibersegurança. No entanto, a criação de um modelo de aprendizado de máquina eficaz requer treinamento extensivo com grandes volumes de dados. O aprendizado federado surge como uma solução para treinamento colaborativo a partir de várias fontes, compartilhando apenas os parâmetros do modelo com um agregador central e mantendo os dados localmente. No entanto, a distribuição de dados entre os participantes do aprendizado federado é fortemente não Independente e Identicamente Distribuído (não-IID), o que prejudica o desempenho do modelo global. Apesar dessa dificuldade, a identificação e a disseminação rápida de novos padrões de ataque à rede são cruciais para melhorar a segurança da rede. Portanto, métodos que favoreçam o aprendizado rápido e acurado são de grande valia nesse tipo de cenário. Além disso, os algoritmos de aprendizado federado precisam lidar com eventuais participantes mal-intencionados, os quais podem intencionalmente prejudicar o treinamento com dados aleatórios ou tendenciosos. Esta tese propõe uma solução para acelerar e melhorar o treinamento de um modelo global em aprendizado federado para o cenário Sistemas de Detecção de Intrusões (Intrusion Detection Systems — IDS). A solução proposta possui um método seleção de participantes baseada em pontuação. Para pontuar a contribuição dos participantes, propõe-se um método de pontuação baseado no ganho de informação que leva em consideração tanto o desempenho individual quanto o desempenho coletivo dos participantes. Em aprendizado federado, a seleção de participantes se destaca pela capacidade de quantificar a contribuição de cada participante para o treinamento do modelo global, permitindo melhor eficiência na seleção. Essa abordagem favorece não apenas a melhoria do desempenho do modelo global, mas também reforça a proteção contra participantes mal-intencionados. A solução proposta também incorpora um termo de momento global para reter conhecimentos adquiridos nas rodadas anteriores de agregação. Isso assegura que eventuais alterações abruptas não desviem o direcionamento do treinamento. Além disso, apresenta-se a meta-heurística Federated Simulated Annealing (FedSA) para selecionar hiperparâmetros para cada rodada de agregação. O FedSA otimiza hiperparâmetros vinculados à convergência do modelo global, reduzindo o número de rodadas de agregação necessárias, aumentando a velocidade de aprendizado e disseminação de novos padrões de ataque. O método proposto de seleção de participantes e o FedSA alcançaram desempenho superior a outras abordagens de aprendizado federado de estado da arte. A solução alcançou mais de 80% de F1-Score e 90% de acurácia no conjunto de teste, mesmo na presença de participantes maliciosos. A avaliação mostra que a abordagem proposta converge em menos de dez rodadas de comunicação, exigindo até 50% menos rodadas de agregação para alcançar aproximadamente 97% de precisão na detecção de ataques em comparação com abordagens convencionais de agregação.

Palavras-chave: Aprendizado Federado, Aprendizado de Máquina, Segurança Cibernética, IDS

Abstract

Machine learning techniques provide accurate solutions for intrusion detection systems and most cybersecurity problems. However, creating effective machine learning models requires extensive training with large amounts of data. Federated learning emerges as a solution for collaborative training from multiple sources, sharing only the model parameters with a central aggregator and maintaining the data locally. Nevertheless, the data distribution among federated learning participants is strongly non-Independent and Identically Distributed (non-IID), which harms the performance of the global model. Despite this issue, identifying and rapidly spreading new network attack patterns is crucial to improving network security. Hence, methods that support rapid and precise learning are significant in such scenarios. In addition, federated learning algorithms must deal with malicious participants who can intentionally disrupt training with random or biased data. This thesis proposes a solution to accelerate and improve the global model training in federated learning for the Intrusion Detection Systems (IDS) scenario. The proposed solution has a score-based participant selection method. We propose a scoring method based on information gain to score the participants contributions. The scoring method takes into account both the individual and collective performance of the participants. In federated learning, participant selection is crucial for its capacity to quantify each participant's contribution to the training of the global model, thereby enabling more efficient and effective selection processes. This approach not only improves the performance of the global model but also strengthens protection against malicious participants. The proposed solution also incorporates a global momentum term to maintain knowledge acquired in previous rounds of aggregation. This ensures that any sudden changes do not divert the training direction. In addition, We proposed a metaheuristic called Federated Simulated Annealing (FedSA) to adaptive select hyperparameters for each aggregation round. The FedSA optimises hyperparameters linked to the global model convergence and reduces the number of necessary aggregation rounds, increasing the speed of learning and dissemination of new attack patterns. Our proposed participant selection method and FedSA metaheuristic outperformed other state-of-the-art federated learning approaches. Our solution achieves more than 80% F1-Score and 90% accuracy in the test set, even in the presence of malicious participants. The evaluation shows that the proposed approach converges in less than ten communication rounds, requiring up to 50% fewer aggregation rounds to achieve approximately 97% attack detection accuracy compared to conventional aggregation approaches.

Keywords: Federated Learning, Machine Learning, Cyber-Security, IDS.

List of Figures

List of Tables

Lista de Abreviaturas e Siglas

CCPA	California Consumer Privacy Act	15
CIC	Canadian Institute for Cybersecurity	55
CLI	Command Line Interface	8
CNN	Convolutional Neural Network	8
CPU	Central Processing Unit	16
CRFL	Certifiably Robust Federated Learning	40
DSSGD	Distributed Selective Stochastic Gradient Descent	43
DDoS	Distributed Denial of Service	55
DNS	Domain Name System	55
EU	European Union	14
eSGD	edge Stochastic Gradient Descent	32
FedAvg	Federated Averaging	11
FedSA	Federated Simulated Annealing	xi
FedSBS	Federated Score-Based Selection	2
FedSGD	Federated Stochastic Gradient Descent	78
FedDyn	Federated Dynamic Regularization	76
FL	Federated Learning	1
FL-LSTM	Federated Learning-based Long Short-Term Memory	8
FedAGM	Federated Averaging with Acceleration of Global Momentum	6
FedCS	Federated Learning with Client Selection	11
GAN	Generative Adversarial Network	37
GDPR	General Data Protection Regulation	1
GRU	Gated Recurrent Unit	8
GPU	Graphics Processing Unit	16

Contents

Chapter 1

Introduction

The number of connected devices on the Internet will reach 27 billion by 2027[1]. Different device types connect to various access networks, leading to the Internet of Everything, a heterogeneous and mutable networking environment [1]. Due to the Internet growth and heterogeneity, the complexity of mapping the vulnerabilities from connected devices also grows [2]. Moreover, the spread of the attacking tools (*e.g., slowloris, hping3, macof*) enables quick implementation of a network attack.

Intrusion Detection System (IDS) effectively identify already-known network attacks. However, traditional IDS either deploy a signature database to recognize known attacks or analyze network behaviour, searching for outliers or anomalous behaviours, which can lead to false positives [3]. Consequently, traditional IDS are ineffective for detecting new attack patterns. In contrast, recent Machine Learning-based IDS (ML-IDS) significantly advances in detecting new attack patterns [4, 5, 6, 7, 8]. Besides learning attacks on an extensive dataset, ML-IDS learns from new traffic coming from the network as long as traffic is appropriately labelled. The more data the machine learning model consumes, the more accurate it becomes [9]. Therefore, an ideal strategy is to create a model that could learn from the traffic of different networks. Still, it is challenging due to strict data protection laws, *e.g.*, General Data Protection Regulation (GDPR).

Federated Learning (FL) emerges as a feasible solution to the privacy problem in collaborative training [10]. FL allows several participants to train a global machine-learning model in a distributed fashion without data sharing, overcoming centralized learning [2]. The FL approach enriches the training, building a global model based on extensive participant data. The centralized approach limits the training to an offline

[1]Available at https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook. Accessed in 10/12/2021

dataset or data obtained from a single monitored network. In FL, participants train local models with their local data and share the local model parameters with a central server. The central server creates an aggregated global model based on those parameters. The aggregation server selects random participants for training at each iteration [10], called the aggregation round. The main goal is sharing knowledge without exposing data stored on participants.

A key challenge in designing federated learning-based IDS is that the network data used for training the global model is highly Non Independent and Identically Distributed (Non-IID) [2]. Non-IID data refers to a dataset in which individual data points are not independently or identically distributed. This leads to potential sample imbalances or correlations impacting machine learning model performance.

In contrast to distributed machine learning, FL assumes that the server does not access the participants' data. The absence of data access makes the training more difficult. The data dependence and the heterogeneous data distribution pose challenges to the optimal convergence of the trained model. The optimization issue delays the learning of new attack patterns.

Furthermore, not all participants contribute to the training of the global model. Therefore, efficiently selecting participants contributes the most to the training and speeds up the global model convergence. However, selecting participants is non-trivial since data always remains private for each participant.

Previous works propose using FL for intrusion detection [11, 12, 13]. Still, these works disregard the FL optimization issues, and the traditional FL algorithm implies delays in learning new attack patterns.

In this thesis, we introduce two significant contributions aimed at enhancing the performance of FL systems. First, we propose FedSA, an adaptive hyperparameter optimization method tailored for FL environments. FedSA employs a variant of Simulated Annealing (SA) to adjust learning rate and local updates dynamically at each aggregation round, thereby effectively balancing computational load at the participant level and mitigating the risk of overfitting. Our second contribution is Federated Score-Based Selection (FedSBS), a participant selection method incorporating global momentum into the global aggregation and local updates. Unlike conventional random selection methods, FedSBS selects participants based on their contributions, enabling a more equitable and informed selection process across multiple rounds. Together, these contributions address key challenges in FL, paving the way for more robust and efficient distributed learning

systems.

1.1 Motivation

Recent works deploy FL to allow a machine learning-based IDS federation to share learning without data sharing [12, 11, 14, 15]. However, none of these works addresses the federated optimization challenges typical of FL. Fast convergence is essential for the intrusion detection system scenario. The global model must be up-to-date as fast as possible. The participant network will be vulnerable if the global model delays learning new attack patterns. The approach to enhance global model convergence in the context of the IDS scenario is critical for effectively detecting and learning new attack patterns. To enable the applicability of FL in IDS scenarios, it is essential to tackle the prevalent optimization challenges inherent in such settings. One key issue is participant selection, which is typically random in traditional FL aggregation algorithms. However, in FL, the aggregation server lacks direct access to participants' data, and the data's quality significantly impacts global model training. Selecting participants with low data quality or including malicious participants can hinder the training process. Thus, the development of an efficient participant selection mechanism becomes imperative.

The success of an IDS relies heavily on its ability to adapt and detect emerging attack patterns. However, cyber threats' inherently dynamic nature demands rapid model convergence, obviating any possibility for time-consuming bottlenecks. Among these, hyperparameter fine-tuning poses a challenge in machine learning environments. In FL, the complexity is exacerbated due to the diverse nature of participants data. The burden of fine-tuning hyperparameters can delay the FL training process, impeding the global model progress and threatening the intrusion detection system. In light of these requirements, the investigation to devise an approach that mitigates the need for extensive hyperparameter fine-tuning takes centre stage.

1.2 Goals

The primary goal of this thesis is to propose a robust and effective collaborative training solution for FL specifically tailored for intrusion detection systems. The proposed solution aims to resolve optimization challenges inherent in FL, ensuring its secure and efficient application in the context of intrusion detection.

1.2.1 Specific Goals

The thesis addresses the optimization challenges related to FL, a mandatory requirement for the IDS scenario. Thus, the specific goals are:

1. Provide a background of collaborative training and FL for IDSes with privacy solution architecture;
2. Provide a landscape of the current FL vulnerabilities and attacks;
3. Provide a solution that supplies fast convergence of collaborative training by;
   (a) Improving federated learning participants selection;
   (b) Developing a convergence concerning aggregation algorithm; and
   (c) Improving training with adaptive hyperparameters. Hence, no fine-tuning is required.
4. Evaluate state-of-the-art FL methods and aggregation algorithms within the context of network traffic prediction scenarios.

1.3 Contributions

This thesis introduces two contributions: FedSA [16, 17], designed for adaptive hyperparameter optimization, and FedSBS [18], focused on participant selection. Both contributions target inherent optimization challenges within the FL systems. In addition, we implemented a global momentum to improve the aggregation. Global momentum plays a crucial role in FL by enhancing the stability and convergence speed of model training, thereby enabling more consistent and efficient learning across decentralized nodes. Besides these two contributions, we also provide a comprehensive review and categorization of current Federated Learning vulnerabilities [19], which highlights the security concern about FL systems. The contributions of this thesis are as follows:

1. **The FedSA metaheuristic:** We have developed an adaptive hyperparameter selection algorithm using the FedSA metaheuristic. The FedSA is a Simulated Annealing (SA) metaheuristic variant adapted for FL scenarios. The hyperparameter selection improves the convergence speed of the FL training. Traditionally, FL environments have static hyperparameters. However, FL has a dynamic scenario

that cannot depend on constant values [20]. FedSA metaheuristic learns the training behaviour and changes the hyperparameters dynamically whenever necessary. We proposed using the FedSA metaheuristic for an adaptive hyperparameter and participant selection algorithm due to its aptitude to adapt to the fluid nature of FL scenarios dynamically. FedSA's capacity to modify hyperparameters based on the learning behaviour ensures optimized convergence speed, which is critical for cyber-security scenarios. In addition to accelerating convergence, FedSA also obviates the traditionally time-consuming step of hyperparameter fine-tuning. This improvement enhances the velocity of disseminating new attack patterns, promoting efficiency in IDS scenario. We evaluate FedSA proposal and demonstrate that the proposed scheme outperforms the conventional aggregation approach. We use CICDDoS2019 and CICIDS2017 as benchmark datasets to simulate a network environment. CICDDoS2019 and CICIDS2017 are network dataset comprising normal and attack network flow samples. ;

2. **The Federated Score-Based Selection:** We present a participant selection method called FedSBS to improve convergence in FL training. Our proposed solution enables more efficient model aggregation by selectively choosing participants based on their contribution and reliability. We employ a participant selection using epsilon greedy to select the participants using a score we created and a mechanism to block to avoid over-selecting participants. We propose a scoring formula to quantify the contribution of each participant. Our scoring formula considers both local and global losses. This way, our score measures the participant contribution individually in the group of selected participants. We intend to give a penalty to participants with low local loss and high global loss and give a reward to the contrary. Our participant selection protocol has a participant blocker mechanism to promote a more balanced distribution of participant selection and avoid over-selection. This mechanism keeps track of the times each participant is selected and determines the probability of blocking a participant from further selection. As a result, participants selected multiple times are increasingly likely to be blocked from further selection. Participant selection is a fundamental aspect of FL due to participants inherent diversity and unpredictability in such distributed scenarios. Efficient participant selection ensures that the learning algorithm leverages meaningful contributions while limiting potential disruption from non-contributive or malicious participants. Malicious participants may intentionally provide misleading updates to disrupt the learning process. We evaluate FedSBS by utilizing the CICIDS2017 dataset and im-

plementing a Dirichlet distribution to simulate an unbalanced data scenario, which is characteristic of the FL environment;

3. **Using global momentum:** The solution uses an exponential moving average to improve the round aggregation performance. This approach stabilizes model performance during training by bridging the gap between local and global objectives. They can leverage the global perspective for local model optimization by broadcasting global momentum back to participants. It is important to highlight that our implementation is inspired by Federated Averaging with Acceleration of Global Momentum (FedAGM) [21] and tailored to fit our optimization protocol. The introduction of a global momentum term in FL is crucial as it accelerates convergence by leveraging past gradients and also aids in mitigating the effects of unbalanced participant contributions. Thus, global momentum enhances learning stability and performance in a distributed, heterogeneous context.

1.4 Roadmap

The thesis proposal is structured as follows. Chapter 2 discusses related work. The literature review in this thesis is bifurcated into two main sections: firstly, exploring other proposals on FL-based IDS, and secondly, examining distinct solutions proposed for the optimization challenge in the context of FL. We conceptualize FL in Chapter 3. This Chapter provides a comprehensive review of FL, describing its reference architectures and research challenges. Chapter 4 describes and categorizes the existing vulnerabilities within the FL landscape. We can categorise FL attacks in model performance and data privacy attacks. Chapter 5 describes our first contribution, the FedSA metaheuristic for adaptive hyperparameter selection. Chapter 6 describes our second contribution, the FedSBS participant selection method. Finally, we conclude the current work in Chapter 7.

Chapter 2

Related Work

Besides proposing a federated learning-based IDS, the thesis proposal also addresses the optimization challenge. Then, the related work is divided into two sections. We point out the Federated Learning-based IDSes in Section 2.1, and Section 2.2 presents the works that aim to address the optimization issue.

2.1 Federated Learning-based Intrusion Detection Systems

Previous works propose collaborative learning through an IDS federation using federated learning, extending the learning perimeter [12, 11, 14, 15].

Nguyen *et al.* propose a federated learning model for cyber-attack detection in an access IoT network. The proposed model considers a random participants' selection for each round [12]. IoT gateways operate as participants in federated learning, and an IoT security service provider acts as an aggregation server for models collaboratively trained. The authors evaluate the proposal in a real smart home environment and successfully detect 95.6% of attacks in approximately 257 ms without triggering false alarms.

Preuveneers *et al.* propose to deploy blockchain technology for sharing participants' parameters in a federated IDS environment [11]. The proposal stores all incremental updates to the machine learning model in the blockchain ledger. Despite the advantages of shared storage, the authors identify that blockchain generates latency in training. Moreover, blockchain also introduces communication and processing overhead. The proposed architecture depends on the network nodes to send all their flows via an agent to the IDS analysis. The proposal requires up to 50 aggregation rounds to achieve 97% of accuracy.

Preuveneers *et al.* evaluate their proposal over the CICIDS2017 dataset.

Chen *et al.* propose the Federated Learning-based Attention Gated Recurrent Unit (FedAGRU) [14] for IDS. The authors deploy a Gated Recurrent Unit (GRU) Neural Network but replace the output layer with a Support Vector Machine (SVM). The Global aggregation is asynchronous, *i.e.*, the server does not wait for all selected participants to send their parameters. By correlating the parameters, the participants compare their model updates with the current global model at each global iteration. The participants only send their parameters if their update is relevant to the training.

Li *et al.* propose a distributed IDS using federated learning in the Satellite-Terrestrial Integrated Network (STIN) scenario [22]. STIN is a valuable supplement to the wireless network allowing large-capacity information transmission service to space access networks and terrestrial networks. Besides using federated learning for distributed IDS, the authors also developed a dataset applied to the STIN challenges and limitations. The authors adapted the federated learning algorithm for the STIN scenario, proposing an efficient processing time synchronization, considering the satellites network limitations. The datasets were crafted in a prototype containing approximately forty nodes and eleven simulated attack types. The authors used Convolutional Neural Network (CNN) as the deep learning model.

Zhao *et al.* proposed an intrusion detection based on commands in Command Line Interface (CLI) [23]. The author used the Long Short-Term Memory (LSTM) model to provide richer semantic information in feature space combined with context. The author compared their proposal with centralized learning. The proposal achieve better performance since the model is built collaboratively. However, according to the simulation results, the performance of Federated Learning-based Long Short-Term Memory (FL-LSTM) and Centralized-LSTM is very close.

Mothukuri *et al.* proposed a federated learning approach using an ensemble to enable anomaly detection on the Internet of Things (IoT) networks [24]. The authors used Gated Recurrent Unit (GRU) neural network models to train the machine learning model on a Modbus network dataset. The authors experimented with both GRU and LSTM, and GRU models outperformed LSTM to achieve a higher accuracy rate and be computationally inexpensive. In the paper, the authors used a Random Forest classifier to ensemble seven global model outputs.

Rey *et al.* propose a federated learning-based IDS using Multilayer Perceptron (MLP) and autoencoder models [25]. The authors used the N-BaIoT, a dataset modelling net-

Table 2.1: Comparative Analysis of Existing Federated Learning-based Intrusion Detection Systems, Including our solution. In this table, 'H.P.' denotes Hyperparameter and 'G.M.' signifies Global Momentum.

Method	Participant Selection	H.P. Selection	G.M.	Baseline	Model
Our Solution	FedSBS	FedSA	Yes	FedAvg, Oort, Wang et al., FedAGM, FedDyn, Slowmo.	MLP
Nguyen et al.	Random	Static	No	Centralized	GRU
Preuveneers et al.	Random	Static	No	Centralized	AutoEncoder
FedAGRU	Random	Static	No	FedAvg, CMFL	GRU
Li et al.	Random	Static	No	Centralized, and FedAvg (Different CNN Architechtures)	CNN
Zhao et al.	Random	Static	No	Centralized, and FedAvg (CNN and LSTM)	CNN and LSTM
Mothukuri et al.	Random	Static	No	Centralized	GRU and LSTM
Rey et al.	Random	Static	No	Centralized	MLP
Rahman et al.	Random	Static	No	Self-learning and Centralized ML	Not provided

work traffic of several real IoT devices while affected by malware. The authors compare centralized, distributed, and federated learning architectures.

Rahman *et al.* proposed a Federated Learning-based scheme for IoT intrusion detection [13]. The author evaluated the proposal considering real scenarios and intrusion attacks. For evaluation, the author compared federated, centralized, and self-learning approaches. Federated learning outperformed the other approaches in almost all communication rounds.

The works mentioned above demonstrated the federated learning massive potential for training a global model from multiple sources with privacy. However, none of these works accelerates the convergence of the global model to speed up the learning of new attack patterns. Table 2.1 provides a comparison of our solution to the state-of-the-art FL-based IDS solutions.

2.2 Federated Learning Optimization Challenge

Unlike the previous works, our proposal is concerned with the fast convergence of the global model. The faster the convergence, the faster the model learns with the participants' data. Other works aim to solve optimization challenges from federated learning [26, 27].

Nguyen *et al.* uses Simulated Annealing to optimize federated learning training [26]. The technique is called Simulated Annealing-based Federated Learning (SAFL). The SAFL performs the local update based on the aggregation server feedback. The selected participants disturb some local parameters based on the Simulated Annealing probability at each aggregation round. The SAFL avoids local optimal or overfitting the local model. The authors used the MNIST, Fashion-MNIST, CIFAR10, and Google speech commands datasets for evaluations.

Smith *et al.* propose MOCHA, an optimization structure for the federated environment, which allows the customization of federated learning through the learning of separated and related models for each device [27]. MOCHA calibrates on the resource restrictions of a participating device, such as network conditions and CPU states of the devices. The method has verifiable theoretical convergence guarantees, but it is limited in scale to massive networks and restricted to convex objectives [27]. For Evaluation, the authors used Google Glass (GLEAM), Human Activity Recognition, and vehicle sensor datasets.

Corinzia *et al.* proposed an algorithm for federated multi-task learning, which extends the FL paradigm to handle real-world federated datasets that show statistical heterogeneity among devices [28]. The algorithm is designed to work with general non-convex models. It uses approximated variational inference to perform learning on the federated network, treating it as a star-shaped Bayesian network. The aggregation server aggregates the model parameters received from each participant and uses them to compute a posterior distribution over the shared model parameters. Variational inference is used to approximate this posterior distribution.

Kim *et al.* proposed FedAGM to use global momentum to maintain knowledge from previous aggregations [21]. The global momentum prevents performance instabilities during collaborative training, reducing the gap between the local and global objective functions. The FedAGM algorithm aims to send the global momentum to the selected participants, enabling the incorporation of global momentum into the local updates at

each participant.

Wang *et al.* propose an algorithm to determine the trade-off between local updates and aggregation rounds [20]. The authors analyze the convergence boundary of federated learning based on the gradient descent from a theoretical perspective and propose a new convergence boundary. The convergence boundary incorporates the data distribution, usually unbalanced, among participants. Determining the ideal frequency of aggregation rounds is possible, saving computational resources. For evaluation, the authors used the MNIST dataset.

Aiming to ensure convergence, Chiu *et al.* propose FedProx, which modifies the global loss function, also including a tunable parameter that restricts how much local updates affect the prevailing model parameters [29]. The FedProx algorithm is adaptively tuned. Model updates are adjusted to have less affect current parameters when training loss increases. Likewise, Huang *et al.* propose the LoAdaBoost Federated Averaging (FedAvg) algorithm, in which participants train the model on their local data and compare the cross-entropy loss with the median loss from the previous training round [30]. If the current cross-entropy loss exceeds the previous one, the model is retrained before global aggregation, increasing learning efficiency.

Previous works implemented participant selection to address FL optimization challenge. Song *et al.* proposed a reputation-based FL for wireless networks that employ a beta distribution function to score the reputation of the participants [31]. The authors also proposed a reputation-based scheduling policy that considers wireless channel problems, such as interference and poor connection, for the selection. Besides evaluating participants' contributions, the reputation schemes have also been evaluated to detect malicious users [31].

Lai *et al.* proposed Oort, a guided participant selection for FL [32]. Oort prioritizes participants with high local loss and can run local training quickly. The authors modeled the participant selection as a multi-armed bandit problem, where each participant is an "arm" of the bandit, and the score obtained is the reward [32]. Then, the participant selection scheme can either explore non-selected participants or exploit the already selected ones.

Nishio *et al.* proposed a participant selection protocol called Federated Learning with Client Selection (FedCS) [33]. FedCS is a framework for Multi-access Edge Computing (MEC) environment. FedCS first requests some resource information for a subset of participants. Based on this information, FedCS selects the maximum participants that

fulfil pre-defined minimum resources. The authors later extended their work to participants with different data distributions [34]. Table 2.2 compares our solution with the state-of-the-art proposals addressing the statistical challenge of FL, where H.P. represents Hyperparameter, B.M. denotes Blocking Mechanism, and G.M. stands for Global Momentum.

2.3 Discussions and Remark

Previous works aim to use federated learning for federated IDS. However, these works do not address a crucial issue implied in the federated learning scenario, the global model convergence. Other works aim to solve the convergence. Nevertheless, these works focus on other applications of Federate Learning, such as image processing [20, 27, 26] or Natural Language Processin (NLP) [26]. Our proposal manages the adaptive selection of hyperparameters related to the global model convergence and the participant selection. None of those mentioned above focuses on providing global-model adaptive changes into hyperparameters based on loss-function values from previous aggregation rounds. Furthermore, our proposal addresses the optimization challenge by providing a participant selection method.

Table 2.2: A comparative evaluation of our solution against state-of-the-art federated learning proposals for optimization challenges.

Method	Participant Selection	H.P Selection	B.M.	G.M	Baseline	Scenario
Our Solution	Selection based on local and global loss	FedSA	Yes	Yes	FedAvg, Oort, Wang et al., FedAGM, FedDyn, Slowmo.	Intrusion Detection
SAFL	Random Selection	Static	No	No	FedAvg	Intrusion Detection, and Speech Recognition
MOCHA	Random Selection	Static	No	No	CoCoA	Human Activity Recognition, and Vehicle Sensor
Corinzia et al	Random Selection	Static	No	No	FedAvg, and FedProx	Image Classification, Natural Language Processing, Vehicle Sensor, and Human Activity Recognition
FedAGM	Random Selection	Static	No	Yes	FedDyn, FedCM, FedAdam, FedProx, and FedAvg	Image Classification
Wang et al.	Random Selection	Static	No	No	FedAvg	Image Classification
FedProx	Random Selection	Static	No	No	FedAvg, and Centralized	Image Classification, and Object Detection
LoAdaBoost	Random Selection	Static	No	No	FedAvg	Healthcare
Song et al.	Selection based on accuracy	Static	No	No	FedAvg	Wireless Channel Problems
Oort	Selection based on local loss	Static	Yes	No	Yogi, and Prox	Image Classification, Natural Language Processing, and Speech Recognition
FedCS	Selection based on participants information	Static	No	No	FedLim	Image Classification

Chapter 3

Federated Learning

Machine learning techniques have shown excellent performance in solving complex problems in the last few years [9, 35]. However, machine learning models require a large dataset for training, especially for deep learning [19]. Deep learning performance considerably increases when exposed to a large amount of data [36]. Collecting data from diverse sources provides a solution to data acquisition [37].

The conventional cloud computing or cloud-centric method involves mobile devices acting as data collection points that transmit the collected data to centralized cloud servers. Subsequently, the cloud servers process the data by performing various analytical and computational tasks [38]. The cloud-centric approach is widely used in several scenarios where the data is generated by local devices and processed by a system in the cloud. It is important to highlight that the processing system must not be machine learning-based [39, 40, 41]. A potential implementation for healthcare monitoring using a cloud-centric approach involves equipping a patient with IoT monitoring devices that generate sensor data. The data is subsequently transmitted to a cloud-based system that utilizes predictive modelling to anticipate potential diseases [39]. Unfortunately, centralized data merging for training a machine learning model harms data privacy protected by personal data protection laws in several countries worldwide.

Increasingly strict private data protection policies limit cloud-centric approaches to extracting knowledge from remote data. Personal data protection laws stipulate rights to data owners and obligations to institutions that hold data. A prominent law is the GDPR[1], in force throughout the European Union (EU), which establishes guidelines on storing and processing personal data in the EU [42]. The GDPR emphasizes the importance

[1] Available at `https://gdpr-info.eu/`. Accessed on 07/11/2022.

of protecting individuals' fundamental rights and freedoms in handling their data [43]. It has inspired other countries[2] to adopt similar guidelines, such as the *Lei Geral de Proteção de Dados* (LGPD) in Brazil, the California Consumer Privacy Act (CCPA) in the United States, and the Personal Information Protection and Electronic Documents Act (PIPEDA) in Canada. In Brazil, the LGPD identifies the entities, either a public or private organization, that carry out any processing operation on the personal data [44]. Among the duties established for these entities is collecting explicit consent from the data owner and providing reports that identify the processing operations applied to the data, including the specification of its storage location, data masking, and data protection measures.

On top of privacy concerns, the cost of uploading raw data to the cloud also presents a significant challenge for the cloud-centric approach. Uploading data from a mobile device in an area with a poor network connection, for example, causes long delays to the training due to low throughput. The cloud-centric approach results in propagation delays that can cause unacceptable latency for real-time decision-making applications, such as anomaly detection [45] — transferring data to the cloud for processing burdens both core and access networks. Overload is even more relevant when considering unstructured data, such as text, voice, or video. The restrictions of sending data to the cloud (*e.g.*, band and delay) are critical when cloud-centric training depends on wireless access networks [37]. Thus, current proposals consider developing mobile applications at the MEC [33], in which training is performed by three distinct actors: devices, edge, and cloud. The training anchored in the MEC model incurs high communication costs and is unsuitable for constant retraining applications [2], such as smart keyboards. Furthermore, outsourcing computing and data processing on edge servers involves transmitting potentially sensitive personal data, exposing privacy-sensitive data.

The aforementioned limitations of the cloud-centric approach have led to the development of FL. This collaborative learning solution addresses issues such as privacy preservation and communication efficiency [10]. By allowing training with real data from mobile devices and preserving privacy-sensitive information, FL enables machine learning algorithms to run collaboratively without transferring private data to a cloud server. As a result, FL is a critical component in ensuring data privacy in distributed environments.

FL has two main entities, the participants — often referred to as clients — that train the machine learning models with their personal data and the aggregation server

[2]Available at `https://www.dlapiperdataprotection.com/`. Accessed on 07/05/2023.

that aggregates the local models, generating the global model [19]. The global model aggregates knowledge of participants' local data in a single model. Participants perform the following tasks: i) every participant must retrieve parameters from the global model, ii) selected participants must update their local models with their data, and then iii) send the updated local parameters of the models to the server. The aggregation round is the process of updating the global model with data stored on participants, uploading the parameters, and performing aggregation. The aggregation server is responsible for controlling the aggregation rounds, selecting a subset of participants, and aggregating the updates provided by the selected participants to improve the global model. The server randomly selects a subset of participants for the model update at each aggregation round. The federated approaches introduce the concept of using local computational resources, such as Central Processing Unit (CPU) or Graphics Processing Unit (GPU), for model training while participants can keep their data secure and private. Thereby, FL presents itself as a powerful approach to preserving privacy. Since data is always processed locally, the global operation aggregates the models without accessing the data stored on participants.

As FL enables the use of large amounts of data while preserving user privacy, it has become a popular solution in many areas, such as cyberattack detection, vehicle networks, smart healthcare, and IoT in general [11, 46, 47, 48, 49]. Unfortunately, the FL approach presents new vulnerabilities and security challenges. For example, a malicious entity may infer the honest data stored on participants despite FL sharing only the parameters of the model. The malicious entity may be a participant or an aggregation server willing to know the data stored on honest participants [50]. In addition, a malicious participant may contaminate the global model with poisoned models and data. The malicious participant may intentionally compromise the global model to mispredict a specific class or degrade the performance of the model [51]. Hence, achieving FL assumptions requires mechanisms to protect participants' private data and the global model performance.

3.1 Collaborative Machine Learning

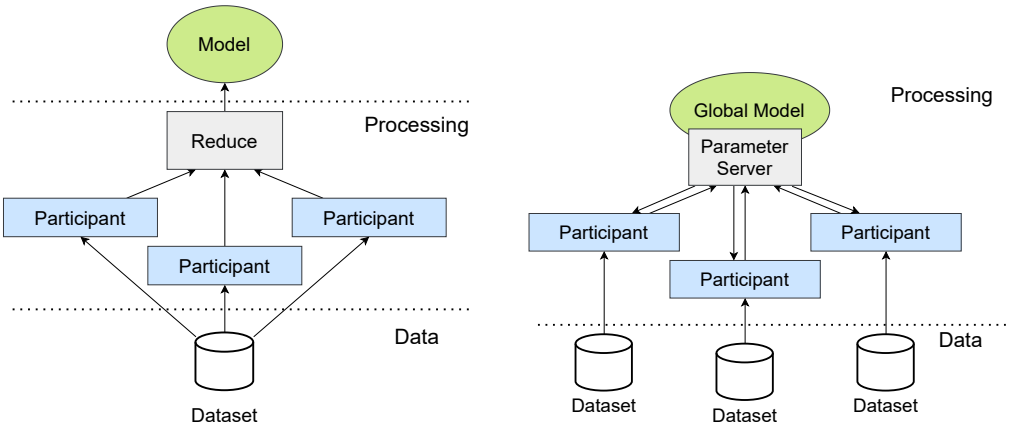
Collaborative machine learning has emerged as an approach that enables distributed processing of a vast amount of available data. This data is necessary to train the ever more sophisticated machine learning models. However, in order to understand collaborative learning, we must first define centralized learning, as it serves as a basis for comparison and contrast to collaborative learning.

Centralized machine learning training is an approach for machine learning models where all data is collected and stored in a central location or server. The owner or an authorized entity typically collects the data and performs the model training. Furthermore, in a centralized approach, the model is trained on the entire data set on a central server or cluster of servers. Such an approach is most useful when the training entity owns or has permission to access the data.

Often, the training in the centralized approach is performed by a cluster of servers in a data center to decrease the training time. Participants can call each server in the cluster. The participant is a node within the cluster that contributes to the processing of the machine-learning model. For training a machine learning model in a cluster, every participant has access to the entire dataset and is responsible for computing a portion of the model. After every participant computes their part of the machine learning model, a reduce function creates the model [37, 4, 52]. MapReduce, Hadoop [53], and Apache Spark [54] are well-known implementations of centralized machine learning with distributed processing [37]. Figure 3.1(a) shows centralized training on a cluster server architecture, in which the participants have full access to a single dataset shared between nodes. Unfortunately, this approach has a few downsides, including privacy concerns around sharing sensitive data with a central cluster of servers and the potential for the training process to become a bottleneck as the dataset grows. The centralized approach may be unfeasible in scenarios where the central entity is now the data owner.

Decentralized training is a machine learning training paradigm in which each participant trains their model to contribute to developing a global model. Decentralized training has at least two entities, the participants and the parameter server [55]. The parameter server is responsible for managing tasks among participants. On the other hand, the participants are responsible for performing the tasks demanded by the parameter server using their local data [55]. The parameter server is fundamental to speeding up the training process and allocating computational resources through an interface to train the model efficiently. The parameter server is also responsible for combining the participant models into a single global model. Figure 3.1(b) shows that each participant can access a local dataset, and the parameter server coordinates the participants. It is worth highlighting that the parameter server cannot access the participant data.

Training in the FL approach is decentralized. The parameter server is called the aggregator server, which has no control or access over the data stored on participants. Its function is to select participants and aggregate the updated parameters received by



(a) In centralized training in a cluster, each participant computes a piece of the model in a data center. The reduced function can be performed by one of the participants.

(b) In decentralized training, each participant has a local dataset, but all nodes compute a model alongside the parameter server to create a global model.

Figure 3.1: Comparison of Centralized and Decentralized Training Architectures. The blue rectangles represent the tasks performed on the participant side and the grey box on the server side. The green ellipse represents the machine learning model.

selected participants. The participant can refuse to participate or even lose connection during the training.

3.2 Federated Learning Model Formalization

The FL system consists of two main entities: the participants, who own the data, and the aggregation server, which owns the global model. Let $N = \{1, ..., n\}$ be the set of participants. Each participant n has its private dataset $D_n, n \in N$, and uses their dataset D_n to train a local model w_n^t at every aggregation round t. In each aggregate round, the aggregation server randomly selects a subset of the participants S^t,$S^t \subset N$. Each selected participant sends only the local model parameters to the aggregation server. Then, the aggregation server aggregates all parameters from the selected participants to generate an updated global model w_G^t, where t is the current aggregation round. The participants update the local models for τ local updates before sending the parameters to the server for global aggregation. After the global aggregation, the aggregation server sends the global model w_G^t to all federated participants. The participants update the global model w_G^t with their local dataset in the aggregation round $t+1$. The aggregation round refers to a specific stage in the training process of the machine learning model where the participating devices send their locally computed model updates to a central server for aggregation. An

underlying assumption is that participants are honest, i.e., they use their actual private data to train and send the proper parameters to the server, and they are not attempting to threaten the training.

The training consists of at least three basic steps in each aggregation round [19]. The first step is Local Update, the second is Participant Selection, and the last is Global Aggregation. Before these three basic steps, the aggregation server creates the initialized global model, generating random parameters, usually from a normal distribution. This initial model is set with random parameter values, usually from a normal distribution, a process called Initialization. The server also specifies the training hyperparameters, such as learning rate (η), local updates number (τ), and mini-batch size (B). The three basic steps of the training are described as follows:

1. **Local Update.** Based on the global model w_G received from the server, the selected participants use their local data and processing power to update the model parameters for τ local iterations, generating w_n. The goal of the participant n in the aggregation round t is to find the optimized parameters w_n that minimize the local loss function $F_n(w_n)$.

2. **Participant Selection.** The server randomly selects the subset S^t,$S^t \subset N$ of participants for training. Only the selected participants will send their parameters to the aggregation server.

3. **Global Aggregation.** The server aggregates the parameters of the selected participants and updates the global model w_G on every participant. The goal of the server is to minimize the global loss function $F(w_G)$. A global aggregation occurs each aggregation round.

The three steps are repeated until the convergence of the global loss function or the algorithm meets a stop condition.

Machine learning models have a set of updated parameters based on training data. A training data sample j has two parts. The first part is the vector x_j, which are the features of the sample j that are the input to the machine learning model; the other part is y_j, the desired output of the model. Each model has a loss function for training the model. The loss function calculates the error of the predicted value based on each sample's desired value y_j. The training process minimizes the loss function based on a training dataset. The loss function is different according to the problem. Besides the

local loss function $F_n(w_n)$, in FL, we have the global loss function, which is defined by Equation 3.1. The global loss function measures the loss of the global model considering all the selected participants, as follows:

$$F(w_G^t) = \sum_{n=1}^{N} \frac{|D_n|}{|D|} F_n(w_n), \tag{3.1}$$

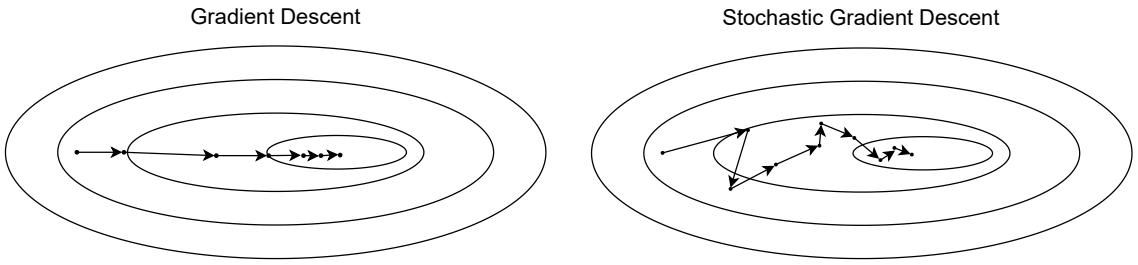
where $|\cdot|$ denotes the cardinality of a set. Assuming that $D = \bigcup_{n=1}^{N} D_n$ and $D_n \cap D_{n'} = \emptyset$ $\forall\ n \neq n'$.

Note that $F(w_G^t)$ cannot be calculated directly without sharing information among participants [20]. The pieces of information shared by the participants are the dataset size $|D_n|$ and the local loss $F_n(w_n^t)$. The optimization problem is then to minimize $F(w_G)$, *i.e.*, to find $w_G^* = \arg\min F(w_G)$.

It is important to highlight that the local and global loss functions are almost identical. It is because the global loss function $F(w_G)$ is the weighted sum of the local loss function of the participants $F_n(w)$, as demonstrated in Equation 3.1. FL works with models based on Stochastic Gradient Descent (SGD) methods [10], such as neural networks, linear regression, and support vector machines. Gradient, in simple terms, means a surface slope direction. Therefore, the gradient descent is the direction to reach a surface minimum point. Hence, the main goal of the gradient descent algorithm is to find the best parameters to minimize the loss function. Thereupon, the model parameters are updated using the partial derivative of the loss function concerning the parameters of the model w_n^t for each sample in the entire dataset. However, this process causes overhead if the dataset has a large number of samples. SGD is a variant of the gradient descent that aims to minimize overhead. Figure 3.2 shows a difference between gradient descent and SGD. In SGD, the parameters are updated in mini-batches at each epoch instead of using the entire dataset samples. Mini-batch denotes the total data used to calculate the gradient at each iteration [56]. That is the reasoning for choosing a subset of participants at each aggregation round in FL instead of asking for the local model of every participant.

In FL, the aggregation server cannot preprocess the dataset, as in centralized models. Besides, federated optimization properties are different from a typical distributed optimization problem, as follows:

1. **Non-IID data.** Local datasets are based on user usage and do not have the same probability distribution, and the samples are dependent. The dependency is due to



(a) The gradient descent uses the entire training dataset to compute the directions to the global minimum.

(b) SGD updates a small data shard, called mini-batch. SGD takes more steps toward the global minimum but is computationally cheap.

Figure 3.2: Comparison of Gradient Descent and Stochastic Gradient Descent. (a) Gradient Descent is a deterministic optimization algorithm that takes large steps to converge to a minimum. (b) Stochastic Gradient Descent is a variant of Gradient Descent that takes smaller steps but is less computationally expensive.

the context of the use of each participant;

2. **Unbalanced Data.** Some users have larger datasets with more samples than others. The local datasets can also be an imbalanced dataset that stands for a dataset with distinct class proportions;

3. **Large number of participants.** The number of participants in a federated optimization is expected to be large. The FL algorithm must handle the massive number of participants. For example, the smart keyboard of Google uses FL for next words prediction with millions of clients [57];

4. **Limited communication.** Mobile and IoT devices, typical of a FL environment, are often disconnected or have low throughput connections.

3.2.1 The Federated Averaging Algorithm

The first and most used aggregation algorithm for FL is FedAvg [10], represented in Algorithm 1. FedAvg convergence has been empirically proven, particularly for problems where the loss function is non-convex [2]. However, FedAvg does not have convergence guarantees and may diverge in practical scenarios when data is heterogeneous [58].

Google researchers implemented FedAvg on Gboard [57]. The Gboard is a smart keyboard for next-word prediction. Since then, other studies have explored FL in a range of scenarios where data is sensitive, for example, developing predictive models for health diagnosis [48] to promote collaboration between hospitals [59] and Government

agencies [60].

Algorithm 1: Federated Averaging pseudo-code [10].

Input: Local mini-batch size B, number of local updates τ, number of participants per aggregation round μ, learning rate η, number of aggregation rounds T

Output: Global model w_G

[*participant n - Update the local model*]
Function *LocalUpdate(n, w)*
  Split the local dataset D_n into mini-batches of size B creating the set B_n
  for *each local_ epoch from 1 to* τ **do**
    for *each* $b \in B_n$ **do**
      $w_n \leftarrow w_n - \eta \nabla F_n(w_n; b)$
    end
  end
  return w_n
[Server side - Performs a global weighted average using the selected local parameters of the models]
INIT w_G
for *each iteration t from 1 to* T **do**
  Randomly selects a subset $S_n \in N$ of size m
  for *each participant* $n \in S_n$ **do**
    $w_n \leftarrow$ **LocalUpdate**(n, w_G)
  end
  $w_G = \sum_{n=1}^{N} \frac{D_n}{D} w_n$
end

The FedAvg algorithm is SGD-based because SGD optimizes the parameter of the model based on a gradient vector $\nabla F_n(w_n)$ pointing to the direction in which the model should evolve. It is simple to perform operations in the gradients of multiple participants. Another point is that deep learning models lean on SGD and variants method to compute parameter optimization [10].

For each aggregation round t, FedAvg algorithm randomly selects a subset of participants, $S^t \in N$, which performs the local update. Each participant computes $\nabla F_n(w_n)$, which are the gradients of their local data for the current model w_n, and the server aggregates these gradients by applying the update:

$$w_G \leftarrow w_G - \eta \sum_{n=1}^{N} \frac{D_n}{D} \nabla F_n(w_n). \tag{3.2}$$

The hyperparameter η is the learning rate and directly influences the convergence speed. A small learning rate implies a smooth trajectory and small weight changes at each

iteration. A very high learning rate implies a more significant weight change, increasing convergence speed. However, it can also lead to fluctuations around a local minimum. An equivalent and commonly used aggregation type is:

$$w_n \leftarrow w_n - \eta \nabla F_n(w_n), \forall n; \tag{3.3}$$

and updated as:

$$w_G \leftarrow \sum_{n=1}^{N} \frac{D_n}{D} w_n. \tag{3.4}$$

Each client locally performs τ gradient descent steps on the current model using its local data, and the server then computes a weighted average of the resulting models. τ controls the local train epochs amount. Hence, three main parameters control the computation amount: i) the portion of participants that perform computation in each aggregation round (parallelization); ii) τ, the number of training iterations each participant performs on their local dataset; iii) and B, the local mini-batch size used for local updates.

Algorithm 1 follows the FL three basic steps aforementioned. In step 1, the server starts the training (lines 11 - 16). Then, in step 2, participant n performs the local training and optimizes its loss function on the local dataset mini-batches (lines 2 - 9). In iteration t (line 17), the server reduces the overall loss by aggregating the average gradients received from the participants. The FL training process will continue until the global loss function achieves a desirable loss or reaches a maximum aggregation round number.

3.3 Federated Learning Reference Architectures

There are three general architectures for a FL system: horizontal FL, vertical FL, and federated transfer learning [9]. We describe each architecture based on a matrix. The rows represent the sample space, and the columns represent the feature space.

3.3.1 Horizontal Federated Learning

Horizontal FL is the most commonly used FL architecture. The main characteristic of this architecture is that the n participants have the same data structure, *i.e.*, the dataset of participants have the same feature space, with a different sample space, as shown in Figure 3.3, where the blue rectangle represents the dataset of participant A, the green

rectangle the dataset of participant B, and the gray rectangle represent the label space of both datasets. An example is when multiple hospitals collaborate to train a machine-learning model for medical diagnosis. Each hospital has its own set of patients with unique medical records, but they all share the same feature space, such as patient age, medical history, and test results. By participating in federated learning, the hospitals can jointly train a model without sharing sensitive patient data directly, thereby maintaining patient privacy.

A conventional assumption is that the participants are honest, while the server is honest-but-curious, which means that the server maintains the functionality of the FL environment but is willing to discover participant data. Therefore, no information leakage from any participant to the server is allowed [61].

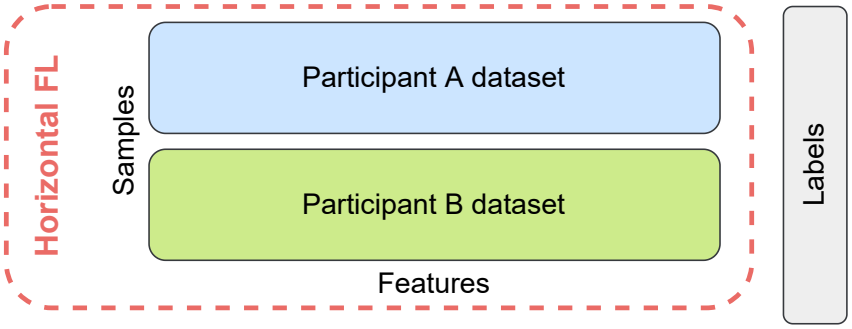


Figure 3.3: Horizontal FL uses datasets with the same feature space but differs in the sample space.

A well-known challenge in the horizontal FL architecture is to protect sensitive data from an honest-but-curious server [9]. Even sharing only the parameters of the model, it is still possible to infer data stored on participants. The most used techniques to protect the parameters of the model are homomorphic cryptography [61] and Secure Multiparty Computation (SMC) [62].

3.3.2 Vertical Federated Learning

Vertical FL or feature-based FL, as shown in Figure 3.4, applies to cases where two datasets share the same sample space but differ in the feature space. For example, two companies operating in the same city may have similar customers but have different information about those customers. Vertical FL is the process of aggregating these different features and computing the loss and gradients with privacy-preserving to build a model with data from both parties collaboratively [9].

Some privacy-preserving machine learning algorithms for vertically partitioned data

have been proposed in the literature, including cooperative statistical analysis [63], association rule mining [64], secure linear regression [65], classification [66], and gradient descent [67].

Some works propose a federated vertical learning scheme to train a privacy-preserving logistic regression model [68, 69]. These works apply Taylor expansion to the loss function and adopt homomorphic cryptography for privacy-preservation gradient descent calculations.

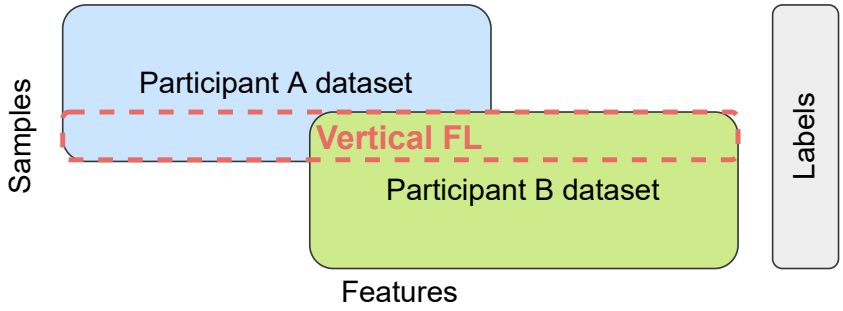


Figure 3.4: In vertical FL, the dataset has some similar samples but with different features. Before starting the training, participants A and B securely select the intersection of the sample spaces.

Suppose companies A and B want to collaboratively train a machine learning model in their business systems, each with its data. Furthermore, Company B also has labeled data that the model needs to predict. For privacy and data security reasons, A and B cannot exchange data directly. A third entity C is involved to ensure data confidentiality during the training process [9]. Entity C is assumed to be honest and not collude with A or B, whereas parties A and B are honest-but-curious about each other. Trusting in entity C is a reasonable assumption, as part C can be performed by authorities such as governments or replaced by a secure compute node [9]. Vertical FL has two parts, as shown in Figure 3.5. In part 1, an alignment between entities is done using cryptography. Since the two companies have different customers, the system uses cryptography-based user Identification (ID) alignment techniques to confirm mutual users in A and B [70, 71]. The system does not use samples of users that do not overlap among the entities during the alignment of entities. Part 2 consists of the encrypted model training. After determining similar users, the model uses the data from those samples to train the machine learning model. We divide the training process into the following four steps [9].

1. Entity C creates homomorphic additive cryptographic pairs of keys and sends the public key to A and B, and encrypts masks for A and B to apply;

2. A and B encrypt and exchange their intermediate results using C's mask for gradient and loss calculations;

3. A and B calculate the encrypted gradients and add an extra mask. B also calculates the encrypted loss. A and B send encrypted values to C;

4. C decrypts and sends the decrypted gradients and loss back to A and B. A and B unmask the gradients and update the model parameters.

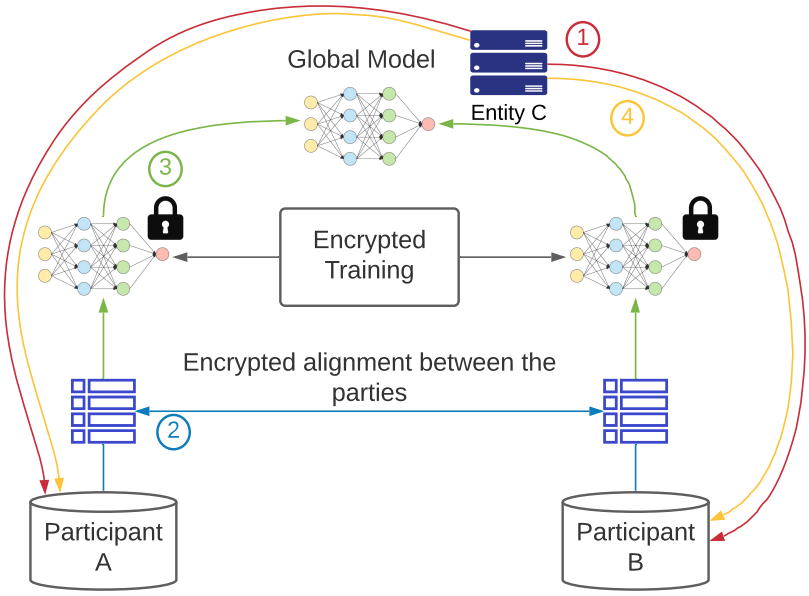


Figure 3.5: The vertical FL system architecture. The entity creates the key pairs in the first step and sends the public key to the participants. In step 2, A and B conduct the security verification to find intersections in their samples, *i.e.*, mutual customers. Participants send their parameters encrypted and masked for aggregation by entity C in step 3. Finally, in step 4, entity C returns the result to the participants.

In short, the vertical FL system helps participants establish a "commonwealth" strategy without affecting data privacy.

A vertical FL system typically assumes honest-but-curious participants. For example, in a two-party case, both parties are not malicious; however, one of the parties may collude with an adversary. An adversary can only learn information from the dishonest participant since the adversary cannot participate. The third participant, C, is introduced to facilitate safe processing between the two parties. In this case, it is an assumption that this third party is not in collusion with either party. Secure multiparty computing provides formal proof of privacy for these protocols [72]. At the end of learning, each party has only the model parameters associated with its features. Therefore, the two parties must collaborate to generate the output model at the inference time.

3.3.3 Federated Transfer Learning

Federated transfer learning applies to scenarios where datasets differ in the samples and the feature space. An example is the case of two different institutions operating in different countries [9]. In this case, transfer learning techniques [73] provide solutions for the entire sample and feature space in a FL environment, as seen in Figure 3.6.

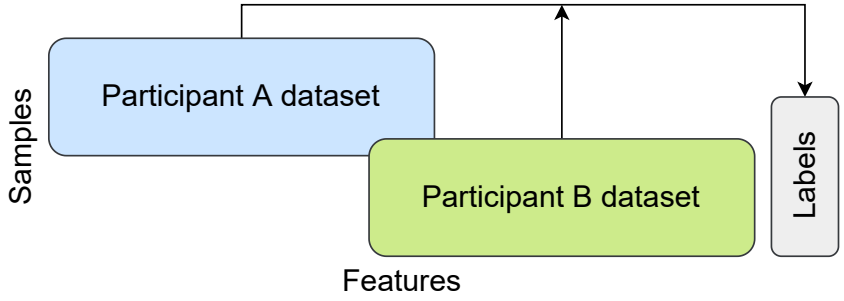


Figure 3.6: In federated transfer learning, participant A wants to learn from the entire dataset of participant B. For this, transfer learning techniques are performed, which transfer knowledge from one model to another without exposing the dataset used to train the source model.

Suppose parties A and B have only a small set of overlapping samples. The main interest is to train a model to predict the labels for the participant A dataset (destination domain) using the participant B model (source domain) knowledge. The architecture described in vertical FL works only for the overlaid dataset samples. Federated transfer learning does not change the general architecture shown in Figure 3.5. However, it changes the intermediate results exchanged between parts A and B. Transfer learning usually involves exchanging common representation from one model to another to minimize the error of the destination domain party, using the knowledge of the source domain party (B, in this case). Therefore, the gradient calculations for parties A and B differ from the vertical FL scenario. Federated transfer learning is an extension of existing FL systems as it deals with problems that exceed the scope of existing FL algorithms.

One possible example is two hospitals, Hospital A and Hospital B, which aim to train a medical diagnosis model for a specific disease. Hospital A has a large dataset of patients with this disease (source domain), while Hospital B has a smaller dataset. Still, it primarily focuses on diagnosing the disease in elderly patients (destination domain). The two hospitals have only a small set of overlapping samples.

The main interest is to train a model that can accurately predict the diagnosis for elderly patients in Hospital B's dataset, leveraging the knowledge from Hospital A's model. In this federated transfer learning scenario, Hospital A first trains a base model using its

extensive patient data. Then, Hospital B fine-tunes this pre-trained model with its local dataset of elderly patients without sharing the raw data with Hospital A.

Through this process, Hospital B can minimize the error in the destination domain while benefiting from the knowledge obtained from the source domain (Hospital A) and maintaining data privacy. Thus, the federated transfer learning extends the capabilities of traditional federated learning by transferring knowledge from one domain to another while addressing unique challenges in each domain.

3.4 Data Sharing and Processing in Federated Learning

Current FL applications use MEC and SMC techniques to ensure low latency and privacy. MEC brings the aggregation server near to the user, while SMC encrypts the parameters of the participants, enabling arithmetic operations.

3.4.1 Multiaccess Edge Computing

MEC is a technology that enables computing resources to be deployed closer to end devices, such as mobile devices and IoT devices. By deploying computing resources closer to end devices, MEC can reduce the data transmission latency. It is particularly important for applications that require real-time data processing or low-latency communications, such as virtual reality, augmented reality, autonomous vehicles, etc. MEC also can support reducing the amount of data that needs to be transmitted over the Internet, which can help to lower the network traffic and reduce congestion. Therefore, MEC allows the model training close to the data sources [2], that is, in devices at the access network. Figure 3.7 shows the MEC traditional architecture.

A collaborative paradigm is widely used for training machine learning models in conventional MEC approaches [2]. The users send their data to the edge servers for the model training instead of sending it directly to the cloud servers, decreasing the communication cost. However, the paradigm still incurs high communication costs for applications that require constant training [74]. Furthermore, data processing on edge servers still involves transmitting potentially sensitive personal data to network edge servers [2]. The leakage possibility may discourage users concerned about their data privacy from participating. Data storage or usage may violate increasingly strict privacy-enforcement laws, such as GDPR. MEC applications are increasingly adopting FL to ensure that training data remains n the device of participants. FL enables complex model training collaboratively

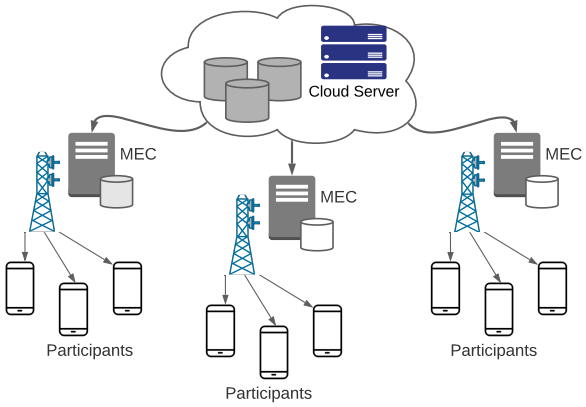


Figure 3.7: MEC architecture, in which servers are located in the carrier's structure, thus offering high throughput and low latency.

among distributed devices without data sharing [74].

3.4.2 Secure Multiparty Computation

FL is closely related to SMC. SMC is a cryptographic protocol that distributes computations among multiple parties, with neither party able to access the data of the other. Secure multiparty computation protocols allow compatible, secure, and private distributed computation [75]. With SMC, multiple parties can collaboratively compute a common function without revealing their private inputs to other parties. For a secure SMC protocol, the parties may learn no other information but the final result. Before FL, previous works proposed algorithms for secure multiparty decision trees for vertically partitioned data [76, 77]. Vaidya and Clifton proposed secure association mining rules [64], secure k-means [78], and a naive Bayes classifier [79] for vertically partitioned data. Du *et al.* proposed secure protocols for linear regression and multiparty classification [66]. Wan *et al.* proposed secure multiparty gradient descent methods [67].

Several SMC approaches were proposed for FL, such as homomorphic encryption [80], pairwise masking [81], and secret sharing [82]. In Homomorphic Encryption, mathematical operations can be performed on encrypted data. Therefore, the aggregation server cannot access the raw values of local models but can aggregate the models, generating an unencrypted global model [80]. Although homomorphic encryption ensures privacy, the scheme is computationally expensive. In pairwise masking, the participants agree with a pairwise mask. The pairwise mask can be exchanged via Diffie-Helman key exchange [81]. However, the approach is insufficient to provide reasonable privacy for the participants. In secret sharing, a secret is broken into multiple parts. Only when the parts are together can the secret be revealed [82]. The secret can be a mask, a key, or parameters. The main

goal of using SMC is to protect the local models against a possible honest-but-curious aggregation server.

3.5 Discussions and Remark

Federated Learning is a research area on the rise due to the growth of available raw data and the need to process them while assuring users' privacy. The federated learning technique develops a collaborative learning model where participants perform part of the learning task locally and contribute to a global model. However, developing collaborative learning models presents several challenges, including statistical data heterogeneity and participant privacy. This chapter reviewed the main federated learning reference architectures and highlighted the difficulties when handling heterogeneous data and devices. We conclude that federated learning still has challenges to reach technological maturity. However, federated learning can potentially protect personal data privacy in environments with sensitive data.

3.6 Research Challenges and Opportunities in Federated Learning

The FL challenges differ from other classical problems, such as distributed learning in data centers or traditional private data analytics. The main three challenges in FL are:

1. Expensive communication: The FL environment comprises a large number of devices, *e.g.*, millions of smartphones, and network communication can be slower than local computing due to limited resources such as bandwidth and energy [9]. Two main aspects must be considered to reduce communication in the federated environment: i) reduce the total number of communication rounds and ii) decrease the size of data transmitted in each aggregation round;

2. Device heterogeneity: Storage, computing, and communication capabilities of each device in the federated environment may differ due to variability in hardware (CPU and memory), network connectivity (2G, 3G, 4G, 5G, and Wi-Fi), and power (battery level) [9]. Furthermore, each network size and system-related restrictions typically result in only a small fraction of devices being active at the same time [83]. It is common for an active device to fail in a given aggregation round due to connectivity or power constraints [83]. Device heterogeneity intensifies challenges such

as stragglers mitigation and fault tolerance. The FL methods must, therefore, predict participant failure, tolerate heterogeneous hardware, and be robust enough to enable that participant failure does not affect aggregation;

3. Federated Optimization: Devices often generate and collect data in a Non-IID manner in the FL environment. Smartphone users, for example, have varied language usage in the context of a next-word prediction task. Also, the number of data points (feature space — x) between devices can vary significantly, and there may be an underlying statistical structure that captures the relationship between devices and their associated distribution [9]. This data generation paradigm violates the Independent and Identically Distributed (IID) data assumptions often used in distributed optimization and can add complexity to problem modelling, theoretical analysis, and empirical evaluation of solutions.

3.6.1 Expensive Communication

Effective communication is crucial to achieving the desired accuracy in FL. Complex deep learning model training, such as in CNN, can comprise millions of parameters in each update [84], resulting in expensive communication and potentially impacting training. The bottleneck is aggravated by network conditions of participating devices and asymmetries in Internet connections, where the transmitting rate is lower than the receiving rate, leading to delays [85, 86]. Therefore, works in the literature [87, 88, 20, 86, 89, 90] aim to improve the communication of FL in three ways: i) increasing the local computation, thus reducing the need for communication rounds; ii) performing the compression of the local model, reducing the size of the data sent to the server; and iii) performing updates based on importance, in which only the parameters that have relevant changes during the local training are sent.

Liu *et al.* proposed the aggregation algorithm Federated Stochastic Block Coordinate Descent (FedBCD), in which each participating device performs several local updates before communicating for global aggregation [91]. Convergence assurance is provided with an approximate calibration of the optimal number of local updates calculated at each aggregation round.

While local update methods can reduce the number of communication rounds, model compression schemes can also reduce the data transmission volume in FL. Examples of compression schemes include “sparsing”, subsampling, and quantization, which significantly reduce the size of the messages communicated in each aggregation round [2]. These

methods have been extensively studied in the literature for distributed training in data center environments, both empirically and theoretically.

Important-based updating is another way to reduce the number of bytes transmitted in FL [2]. This technique assumes that most parameters of a deep neural network model are sparsely distributed and their values near zero [92]. Thus, Tao *et al.* proposed the edge Stochastic Gradient Descent (eSGD) algorithm, which sends only a small fraction of important gradient to the aggregation server [90]. The eSGD algorithm tracks the loss values in two consecutive aggregation rounds. Suppose the loss value of the current aggregation round is less than the previous one. In that case, it implies that the current training gradients and parameters are important for minimizing the training loss. Thus, their respective hidden weights are assigned a positive value.

3.6.2 Device Heterogeneity

In the FL environment, there is significant variation in the device characteristics of network participants, as these devices may have different hardware, network connectivity, and battery levels. These system characteristics make issues such as delays significantly more prevalent than in typical data center environments. To solve this problem, several solutions have been proposed.

Typically, the participant selection in FL is random [9]. FL training progress is limited by the training time of the slowest participating devices [19], *i.e.* stragglers. Therefore, participant selection protocols are investigated to solve the training bottleneck in FL. Kang *et al.* considers system overheads incurred by each device when designing incentive mechanisms to encourage devices with high-quality data to participate in the training process [93]. While these methods primarily focus on the variability of systems to perform live sampling, it is advantageous to consider live sampling a set of small but sufficiently representative devices based on the statistical data structure. From a security perspective, the malicious participant can forge the results to receive the incentive and be selected using their poisoned or mislabeled data.

Traditional data center configurations are based on synchronous and asynchronous schemes. In the synchronous scheme, participants wait for each other to sync — in the asynchronous scheme, participants run independently without synchronization [58]. Synchronous schemes are simple and guarantee a trivial equivalent serial computational model, but they are also more susceptible to delays due to device variability. Asynchronous schemes are approaches used to mitigate stragglers in heterogeneous environments, par-

ticularly in shared memory systems. However, they rely on bounded delay assumptions to control the degree of staleness. The FedAvg [10] algorithm synchronously aggregates parameters. It is, therefore, susceptible to the lag effect as each training round progresses at the speed of the slowest device as the server waits for all selected devices to complete their local training before global aggregation. The training will be affected if various participants face network problems.

Sprague *et al.* have empirically found that an asynchronous approach is robust for participants joining during the aggregation round in progress, as well as when the federation involves participating devices with heterogeneous processing capabilities [94]. Nevertheless, training models on Non-IID data often results in a significant slowdown in the global model convergence. Similarly, Xie *et al.* proposed the FedAsync algorithm, where each newly received local update is adaptively weighted according to its degree of obsolescence, which is defined as the difference between the current aggregation round and the aggregation round to which the received update belongs [95]. Thus, an obsolete update from a straggler is still considered but receives less importance. Furthermore, the authors prove the convergence guarantee for a restricted group of non-convex problems. However, FedAsync accepts every update from honest and malicious participants. FedAsync has no mechanism to identify a malicious participant; every participant is considered honest. Despite the potential benefits of asynchronous FL, synchronous methods remain more prevalent due to their relative immaturity and lack of reliability.

3.6.3 Federated Optimization

Distributed machine learning and FL are both approaches for performing machine learning on data distributed across multiple devices. However, there is a key difference between these two approaches. In distributed machine learning, a central server can access the entire training dataset and subsample it into smaller subsets with similar distributions. The central server can then forward these subsets to participating nodes for distributed training. Some frameworks, such as Apache Spark and Apache Hadoop, are used for this purpose. On the other hand, FL is a method where the central server cannot access the data. In the latter, the training data remains on the devices that generated it, and only the model parameters are shared among the devices.

Previous works on machine learning aim to model statistical heterogeneity using meta-learning and multitasking learning methods. Meta-learning consists of machine learning algorithms applied to metadata [96]. Multitasking learning is a transfer of learning ap-

proach that improves generalization by using the information in the training parameters of related tasks as an inductive bias [97]. These ideas were recently extended to the FL environment [58].

Smith *et al.* proposed MOCHA, an optimization framework for FL that enables the development of customized models for each device while utilizing a shared representation via multitasking learning [27]. MOCHA is calibrated according to the resources of participating devices, such as network conditions and processing load. Although multitasking learning effectively captures internal relationships in local models to improve performance and increase the effective sample size for each node, it has limitations in scalability to massive networks, and it is restricted to convex problems [27].

Chapter 4

Vulnerabilities in Federated Learning

FL is susceptible to attacks against collaborative training. This chapter categorizes the main attacks on FL as model performance and privacy attacks. The chapter discusses FL attacks and proposed countermeasures to address these attacks. While we provide a comprehensive overview of the current vulnerabilities in Federated Learning, it is crucial to note that our framework addresses issues related to model performance attacks.

4.1 Model Performance Attacks

This kind of attack aims to directly or indirectly affect the performance of the global model. Malicious participants can send incorrect or corrupted parameters to bias the global model during global aggregation. Attacks on the model performance can be targeted attacks (backdoor attacks) and untargeted attacks (byzantine attacks) [98]. The malicious participant can backdoor subtasks by sending poisoned models. A backdoor subtask makes the global model fail to predict a particular class, leading to mispredictions [51]. The Byzantine attack goal is to cause the collapse of the global model. Unlike backdoor attacks, byzantine attacks do not intend the misprediction on a specific task. Consequently, when attackers try to poison the training, the aggregation server will update the global model incorrectly, and the entire collaborative training will be compromised [99]. Besides, a participant may want to get the global model but is unwilling to contribute, sending random parameters to aggregation (free-riding). Even unintentionally, free-riding can be harmful to collaborative training.

4.1.1 Data Poisoning Attacks

The objective of the FL is to preserve the privacy of data stored on participants by training machine learning models locally and transmitting only the model parameters to a central server for aggregation. However, the server cannot guarantee that the participants have used actual data during their training process, rendering the system vulnerable to attacks by malicious participants [100]. By poisoning the global model, an attacker can introduce mislabeled data, which biases the training of the global model and produces falsified parameters [101]. Mislabeled data refers to instances in a dataset where the label assigned to a data point is incorrect, *i.e.*, the data is labelled with a category or class that does not accurately reflect its true identity or characteristics. One potential method of carrying out such an attack is to generate a series of counterfeit samples and incorporate them into the local model update, thereby impeding or sabotaging the convergence of the global model.

This attack only affects collaborative training if participants collude to poison the global model. The server randomly selects the participants for aggregation; then, a malicious participant has a $\frac{1}{N}$ chance of being selected. Even if selected once or twice, a single malicious participant cannot harm the training by poisoning its dataset [2]. The malicious participant may collude, infect other participant machines, or use a Sybil attack [102, 19]. The Sybil attack involves a malicious participant attempting to amplify the impact of data poisoning by generating multiple false identities of legitimate participants, all containing poisoned data. Only two false participants can collapse the entire training [101].

Similar to the Sybil attack, Distributed Backdoor Attack (DBA) is a threat assessment framework that exploits the distributed nature of FL to manipulate a subset of training data by injecting adversarial triggers in a distributed manner [103]. In DBA, a global trigger pattern is decomposed into different local patterns and embedded into the training set of different adversarial parties. A trigger pattern can be a specific image or a sequence of pixels that can manipulate the machine learning model's behaviour when added to the training data. For example, a trigger pattern in an image classification task could be a small, specific shape or pattern (such as a red square) added to a corner of an image. When this trigger pattern is present in the image, the model will misclassify the image to a targeted class. DBA aims to create backdoors in the FL model that will make arbitrarily incorrect predictions on the test set with the same trigger embedded. Compared to standard centralized backdoors, DBA is more persistent and stealthy against FL on diverse datasets such as finance and image data [103].

Data poisoning attacks in FL can be executed through Generative Adversarial Networks (GANs) [104]. The attacker initially trains the GAN to replicate the training samples of other participants and subsequently leverages these replicated samples to generate poisoned updates. Such a poisoning attack is characterized by increased generality and efficacy, making it challenging to identify and mitigate [104].

Clean-label and dirty-label are the two main categories of data poisoning attacks [102]. In clean-label attacks, the adversary is assumed to be unable to modify the labels of training data due to a certification process that verifies the correct class of the data and requires imperceptible modifications to any data samples. Conversely, in dirty-label attacks, the adversary can deliberately mislabel a set of data samples with a desired target label, leading to the misclassification of future data when these samples are introduced into the training set.

Possible Solution: Robust aggregation and differential privacy are the most common defenses against data poisoning attacks. Robust aggregation in FL refers to aggregating model updates from multiple participants in a way that is resistant to various types of attacks or noise. Robust aggregation techniques are used to aggregate the updates robustly to such noise and attacks. Another solution to mitigate model poisoning attacks involves incorporating differential privacy mechanisms. This approach entails the introduction of noise to the model updates of each participant, thereby ensuring that the updates do not disclose any information about the participant's private data. The differential privacy prevents attackers from generating poisoned updates, similar to the GAN poisoned dataset.

Fung *et al.* introduced a robust aggregation method to identify and mitigate Sybil attacks [101]. The proposed defense strategy is called FoolsGold. The authors have discovered the differentiation of honest participants from malicious ones by analyzing their updated gradients. In FL, the training data of each participant has a unique distribution and is not shared. Sybil attackers share a common goal and will contribute updates that are similar to each other, unlike honest participants. FoolsGold uses this assumption to modify each aggregation round's local learning rate to minimize malicious participants' impact. The proposal is to maintain the learning rate of participants who provide unique updates while reducing the learning rate of customers who repeatedly contribute similar gradient updates.

Fang *et al.* proposed a defence strategy for detecting and mitigating poisoning datasets in FL [105]. The proposed defence is based on identifying malicious participants by an-

alyzing their parameter updates, which have unique characteristics compared to honest participants. The defence strategy is designed to differentiate between malicious and honest participants using Principal Component Analysis (PCA) for dimensionality reduction and pattern visualization. After computing the gradients in a model update and comparing them to the global model, only the subset of models corresponding to the participants suspected to be the source of a poisoning attack is extracted. This subset is added to a global list. Then, the standardized list is fed into PCA, generating a two-dimensional data visualization. The results show that the defence can effectively identify malicious participants, even in scenarios with a few participants, and remains robust to gradient drift. Similarly, Tolpegin *et al.* propose a method for identifying malicious participants by comparing their updates and either blacklisting them or disregarding their updates in future aggregation rounds [106]. Finally, Sun *et al.* proposed to add Gaussian noise with small standard deviations to the aggregation to mitigate the backdoor attacks [107].

4.1.2 Model Poisoning Attacks

In model poisoning attacks, the malicious participant attempts to manipulate the local model updates before sending them to the aggregation server to poison the global model directly [100]. The adversary aims to cause the global model to misclassify specific inputs with high confidence, which is achieved by manipulating the training process. The attacker can even scale up their parameters to prevail over the averaging, increasing their influence on the global model. Previous works [108, 109, 110, 51] have demonstrated that model poisoning attacks are significantly more effective than data poisoning attacks. Furthermore, these attacks can be executed with just a single attempt, making them a serious threat to the security and integrity of the global model [108].

Bagdasaryan *et al.* introduced a highly effective model poisoning attack for FL [109]. By sharing their poisoned model that contains a backdoor to bias the global model to misclassify, a malicious participant can compromise one or more participants using the proposed constrain-and-scale backdoor. This algorithm allows attackers to create a model with high accuracy in both the main task and the backdoor, enabling the global model to be manipulated without modifying the local dataset of the malicious participant. For instance, in a sentiment analysis task, a backdoor attack could bias the model to classify all reviews containing a particular keyword as positive, regardless of their actual sentiment. Bagdasaryan *et al.* showed that their approach is more effective than dataset poisoning attacks. According to the authors, only eight participants are sufficient to compromise an

entire FL environment with high accuracy in the malicious classification. However, the aggregation server can detect malicious participants by comparing the received models, as poisoned models will have large parameter values compared to other participants.

Zhou *et al.* proposed a deep model poisoning that can be stealthy among benign models[110]. The proposed attack trains a mini-batch for the main task and backdoor sub-task. In this way, the poisoned models will have similar parameters to benign ones, making detection by model comparison complex. The authors reported that some neurons are more important for the main task, and others are more important for the backdoor sub-task. The authors found that calculating the second-order derivative makes it possible to find the neurons that considerably affect the loss function. Then, capturing the second-order derivative, the Hessian matrix can measure the distance and direction of the update [110]. Hence, the authors proposed to find the neurons important to the main task and use a regularization term to penalize SGD, avoiding updating those neurons.

Wang *et al.* proposed a new class of backdoor attack called edge-case backdoor [51]. The edge-case backdoor targets underrepresented input data for misclassification. For example, in an image classification task, the malicious participants can label samples of people using a kilt[1] as "airplane". In an image dataset, it is relatively common to have ordinary people, but people wearing kilts are rare. The edge-case backdoor trains the poisoned model using Projected Gradient Descent (PGD) to reduce the detection probability. Using PGD, the model of the malicious participant does not differ much from the global one at every aggregation round. Finally, before sending the model to the aggregation server, the malicious participant scales its parameter by a scalar to cancel the contribution of the honest participants [109].

Possible Solution: The main defense for model poisoning attacks is robust aggregation [98]. There are two possible ways to achieve robust aggregation [98]. One approach involves evaluating the performance of local models using a validation dataset and avoiding those with significantly poor performance. The other approach involves comparing local models from each participant to detect any statistical differences with the updates made by other local workers. Typically, malicious participants have different goals than honest participants, leading to differences in their local models. Therefore, identifying statistically different local models can help prevent model poisoning attacks.

Andreina *et al.* proposed a robust aggregation called Backdoor detection via Feedback-based FL (BAFFLE) [111]. BAFFLE introduced a validation phase in the collaborative

[1]Kilt is a traditional Scottish garment.

training. In this phase, the aggregation server sends the global model to a random participant set for validation using their local dataset. The participants in this set will vote if the aggregation is genuine. Based on the feedback of the participants, the aggregation server accepts or rejects the global model.

Shen *et al.* proposed a mechanism called AUROR for robust aggregation [112]. The authors observed that the parameters of most honest participants have a similar distribution. On the other hand, malicious participants present an anomalous distribution. AUROR uses k-means to cluster the participants' updates at each aggregation round and discard outliers. Xie *et al.* proposed a general framework called Certifiably Robust Federated Learning (CRFL) [113]. CRFL clips and smooths the local parameters using parameter smoothing [114] before aggregation.

Pillutla *et al.* introduced a new approach called Robust Federated Aggregation (RFA) to make FL more robust in settings where some participants may send corrupted updates to the aggregation server [115]. RFA relies on a robust aggregation based on the geometric median of the parameters and preserves the privacy of participating devices through secure multi-party computation. The paper establishes RFA's convergence for least squares estimation of the global model. It provided experimental results with linear models and deep networks for three tasks in computer vision and natural language processing. RFA outperforms classical aggregation approaches in terms of robustness when the level of malicious participants is high and competitive in low corruption regimes. Similarly, Yin *et al.* presented a robust aggregation algorithm against model poisoning [116]. The focus is on achieving optimal statistical performance, and the authors analyzed two robust distributed gradient descent algorithms based on median and trimmed mean operations. Their median-based aggregation algorithms also improved communication by requiring fewer aggregation rounds for convergence.

Mhamdi *et al.* introduced a new approach called Bulyan, which reduces the space for adversarial attacks and achieves convergence as if only non-Byzantine gradients had been used to update the model [117]. Bulyan combined Krum and a variant of the trimmed mean. Krum is a distributed algorithm for "K-reverse nearest neighbour Using Minimization". The Krum algorithm operates by collecting model updates from a subset of participants and then selecting the updates that are closest to the median using the Euclidean distance as a metric for comparison. Specifically, Krum chooses the K updates farthest away from the other updates and then averages them to produce the final aggregated update. Bulyan is shown to avoid convergence to ineffectual models and

achieves comparable performance to a non-attacked averaging scheme. However, the authors acknowledge that finding the best direction for non-convex loss functions remains a challenging problem.

4.1.3 Free-Riding Attacks

In FL, free-riding is a deceptive attack where a participant tries to exploit the benefits of the global model without investing sufficient resources in the training process. Essentially, a free-rider selects a smaller subset of their dataset for training or uses random noise instead, conserving their computational resources. This behaviour results in the honest participants having to contribute more resources to the global model training process. Consequently, the poor quality of data the free-rider provides compromises the overall model performance.

Possible Solution: Various solutions have been proposed to mitigate the issue of free-riding in FL. One commonly suggested solution is to utilize blockchain technology to track participant updates and ensure their contributions. However, this approach can lead to potential data privacy attacks. An alternative approach is incentivizing participants to contribute by implementing reward mechanisms that benefit participants with contributions and penalizing those who do not.

Kim *et al.* proposed BlockFL, a FL architecture that leverages blockchain technology to exchange and verify local model updates, thereby addressing the problem of free-riding in FL [118]. Each participant trains and sends the trained local model to its associated miner in the blockchain and then receives a reward proportional to the number of samples of trained data. Then, the proposed framework avoids free-riding participants and encourages all participants to contribute to the learning process. A similar model, also based on blockchain, is introduced by Weng *et al.*, aiming to provide data confidentiality, computational auditability, and incentives for participants in FL [119]. However, using blockchain technology implies implementing and maintaining miners to operate the blockchain. Furthermore, consensus protocols used in blockchain networks, such as Proof-of-Work (PoW), tend to cause long delays in information exchange and, thus, are inappropriate for implementing dynamic FL models.

Another approach to avoid free-riding is using incentive mechanisms for participants contributing to the collaborative training [120]. The incentive mechanism compensates for the effort of a contributing participant. Richardson *et al.* define the influence of the participant as the effect of its contribution on the loss function of the FL model. The

participants receive incentives according to their influence. The total reduction in the loss function bounds the expenses. The aggregation server has to obtain the required budget for the rewards. Huang *et al.* model their incentive framework using game theory [121]. The idea is to monetize the global model and allocate the profits to each participant according to their contributions.

4.2 Data Privacy Attacks

One of the main goals of FL is to protect the participant's privacy in collaborative training. However, data privacy attacks can infer the data stored on participants. Any entity possessing local models can infer their data [50]. The aggregation server is the most likely entity to perform such an attack. In particular, this threat is against the FL privacy assumption because the data stored on participants may be leaked.

4.2.1 Model Inversion and Gradient Inference

Model inversion is the attack in which an adversary possessing a trained model uses its parameters to predict the dataset used as input to train that model, thus characterizing an attack on the privacy of a participant [50]. The attacker seeks to take advantage of the correlation between the target, which would be the unknown features and the result predicted by the model. This attack can be performed by the aggregation server that has the updated local models of the participants. The model inversion harms blockchain-based proposals because the models are stored in clear text on the chain. Every blockchain client has a copy of the chain and can be able to perform model inversion to reveal data stored on participants. Fredrikson *et al.* proposed the model inversion attack to retrieve images from a facial recognition model [122]. The authors developed a new class of model inversion attacks that exploits the training parameters revealed with the predictions.

Possible Solution: The main solution for model inversion is using cryptography, differential privacy, and generating artificial data with Generative Adversarial Networks (GAN). Triastcyn and Faltings proposed a framework called Federated Generative Privacy (FedGP) [123]. The main idea of this approach is to train GANs on data stored on participants to produce an artificial dataset to replace the actual participants' dataset. As some participants may have insufficient data to train a GAN locally, the authors proposed a federated GAN model. Thus, user data always remains on their devices. In addition, federated GAN generates a dataset following all data stored on participants' distribution

rather than a single one, which increases privacy. Figure 4.1 shows the architecture of the framework. The authors evaluated the protection by running the model inversion attack and showed that federated GAN reduces information leakage.

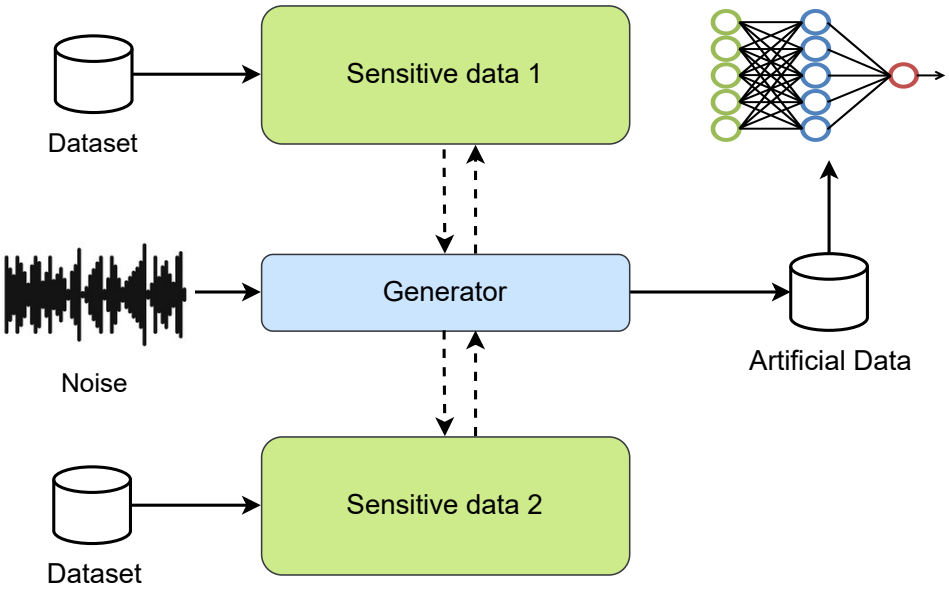


Figure 4.1: FedGP architecture for two participants. Sensitive data trains a GAN, producing an artificial private dataset. The green rectangle represents each participant's data, and the blue one represents the generator model. Adapted from [123].

Zhang *et al.* proposed a homomorphic encryption scheme to preserve the local model parameters of FL [80]. The authors proposed Privacy-Enhanced Federated Learning (PEFL) to protect gradients from an untrusted server. PEFL enhances privacy by encrypting the gradients of local models with Paillier homomorphic cryptosystem. The proposal uses Distributed Selective Stochastic Gradient Descent (DSSGD) [124] algorithm in the local update to reduce the computational costs of the cryptographic system. In addition, encrypted gradients are used for server-side secure sum aggregation, as shown in Figure 4.2. In this way, the untrusted server only learns the aggregated statistics of the updates, keeping local data protected. The authors theoretically prove that the scheme is secure. Evaluations demonstrate that PEFL has low computational costs and, at the same time, achieves a high-accuracy global model. Zhang *et al.* also analyze the time to encrypt the parameters of a fragment of the weights, using DSSGD, and conclude that it is short, sometimes less than one second [80]. However, the authors use state-of-the-art equipment in the evaluations, which does not correspond to the reality of the IoT and mobile environments.

Titcombe *et al.* proposed a mechanism called NoPeekNN to protect the participants against model inversion by adding noise to the intermediate data representation [125]. The

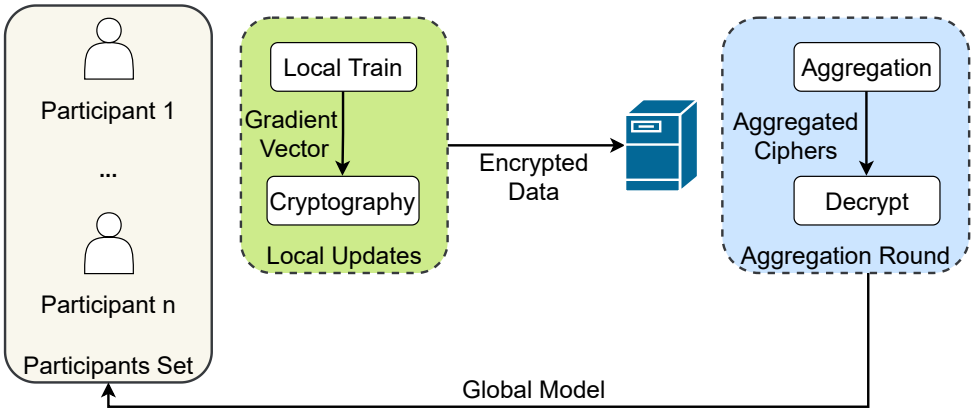


Figure 4.2: In the PEFL Architecture, the participants send the data encrypted using homomorphic encryption, and the server performs the aggregation, returning the aggregated weights to all the participants. Adapted from [80].

authors used additive Laplacian noise to obscure the private data stored on participants. The noise increases privacy but decreases the performance of the model. On the other hand, Qi *et al.* proposed a privacy-preserving method for FL training using differential privacy [126]. The authors used Local Differential Privacy (LDP) for the model gradients before uploading them to the server to protect privacy. LDP is a privacy-preserving technique that adds random noise to individual data points before releasing them or computing aggregate statistics. This way, it can protect the privacy of individuals in a dataset by making it difficult to link their specific data points to their identities.

4.2.2 GAN Reconstruction Attack

The GAN reconstruction attack is a class of FL privacy attacks that is even more effective than model inversion attacks [127]. Model inversion attacks struggle to infer data stored on participants when the deep learning structure is more complex. Hitaj *et al.* introduced the GAN reconstruction attack and showed that a malicious participant could reconstruct the data of the participants [127]. In this attack, the malicious participant creates a replica of the global model to be the discriminator and then trains a generator to create replicas of the data stored on participants. The malicious participant inputs data produced by the generator into the discriminator, calculates the loss of the discriminator outputs and then updates the generator. The malicious participant could infer the data stored in the participants even by applying moderate differential privacy. It is essential to highlight that the more differential privacy is used, the lower the accuracy is.

Possible Solution: Secure multiparty computation or mechanisms for malicious participant detection are the primary defence for FL from GAN reconstruction. Chen *et*

al. proposed a mechanism to protect collaborative training against GAN reconstruction attacks using secure multiparty computation [128]. For this sake, the authors used an improved Du-Atllah scheme, a method for multiple parties to perform calculations without knowledge of the raw data [129]. To achieve privacy, each party adds a mask to their data in a way that the other party mask can cancel the mask during the calculation. The proposed mechanism needs a Trusted Third Party to generate the masks. Since participants do not have full access to the global model during training, the aggregation server must communicate with participants several times at each local update to assist the local training. Thus, neither the participants nor the aggregation server can access the data [128]. The advantage is that the data is protected against the honest-but-curious server and the other malicious participants. The downside is the increased communication cost and the need for a new trusted party.

Yan *et al.* protected the collaborative training against GAN attacks by changing the parameter exchanging protocol [130]. In the proposal, every honest participant embeds an additional layer in the local model called the buried point layer. During the training, when a malicious participant sends its model for aggregation, the aggregation server can identify it due to the absence of the buried point layer. When a malicious participant is detected, communication with the participant is blocked.

Chapter 5

The Federated Simulated Annealing (FedSA) Metaheuristic

The FedSA [17] is a SA variant for optimizing the federated learning hyperparameters. The proposal optimizes the learning rate and the number of local updates. The FedSA metaheuristic can optimize participant selection compared to the traditional random participant selection method. Before we describe the FedSA metaheuristic, it is crucial to understand how SA works.

5.1 Simulated Annealing

The SA metaheuristic aims to find the global minimum of a function based on principles of statistical mechanics. SA is a metaheuristic for tackling Non-deterministic Polynomial-time hard (NP-hard) optimization problems. An NP-hard problem is a class of problems for which no polynomial-time algorithm (*i.e.*, an algorithm that runs in $O(n^k)$ time for some constant k) is known [131]. The term describes problems that are at least as hard as the hardest problems in NP, the class of decision problems that can be verified in polynomial time [131].

SA seeks an optimal solution s derived from an initial randomly generated solution. At each iteration, the metaheuristics generate a random solution ς' in the neighbourhood of the currently best solution. An objective function $f(\varsigma)$ evaluates the performance of the candidate solution ς'. An objective function, also known as a cost function or loss function in optimization and machine learning contexts, is a function that the algorithm or the metaheuristic aims to optimize. The function transforms a given input into a single scalar output that quantitatively represents the quality, cost or loss associated

with that input. The goal is to find the combination of input variables that maximizes or minimizes the output value, depending on the problem. Solutions that minimize the objective function are always accepted. Otherwise, the solution ς' is only accepted within a certain probability. The probability depends on the current simulated temperature parameter T and the degradation of ΔE of the objective function. The ΔE represents the difference between the value of the objective function of the current solution $f(\varsigma)$ and the candidate solution $f(\varsigma')$, hence

$$\Delta E = f(\varsigma') - f(\varsigma). \tag{5.1}$$

Therefore, the acceptance probability function is given by

$$P(\Delta E, T) = \exp\left(-\frac{\Delta E}{T}\right), \tag{5.2}$$

where the lower the temperature value, the lower the probability that a solution worse than the current best one is to be accepted. $P(\Delta E, T)$ assesses the probability of transitioning from the current state ς to a worse candidate state ς'. The acceptance probability follows the Boltzmann distribution [132]. The parameter α, where $0 < \alpha < 1$, is a metaheuristic constant responsible for the gradual reduction of the simulated temperature.

The SA meta heuristic has two while loops. The first loop is responsible for reducing the Temperature. The loop, also called the outer loop, runs the second loop until the Temperature reaches the minimum Temperature tolerable T_{tol}. The second one, also called the inner loop, iterates using the same Temperature for a specific iteration number, managed by the variable T_{iter}.

Algorithm 2 is the pseudo-code of SA. The algorithm takes as inputs an initial temperature T_{init}, a number of iterations at each temperature level T_{iter}, the cooling factor α, and a temperature tolerance — or stopping temperature T_{tol}. The algorithm aims to return the optimal solution ς for the given problem. In Line 2 the function `INIT` generates a random solution. Usually, we use the `INIT` in SA to generate the first solution. In line 3, we initialize i to 0, which counts the iterations. The first while loop at Line 4 keeps iterating while the temperature T is above the tolerance T_{tol}. The second while loop in Line 5 iterates T_{iter} times within each temperature level. In Line 7, the function `GEN_NEIGHBOR` generates a neighbour solution of the best solution. The neighbour solution can be distinct for each type of problem. In line 9, the algorithm computes the change in the objective function ΔE. In line 8, the acceptance probability p is calculated. Lines 10 — 15 check if the change in the objective function is less than 0. Therefore, if ΔE is

Algorithm 2: Simulated Annealing Metaheuristic.

Input: T_{init}, T_{iter}, α, T_{tol}
Output: ς

```
1  T ← T_init
2  ς ← INIT()
3  i ← 0
4  while T < T_tol do
5  |   while i < T_iter do
6  |   |   i ← i + 1
7  |   |   ς' ← GEN_NEIGHBOR(ς)
8  |   |   ΔE ← f(ς') − f(ς)
9  |   |   p = exp(−ΔE/T)
10 |   |   if ΔE < 0 then
11 |   |   |   ς ← ς'
12 |   |   end
13 |   |   else if unif(0, 1) < p then
14 |   |   |   ς ← ς'
15 |   |   end
16 |   end
17 |   T ← T × α
18 |   i ← 0
19 end
20 return ς
```

less than 0, the new solution is better and becomes the best solution. Otherwise, the new solution is conditionally accepted based on a probabilistic criterion determined by the value of p. Line 17 reduces the temperature T by multiplying it with the cooling factor α. In Line 18, we reset i to 0 for the next temperature level. Finally, the algorithm returns the best-found solution ς in Line 20.

A significant advantage of SA lies in its flexibility, allowing application to a diverse array of optimization problems, encompassing both combinatorial and continuous domains [133]. SA is relatively straightforward to implement and does not require a deep understanding of the problem domain [134]. On the other hand, SA can be slow, particularly for complex problems with large search spaces or expensive objective functions [134].

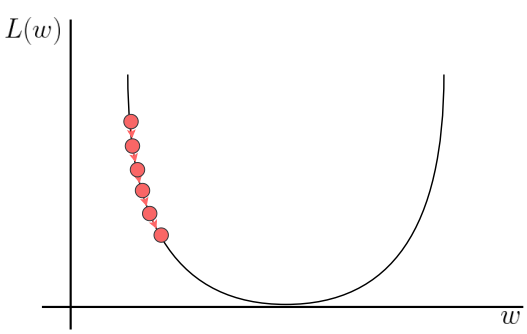
5.2 FedSA Architecture

The federated-learning hyperparameters' optimization scenario differs from the traditional NP-hard problems solved with SA. In traditional NP-hard problems, *e.g.*, the travelling salesman problem, the loss of a given solution ς is the same regardless of the algorithm

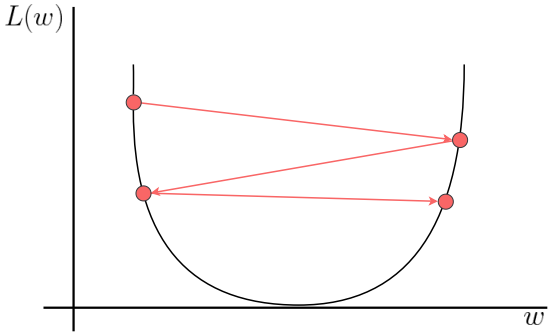
epoch. Unlikely, in the federated learning scenario, the solution ς loss in iteration i may differ from the same solution loss in iteration $i + 1$. In the travelling salesman problem, you are given a list of cities and the distances between each pair of cities. The problem is to find the shortest possible route that visits each city exactly once and returns to the original city. The objective function, the total distance travelled, is fixed: if you re-evaluate the same route at different times (epochs), the distance will remain the same. On the other hand, in the FL hyperparameter optimization scenario, the objective function (also called loss function) can change over time as the global model evolves. Here, the objective function is likely related to the global model's performance, specifically the loss, which can be influenced by the hyperparameters used in each participant local update. Due to the dynamic aggregation of models and the evolving nature of the data across nodes, a set of hyperparameters that produced a global model with low loss in one epoch may not necessarily produce an equally good model in the next epoch. The performance is epoch-dependent and fluctuates as the training progresses. In other words, the training process is stochastic since the same solution returns distinct losses. Hence, the best solution cannot stand for the entire training process. Therefore, we propose a variation of the SA for federated-learning scenarios, which we call FedSA.

The FedSA optimizes the learning rate and the number of local updates. The learning rate stands for the model convergence speed —– a low learning rate delays convergence or traps the model in a local minimum. At a local minimum, the disturbance in candidate solutions is insufficient to generate a change that causes the algorithm to reach another local minimum value. However, a high learning rate may increase learning speed, but it also may lead to fluctuations around the global minimum without refining the solution to the global minimum. The main goal of FedSA is to find an optimal value at each aggregation round. The proposal also uses a neighbourhood structure to find the best value adaptively. Figure 5.1 contrasts a static learning rate with the dynamic learning rate adjustments enabled by FedSA.

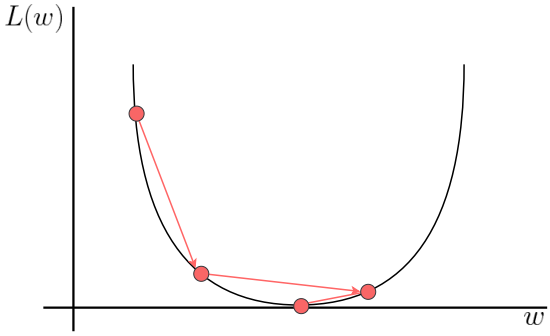
Besides defining the learning rate, the proposal also defines the number of local updates. The number of local updates indicates how many iterations the participants' local models perform on their local datasets before a global aggregation round. In this case, the main goal of FedSA is to define when to increase or decrease the local computation among the selected participants, and FedSA deploys the neighbour structure to decide when to increase or decrease the local computation. A high number of local updates can lead to overfitting, as models may become specialized to their local datasets, reducing their generalizability. On the other hand, a low number of local updates may result in



(a) Convergence behaviour with a small learning rate. Note the slow progression towards the global minima, illustrating the trade-off of stability for speed.



(b) Convergence behaviour with a large learning rate. The plot reveals a fluctuation over the global minima, indicating an instability in the learning process.



(c) Convergence behaviour of the dynamic learning rate adjustment using FedSA. The learning rate adapts at each aggregation round, balancing speed and stability.

Figure 5.1: Compared to the static learning rates, the dynamic learning rate adjustment in FedSA offers an adaptive solution for optimizing convergence.

underfitting and insufficient learning, failing to capture the underlying distribution of the data. Therefore, utilizing FedSA for dynamic local updates mitigates these problems and optimizes computational efficiency at the participant level in Federated Learning.

We also define a neighbourhood structure, enabling FedSA to perform a non-random participant selection at each aggregation round. Using FedSA is not the best solution for participant selection, but it is a better solution than random selection. The neighbourhood structure is based on the ID of the participants and the global loss function. Employing participant selection over random selection in FL ensures that the most informative and relevant local updates are aggregated, enhancing performance. The non-random approach also increases computational efficiency by focusing on participants who contribute meaningfully to training rather than randomly including potentially noisy or irrelevant data.

5.3 Simulated Annealing for Federated Learning

The proposed FedSA metaheuristic aims to find the best combination of participant selection, learning rate, and local updates for each aggregation round, as shown in Algorithm 3. Therefore, the FedSA expects as input the set of participants IDs (μ), the learning rate values limits (η), and the number of local updates limits (maximum and minimum values), given by the tuple τ.

Algorithm 3: Federated Simulated Annealing Metaheuristic.

Input: η, τ, μ, T_{init}, epochs, α
Output: *best_solution*

$T \leftarrow T_{init}$
best_solution $\leftarrow$ **INIT**$(\boldsymbol{\eta, \tau, \mu})$
best_loss $\leftarrow$**AGGREGATE**$(\boldsymbol{best\_solution})$
for $i <$ *epochs* **do**
  current_solution $\leftarrow$ **GEN_NEIGHBOR_SOLUTION**(*best_solution*)
  current_loss $\leftarrow$**AGGREGATE**$(\boldsymbol{current\_solution})$
  $\Delta E \leftarrow \boldsymbol{current\_loss - best\_loss}$
  $p = \exp(-\frac{\Delta E}{T})$
  if *best_loss* $<$ *current_loss* **then**
    best_solution $\leftarrow$ *current_solution*
    best_loss $\leftarrow$ *current_loss*
  end
  else if $\boldsymbol{unif}(0, 1) < p$ **then**
    best_solution $\leftarrow$ *current_solution*
    best_loss $\leftarrow$ *current_loss*
    $T \leftarrow$ **COOL**$(\boldsymbol{T, \alpha})$
  end
  best_loss_test $\leftarrow$ **AGGREGATE**$(\boldsymbol{best\_solution})$
  if *best_loss_test* $<$ *best_loss* **then**
    best_solution $\leftarrow$ **INIT**$(\boldsymbol{\eta, \tau, \mu})$
  end
end
return *best_solution*

The η tuple comprises the interval between the smallest and the highest value that the learning rate may assume during training. The τ tuple comprises the interval of possible values for the number of local updates. The μ contains a list with the IDs of all participants. Every participant associates with an index, a unique identifier, and indicates the sequence that participants joined the federation if FedSA is used for participant selection. FedSA bootstrapping is to perform the parameter selection at random, *i.e.*, the participants' subset selection, the learning rate, and local updates are random. The

subsequent selections are neighbouring solutions to the best solution generated in each aggregation round. Other parameters of the algorithm are the initial temperature (T_{init}), the maximum number of epochs ($epochs$), and a cooling value (α).

We model a solution ς as a tuple containing the number of local updates, the learning rate, and selected participant indexes (IDs). We add a new stage at the end of each iteration of the conventional SA, a new aggregation round, to assess the best solution. Evaluating the best solution allows adding randomness if the best solution degrades, avoiding local minima.

Traditionally, there is an inner loop that is responsible for temperature reduction. For FedSA, we replace the inner loop with a new temperature reduction method. In our proposed variant, the temperature is only reduced when a worse solution is selected. Therefore, the probability $P(\Delta E, T)$ decreases whenever a worse solution is accepted. The idea is enabling a trade-off between diversity and greed. Hence, the algorithm tends to be greedy only when it accepts generalist solutions. By reducing the temperature only when a worse solution is accepted, we provide a trade-off between exploration and exploitation. This enables the algorithm to be greedy precisely when generalist solutions emerge, thereby accelerating convergence without sacrificing solution quality.

The function `INIT` generates a random candidate solution following a uniform distribution. The probability density function is $\frac{1}{b-a}$, where a and b are lower and higher threshold values, respectively. Those thresholds can be passed to FedSA as a hyperparameter. The function `AGGREGATE` represents a global aggregation round and returns the loss of the global model. The function `GEN_NEIGHBOR_SOLUTION` generates a neighbour solution of the current best solution known. The generation of the best solution's neighbour depends on the variable type. The number of local updates is an integer value. Hence, our function selects a subsequent value of the current value as a neighbour.

Figure 5.2 (1) and (2) are a graphical representation of the neighbour selection structure. Each square is an integer value that represents a number of local updates. The blue square represents the so-far best value. The scenario in Figure 5.2 (1) represents a normal situation where the metaheuristic selects neighbour value to the current best value. The subsequent value may be in a positive direction, simply by adding $+1$ to the current value, or in a negative direction, reducing -1 to the current value. The second scenario is when the current best value is the first or last in the list, i.e., a hyperparameter's maximum or minimum value. In this case, the direction of the current iteration would cause the neighbouring solution to extrapolate the border. Then, under this condition, the search

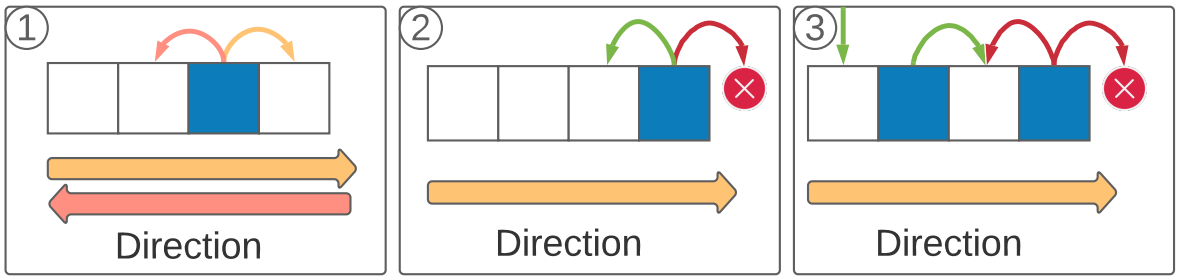


Figure 5.2: Neighbor structure for integer values. The thick orange arrow represents the positive direction, and the salmon represents the negative direction. The thin green arrows are acceptable permutations, and the red ones are forbidden.

direction is flipped.

The participant set selection follows a similar structure. Each participant has an integer number as an ID, and selecting a neighbour means incrementing or decrementing the current ID as shown in Figures 5.2 (1) and (2). Nevertheless, as we are selecting a new set of participants, our proposed method could cause a participant to be selected more than once. In this case, the function performs one of two actions presented in Figures 5.2 (3): the first is to flip the direction and the second is randomly choosing another not yet selected participant. The function will first choose to flip the direction, but if the participant ID of the flipped direction has already been selected, it randomly selects another participant.

The learning rate is a float value, requiring a different method to calculate a neighbour value. Hence, we use

$$\eta \leftarrow \eta_\beta \pm \epsilon \times unif\left(\eta_a, \eta_b\right), \tag{5.3}$$

where, η is a hyperparameter represented as a floating-point number, η_β is the best value so far, η_a and η_b are the smallest and largest values that the hyperparameter assumes. Then, a value is generated following a continuous uniform distribution between the limits values of the hyperparameter. The value is multiplied by a constant ϵ, where $0 < \epsilon < 1$. The constant ϵ is the step for the neighbouring value: the larger the constant, the greater the step. The product of the uniform distribution and ϵ is added or subtracted, depending on the iteration direction. The direction determines whether the term $\epsilon \times unif\left(\eta_a, \eta_b\right)$ should be added to or subtracted from η_β.

After running the neighbourhood calculation, we set the candidate solution as a set composed of i) The selected participants subset S_t; ii) the learning rate η; and iii) the

number of local updates τ. Then, we calculate the loss function for the new candidate solution (line 6) and evaluate whether the new solution should replace the previous solution (lines 7-17).

At the end of every FedSA iteration, an aggregation round is performed (line 18). The new aggregation validates the best solution known. When the known solution is no longer the best, a completely random solution is generated with the function `INIT` (line 20) to prevent the algorithm from being stuck in a single region. The best solution may no longer be the best, *e.g.*, the best learning rate may change with some iteration, or a participant's dataset may no longer reduce loss. Thus, the FedSA constantly evaluates the best solution to avoid getting stuck with a solution that is no longer the best. The landscape of optimal solutions in FL is not static but changes dynamically over time. Factors like data distribution shifts among participants can affect what constitutes an "optimal" hyperparameter setting. Therefore, by re-evaluating the best-known solution at the end of each FedSA iteration, our algorithm ensures adaptability and resilience to these dynamic shifts, preventing stagnation and continuously optimizing performance.

5.4 FedSA: Evaluation and Results

We compare the FedSA with FedAvg as the baseline to evaluate our proposal. The primary purpose of the evaluation is to assert the convergence time reduction of FedSA upon the baseline proposal, FedAvg. At this point, we used FedSA to select the participants.

It is pertinent to highlight that the evaluation and results discussed in this section are based on our earlier work, referenced in [16], carried out before we developed our participant selection method. The assessment of the complete framework can be found in Section 6.3. Additionally, it should be noted that in the evaluations presented in Section 6.3, we were able to refine our approach to simulating non-IID data, which augmented the complexity of the global model, subsequently impacting performance.

We develop a Python-based simulator that generates the participants' processes and shares the dataset among all participants. The simulator gets a dataset and splits it into training and evaluation sets. It is essential to notice that the training and validation have the same proportion of normal and attack samples. The training set contains 70% of the dataset, while the evaluation set contains 30% of the samples. The experiments run with a confidence interval of 95% to ensure statistical relevance. Following a normal distribution, the simulator randomly splits the training set in shards to the n participants.

Since the shards are composed of random samples selected from the training set, it is not guaranteed to contain attack samples. The shards are the same size and represent the participants local datasets. Each participant trains its local model using their local datasets without sharing data. Thus, each participant is unawarded by the other participants' data. At each aggregation round, the selected participants send their models' parameters for aggregation, Equation 3.1, generating the global model w_G^t. The aggregation server uses the validation set to calculate the global loss. The simulations run in a computer equipped with an Intel (R) Xeon Phi (TM) CPU 7250 @ 1.40GHz processor and 128GB of RAM.

We employ TensorFlow[1] to build the machine learning models. The simulator implements FedAvg and the FedSA proposal. Our machine learning model is a Multilayer Perceptron (MLP), deploying two hidden layers. The first hidden layer contains 50 neurons, and the second has 100 neurons. We apply Rectified Linear Units (ReLU) as the activation function in hidden layers and Softmax in the output layer. We empirically choose the neural network configuration, performing fine-tuning with the entire dataset in a centralized machine learning evaluation.

We evaluate the proposals using the CICDDoS2019 and CICIDS2017 dataset [135] to generate traffic on each local network. CICDDoS2019 and CICIDS2017 datasets are from the Canadian Institute for Cybersecurity (CIC) and present normal and attack flows. The CICDDoS2019 dataset has normal flows and the most up-to-date common Distributed Denial of Service (DDoS) attacks, which resemble the actual real-world data. The testbed used in the CICDDoS2019 had one web server, one firewall, and four hosts. The DDoS attacks arrive from an external network. The dataset captures two-day traffic and has twelve DDoS attacks, including Network Time Protocol (NTP), Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), Microsoft Structured Query Language (MSSQL), NetBIOS, Simple Network Management Protocol (SNMP), Simple Service Discovery Protocol (SSDP), User Datagram Protocol (UDP), UDP-Lag, WebDDoS, SYN, and Trivial File Transfer Protocol (TFTP). The CICIDS2017 is a dataset containing a variety of common network attacks. The CICIDS2017 testbed consists of a firewall and twelve hosts, and the attackers are located in a separate network. The CICIDS2017 dataset has five days of captured traffic and eight network attacks, including Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. The CICIDS2017 dataset is unbalanced, containing around 80% of

[1]Available at `https://www.tensorflow.org/library`. Accessed on 31/12/2021.

normal flows and 20% of attack flows. Besides, we apply CICFlowMeter[2] to transform network packets into bidirectional network flows. The CICFlowMeter tool generates 80 different features, such as flow duration, number of packets, number of bytes, and average packet size. We discard source and destination IP addresses, source and destination ports, and transport protocol to avoid overfitting. Both CICDDoS2019 and CICIDS2017 are unbalanced datasets, containing around 80% of normal flows and 20% of attack flows.

5.4.1 Simulation Scenario

We evaluate the accuracy, precision, sensibility, specificity, and loss function to measure the performance of FedSA and FedAvg. We use a scenario with 100 participant IDSes, and only 30% of the participants are selected at each aggregation round. It is important to note that the proportion of selected participants affects the learning. Besides evaluating the metrics above, we measure whether the proportion of selected participants affects the classification performance.

We simulate two scenarios varying the proportion of selected participants to evaluate the impact on training. The main goal is to evaluate the impact of the chosen participants' ratio on the global model's performance and convergence. It is important to note that aggregating the parameter of all participants is not scalable since the growth in the number of participants may negatively impact the server processing capacity. In the first scenario, we consider a federation consisting of 100 participants, where 50 of them are selected for training (S^t), comprising 50% of the entire participant pool. In the second scenario, we increase the total number of participants to 150, selecting 40 for training, thereby altering the proportion of S^t from 50% to $\approx$ 27%. By varying the number of selected participants, our objective is to evaluate the impact of the participant selection ratio on the convergence of the proposed methods. This approach allows us to assess the performance of our proposal with both large and small subsets of selected participants.

FedAvg selects the same number of participants as FedSA, albeit at random. FedAvg employ a learning rate of $\eta = 0.1$ with a decay at each iteration of $\gamma = \frac{0.1}{i}$, in which i is the number of the iteration. There is no established method for determining the optimal learning rate value; it is typically obtained through a fine-tuning process or empirical selection. On the other hand, previous proposals aim to improve the convergence of the neural network using adaptive learning rates [136, 137, 138, 139]. At each iteration, the learning rate updates to:

[2]Available at `https://www.unb.ca/cic/research/applications.html`. Accessed on 31/12/2021.

$$\eta^t \leftarrow \eta^{t-1} \times \frac{1}{1+\gamma i}. \tag{5.4}$$

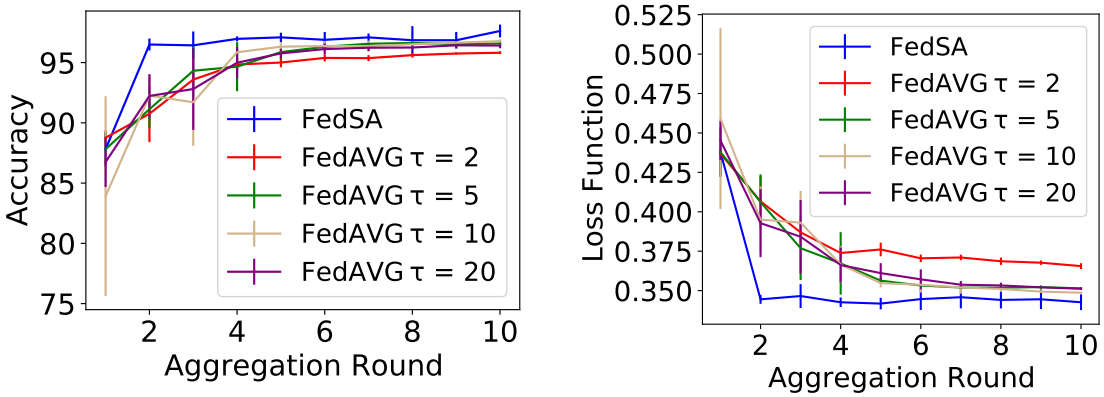
For the FedAvg evaluation, the learning rate dynamically decrements during the communication rounds according to the decay. In FedSA, the learning rate adaptively changes during the training, which means that the learning rate value depends on the previous learning rate value. The local update is the only hyperparameter that remains static for FedAvg. Then, we evaluate scenarios with different local update numbers.

Another crucial point is to evaluate whether the FedSA parameters impact the training. We evaluate the FedSA with different combinations of T and α to evaluate the performance impacts of the FedSA hyperparameters. We also compare federated learning against centralized learning strategies to evaluate the overhead added by the federation procedure.

5.4.2 CICIDS2017 Evaluations

First, we evaluate the impact of the local updates over the aggregation rounds of FedAvg compared against the FedSA proposal. We perform ten aggregation rounds because neither FedSA nor FedAvg improves performance after the tenth round. FedSA chose the number of local updates at each aggregation round in the interval of 1 to 20 local updates. Figure 5.3 shows that the proposal reaches 96% of accuracy in 2 rounds, while, to achieve the same accuracy, FedAvg takes eight aggregation rounds with ten local updates. The randomness of FedAvg while choosing the participants' subset for training may lead to the selection of participants that do not contribute to the model. Besides, the gradual reduction of the learning rate delays the convergence of the global model [140]. The adaptive number of the local updates is essential to the global models' convergence since the need for local updates may diverge for each dataset and sometimes for each aggregation round. We also observe, Figure 5.3(b), that global loss significantly reduces in the first two iterations using FedSA, followed by minor changes in the subsequent rounds. Then, we assume that three aggregation rounds are a fair trade-off between classification performance and processing for a real-world environment.

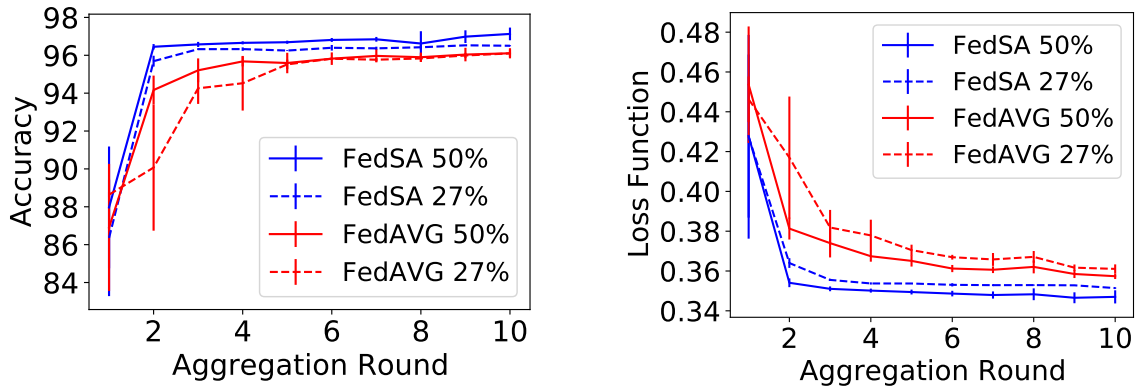
Figure 5.4 shows the proposal and the baseline in two different scenarios. The first scenario is a federation with 100 participants, and at each aggregation round, the server selects 50% for the training process. The second scenario is a federation with 150 participants, but just 27% join the training process. The main focus of the test is to certify if



(a) Comparison of the FedSA and FedAvg accuracy.

(b) Cross-entropy loss function evolution over the aggregation rounds.

Figure 5.3: Accuracy and loss regarding the validation dataset. We can observe that FedAvg achieved 95,5% accuracy in the 6^{th} aggregation round ($\tau = 10$), the same result as FedSA achieved in 2^{nd} aggregation rounds. FedSA converged in the 5^{th} aggregation while FedAvg in the 8^{th}. The acceleration in the convergence is related to the FedSA selection of the hyperparameters and participants.

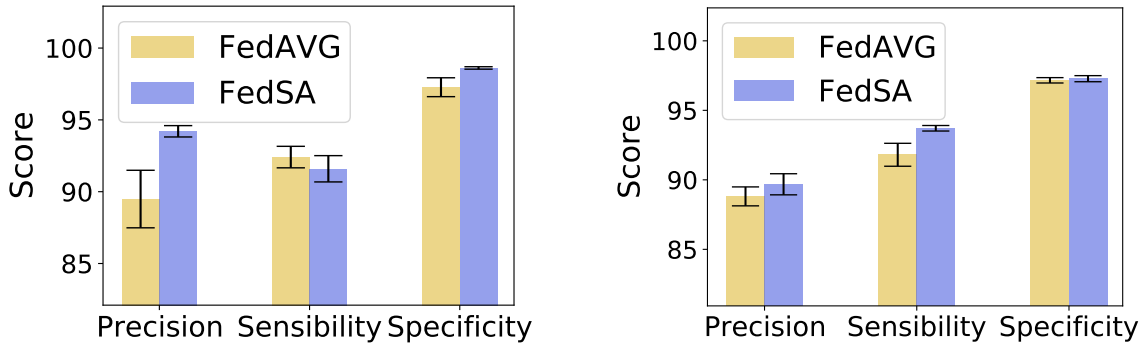


(a) Varying participants' selection ratio for the model training

(b) Loss function evolution over the aggregation rounds.

Figure 5.4: The global model performance for different participants' selection proportions. Comparison of two scenarios: 150 participants and 40 selected at each aggregation round (27% selection scenario); 100 participants and 50 selected at each aggregation round (50% selection scenario). We can observe that both scenarios in FedAvg converged in the 8^{th} aggregation round, achieving the same result by the end. For FedSA, both scenarios converged in the 5^{th} aggregation round, whereas the 50% selection scenario was slightly better.

selecting a different proportion of participants affects the convergence of the model. For the sake of fairness, FedAvg applies the number of local updates that were best performed in previous evaluations. Figure 5.3 shows that ten local updates work better than two, five, and twenty local updates for FedAvg using the CICIDS2017 dataset. Selecting fewer



(a) 100 participants and 50 user selected to S^t subset.

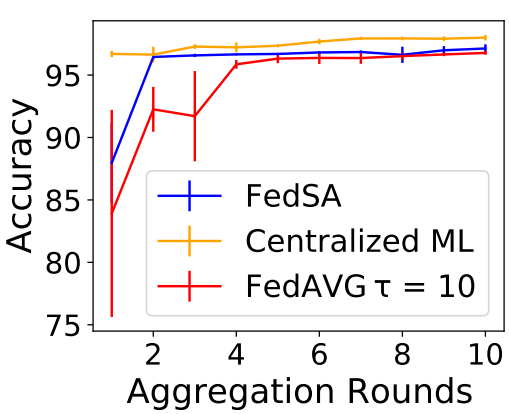
(b) 150 participants and 40 user selected to S^t subset.

Figure 5.5: The global model metrics for FedSA and FedAvg in the scenario of 50% and ≈ 27% of participants' subset selection. We can observe that FedSA had 4% more precision in the scenario selecting more participants. However, we can also mark that even in the 27% selection scenario, FedSA has better performance than FedAvg in the 50% selection scenario.

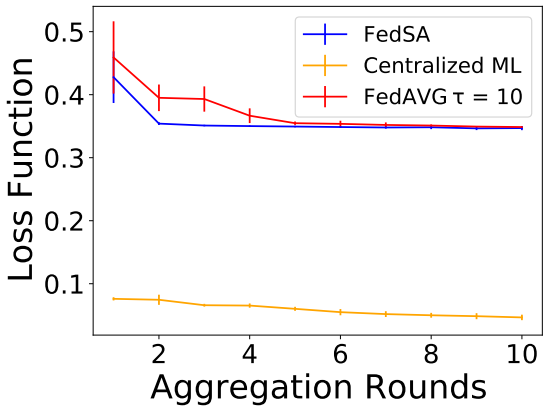
participants does not affect the convergence, as shown in Figure 5.4. It is important to note that the FedSA achieves 96% accuracy even when selecting a small proportion of participants in just three aggregation rounds. In comparison, FedAvg needs eight aggregation rounds. In addition, Figure 5.4(a) shows that the scenario with more participants (50%) obtains a slightly better accuracy than the scenario in which 27% of participants are selected to train the global model. We hypothesize that selecting fewer participants does not burden the classification performance, but it is still crucial to consider other metrics, such as precision, sensibility, and specificity.

The validation dataset, similar to the participant's local dataset, is also unbalanced. Precision evaluates how many attack samples the model predicted correctly. Sensitivity, or Recall, measures the proportion of actual attack cases predicted as an attack by the model. On the other hand, specificity measures the proportion of actual normal traffic predicted as normal traffic by the model. Figure 5.5 shows the precision, sensibility, and specificity for FedSA and FedAvg in two scenarios. The first scenario, shown in Figure 5.5(a), has 100 participants, and 50 participants are selected at each aggregation round to train the global model. FedSA presents better precision and specificity, but worst sensibility, *i.e.*, the FedAvg detected more attack samples in the validation dataset than FedSA. However, FedSA had better precision, *i.e.*, our proposal presents fewer false-positive samples than FedAvg. False-positive is a significant problem for the IDS scenario since classifying a normal flow as an attack flow will cause undesirable false alarms.

The evaluations show that selecting fewer participants for the training does not com-



(a) Comparison of centralized machine learning and federated learning accuracy for FedSA and FedAvg.



(b) The loss function for the centralized machine learning model and the federated learning global models.

Figure 5.6: For centralized machine learning (non-federated learning), one aggregation round equals ten epochs. As expected, centralized learning is slightly better than the federated learning approach. The loss of centralized learning is considerably better than the federated learning losses. This low loss is due to the access to the entire dataset.

promise the convergence but reduces the model's efficiency. The global model achieves reasonable specificity in both scenarios, showing that the proposed algorithm correctly detects normal flows.

Another essential evaluation is a comparison of federated learning and centralized machine learning. We measure the accuracy and loss function of centralized machine learning, FedSA, and FedAvg for the evaluation. Centralized machine learning performs better because the model training accesses the entire dataset. A centralized machine learning approach implies that all the participants' data is centralized in a single dataset for training a single machine learning model. Besides, while training a collaborative model, the data sharing may add some noise to the training. Each aggregation round is equivalent to ten iterations of centralized machine learning. We used ten iterations per aggregation round because we use ten local updates for FedAvg.

Figure 5.6 shows that the federated learning global models achieve an accuracy value close to one from the centralized machine learning model. Although centralized machine learning performs better than federated learning, the centralized approach relies on collecting all samples from the participants' dataset, harming participant privacy. FedSA converges faster than FedAvg, requiring just two aggregation rounds.

The FedSA metaheuristic has two hyperparameters, cooling (α) and temperature (T). The temperature determines the acceptance probability of a solution worse than the best.

High temperatures lead to the acceptance of random solutions, and low temperatures lead to greedy behaviour, refining an already good and accepted solution. The cooling hyperparameter is responsible for decreasing the temperature periodically. Hence, we used a high temperature 0.8 and a low cooling 0.05 in our evaluations. The following evaluation aims to assess the impact of the FedSA hyperparameters on the global model accuracy. This evaluation intends to show that the FedSA hyperparameters do not drastically change the global models' accuracy. We perform the FedSA hyperparameters evaluations using ten aggregation rounds.

Table 5.1: Evaluating FedSA with different cooling and initial temperature. We evaluated the global models' accuracy several times for each hyperparameter combination in a five aggregation rounds scenario and calculated the mean (μ) and standard deviation (σ).

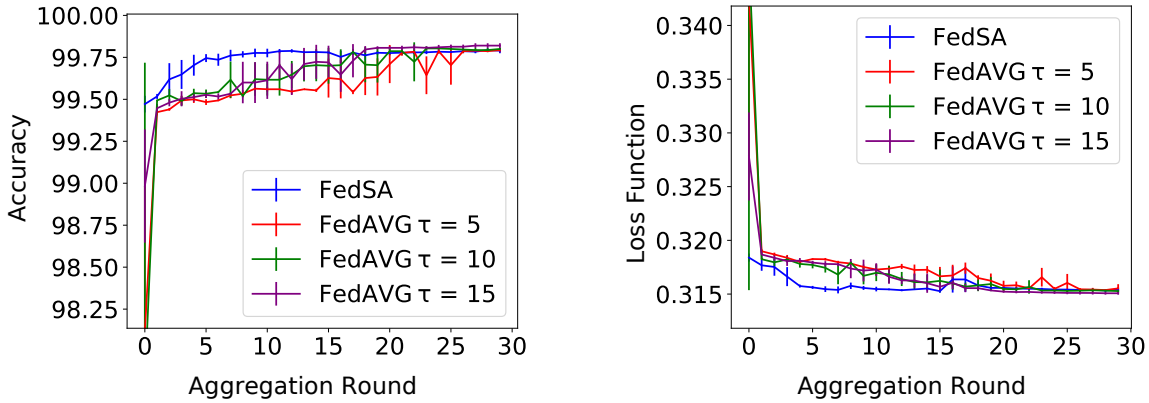
	T = 0.1	T = 0.4	T = 1
cooling = 0.05	$\mu = 96.84$ $\sigma = 0.08$	$\mu = 96.74$ $\sigma = 0.02$	$\mu = 96.66$ $\sigma = 0.01$
cooling = 0.4	$\mu = 96.71$ $\sigma = 0.04$	$\mu = 96.74$ $\sigma = 0.09$	$\mu = 96.67$ $\sigma = 0.07$
cooling = 0.9	$\mu = 96.80$ $\sigma = 0.05$	$\mu = 96.61$ $\sigma = 0.10$	$\mu = 96.56$ $\sigma = 0.06$

Table 5.1 reveals combinations of FedSA hyperparameters. We perform tests for each combination several times for five aggregation rounds. Then, we calculated the mean (μ) and the standard deviation (σ) of each combination. We deploy only five aggregation rounds because, in the previous evaluation, the FedSA global model, at the fifth round, achieved a similar result to the last aggregation round. Thus, we assume the global model convergence at the fifth aggregation round.

Based on the results of Table 5.1, we show that the selection of FedSA input hyperparameters does not interfere the final results. Table 5.1 shows that even drastically changing the FedSA's hyperparameters, the model achieved more than 96% accuracy in 5 aggregation rounds. Therefore, FedSA hyperparameters require no fine-tuning process. Nonetheless, traditional Federated Learning hyperparameters, such as learning rate and local update number, drastically change the global model accuracy, requiring a fine-tuning process.

5.4.3 CICDDoS2019 Evaluations

For CICDDoS2019, we evaluate the impact of local update numbers over the aggregation rounds. We modify the local update number of FedAvg and compare it with the FedSA



(a) Accuracy of FedSA proposal compared to FedAvg.

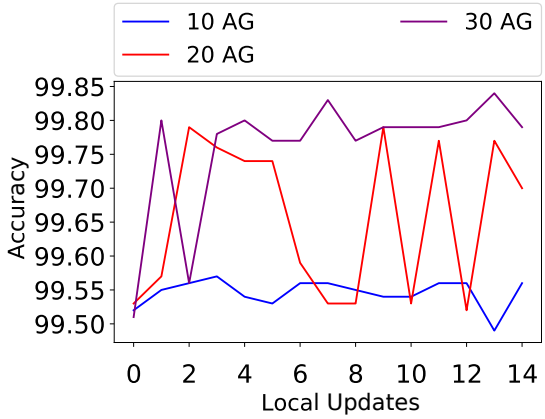
(b) Loss of the cross entropy function.

Figure 5.7: Accuracy and loss function regarding the validation set. Using the CICDDoS2019, FedSA was slightly better than FedAvg. We can observe that FedSA is faster than FedAvg as it converges with fewer aggregation rounds.

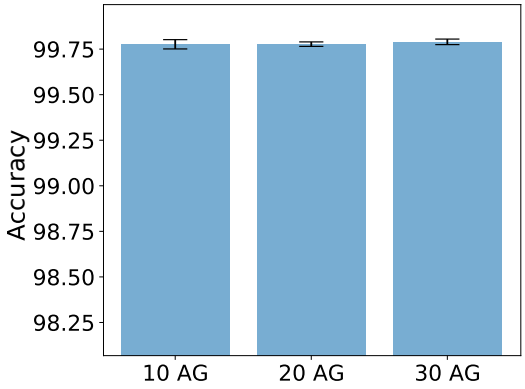
proposal. Figure 5.7 shows the FedAvg with 5, 10, and 15 local updates, and FedSA adaptive varying between 1 and 15 local updates. It is possible to notice that the proposal converges in 5 aggregation rounds, while FedAvg's achieved the same result in 15 aggregation rounds, using 15 local updates. FedAvg scenarios for 5 or 10 local updates converge after 23 aggregation rounds. We can observe that FedSA and FedAvg achieved 99% accuracy in the first couple of aggregation rounds using the CICDDoS2019 dataset. Probably, there is some feature in the dataset that can define the sample class, i.e., has high information gain for the objective classes.

Figure 5.8(a) shows the accuracy of FedAvg in three different scenarios. The first scenario uses 10 aggregation rounds, the second 20, and the third 30. In each scenario, we vary the number of local updates from 1 to 15. This step is a manual fine-tuning of FedAvg hyperparameters. This analysis assesses whether FedSA's choices about the number of local updates impact the accuracy of the solution when compared to using FedAvg. Thus, Figure 5.8(b) presents a bar graph with the proposal's accuracy in the same three evaluation scenarios presented in Figure 5.8(a). The proposal obtained better results than the FedAvg algorithm without performing several fine-tuning tests to identify the best set of hyperparameters.

Figure 5.9 shows the proposal's precision, sensitivity, and specificity bar in the scenario with only ten aggregation rounds for FedSA and FedAvg. We use ten aggregation rounds because it is necessary for the proposal to converge. For a fair comparison with the FedSA, we chose the best number of local updates evaluated in the FedAvg's fine-tuning



(a) We set a window of local updates of 1 to 15 and evaluate the FedAvg accuracy. We use three different scenarios, 10, 20, and 30 global aggregations.



(b) Accuracy of the global model using the FedSA proposal in three different scenarios.

Figure 5.8: The global model accuracy with different values of local updates τ, considering scenarios with 10, 20 or 30 global aggregations.

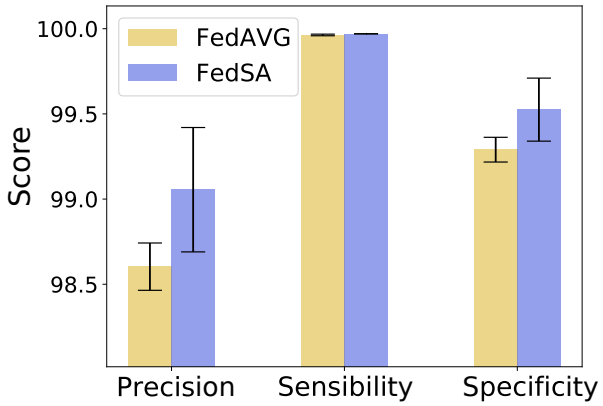


Figure 5.9: Accuracy, Sensitivity, and Specificity of the global model in the scenario of 10 global aggregations using FedSA and FedAvg.

test, described in Figure5.8(a). The FedSA proposal achieved better accuracy, *i.e.*, the amount of correctly classified attacks. Likewise, the proposal achieved a slightly better specificity than FedAvg, *i.e.*, the rate of attacks classified correctly.

5.4.4 FedSA Final Remark

We conclude that using FedSA, we get the model to converge $\approx$ 50% faster than the conventional. We conducted the evaluations five times and calculated the mean and confidence interval of the evaluations. Using the CICDDoS2019 dataset, FedAvg and FedSA achieved 99% accuracy in a few aggregation rounds. We could not demonstrate

that FedSA was faster than FedAvg with the CICDDoS2019 dataset. However, we were able to show that federated learning worked very well for the federated IDS scenario. The CICDDoS2019 dataset is quite massive and takes time to evaluate.

Chapter 6

The Federated Score-Based Selection Method

Federated optimization, discussed in Section 3.6.3, is a critical issue in FL. The federated optimization occurs because the data samples collected from the participants in federated learning are statistically different. This statistical difference can make the global model biased towards specific devices or regions. Furthermore, participants could behave maliciously and intentionally attempt to decrease the performance of the global model, as discussed in Chapter 4. Malicious participants can send incorrect or corrupted parameters to bias the global model during global aggregation.

There are two primary approaches in federated optimization: participant selection [32, 31, 33] and novel aggregation algorithms [16, 141, 142, 21, 143, 144, 145]. Participant selection aims to ensure that the selected devices have data representative of the global population, contributing to the global model performance. Conversely, novel aggregation algorithms have been proposed to enhance the aggregation of the global model, thus mitigating federated optimization. These algorithms aim to improve the performance of federated learning by effectively aggregating the locally trained models from multiple devices.

This chapter presents a method to address the federated optimization in federated learning, which involves a two-fold solution: participant selection and an aggregation algorithm leveraging global momentum. Our proposed solution enables more efficient model aggregation by selectively choosing participants based on their contribution and reliability and leveraging global momentum to improve the convergence and stability of the federated optimization process. We develop a participant selection using epsilon greedy to select the participants using a score we created and a mechanism to avoid over-selecting

participants. We propose an Information Gain (IG) variant to score each participant. Our IG variant considers both local and global losses. This way, our score measures the participant contribution individually in the group of selected participants. We intend to give a penalty to participants with low local loss and high global loss and give a reward to the contrary. Our participant selection method has a participant blocker mechanism to promote a more balanced distribution of participant selection and avoid over-selection. This mechanism keeps track of the times each participant is selected and determines the probability of blocking a participant from further selection. As a result, participants selected multiple times are increasingly likely to be blocked from further selection.

We propose the incorporation of global momentum in the global aggregation to enhance the optimization process. Global momentum enables stable model performance during training by minimizing the gap between the local and global objective functions. After computing the global momentum at each aggregation round, the aggregation server broadcasts the global momentum back to the participants. Thus, the participants can utilize it as local momentum. This approach allows local participants to take advantage of the global perspective while optimizing their models. Our implementation of global momentum is inspired by the FedAGM [21] approach and has been adapted to suit our optimization method. Overall, using global momentum can significantly improve the performance of the global model by promoting stability and reducing performance discrepancies between local and global objectives [21]

6.1 Aggregation with Global Momentum

We describe how the aggregation algorithm utilizes global momentum to prevent performance instabilities during training, thereby reducing the gap between the local and global loss functions [21].

Momentum is a well-known optimization technique in machine learning that accelerates convergence towards the minimum of a loss function. Instead of relying on the gradient of the current step to update model parameters, momentum also considers past gradients. The momentum technique also helps overcome local minima or saddle points, thus facilitating better model generalisation. Figure 6.1 illustrates the model parameters optimization trajectory for two scenarios: one without momentum (depicted in red) and one employing momentum (depicted in blue).

We propose the incorporation of global momentum in the global aggregation to en-

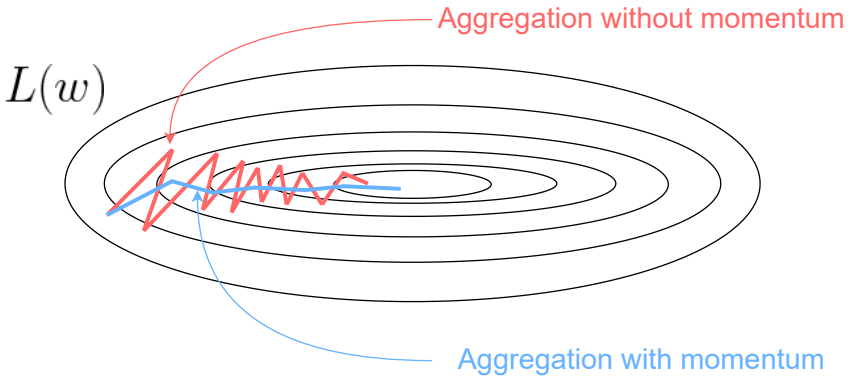


Figure 6.1: Comparative analysis of training a model with and without Momentum. The inclusion of momentum not only accelerates convergence towards the minimum but also aids in overcoming local minima, leading to more efficient and effective training.

hance the optimization process. Unlike traditional momentum, which is computed locally at each participant, Global Momentum is calculated on the server side during the aggregation phase. This centralized momentum term encapsulates global trends across all participating nodes and is then disseminated back to the individual participants for the subsequent training iteration. By doing so, Global Momentum enables a more harmonized and efficient convergence across the network, making the collective learning process more effective and robust. Global momentum enables stable model performance during training by minimizing the gap between the local and global loss functions.

After computing the global momentum at each aggregation round, the aggregation server broadcasts the global momentum back to the participants. Then, the participants can utilize it as local momentum. This approach allows local participants to take advantage of the global perspective while optimizing their models. Overall, using global momentum can significantly improve the performance of the global model by promoting stability and reducing performance discrepancies between local and global objectives [21]. In this section, we explain how the aggregation algorithm uses global momentum. Using global momentum helps prevent performance instabilities during training [21].

The server initializes a global model and sends it to all the participants in the federation. Let $\mathbf{w}_G^t$ denote the global model at aggregation round t, and the global model accelerated by the momentum by $\mathbf{w}_a^t = -\left(\mathbf{w}_G^{t+1} - \mathbf{w}_G^t\right)$. Then, $\mathbf{w}_a^t$ is the global gradient information of the current global aggregation. Each selected participant updates their local model using its local data. The local model of participant i is denoted by $\mathbf{w}_i$. Therefore, it applies a regularization term to the local loss function to reduce the bias and variance of local updates [21]. The regularization term at participant i is denoted by $\mathbf{r}_i(\mathbf{w}_i) = \lambda(\mathbf{w}_i - \mathbf{w}_a^t)$, where λ is a hyperparameter that controls the strength

of regularization. A larger λ means more regularization, which can improve the stability and consistency of local updates. A smaller λ means less regularization, allowing more flexibility for local updates. Therefore, the local loss function is defined by

$$\arg\min_{\mathbf{w_i}} L_i\left(\mathbf{w_i}\right) = \omega f_i\left(\mathbf{w}_i\right) + \mathbf{r}_i(\mathbf{w}_i) \tag{6.1}$$

where $f_i\left(\mathbf{w}_i\right)$ is the objective function, such as cross-entropy or mean squared error. The coefficient ω determines the relative significance of the objective function.

The server collects the local gradients from the selected participants and computes a global gradient by averaging them. The server updates the global model using a momentum-based method incorporating past global gradients. The global update is given by $\mathbf{w}_G^{t+1} = \frac{1}{|S^t|}\sum_{k\in S^t}\mathbf{w}_i^{t+1}$, where $\mathbf{w}_i^{t+1} = \mathbf{w}_G^t - \tau(\mathbf{w}_G^t + \mathbf{r}_i(\mathbf{w}_i))$. The coefficient τ is the learning rate, and S^t is the selected participant subset at aggregation round t.

The main idea of global momentum is to use the global gradient as an acceleration term that helps the participants converge faster and more stably. This method does not require additional communication costs or storage of past models. The evaluations of FedSBS with global momentum are reported in Section 6.3.

6.2 Participant Scoring Method

Our participant scoring method takes inspiration from the IG equation. IG quantifies the extent of information a subset contributes towards making accurate class predictions [146]. The primary use of IG is for feature selection and decision tree creation. We define the IG as:

$$IG = Entropy(\mathbb{P}) - \sum_{t=1}^{N} \varphi Entropy(S^t), \tag{6.2}$$

where N is the number of subsets and φ weights the entropy of the subset S^t. Typically, φ is quantified as the ratio of the number of samples in the subset S^t to the total number of samples in the complete dataset $\mathbb{P}$, $\forall\ S^t \in \mathbb{P}$. This relationship can be formalized as $\varphi = \frac{|S^t|}{|N|}$. Therefore, the IG measures the reduction in entropy of a given subset, *i.e.*, the uncertainty associated with a subset of randomly selected samples [147]. We define the entropy as:

$$Entropy = -\sum_{i} p_i \log_2 p_i \tag{6.3}$$

where p_i denotes the probability of the i^{th} outcome in the dataset or probability distribution under consideration. It serves as a measure of how frequent or likely this particular outcome is concerning all other possible outcomes.

In traditional applications of IG, such as decision tree construction, the selected set of feature S^t is not necessarily identical to the set feature $\mathbb{S}_{t+1}$. However, in the participant selection scenario, the intersection $S^t \cap \mathbb{S}_{t+1}$ can occur, allowing for the possibility of a participant being consecutively selected across multiple aggregation rounds.

The main goal of the participant selection scenario for class prediction is to reduce cross-entropy. Cross-entropy is a measure used to evaluate the dissimilarity between two probability distributions, typically the true distribution and the estimated distribution. In machine learning, particularly in classification problems, cross-entropy is commonly employed as a loss function of the model, denoted by $L(\mathbf{w})$. The goal is to minimize this loss function, driving the estimated probability distribution of the model closer to the true distribution.

Consequently, we can reformulate the IG as:

$$IG' = L(\mathbf{w}_G^t) - \varphi L_i(\mathbf{w}_i) \tag{6.4}$$

where $L(\mathbf{w}_G^t)$ is the global loss function, and $L_i(\mathbf{w}_i)$ is the local loss function of participant i. We removed the sum from the equation to be able to determine the IG of each participant individually.

Equation 6.4 fails to meet the necessary criteria for participant selection, as it yields a high value for scenarios with high entropy in $\mathbb{P}$ and low entropy in S^t subsets. Optimal participant selection is characterized by choosing individuals who attain a low $L(\mathbf{w}_G^t)$, reflecting collective performance, while simultaneously exhibiting a high $L_i(\mathbf{w}_i)$. The high local loss indicates a necessity for further epochs to minimize their individual loss effectively. It means we strive to achieve a low global loss to optimize the overall model performance. Concurrently, participant selection within FL operates with a contrasting goal. The aim is to identify participants associated with a high loss value and work on minimizing it. While the overall FL training targets global loss reduction, the participant selection process focuses on minimizing individual or local losses. As a result, an effective

scoring method for participant selection must strike a balance between lower global loss and higher local loss. Therefore, we use the negative log of both local and global loss functions to achieve this property. Thus, our participant scoring method I definition is as follows:

$$\begin{aligned} I = & -\ln(L(\mathbf{w}_G^t)) - \varphi \times -\ln(L_i(\mathbf{w}_i)) \\ = & -\ln(L(\mathbf{w}_G^t)) + \varphi \ln(L_i(\mathbf{w}_i)) \end{aligned} \tag{6.5}$$

As well as the traditional IG, our participant scoring method returns a higher value for participants who can contribute to the training and a low value for those who cannot. Based on Equation 6.5, the participants with a lower local loss than the global loss are scored with a negative number. On the other hand, participants with a higher local loss than the global loss have a positive score number, usually greater than one. It is essential to highlight that our participant scoring method cannot replace the traditional IG. FedSBS is a scoring equation based on IG for the FL participant selection scenario.

We use the entropy of each participant dataset to determine the weighting of their individual local loss. Our proposal uses the dataset entropy to punish participants with an imbalanced dataset. Every dataset that the target class has an uneven distribution is called an imbalanced dataset. In such a case, using the entropy to weight $\ln(L_n(\mathbf{w}_n))$ may have a different effect if $\ln(L_n(\mathbf{w}_n))$ is positive or negative. Furthermore, if $\ln(L_n(\mathbf{w}_n))$ is positive, then $\varphi = entropy$; otherwise, $\varphi = 1 - entropy$. It is necessary to verify the latter because weights have different effects on negative and positive numbers.

$$\varphi = \begin{cases} 1 + \sum p_c \log_2(p_c), & \textbf{if } \ln(L_i(\mathbf{w}_i)) < 0 \\ -\sum p_c \log_2(p_c), & \textbf{if } \ln(L_i(\mathbf{w}_i)) \geqslant 0 \end{cases} \tag{6.6}$$

Equation 6.6 defines φ, where p_c is the proportion of class c in participant's i dataset.

6.2.1 Participant Selection Method

Alongside utilizing the participant scoring method, our method addresses several critical elements to improve the robustness and efficiency of participant selection:

1. **Balancing Exploration and Exploitation**: Our method strategically navigates the trade-off between randomness and greed. It leverages knowledge from existing,

high-score participants while maintaining a level of randomness to explore potential contributions from other participants. This balance optimizes the learning process and enhances model performance.

2. **Ensuring Diversity**: Our approach actively prevents the over-selection of participants. Over-selected participants, those who are frequently chosen, could bias the training process. By preventing this, our method ensures a diverse range of participants contributing to the model, which leads to a more robust and generalizable global model.

3. **Prioritizing Scalability**: Our solution is designed to be scalable. Recognizing that selecting all participants in large-scale scenarios is not feasible, our method only relies on exploring a subset of participants. It thus provides an efficient and effective selection process that scales with the size of the participant pool.

By addressing these elements, FedSBS resolves significant challenges of participant selection in FL and improves the global model's accuracy and reliability.

FedSBS uses the epsilon greedy approach to balance between the exploration of selecting new participants and exploiting the already known gain of selecting participants. We opted for the epsilon greedy approach in FedSBS due to its efficacy and computational simplicity, ensuring an efficient participant selection process without incurring significant computational overhead. The epsilon greedy policy selects the current best participant with the probability of $1 - \epsilon$ and a random participant selection with the probability of ϵ, where $0 < \epsilon < 1$.

To encourage high exploration at the beginning of training, we start with $\epsilon = 1$. As training progresses, we aim to reduce ϵ to a predefined value b by the end of training to control the algorithm's greediness. To achieve this transition smoothly, we define the decay ϑ as follows:

$$\vartheta = \sqrt[t]{b} \tag{6.7}$$

where t represents the total number of aggregation rounds, and b the last desireable value of ϵ. This strategy allows a balance between exploration and exploitation throughout the training process.

Our participant selection method incorporates a participant blocker mechanism to prevent the over-selection of participants. This mechanism observes the number of times

each participant is selected. It applies a Boltzmann distribution to determine the likelihood of a participant being blocked from further selection after their initial selection. In essence, if a participant has been chosen once, their probability of being blocked from further selection increases with each subsequent selection, promoting a more balanced distribution of participant selection and reducing the likelihood of over-reliance on a small subset of participants. Therefore, the selection of a participant who has already been chosen is determined by

$$P(\Omega_n, T) = \exp\left(-\frac{\Omega_n}{T}\right) \tag{6.8}$$

where Ω_n denotes the frequency with which participant n has been selected in previous rounds, and T serves as the temperature hyperparameter. Notably, the likelihood of selecting a participant is elevated during the early rounds when the temperature parameter is high. As much as a participant is selected, the probability of being selected again diminishes, thereby fostering a balanced distribution of participant selection over multiple rounds.

The pseudo-code for FedSBS is presented in Algorithm 4. The pseudo-code requires input parameters, including a temperature T for computing the probability of re-selecting a participant, a minimum epsilon value ϵ_{min}, a set of participants S, and the number of aggregation rounds (*epochs*), and the number of selected participants s. Its output is the global model at the t^{th} aggregation round. Lines (1 - 4) are assigned to initialize specific variables. The variable ϵ is pre-set to one for subsequent utilization in the epsilon-greedy algorithm, while ϑ is defined as the decay coefficient for the epsilon value. Concurrently, the score variable I is instantiated at zero for all participants because the score computation is unfeasible during the first aggregation round. The variable Ω monitors and quantifies the selection frequency for each participant within the training process. It is critical to underscore that the participant's selection frequency, as quantified by Ω, inversely influences the likelihood of their future selection. This built-in safeguard ensures a balanced participant representation by limiting the dominance of frequently selected participants in the training process. The `INIT` function initializes the Ω and I sets, each containing the participant's ID and the value 0. Lines (5-21) illustrate the epsilon greedy algorithm.

In Line 7, the code generates a random value between 0 to 1. If this value falls below ϵ, it selects a random participant. Since ϵ starts with the value 1 (Line 1), the initial selection always occurs randomly. The code then utilizes the `SelectedParticipant` function (Line 8) to select a participant randomly, and this function, described in Algorithm 5,

Algorithm 4: Federated Score-Based Selection pseudo-code.

Input: T, ϵ_{min}, N, *epochs*, s
Output: w_G^t

```
1  ϵ ← 1
2  η ← ∛ϵ_min
3  I ←INIT()
4  Ω ←INIT()
5  for t < epochs do
6      for i < s do
7          if unif(0,1) < ϵ then
8              n ← SelectParticipant(T, Ω, "random" , N, I)
9              Ω_n ← Ω_n + 1
10             S^t.insert(n)
11         end
12         else
13             n ← SelectParticipant(T, Ω, "greedy" , N, I)
14             Ω_n ← Ω_n + 1
15             S^t.insert(n)
16         end
17         i ← i + 1
18     end
19     for each participant n ∈ S^t do
20         w_n ← LocalUpdate(n, w_G)
21         I_n ← − ln(L(w_G)) + φ_n ln(L_n(w_n))
22     end
23     w_G^t ← Σ_{n=1}^{N} (D_n/D) w_n
24     ϵ ← ϵ × η
25 end
26 return w_G
```

determines the probability of re-selecting a participant. Following this, the code updates the selection count stored in Ω (Line 9) and adds participant n to the selected participants set S^t (Line 10).

If the random value generated in Line 6 surpasses ϵ, the code selects the best participant based on our participant scoring method I, using the `SelectParticipant` function (Line 13). It also updates Ω and includes participant n in the selected participant set S^t (Lines 14 and 15). The epsilon greedy algorithm provides a trade-off between exploring new participants and exploiting the best participants selected in previous rounds based on the score.

Each participant performs a local update and sends the local model and its local loss to the server in Lines 19 and 20. In line 21, the code updates each participant's score

I according to our participant scoring method. Finally, the server performs the global aggregation in line 23 and decays the epsilon in line 24.

Algorithm 5: Participant selection function pseudo-code.

```
   Input: T, Ω, selection_type, N, I
   Output: n
 1 if selection_type = "random" then
 2 |  n ← RANDOM(N)
 3 end
 4 else
 5 |  n ← MAX(I)
 6 end
 7 b ← 1
 8 while b = 1 do
 9 |  P(Ω_n, T) = exp(−Ω_n / T)
10 |  if unif(0, 1) < P(Ω_n, T) then
11 |  |  b ← 0
12 |  end
13 |  else
14 |  |  if selection_type = "random" then
15 |  |  |  n ← RANDOM(N)
16 |  |  end
17 |  |  else
18 |  |  |  n ← MAX(I)
19 |  |  end
20 |  end
21 end
22 return n
```

Algortihm 5 describes the function `SelectParticipant` that determines the probability of re-selecting a participant during the federated learning process and then selects a participant. The pseudo-code requires input parameters such as temperature T, and Ω, which counts the number of times each participant has been selected. It also receives a string value, *selection_type*, which determines if the selection should be greedy or random. Another input parameter is the participant set N, which contains all participants. The final input, denoted as I is a list that archives the scores of the participants. Lines 1-6 discern between a greedy (Line 5) and a random (Line 2) selection and execute the selection accordingly. The variable b, instantiated in Line 7, sets the duration for which the algorithm will attempt to select a non-blocked participant within the while loop. In Line 9, the algorithm calculates the probability of accepting a selected participant, a value between 0 and 1. Note that a never-before-selected participant will always have a selection probability of 100%, according to the equation. The algorithm then generates a random

number between 0 and 1. If the selection probability surpasses this random number (Line 10), the algorithm selects the participant, resets variable b to 0 (Line 11), and breaks the while loop. If the selection probability is less than the random number, the participant is considered blocked, and the algorithm repeats the selection process, randomly choosing another participant (Line 15) or greedily (Line 18) and reassessing the probability.

In summary, FedSBS presents a robust and adaptive method for participant selection within federated learning environments. This is evidenced by our selection strategy, which demonstrates the potential to enhance global model performance. The approach is further characterized by its adaptability, as evidenced by the dynamic scoring of participants upon each selection, ensuring continuous alignment with evolving model requirements. Having laid the theoretical groundwork for FedSBS, the next section will provide an empirical evaluation of our method.

6.3 FedSBS: Evaluation and Results

In this section, we evaluate the proposed FedSBS in network intrusion detection use case. The main goal of the evaluation is to compare our participant selection method FedSBS with state-of-the-art aggregation algorithms and participant selection methods. Another purpose of the evaluation is to compare the performance of FedSBS with state-of-the-art participant selection method in an environment with malicious participants. We compare FedSBS with the state-of-the-art aggregation algorithm to demonstrate the effectiveness of FedSBS in addressing the statistical challenge in FL. We aim to emphasize the significance of the participant selection method in mitigating the statistical challenge, as the new aggregation algorithm has been put forth as a solution to this challenge. In the evaluations, we intentionally excluded the use of FedSA to ensure that the observed contributions and performance solely derive from FedSBS.

We have developed a Python-based simulator[1] using the CICIDS2017 dataset [135]. The CICIDS2017 [135] is an open-source network dataset created by the Canadian Institute for Cybersecurity (CIC). We use PyTorch to implement our participant selection method and the baselines. Our machine learning model consists of a multilayer perceptron with two hidden layers. The first hidden layer comprises 50 neurons, while the second contains 100 neurons. Our simulator ensures that the dataset is split into three parts with an equal balance of classes - 80% of attack traffic labels and 30% of normal traffic labels.

[1] Available at https://github.com/helioncneto/FederatedLearningSimulator.

The first part of the dataset is used for training, the second, for validation, and the third, for testing. The training set is used to train the machine learning models. The validation set is a separate portion of the dataset not used to train the model. It provides an unbiased evaluation of the model during the aggregation rounds. The test set is used to assess the performance of the model after its finalization. The training, validation, and testing sets comprise 90%, 5%, and 5% samples. The simulator applies the Dirichlet distribution to divide the training set for each participant to simulate Non-IID data [148, 149, 21]. We selected the Dirichlet distribution to distribute data among participants due to its flexibility in generating a range of participant data profiles [148], from highly similar to highly distinct, effectively capturing the diverse and non-uniform class distributions typical in real-world FL scenarios.

We use Dirichlet distribution to simulate Non-IID data. The Dirichlet distribution is a multivariate distribution that generates a probability distribution over a set of k categories, where k is a positive integer. Each draw from the Dirichlet distribution produces a probability vector, *i.e.*, a vector of k probabilities that sum to 1, where the values in the vector represent the probabilities of each category. We generate multiple independent samples from the Dirichlet distribution with different sets of parameters, representing different underlying distributions to generate Non-IID data. The resulting probability vectors will differ for each observation, reflecting the different underlying distributions. To replicate data heterogeneity characteristic of non-IID data, we utilize a Dirichlet distribution with parameters 0.3, 0.6 to sample the proportions of labels, as inspired by the methodology outlined in [148].

In our simulations, each participant holds a different amount of data, and the simulator randomly selects the amount for each participant. It is important to highlight that we run each evaluation for 100 aggregation rounds because the global model performance remains relatively stable after 90 rounds of aggregation. We conduct multiple runs for each evaluation to ascertain statistical significance. We calculated the mean alongside a 95% confidence interval. Each run utilized a distinct data distribution among the participants. For each evaluation, we computed the mean and standard deviation to assess the variance in the results. For each aggregation round, we select 30% of participants for the aggregation in every evaluation. Our previous work found that selecting more participants does not significantly impact the global model performance [16].

We evaluate FedSBS in comparison to three aggregation algorithms, Federated Dynamic Regularization (FedDyn) [150], FedAGM [21], and Slow Momentum (SlowMo) [151].

Additionally, we compare it with FedAvg [57], the first aggregation algorithm. The objective is to conduct an evaluation of our participant selection mechanism alongside state-of-the-art aggregation algorithms, thereby underscoring the critical role that participant selection plays in optimizing the overall FL performance. Furthermore, we compare FedSBS with two other participant selection methods, Oort [32] and Wang *et al.* [152]. The idea is to compare our participant selection mechanism with state-of-the-art participant selection mechanisms. Moreover, we evaluate our participant selection method in an environment with malicious participants, using the participant selection methods mentioned above as baselines. The attacker model is described in Section 6.3.1.

6.3.1 Attacker Model

The attacker performs a byzantine attack, using random data points to harm the training and attempting to disrupt the training process [19, 98]. Byzantine attacks in federated learning occur when malicious participants, or Byzantine nodes, inject random data into the system to cause the collapse of the global model. These attacks can occur when the malicious nodes are part of the overall learning system. These attacks can significantly impact the integrity and performance of the global model.

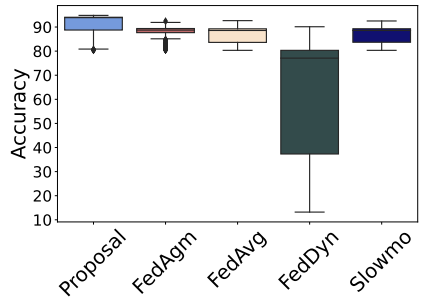
The attacker is a participant in the federated learning environment and uses this access to inject fake data points into the local dataset. These fake data points can be crafted in such a way as to bias the training results and cause inaccurate or undesirable outcomes. In our scenario, we consider three distinct types of malicious participants: I) constant malicious participant, II) p-malicious participant, and III) aggregation k malicious participant. The *constant malicious participant* is a type of malicious participant that behaves maliciously throughout the training process. The *p-malicious participant* utilizes their actual data but also may use fake data with a probability p that follows a Boltzmann distribution. The adoption of a Boltzmann distribution in characterizing the behaviour of the p-malicious participant is predicated on the need for a stochastic model that captures the inherent variability and unpredictability of adversarial actions within the federated learning ecosystem. Moreover, the distribution's parameters can be tuned to reflect the fluctuating intensity of the attacks, thereby offering a versatile and realistic model of the attacker's profile. Finally, the *aggregation k malicious participant* maintains honest behaviour until it reaches aggregation round k, at which point it begins using fake data.

6.3.2 Evaluation of Novel Aggregation Algorithm

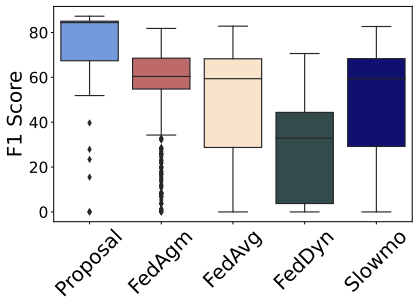
We first evaluate FedSBS using state-of-the-art aggregation algorithms as the baseline. This evaluation compares the performance of FedSBS, a participant selection method, with those aggregation algorithms. Both approaches are intended to enhance the performance of the global model in FL, but they do so with different approaches. The state-of-the-art aggregation algorithms we use as baseline are FedAGM [21], FedDyn [150], SlowMo [151], and FedAvg [10] (the first aggregation algorithm). In our evaluation, we included FedAvg as a representative of simpler aggregation algorithms, encompassing approaches like Federated Stochastic Gradient Descent (FedSGD). FedAGM [21] is a federated learning aggregation algorithm that enhances stability and convergence by providing participants with an accelerated model, informed by global gradients, to guide local updates, effectively reducing bias and improving update consistency. FedDyn [150] optimizes FL training by orchestrating cooperation among a random subset of devices each round, employing dynamic regularizers to align local and global solutions, thereby enhancing training efficiency and robustness across diverse device environments in the face of device heterogeneity, partial participation, and unbalanced data. SlowMo [151] enhances the generalization performance of the global model by synchronizing participants periodically for a momentum update after several aggregation rounds of a base optimizer, thereby achieving efficient training with theoretical convergence guarantees, even in non-convex scenarios.

Subsequently, we evaluate the accuracy, precision, sensitivity, specificity, and F1-score for the validation set in each aggregation round and for the test set at the end of collaborative training. Precision evaluates how many attack samples the model predicted correctly. Sensitivity, or Recall, measures the proportion of actual attack cases predicted as an attack by the model. On the other hand, specificity measures the proportion of actual normal traffic predicted as normal traffic by the model. The F1 score combines precision and sensitivity by taking their harmonic mean. In the IDS scenario, a high F1 score would indicate that the model accurately detects attack traffic (high precision) while also detecting most of the attack traffic (high sensitivity). In these evaluations, it is pertinent to note that we did not include any malicious participants to focus solely on assessing the performance of the methods.

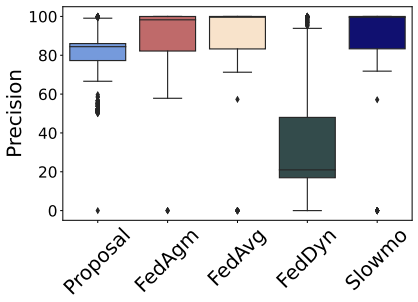
Figure 6.2 presents an analysis of the FedSBS against novel aggregation algorithms and the FedAvg for detecting attacks on network traffic. We used boxplots to represent the accuracy, F1 score, precision, sensitivity, and specificity scores. We have many baselines,



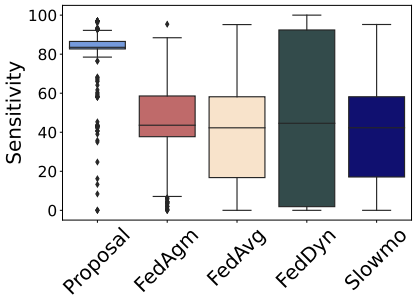
(a) Accuracy in the validation set.



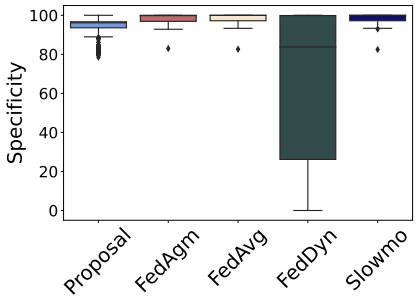
(b) F1 score in the validation set.



(c) Precision in the validation set.



(d) Sensitivity in the validation set.



(e) Specificity in the validation set.

Figure 6.2: The distribution of metrics on the validation dataset. Our proposed method demonstrates less variation and achieves over 80% accuracy in detecting attacks and normal traffic. In contrast, the baseline algorithms better detect normal traffic than attack traffic.

and a line plot would have been unfeasible due to its cluttering effect. While using boxplots means that we lose the temporal relationship between the different baselines, it is a better approach to represent the performance of the algorithms clearly and understandably.

Our proposed FedSBS achieves an average accuracy of 91%, higher than all the other baselines. Our algorithm also varied less than all the other baselines except for FedAGM. Interestingly, we observed that FedDyn has the most variation in every evaluation. It is important to emphasize that throughout the training, as assessed using the validation set, the variation in the F1 score of our proposal is less pronounced than the aggregation

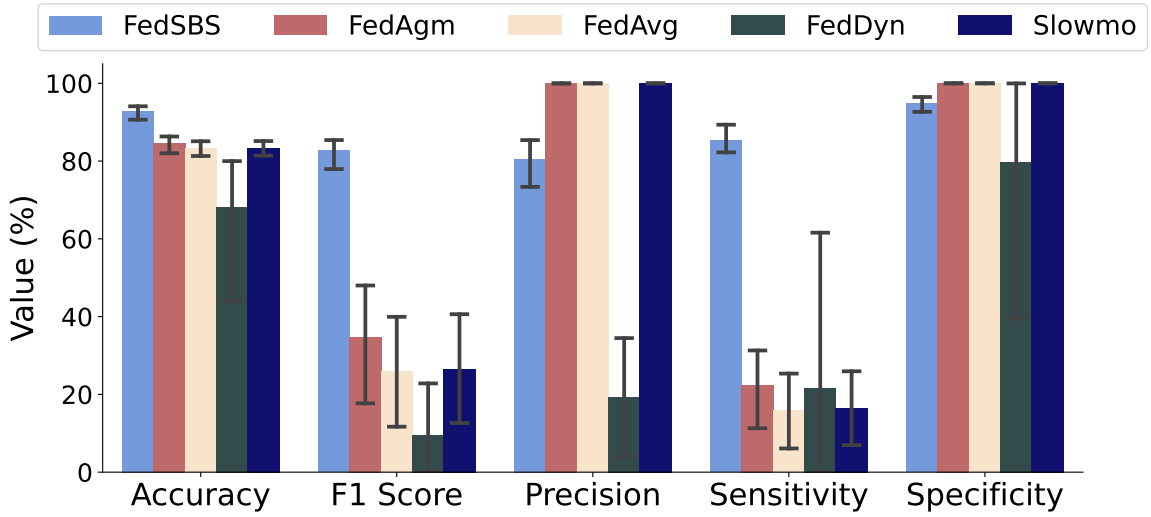


Figure 6.3: Performance metrics for the test set, comparing FedSBS participant selection with aggregation algorithms. FedSBS demonstrates the highest accuracy and F1-score, 92.8%, and 82.7%, respectively, with a low error bar compared to the aggregation algorithms.

algorithms under evaluation. Our participant selection aims to select participants with more contributions, which is crucial to achieving this performance. It is important to highlight that the participant selection of the baseline algorithm is random.

While our precision is smaller than the other baselines, this was expected since we classified more samples as attack samples, leading to more false positives. However, it is essential to note that our participant selection method also had more true positives than the other baselines. Finally, FedSBS could have been better than the baseline in terms of specificity because the baseline algorithms failed to detect attacks effectively, classifying almost everything as normal traffic.

The accuracy of the baseline algorithms is good because the validation set has more normal traffic than attack traffic. While the baselines effectively detect normal traffic, they perform poorly in detecting attacks, a critical objective of any intrusion detection system.

Besides evaluating the validation set, we also evaluate the final global model using the test set. The test set is an important part of the machine learning process as it provides a way to evaluate the performance of a trained model. The test set contains data samples the model has not seen during training and validation. Its performance on this unseen data is used to measure the generalization ability of the global model. It helps ensure the model has learned patterns applicable to unseen data and not just memorized the training and validation data.

Figure 6.3 compares FedSBS against several state-of-the-art aggregation algorithms across crucial performance metrics in the test set. The results are presented as mean values and a confidence interval of 95%, calculated over multiple runs for each method. Our proposed solution demonstrates superior performance across several metrics. In terms of accuracy, our method achieves a mean of 92.89% with a low error bar compared to all baseline approaches. FedSBS also achieves the highest F1 score with a mean of 82.7% with a low error bar compared to all baseline approaches, indicating a solid balance between precision and recall. While the precision of our method (80.53%) is lower compared to some of the baselines, FedSBS significantly outperforms them in terms of sensitivity, with a mean of 85.32%. This highlights that our method better identifies attack traffic, reducing the false-negative rate. Lastly, FedSBS achieves a mean specificity of 94.75%, which is competitive with the baseline approaches. Our proposed method outperforms the baseline approaches across multiple performance metrics. It consistently exhibits lower error bar, indicating the robustness and reliability of our participant selection for the statistical challenge.

In conclusion, evaluating our participant selection method demonstrates the significant impact of an effective participant selection strategy on addressing the federated optimization inherent in federated learning. Our participant selection method outperforms the baseline approach (FedAvg) and the novel aggregation algorithms, emphasizing the importance of participant selection for FL. The results highlight the robustness and reliability of our participant selection method in tackling the complexities of unbalanced datasets.

In conclusion, evaluating our participant selection method against state-of-the-art aggregation algorithms demonstrates the significant impact of an effective participant selection strategy on addressing the federated optimization inherent in federated learning. Our participant selection method outperforms the baseline approach (FedAvg) and the novel aggregation algorithms, emphasizing the importance of participant selection for FL. The results highlight the robustness and reliability of our participant selection method in tackling the complexities of unbalanced datasets.

6.3.3 Evaluation of Participant Selection Methods

In this subsection, we examine the performance of FedSBS in comparison with two state-of-the-art baselines, Wang *et al.* [152] and Oort [32]. The evaluation aims to provide a comprehensive analysis of the effectiveness of our proposal, particularly concerning unbal-

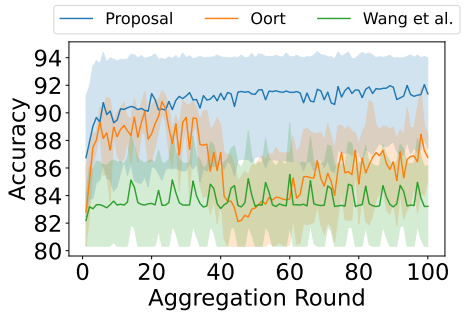
anced datasets. By assessing key metrics such as accuracy, F1 score, precision, sensitivity, and specificity over multiple aggregation rounds, we aim to demonstrate the effectiveness of our proposal in addressing the challenges posed by unbalanced data, highlighting its potential for real-world applications in IDS.

Figure 6.4 compares our proposal and the baseline participant selection methods over 100 aggregation rounds, demonstrating the superior performance of our proposal in terms of F1 score and sensitivity. These metrics are crucial for unbalanced datasets, such as those typically encountered in IDS. F1 score and sensitivity ensure that minority classes, in this case, attack traffic, are not overlooked, providing a balanced evaluation of precision and recall. A high F1 score is crucial for an effective IDS, as it must be adept at detecting and predicting attacks to maintain a secure network environment.

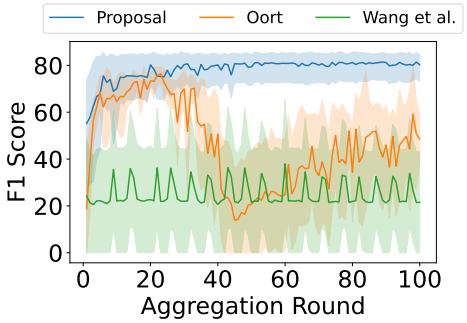
Although FedSBS exhibits lower precision and specificity due to its more assertive approach in classifying samples as attacks, resulting in a higher false-positive rate, it still surpasses the baselines regarding f1 score and sensitivity. Notably, Oort demonstrates higher precision but exhibits greater variability than FedSBS, as evidenced by the confidence interval. Furthermore, FedSBS is the only method whose confidence interval does not reach 0. This demonstrates that FedSBS has a better balance between detecting genuine attacks and minimizing false alarms, a crucial consideration in maintaining an efficient and reliable IDS.

In contrast, the baselines exhibit higher specificity due to their tendency to classify a larger proportion of samples as normal traffic, thereby revealing a bias toward the more populated class. While this approach may result in fewer false positives, it also increases the likelihood of missed attacks, leading to potentially severe consequences regarding network security.

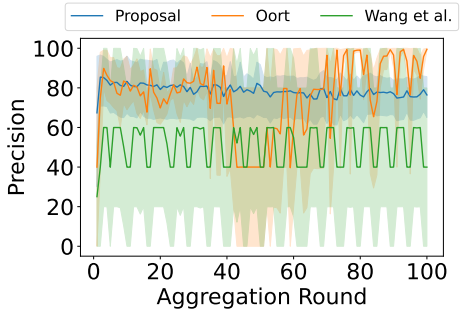
Figure 6.5 compares the performance metrics for three methods — FedSBS, Oort [32], and Wang *et al.* [152] in the test set. FedSBS demonstrates superior accuracy, F1 score, precision, and sensitivity performance compared to the other two approaches. It indicates that our proposed method provides a more balanced and reliable classification across classes, making it a potentially better choice for unbalanced problems. On the other hand, both Oort [32] and Wang *et al.* [152] methods achieve high specificity values. This high specificity can be attributed to their inclination to classify samples favoring the predominant class. While this may result in an better performance in identifying true negatives, it also suggests that these methods struggle to identify positive samples accurately. Consequently, the high Specificity values for Oort [32] and Wang *et al.* [152] should



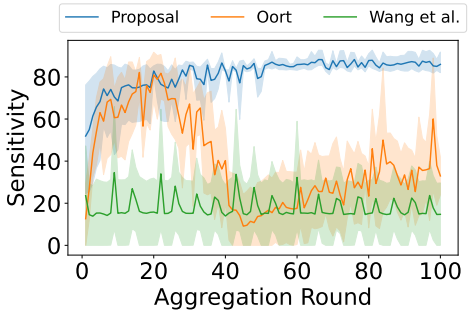
(a) Accuracy in the validation set.



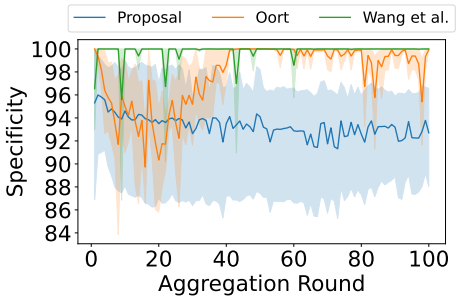
(b) F1 score in the validation set.



(c) Precision in the validation set.



(d) Sensitivity in the validation set.



(e) Specificity in the validation set.

Figure 6.4: Performance of FedSBS and baseline participant selection methods across aggregation rounds. FedSBS has a superior F1 score and sensitivity. FedSBS exhibits lower precision and specificity due to a more assertive approach in classifying samples as attacks, leading to a higher false-positive rate. In contrast, the baselines maintain higher specificity by leaning toward classifying a larger proportion of samples as normal traffic, thus revealing a bias toward the more populated class.

be interpreted cautiously, as their overall performance may be sub-optimal in applications where with a predominant class, such as IDS scenario. It is important to highlight that the performance of baseline methods is considerably worse compared to the results obtained from the validation data.

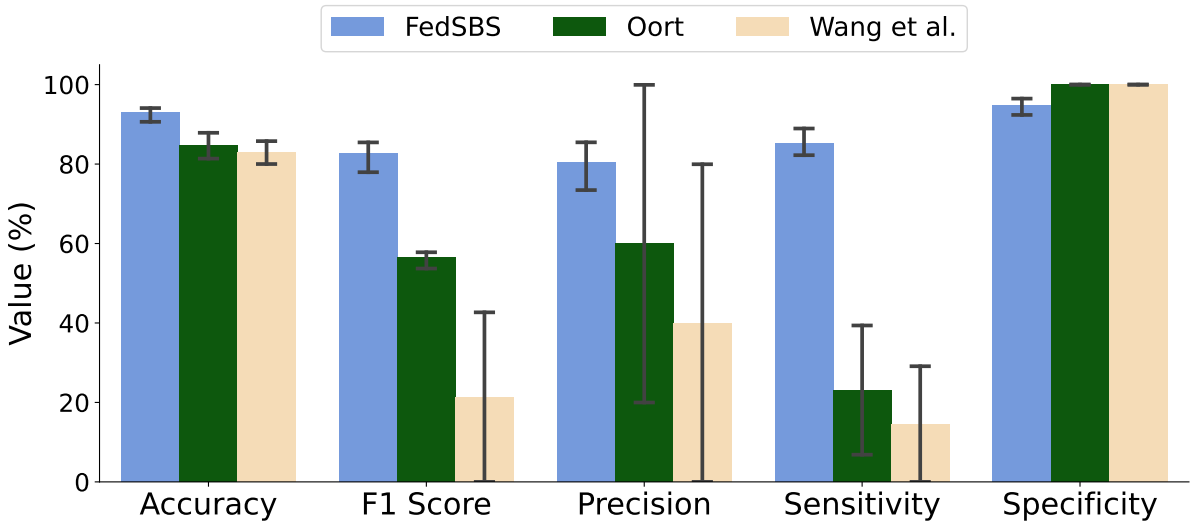


Figure 6.5: Comparison of the performance metrics for FedSBS, Oort, and Wang *et al.* methods. The proposal method outperforms the baseline methods regarding accuracy, F1-Score, precision, and sensitivity. Oort and Wang *et al.* achieve high specificity, which can be attributed to their tendency to classify samples in favor of the predominant class.

6.3.4 Evaluation with Malicious Participants

We present the evaluation of participant selection methods with malicious participants. We evaluate the performance of FedSBS and compare it against two baselines — Oort and Wang *et al.* — in the presence of malicious participants. The key performance metrics examined in this evaluation include accuracy, F1 score, precision, sensitivity, and specificity.

To evaluate the robustness of our method in the face of adversarial activities, we examine two distinct ratios of malicious to benign participants: 20% and 60%. This evaluation is designed to demonstrate the resilience and efficacy of the FedSBS approach, as well as that of baseline methods, across varied adversarial landscapes. A lower rate of 20% was chosen to simulate a more common, real-world scenario where the majority of the network participants are benign. This rate provides a more conservative estimate and allows us to assess how well our methods perform when participants collude to attack the global model. On the other end, a 60% malicious rate represents an extreme case that pushes the robustness of the federated system to its limits. While exceeding 50% may be considered improbable in many real-world applications, we argue that it serves as a valuable stress test. This extreme scenario provides critical insights into the upper bounds of our method ability to withstand adversarial activity. To this end, we consider three specific scenarios:

1. Balanced Malicious Profiles: In this scenario, three malicious participant profiles are evenly distributed. The malicious-p participants are adversarial with a 50% probability, whereas the aggregation k malicious participants become adversarial only after the 50^{th} aggregation round.

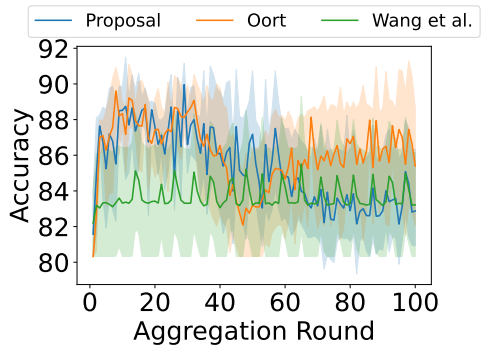
2. Variable Probability Malicious Participants: We evaluate solely the malicious-p participants. We vary the probability p of a participant being malicious to quantify the level of disruption that varying levels of malicious activity can introduce into the system.

3. Variable Aggregation Round Malicious Participants: We concentrate on aggregation k malicious participants. The variable k is adjusted to assess the effect of malicious activity at different points in the aggregation process.

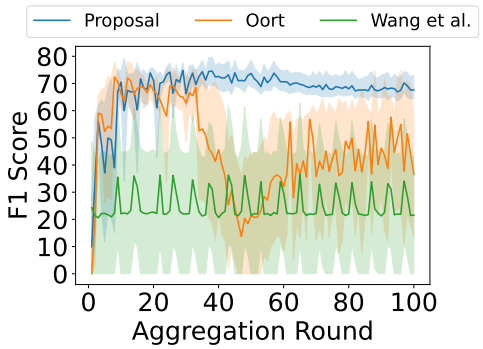
In the balanced malicious profiles scenario, the probability of a malicious participant behaving maliciously is set at 50%, thereby establishing an equilibrium in which the propensity for malicious activity is counterbalanced by an equivalent likelihood of benign participation, creating an environment of uncertainty. It allows us to test the resilience of FedSBS and baseline methods under conditions where the malicious participants are equally likely to be either benign or malicious participants. This evaluation will provide insights into each method, ultimately guiding the selection of the most appropriate approach for ensuring reliable and secure participant selection in the presence of malicious actors. It is a complex scenario for assessing whether our methods can distinguish between honest and adversarial participants when the odds are evenly split. Nonetheless, in the balanced malicious profiles scenario, our choice of the participant being malicious after 50^{th} round serves two purposes. First, it provides ample time for the aggregation process to stabilize, offering a more clear-cut assessment of the impact of adversarial activity. Second, it reflects a more realistic attack model where adversaries might wait to gain trust or accumulate sufficient scores before acting maliciously. By choosing a later aggregation round, we aim to assess the robustness of our methods in a longer-term operational setting.

For the Balanced Malicious Profiles evaluation, Figures 6.6 and 6.7 compare FedSBS, Oort and Wang *et al.*, evaluating their performance across five critical metrics: accuracy, F1 score, precision, sensitivity, and specificity. A key feature of FedSBS is the participant blocker, which mitigates malicious participants impact on the collaborative training process. The participant blocker contributes to the stable training and superior effectiveness

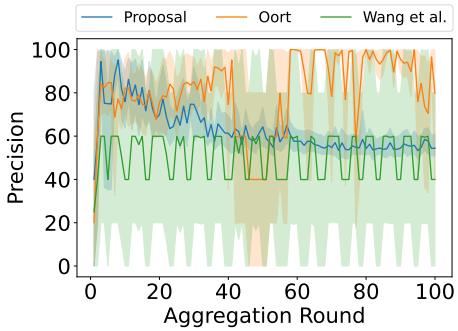
of our model, even in scenarios with a significant presence of malicious participants (20% in Figure 6.6 and 60% in Figure 6.7).



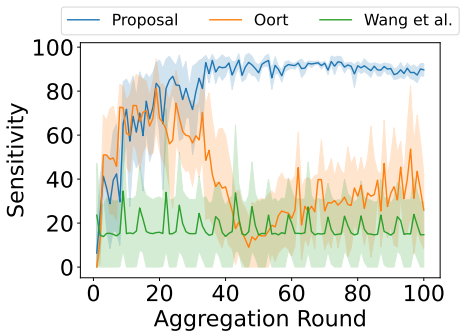
(a) Accuracy in the validation set.



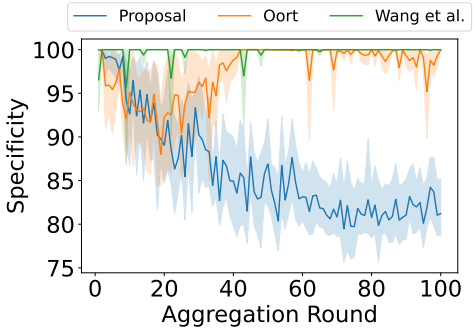
(b) F1 score in the validation set.



(c) Precision in the validation set.



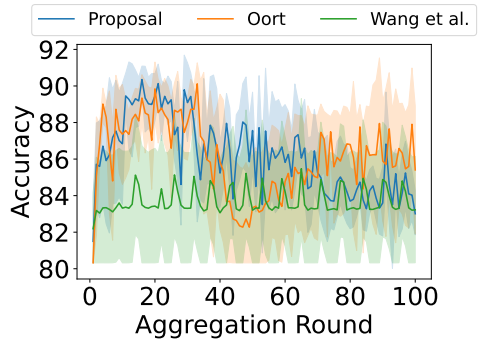
(d) Sensitivity in the validation set.



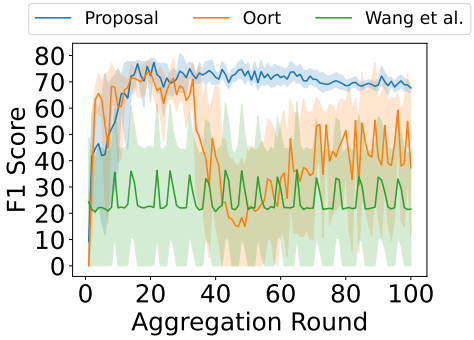
(e) Specificity in the validation set.

Figure 6.6: Performance comparison of FedSBS, Oort, and Wang *et al.*, considering the Balanced Malicious Profile with 20% of the malicious participants. Oort achieved the highest accuracy, but its performance is less meaningful due to the unbalanced dataset. FedSBS outperformed the baselines in F1 score and sensitivity, with lower variability, while the baselines excelled in specificity by predominantly classifying samples as normal traffic.

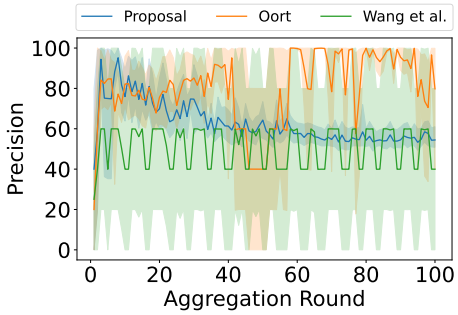
FedSBS provides a trade-off between precision and sensitivity, as evidenced by its outstanding F1 score performance. In contrast, Oort and Wang *et al.* exhibit high variance,



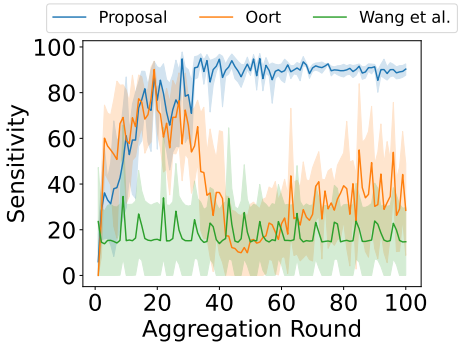
(a) Accuracy in the validation set.



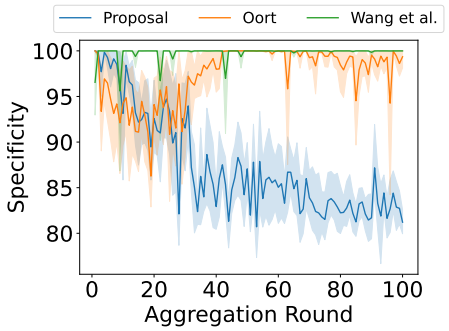
(b) F1 score in the validation set.



(c) Precision in the validation set.



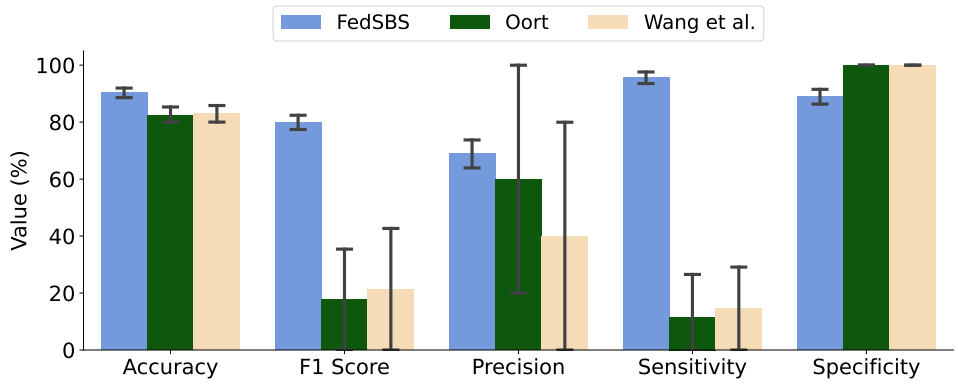
(d) Sensitivity in the validation set.



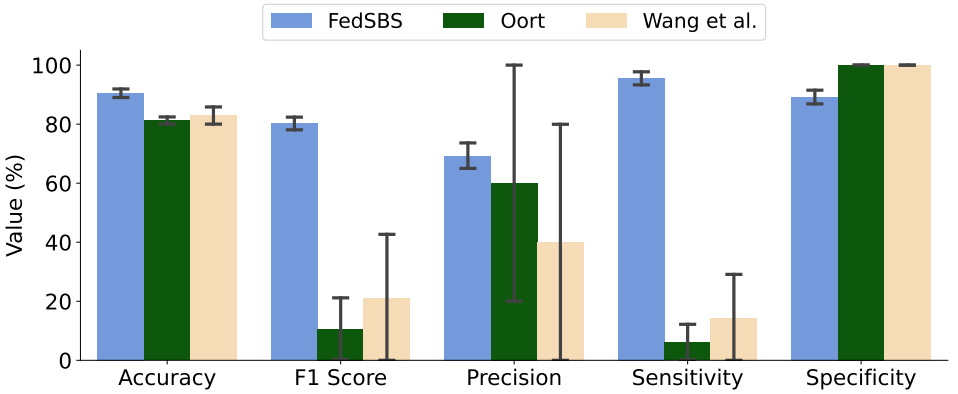
(e) Specificity in the validation set.

Figure 6.7: Line graphs comparing the performance of FedSBSl, Oort, and Wang *et al.*, considering the Balanced Malicious Profiles evaluation, with 60% malicious participants. FedSBS demonstrates superior effectiveness and adaptability, outperforming the baselines in crucial metrics for unbalanced data and maintaining lower variability.

with some models excelling in specificity but struggling in sensitivity. The resilience of FedSBS to the attacker model, as detailed in Section 6.3.1, further highlights its adaptability and robustness in challenging scenarios. By incorporating the participant blocker and our participant selection method, our model effectively avoids malicious participants, resulting in improved performance and stability compared to the baselines.



(a) 20% malicious participant.



(b) 60% malicious participant.

Figure 6.8: Comparison of test set performance metrics for FedSBS, Oort, and Wang *et al.* methods in the Balanced Malicious Profiles evaluation varying proportions of malicious participants (20% and 60%). The figure highlights the robust performance of the FedSBS method across all metrics, demonstrating its resilience even in the presence of adversarial actors.

Figure 6.8 compares the test set performance metrics for FedSBS, Oort, and Wang *et al.*, evaluated under different proportions of malicious participants (20% and 60%). Notably, FedSBS method exhibits superior performance across key metrics, particularly in F1-Score, Precision, and Sensitivity. These metrics are critical when dealing with unbalanced datasets, as they provide a more comprehensive understanding of the model's performance in identifying both positive and negative classes.

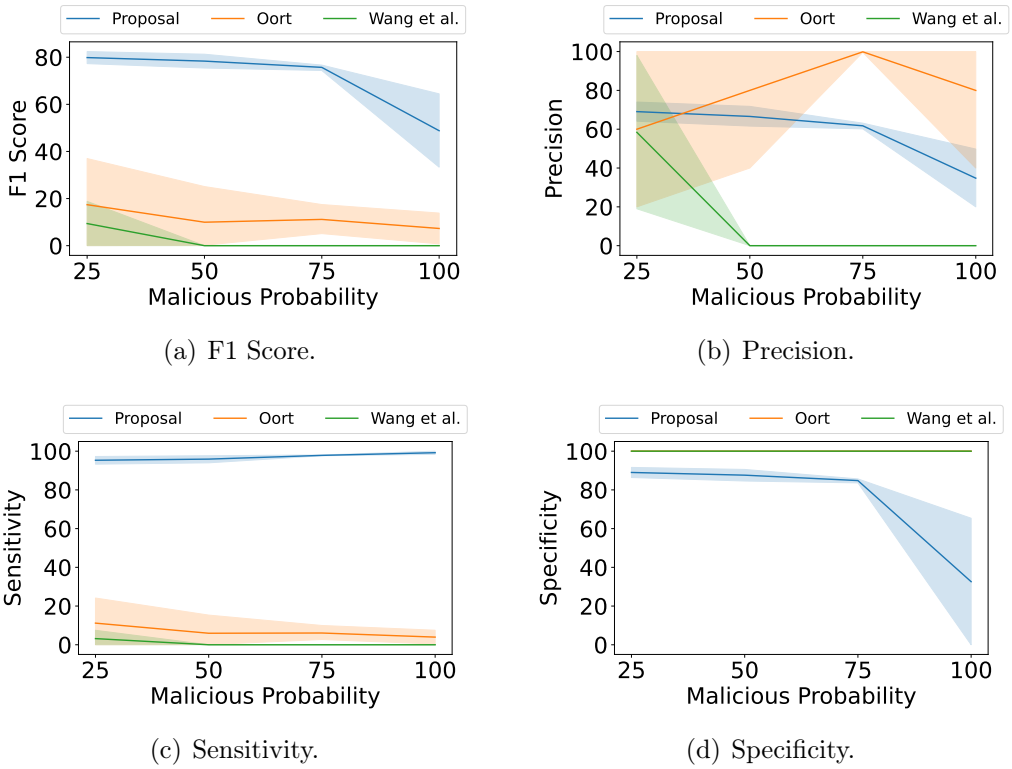
A higher F1-Score indicates that our method maintains a balanced performance in identifying true positives and true negatives, even in the presence of adversarial actors. Precision measures the proportion of true positive predictions among all positive predictions, sensitivity evaluates the algorithm's ability to identify positive instances correctly (attack traffic), and specificity is the ability to identify false instances (normal traffic).

Those metrics play a crucial role in determining the robustness of a method in handling unbalanced datasets because they can express the performance of the model on both classes.

The test set results demonstrate that our method surpasses the baselines in performance in the Balanced Malicious Profiles evaluation, even when evaluated on previously unseen data. When comparing FedSBS to the baselines, it is evident that our approach demonstrates superior performance on the test set. This outstanding performance highlights the effectiveness of our method in addressing the challenges posed by unbalanced datasets and adversarial participation, making it a promising solution for ensuring reliable and secure participant selection in such scenarios.

In subsequent evaluations, we assess the isolated impact of individual attack profiles to comprehend their influence on the holistic training process. This approach contrasts with the '*Balanced Malicious Profiles*' scenario, where the model's performance is evaluated at each aggregation round using a validation set, followed by a final evaluation with the test set. In the following cases, we utilize the test set to evaluate the global model while varying the critical variables associated with the profiles under examination. Specifically, in the '*Variable Probability Malicious Participants*' scenario, we manipulate the probability p of a participant acting maliciously. Conversely, in the 'Variable Aggregation Round Malicious Participants' scenario, we adjust the aggregation round k at which a participant commences malicious behaviour. This methodical variation allows us to observe the model's performance degradation at each alteration stage.

As illustrated in Figure 6.9, FedSBS maintains a slight reduction of performance across all metrics as the malicious probability is increased in the scenario where the malicious participant rate is 20%. When faced with continuous malicious participants — 100% malicious probability, the performance of our proposal and the baselines is significantly compromised. The results indicate that our approach achieved approximately 50% in F1 score, 40% in precision, 98% in sensitivity, and 38% in specificity when participants act maliciously all the time. We observe that when the participant acts maliciously throughout all the training, our system effectively detects attacks, albeit with a high rate of false positives. It is also noteworthy that Oort exhibits enhanced precision compared to FedSBS and Wang *et al.*, especially in malicious probability 75%. The near 100% precision of Oort indicates that the model might be overlooking many other positive instances (attack traffic), indicated by the low sensitivity (recall), resulting in a low F1 score. Notably, our proposal outperformed the baseline F1 score when participants had a



(a) F1 Score.

(b) Precision.

(c) Sensitivity.

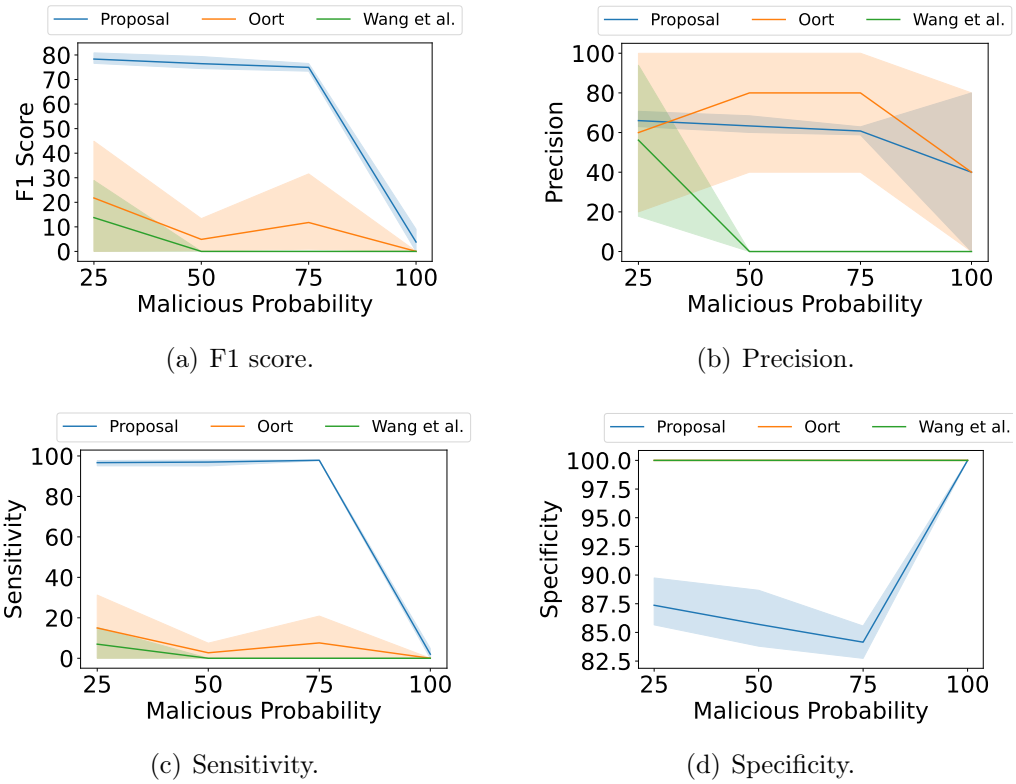
(d) Specificity.

Figure 6.9: An evaluation of Our proposal, Oort, and Wang *et al.* methods across the metrics F1 Score, Sensitivity, Specificity, and Precision in the face of different malicious probability levels with a confidence interval of 95%. In this scenario, 20% of participants are malicious. Collectively, they offer a holistic view of each method's resilience and performance when challenged with adversarial conditions.

probability of up to 75% to behave maliciously. The F1 score is particularly important in the context of collaborative IDS, as our objective is to strike an optimal balance between accurately identifying true positives and concurrently minimizing both false negatives and false positives.

Figure 6.10 shows the performance of our proposed model compared with the state-of-the-art participant selection techniques, Wang *et al.* and Oort, in an environment where 60% of participants are malicious. The precision of our approach achieves around 70%, exhibiting a slight decrement as the malicious probability escalates to 75%, followed by a considerable decline when extending to 100% of malicious probability. Noteworthy, the confidence interval demonstrates that our method has a lower precision variance than Oort, which has superior precision. Moreover, our approach outperforms the benchmark methods concerning the F1 score. Similar to previous evaluations, the baselines report specificity near 100%, indicating a propensity for bias towards the predominant class.

The following evaluations concern the 'Variable Aggregation Round Malicious Partic-



(a) F1 score.

(b) Precision.

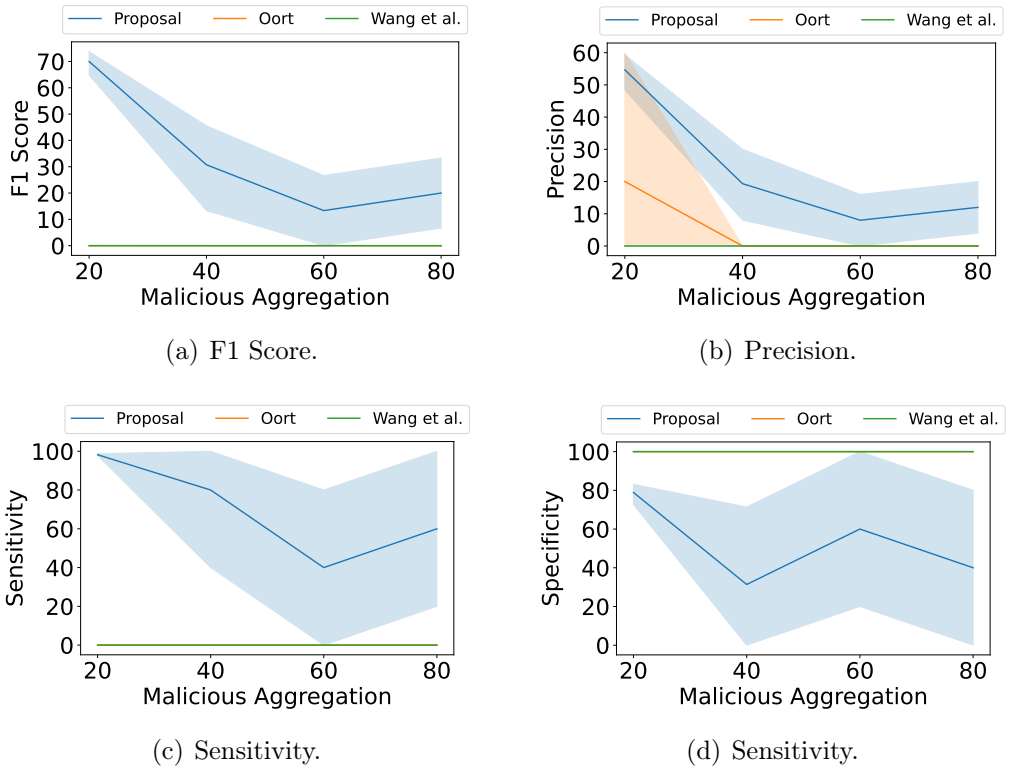
(c) Sensitivity.

(d) Specificity.

Figure 6.10: Comparative analysis of the FedSBS proposal against Oort and Wang *et al.* methods, depicting precision, sensitivity, specificity, and f1 score across varying probabilities of malicious activity. The FedSBS model demonstrates superior f1 score and sensitivity, with specificity up to 80% and precision around 70%. The results indicate its efficacy in accurately detecting and minimizing false negatives in network security applications. In this evaluation, 60% of the participants are malicious.

ipants' scenario. In this scenario, the evaluation exclusively concentrates on participants who commence malicious activities after a predetermined aggregation round. The objective is to evaluate and compare the performance of our proposed method against the baseline methods in mitigating the impact of delayed adversarial behaviour within the method.

Figure 6.11 illustrates the effectiveness of our proposed method in scenarios where malicious participants begin their adversarial actions at different aggregation rounds, set within an environment that includes a 20% proportion of malicious participants. Conversely, if the malicious entities commence their malicious actions after gaining trust, the inflicted damage is more severe. Notably, the F1 score of our proposal stands at 70% when malicious behaviour is initiated post the 20th aggregation. This performance deteriorates drastically to 10% F1 score, with the confidence interval narrowing to zero, when malicious behaviour begins after the 60th aggregation. It indicates that our method can



(a) F1 Score.

(b) Precision.

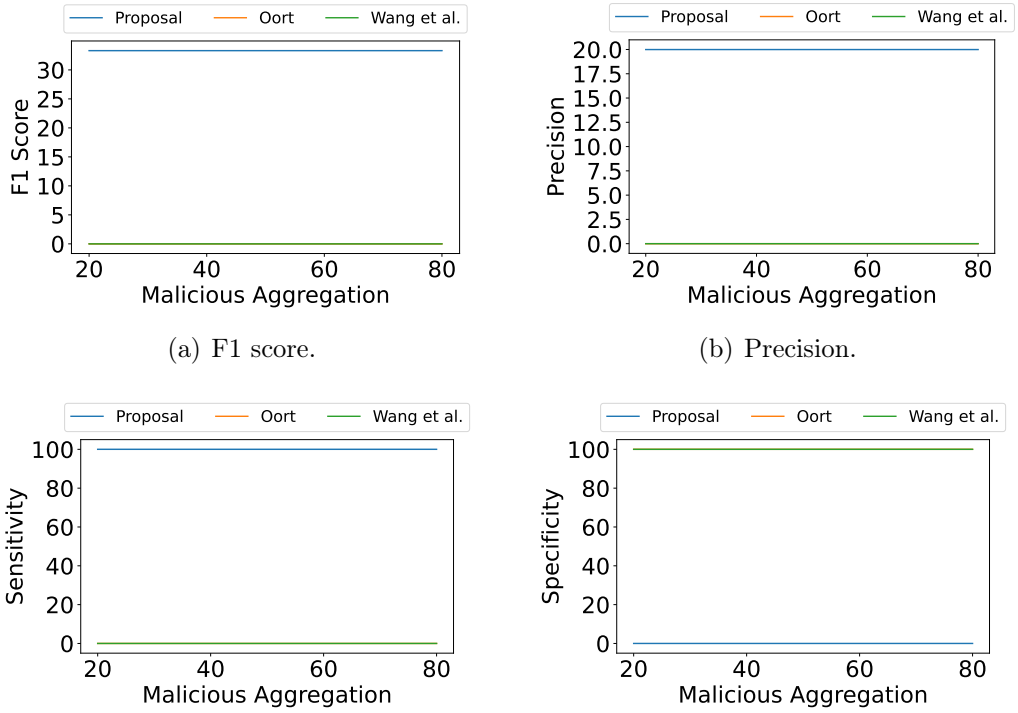
(c) Sensitivity.

(d) Sensitivity.

Figure 6.11: Comparative analysis of Proposal, Oort, and Wang *et al.* varying the aggregation the malicious participants start acting malicious. The Proposal method consistently demonstrates resilience, maintaining higher performance levels relative to the other methods under a scenario 20% of participants are malicious.

avoid selecting malicious participants during the training when they act maliciously at the beginning of the training. Moreover, comparative analysis indicates that our method outperforms the baseline methods, underscoring the robustness of our approach against adversarial participants.

Figure 6.12 presents our proposed method performance compared to baseline methods as a function of the aggregation round in which participants commence malicious behaviour within a context where 60% of participants are malicious. In this scenario, the aggregation round at which participants begin to exhibit malicious behaviour is indifferent to the performance, owing to the predominance of malicious actors within the environment. Consequently, every method exhibits suboptimal performance, with F1 scores below 40%. In our analysis, it is evident that our method tends to classify most of the traffic as an attack, in contrast to Oort and Wang *et al.*, which are inclined to classify most traffic as normal. This observation is substantiated by our method's high sensitivity and low specificity, while Oort and Wang *et al.* exhibit the opposite pattern.



(a) F1 score.

(b) Precision.

(c) Sensitivity.

(d) Specificity.

Figure 6.12: Performance evaluation of Proposal, Oort, and Wang *et al.* methods under a scenario with 60% malicious participants. It is evident that the performance of all methods is compromised irrespective of the onset of adversarial behaviour. Notably, our proposal consistently outperforms the baseline methods in terms of F1 score and precision.

6.3.5 Discussions and Remark

This Chapter proposed a FL training method, incorporating FedSBS participant selection and a global momentum aggregation algorithm, which tackles statistical challenges, particularly in adversarial scenarios. The participant scoring, epsilon-greedy selection, and participant blocking strategies enhance method robustness. Besides, the global momentum term accelerates convergence, resulting in superior overall performance compared to competing state-of-the-art methods. Evaluations, especially in scenarios with malicious participants, demonstrated the proposed method's resilience, achieving notable F1-Score and accuracy even in the presence of malicious participants. Comparisons with state-of-the-art methods showcased our approach's superiority, attaining 92% accuracy and 82% F1-Score in scenarios without malicious participants, while our method achieved over 80% F1-Score and 90% accuracy on the test set in the balanced malicious profile, showcasing its resilience against attacks. The proposed method's robustness, scalability, and performance position it as a promising solution for secure and efficient federated learning systems. Future research may explore advanced privacy-preserving techniques, including

homomorphic encryption and parameter masking, to further enhance the security and confidentiality of FL training processes.

Chapter 7

Conclusion

Federated learning for IDS shows promising results in detecting new attack patterns. However, federated learning has several optimization problems since the aggregation server does not access the data. The random participant selection approach of the traditional federated learning aggregation algorithm FedAvg provides no warranty that the selected participants contribute with high-quality data to optimize the global model. Also, the essential hyperparameters for the federated learning convergence, the number of local updates, and the learning rate have fixed values in the FedAvg algorithm. Furthermore, federated learning indirectly allows the aggregator server to infer participant data.

The thesis proposal presented a metaheuristic and a participant selection method for the federated IDS scenario. Our proposal is not only concerned with using federated learning to detect new attacks but also to overcome optimization issues intrinsic to federated learning. A FL-based IDS with fast convergence provides fast learning of new attack patterns and encourages new participants to contribute.

This thesis has explored and innovated on various aspects of federated learning and IDS, culminating in proposing a novel participant selection method and the FedSA metaheuristic. The challenges and statistical complexities inherent in federated learning environments have been addressed with efficacy, even in the presence of malicious participants.

Our participant selection method leverages a variant of information gain for scoring participants and employs an epsilon-greedy strategy for the selection process. We introduced a participant blocker mechanism to prevent over-selected participants from dominating the learning process and compromising the system's resilience. In tandem with the adoption of a global momentum term to facilitate accelerated convergence, these innovations significantly improved the overall performance of our federated learning sys-

tem. Our proposal evaluation underscored the efficacy of our approach across a diverse range of scenarios, including scenarios with malicious participants. Despite the presence of adversarial actors, constituting 20% and 60% of the participant pool in distinct scenarios, our method successfully attained an f1-score exceeding 80% and an accuracy of 90% on the test dataset, thereby highlighting its robustness. We extended our assessment to scenarios devoid of malicious participants and juxtaposed our methodology against contemporary aggregation algorithms and participant selection techniques. Here, our strategy excelled, registering an impressive accuracy of 92% and an F1-Score of 82%, surpassing baseline performances.

The FedSA we propose streamlines the federated learning workflow, enabling dynamic hyperparameter selection and accelerating the convergence of the global model. It eliminates the necessity for time-consuming fine-tuning, as it adaptively searches for optimal hyperparameters during training. The results underscored the efficiency of FedSA in optimizing the training process, achieving an approximate accuracy of 97% with the CICIDS2017 dataset. By contrast, the baseline FedAVG required nearly double the aggregation rounds to arrive at the same outcome. Furthermore, the evidence revealed that despite selecting a smaller fraction of participants for training, the FedSA method was able to deliver 96% accuracy, significantly outstanding the baseline. Specifically, with FedSA selecting only 27% of participants, it reached 96% accuracy, while FedAvg, even when engaging 50% of participants, required more aggregation rounds to achieve the same accuracy. Thus, our proposed solution reached peak accuracy in a few aggregation rounds, providing a 50% speed boost compared to the traditional approach.

Furthermore, the vulnerability review presented herein contributes to understanding the systemic frailties to which such systems could be exposed. We can design more secure and efficient federated learning systems by addressing these vulnerabilities. In our review of vulnerabilities in FL, we classified attacks into two broad categories: Model Performance Attacks and Data Privacy Attacks. Each attack category was analyzed, and potential solutions were suggested to mitigate these vulnerabilities. We also highlight various research works that aim to address these security challenges. Then, we assess the strengths and weaknesses of the approaches, providing a robust understanding of the current state of security in FL and opening avenues for future research.

In conclusion, our exploration of federated learning and intrusion detection has yielded effective, robust solutions. These solutions open the path to more secure and efficient federated learning systems by overcoming statistical challenges and optimizing the learning

process. Future research will consider enhancing privacy-preserving measures. As delineated in Chapter 4, the potential for information leakage from local model parameters exists, and the aggregation server may adopt the role of an honest but curious entity. This will involve exploring techniques such as homomorphic encryption and parameter masking strategies and tailoring the neighbourhood structure to participants with similar model parameters.

References

[1] PETROLO, R.; LOSCRI, V.; MITTON, N. Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms. *Transactions on emerging telecommunications technologies*, Wiley Online Library, v. 28, n. 1, p. e2931, 2017.

[2] LIM, W. Y. B.; LUONG, N. C.; HOANG, D. T.; JIAO, Y.; LIANG, Y. C.; YANG, Q.; NIYATO, D.; MIAO, C. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*, IEEE, 2020.

[3] ALDWEESH, A.; DERHAB, A.; EMAM, A. Z. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, v. 189, p. 105124, 2020. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705119304897>.

[4] LOPEZ, M. A.; MATTOS, D. M.; DUARTE, O. C. M.; PUJOLLE, G. Toward a monitoring and threat detection system based on stream processing as a virtual network function for big data. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 31, n. 20, p. e5344, 2019.

[5] SANZ, I. J.; MATTOS, D. M. F.; DUARTE, O. C. M. B. Sfcperf: An automatic performance evaluation framework for service function chaining. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. [S.l.: s.n.], 2018. p. 1–9.

[6] LIU, H.; LANG, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *applied sciences*, Multidisciplinary Digital Publishing Institute, v. 9, n. 20, p. 4396, 2019.

[7] de Souza, L. A. C.; Antonio F. Rebello, G.; Camilo, G. F.; Guimarães, L. C. B.; Duarte, O. C. M. B. Dfedforest: Decentralized federated forest. In: *2020 IEEE International Conference on Blockchain (Blockchain)*. [S.l.: s.n.], 2020. p. 90–97.

[8] Silva, J. V. V.; Lopez, M. A.; Mattos, D. M. F. Attackers are not stealthy: Statistical analysis of the well-known and infamous kdd network security dataset. In: *2020 4th Conference on Cloud and Internet of Things (CIoT)*. [S.l.: s.n.], 2020. p. 1–8.

[9] YANG, Q.; LIU, Y.; CHEN, T.; TONG, Y. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, v. 10, n. 2, p. 1–19, 2019. ISSN 21576912.

[10] Brendan McMahan, H.; MOORE, E.; RAMAGE, D.; HAMPSON, S.; Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*. [S.l.: s.n.], 2017. v. 54.

[11] PREUVENEERS, D.; RIMMER, V.; TSINGENOPOULOS, I.; SPOOREN, J.; JOOSEN, W.; ILIE-ZUDOR, E. Chained anomaly detection models for federated learning: An intrusion detection case study. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 8, n. 12, p. 2663, 2018.

[12] Nguyen, T. D.; Marchal, S.; Miettinen, M.; Fereidooni, H.; Asokan, N.; Sadeghi, A. DÏot: A federated self-learning anomaly detection system for iot. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.: s.n.], 2019. p. 756–767.

[13] RAHMAN, S. A.; TOUT, H.; TALHI, C.; MOURAD, A. Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network*, v. 34, n. 6, p. 310–317, 2020.

[14] CHEN, Z.; LV, N.; LIU, P.; FANG, Y.; CHEN, K.; PAN, W. Intrusion detection for wireless edge networks based on federated learning. *IEEE Access*, v. 8, p. 217463–217472, 2020.

[15] WANG, L.; WANG, W.; LI, B. Cmfl: Mitigating communication overhead for federated learning. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.: s.n.], 2019. p. 954–964.

[16] NETO, H. N. C.; DUSPARIC, I.; MATTOS, D. M. F.; FERNANDES, N. C. Fedsa: Accelerating intrusion detection in collaborative environments with federated simulated annealing. In: *Proceedings of the 2022 IEEE Conference on Network Softwarization (NetSoft)*. [S.l.: s.n.], 2022. p. 1–5.

[17] NETO, H. N. C.; MATTOS, D. M.; FERNANDES, N. C. Fedsa: Arrefecimento simulado federado para a aceleração da detecção de intrusão em ambientes colaborativos. In: SBC. *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*. [S.l.], 2021. p. 280–293.

[18] NETO, H. N. C.; FERNANDES, N. C.; MATTOS, D. M. F. Fedsbs: Seleção de participantes baseado em pontuação para aprendizado federado no cenário de detecção de intrusão. *XXIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais-SBSeg*, 2023.

[19] NETO, H. N. C.; HRIBAR, J.; DUSPARIC, I.; MATTOS, D. M. F.; FERNANDES, N. C. A survey on securing federated learning: Analysis of applications, attacks, challenges, and trends. *IEEE Access*, v. 11, p. 41928–41953, 2023.

[20] WANG, S.; TUOR, T.; SALONIDIS, T.; LEUNG, K. K.; MAKAYA, C.; HE, T.; CHAN, K. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications*, v. 37, n. 6, p. 1205–1221, 2019. ISSN 15580008.

[21] KIM, G.; KIM, J.; HAN, B. Communication-efficient federated learning with acceleration of global momentum. *arXiv preprint arXiv:2201.03172*, 2022.

[22] LI, K.; ZHOU, H.; TU, Z.; WANG, W.; ZHANG, H. Distributed network intrusion detection system in satellite-terrestrial integrated networks using federated learning. *IEEE Access*, v. 8, p. 214852–214865, 2020.

[23] ZHAO, R.; YIN, Y.; SHI, Y.; XUE, Z. Intelligent intrusion detection based on federated learning aided long short-term memory. *Physical Communication*, v. 42, p. 101157, 2020. ISSN 1874-4907. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1874490720302342>.

[24] MOTHUKURI, V.; KHARE, P.; PARIZI, R. M.; POURIYEH, S.; DEHGHANTANHA, A.; SRIVASTAVA, G. Federated learning-based anomaly detection for iot security attacks. *IEEE Internet of Things Journal*, p. 1–1, 2021.

[25] REY, V.; SáNCHEZ, P. M. S.; CELDRáN, A. H.; BOVET, G.; JAGGI, M. *Federated Learning for Malware Detection in IoT Devices*. 2021.

[26] NGUYEN, L. T.; KIM, J.; SHIM, B. Gradual federated learning with simulated annealing. *IEEE Transactions on Signal Processing*, v. 69, p. 6299–6313, 2021.

[27] SMITH, V.; CHIANG, C.-K.; SANJABI, M.; TALWALKAR, A. S. Federated multi-task learning. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2017. p. 4424–4434.

[28] CORINZIA, L.; BUHMANN, J. M. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

[29] Chiu, T.; Shih, Y.; Pang, A.; Wang, C.; Weng, W.; Chou, C. Semi-supervised distributed learning with non-iid data for aiot service platform. *IEEE Internet of Things Journal*, p. 1–1, 2020.

[30] HUANG, L.; YIN, Y.; FU, Z.; ZHANG, S.; DENG, H.; LIU, D. Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data. *Plos one*, Public Library of Science San Francisco, CA USA, v. 15, n. 4, p. e0230706, 2020.

[31] SONG, Z.; SUN, H.; YANG, H. H.; WANG, X.; ZHANG, Y.; QUEK, T. Q. Reputation-based federated learning for secure wireless networks. *IEEE Internet of Things Journal*, IEEE, v. 9, n. 2, p. 1212–1226, 2021.

[32] LAI, F.; ZHU, X.; MADHYASTHA, H. V.; CHOWDHURY, M. Oort: Efficient federated learning via guided participant selection. In: *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*. [S.l.: s.n.], 2021. p. 19–35.

[33] Nishio, T.; Yonetani, R. Client selection for federated learning with heterogeneous resources in mobile edge. In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. [S.l.: s.n.], 2019. p. 1–7.

[34] YOSHIDA, N.; NISHIO, T.; MORIKURA, M.; YAMAMOTO, K.; YONETANI, R. Hybrid-fl for wireless networks: Cooperative learning mechanism using non-iid data. In: IEEE. *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. [S.l.], 2020. p. 1–7.

[35] NETO, H. N. C.; LOPEZ, M. A.; FERNANDES, N. C.; MATTOS, D. M. Minecap: super incremental learning for detecting and blocking cryptocurrency mining on software-defined networking. *Annals of Telecommunications*, Springer, p. 1–11, 2020.

[36] ALOM, M. Z.; TAHA, T. M.; YAKOPCIC, C.; WESTBERG, S.; SIDIKE, P.; NASRIN, M. S.; HASAN, M.; ESSEN, B. C. V.; AWWAL, A. A. S.; ASARI, V. K. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, v. 8, n. 3, 2019. ISSN 2079-9292.

[37] MEDEIROS, D. S. V.; NETO, H. N. C.; LOPEZ, M. A.; MAGALHÃES, L. C. S.; FERNANDES, N. C.; VIEIRA, A. B.; SILVA, E. F.; MATTOS, D. M. F. A survey on data analysis on large-scale wireless networks: online stream processing, trends, and challenges. *Journal of Internet Services and Applications*, SpringerOpen, v. 11, n. 1, p. 1–48, 2020.

[38] LI, P.; LI, J.; HUANG, Z.; LI, T.; GAO, C.-Z.; YIU, S.-M.; CHEN, K. Multi-key privacy-preserving deep learning in cloud computing. *Future Generation Computer Systems*, v. 74, p. 76 – 85, 2017. ISSN 0167-739X.

[39] VERMA, P.; SOOD, S. K.; KALRA, S. Cloud-centric iot based student healthcare monitoring framework. *Journal of Ambient Intelligence and Humanized Computing*, Springer, v. 9, p. 1293–1309, 2018.

[40] BUTT, U. A.; MEHMOOD, M.; SHAH, S. B. H.; AMIN, R.; SHAUKAT, M. W.; RAZA, S. M.; SUH, D. Y.; PIRAN, M. J. A review of machine learning algorithms for cloud computing security. *Electronics*, MDPI, v. 9, n. 9, p. 1379, 2020.

[41] ARULANTHU, P.; PERUMAL, E. An intelligent iot with cloud centric medical decision support system for chronic kidney disease prediction. *International Journal of Imaging Systems and Technology*, Wiley Online Library, v. 30, n. 3, p. 815–827, 2020.

[42] Parlamento Europeu e Conselho da União Européia. *Regulamento (UE) 2016/679*. 2016. Https://eur-lex.europa.eu/legal-content/PT/TXT/PDF/?uri=CELEX:32016R0679&from=PT.

[43] OLIVEIRA, M. T. D.; REIS, L. H. A.; VERGINADIS, Y.; MATTOS, D. M. F.; OLABARRIAGA, S. D. Smartaccess: Attribute-based access control system for medical records based on smart contracts. *IEEE Access*, v. 10, p. 117836–117854, 2022.

[44] BRASIL. *Lei nº 13.709, de 14 de agosto de 2018. Institui a Lei Geral de Proteção de Dados Pessoais (LGPD)*. 2018.

[45] Gupta, A.; Jha, R. K. A survey of 5g network: Architecture and emerging technologies. *IEEE Access*, v. 3, p. 1206–1232, 2015.

[46] Chen, M.; Semiari, O.; Saad, W.; Liu, X.; Yin, C. Federated echo state learning for minimizing breaks in presence in wireless virtual reality networks. *IEEE Transactions on Wireless Communications*, v. 19, n. 1, p. 177–191, 2020.

[47] Samarakoon, S.; Bennis, M.; Saad, W.; Debbah, M. Federated learning for ultra-reliable low-latency v2v communications. In: *2018 IEEE Global Communications Conference (GLOBECOM)*. [S.l.: s.n.], 2018. p. 1–7.

[48] BRISIMI, T. S.; CHEN, R.; MELA, T.; OLSHEVSKY, A.; PASCHALIDIS, I. C.; SHI, W. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, v. 112, p. 59 – 67, 2018. ISSN 1386-5056.

[49] WU, Q.; CHEN, X.; ZHOU, Z.; ZHANG, J. Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring. *IEEE Transactions on Mobile Computing*, v. 21, n. 8, p. 2818–2832, 2022.

[50] FREDRIKSON, M.; LANTZ, E.; JHA, S.; LIN, S.; PAGE, D.; RISTENPART, T. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In: *23rd {USENIX} Security Symposium ({USENIX} Security 14)*. [S.l.: s.n.], 2014. p. 17–32.

[51] WANG, H.; SREENIVASAN, K.; RAJPUT, S.; VISHWAKARMA, H.; AGARWAL, S.; SOHN, J.-y.; LEE, K.; PAPAILIOPOULOS, D. Attack of the tails: Yes, you really can backdoor federated learning. In: LAROCHELLE, H.; RANZATO, M.; HADSELL, R.; BALCAN, M.; LIN, H. (Ed.). *Advances in Neural Information Processing Systems*. [S.l.]: Curran Associates, Inc., 2020. v. 33, p. 16070–16084.

[52] ASSEFI, M.; BEHRAVESH, E.; LIU, G.; TAFTI, A. P. Big data machine learning using apache spark mllib. In: *2017 IEEE International Conference on Big Data (Big Data)*. [S.l.: s.n.], 2017. p. 3492–3498.

[53] DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, ACM, v. 51, n. 1, p. 107–113, 2008.

[54] ZAHARIA, M.; XIN, R. S.; WENDELL, P.; DAS, T.; ARMBRUST, M.; DAVE, A.; MENG, X.; ROSEN, J.; VENKATARAMAN, S.; FRANKLIN, M. J. Apache spark: a unified engine for big data processing. *Communications of the ACM*, ACM, v. 59, n. 11, p. 56–65, 2016.

[55] HO, Q.; CIPAR, J.; CUI, H.; LEE, S.; KIM, J. K.; GIBBONS, P. B.; GIBSON, G. A.; GANGER, G.; XING, E. P. More effective distributed ml via a stale synchronous parallel parameter server. In: *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013. p. 1223–1231. Disponível em: <http://papers.nips.cc/paper/4894-more-effective-distributed-ml-via-a-stale-synchronous-parallel-parameter-server.pdf>.

[56] BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010*. Heidelberg: Physica-Verlag HD, 2010. p. 177–186. ISBN 978-3-7908-2604-3.

[57] HARD, A.; RAO, K.; MATHEWS, R.; RAMASWAMY, S.; BEAUFAYS, F.; AUGENSTEIN, S.; EICHNER, H.; KIDDON, C.; RAMAGE, D. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[58] Li, T.; Sahu, A. K.; Talwalkar, A.; Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, v. 37, n. 3, p. 50–60, 2020.

[59] LI, W.; MILLETARÌ, F.; XU, D.; RIEKE, N.; HANCOX, J.; ZHU, W.; BAUST, M.; CHENG, Y.; OURSELIN, S.; CARDOSO, M. J.; FENG, A. Privacy-preserving federated brain tumour segmentation. In: *Machine Learning in Medical Imaging*. Cham: Springer International Publishing, 2019. p. 133–141. ISBN 978-3-030-32692-0.

[60] VERMA, D.; JULIER, S.; CIRINCIONE, G. Federated ai for building ai solutions across multiple agencies. *arXiv preprint arXiv:1809.10036*, 2018.

[61] Phong, L. T.; Aono, Y.; Hayashi, T.; Wang, L.; Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, v. 13, n. 5, p. 1333–1345, 2018.

[62] BONAWITZ, K.; IVANOV, V.; KREUTER, B.; MARCEDONE, A.; MCMAHAN, H. B.; PATEL, S.; RAMAGE, D.; SEGAL, A.; SETH, K. Practical secure aggregation for privacy-preserving machine learning. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2017. (CCS '17), p. 1175–1191. ISBN 9781450349468.

[63] Wenliang Du; Atallah, M. J. Privacy-preserving cooperative statistical analysis. In: *Seventeenth Annual Computer Security Applications Conference*. [S.l.: s.n.], 2001. p. 102–110.

[64] VAIDYA, J.; CLIFTON, C. Privacy preserving association rule mining in vertically partitioned data. In: *KDD '02*. New York, NY, USA: Association for Computing Machinery, 2002. p. 639–644. ISBN 158113567X.

[65] KARR, A. F.; LIN, X.; SANIL, A. P.; REITER, J. P. Privacy-preserving analysis of vertically partitioned data using secure matrix products. *Journal of Official Statistics*, v. 25, n. 1, p. 125, 2009.

[66] DU, W.; HAN, Y. S.; CHEN, S. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In: SIAM. *Proceedings of the 2004 SIAM international conference on data mining*. [S.l.], 2004. p. 222–233.

[67] WAN, L.; NG, W. K.; HAN, S.; LEE, V. C. Privacy-preservation for gradient descent methods. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2007. p. 775–783.

[68] HARDY, S.; HENECKA, W.; IVEY-LAW, H.; NOCK, R.; PATRINI, G.; SMITH, G.; THORNE, B. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[69] NOCK, R.; HARDY, S.; HENECKA, W.; IVEY-LAW, H.; PATRINI, G.; SMITH, G.; THORNE, B. Entity resolution and federated learning get a federated resolution. *arXiv preprint arXiv:1803.04035*, 2018.

[70] LIANG, G.; CHAWATHE, S. S. Privacy-preserving inter-database operations. In: SPRINGER. *International Conference on Intelligence and Security Informatics*. [S.l.], 2004. p. 66–82.

[71] SCANNAPIECO, M.; FIGOTIN, I.; BERTINO, E.; ELMAGARMID, A. K. Privacy preserving schema and data matching. In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. [S.l.: s.n.], 2007. p. 653–664.

[72] GOLDREICH, O.; MICALI, S.; WIGDERSON, A. How to play any mental game, or a completeness theorem for protocols with honest majority. In: _____. *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. New York, NY, USA: Association for Computing Machinery, 2019. p. 307–328. ISBN 9781450372664. Disponível em: <https://doi.org/10.1145/3335741.3335755>.

[73] Pan, S. J.; Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, v. 22, n. 10, p. 1345–1359, 2010.

[74] Wang, X.; Han, Y.; Leung, V. C. M.; Niyato, D.; Yan, X.; Chen, X. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys Tutorials*, v. 22, n. 2, p. 869–904, 2020.

[75] BOGETOFT, P.; CHRISTENSEN, D. L.; DAMG, I. Multiparty Computation Goes Live. *Review Literature And Arts Of The Americas*, p. 1–13, 2009.

[76] VAIDYA, J.; CLIFTON, C.; KANTARCIOGLU, M.; PATTERSON, A. S. Privacy-preserving decision trees over vertically partitioned data. *ACM Trans. Knowl. Discov. Data*, Association for Computing Machinery, New York, NY, USA, v. 2, n. 3, out. 2008. ISSN 1556-4681.

[77] DU, W.; ZHAN, Z. Building decision tree classifier on private data. *Syracuse University - Electrical Engineering and Computer Science*, 2002.

[78] VAIDYA, J.; CLIFTON, C. Privacy-preserving k-means clustering over vertically partitioned data. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery, 2003. (KDD '03), p. 206–215. ISBN 1581137370. Disponível em: <https://doi.org/10.1145/956750.956776>.

[79] VAIDYA, J.; CLIFTON, C. Privacy preserving naive bayes classifier for vertically partitioned data. In: SIAM. *Proceedings of the 2004 SIAM international conference on data mining*. [S.l.], 2004. p. 522–526.

[80] Zhang, J.; Chen, B.; Yu, S.; Deng, H. Pefl: A privacy-enhanced federated learning scheme for big data analytics. In: *2019 IEEE Global Communications Conference (GLOBECOM)*. [S.l.: s.n.], 2019. p. 1–6.

[81] GORYCZKA, S.; XIONG, L. A comprehensive comparison of multiparty secure additions with differential privacy. *IEEE Transactions on Dependable and Secure Computing*, v. 14, n. 5, p. 463–477, 2017.

[82] LI, Y.; ZHOU, Y.; JOLFAEI, A.; YU, D.; XU, G.; ZHENG, X. Privacy-preserving federated learning framework based on chained secure multiparty computing. *IEEE Internet of Things Journal*, v. 8, n. 8, p. 6178–6186, 2021.

[83] BONAWITZ, K.; EICHNER, H.; GRIESKAMP, W.; HUBA, D.; INGERMAN, A.; IVANOV, V.; KIDDON, C.; KONEČNÝ, J.; MAZZOCCHI, S.; MCMAHAN, H. B. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[84] HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016.

[85] Wang, L.; Wang, W.; Li, B. Cmfl: Mitigating communication overhead for federated learning. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. [S.l.: s.n.], 2019. p. 954–964.

[86] KONEČNỲ, J.; MCMAHAN, H. B.; YU, F. X.; RICHTÁRIK, P.; SURESH, A. T.; BACON, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[87] LIU, Y.; KANG, Y.; ZHANG, X.; LI, L.; CHENG, Y.; CHEN, T.; HONG, M.; YANG, Q. A communication efficient vertical federated learning framework. *arXiv preprint arXiv:1912.11187*, 2019.

[88] Yao, X.; Huang, C.; Sun, L. Two-stream federated learning: Reduce the communication costs. In: *2018 IEEE Visual Communications and Image Processing (VCIP)*. [S.l.: s.n.], 2018. p. 1–4.

[89] CALDAS, S.; KONEČNY, J.; MCMAHAN, H. B.; TALWALKAR, A. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[90] TAO, Z.; LI, Q. esgd: Communication efficient distributed deep learning on the edge. In: {*USENIX*} *Workshop on Hot Topics in Edge Computing (HotEdge 18)*. [S.l.: s.n.], 2018.

[91] LIU, Y.; ZHANG, X.; KANG, Y.; LI, L.; CHEN, T.; HONG, M.; YANG, Q. Fedbcd: A communication-efficient collaborative learning framework for distributed features. *IEEE Transactions on Signal Processing*, v. 70, p. 4277–4290, 2022.

[92] STROM, N. Scalable distributed dnn training using commodity gpu cloud computing. In: *Sixteenth Annual Conference of the International Speech Communication Association*. [S.l.: s.n.], 2015.

[93] Kang, J.; Xiong, Z.; Niyato, D.; Xie, S.; Zhang, J. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, v. 6, n. 6, p. 10700–10714, 2019.

[94] SPRAGUE, M. R.; JALALIRAD, A.; SCAVUZZO, M.; CAPOTA, C.; NEUN, M.; DO, L.; KOPP, M. Asynchronous federated learning for geospatial applications. In: *ECML PKDD 2018 Workshops*. [S.l.]: Springer International Publishing, 2019. p. 21–28. ISBN 978-3-030-14880-5.

[95] XIE, C.; KOYEJO, S.; GUPTA, I. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019.

[96] VILALTA, R.; DRISSI, Y. A perspective view and survey of meta-learning. *Artificial intelligence review*, Springer, v. 18, n. 2, p. 77–95, 2002.

[97] CARUANA, R. Multitask learning. *Machine learning*, Springer, v. 28, n. 1, p. 41–75, 1997.

[98] LIU, P.; XU, X.; WANG, W. Threats, attacks and defenses to federated learning: issues, taxonomy and perspectives. *Cybersecurity*, SpringerOpen, v. 5, n. 1, p. 1–19, 2022.

[99] JERE, M. S.; FARNAN, T.; KOUSHANFAR, F. A taxonomy of attacks on federated learning. *IEEE Security & Privacy*, v. 19, n. 2, p. 20–28, 2021.

[100] GOLDBLUM, M.; TSIPRAS, D.; XIE, C.; CHEN, X.; SCHWARZSCHILD, A.; SONG, D.; MADRY, A.; LI, B.; GOLDSTEIN, T. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2022.

[101] FUNG, C.; YOON, C. J.; BESCHASTNIKH, I. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[102] LYU, L.; YU, H.; YANG, Q. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133*, 2020.

[103] XIE, C.; HUANG, K.; CHEN, P.-Y.; LI, B. Dba: Distributed backdoor attacks against federated learning. In: *International conference on learning representations*. [S.l.: s.n.], 2020.

[104] ZHANG, J.; CHEN, J.; WU, D.; CHEN, B.; YU, S. Poisoning attack in federated learning using generative adversarial nets. In: *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. [S.l.: s.n.], 2019. p. 374–380.

[105] FANG, M.; CAO, X.; JIA, J.; GONG, N. Z. Local model poisoning attacks to byzantine-robust federated learning. In: *Proceedings of the 29th USENIX Conference on Security Symposium*. [S.l.: s.n.], 2020. p. 1623–1640.

[106] TOLPEGIN, V.; TRUEX, S.; GURSOY, M. E.; LIU, L. Data poisoning attacks against federated learning systems. In: CHEN, L.; LI, N.; LIANG, K.; SCHNEIDER, S. (Ed.). *Computer Security – ESORICS 2020*. Cham: Springer International Publishing, 2020. p. 480–501. ISBN 978-3-030-58951-6.

[107] SUN, Z.; KAIROUZ, P.; SURESH, A. T.; MCMAHAN, H. B. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

[108] BHAGOJI, A. N.; CHAKRABORTY, S.; MITTAL, P.; CALO, S. Analyzing federated learning through an adversarial lens. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2019. p. 634–643.

[109] BAGDASARYAN, E.; VEIT, A.; HUA, Y.; ESTRIN, D.; SHMATIKOV, V. How to backdoor federated learning. In: *Proceedings of Machine Learning Research*. Online: PMLR, 2020. v. 108, p. 2938–2948.

[110] ZHOU, X.; XU, M.; WU, Y.; ZHENG, N. Deep model poisoning attack on federated learning. *Future Internet*, v. 13, n. 3, 2021. ISSN 1999-5903. Disponível em: <https://www.mdpi.com/1999-5903/13/3/73>.

[111] ANDREINA, S.; MARSON, G. A.; MöLLERING, H.; KARAME, G. Baffle: Backdoor detection via feedback-based federated learning. In: *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. [S.l.: s.n.], 2021. p. 852–863.

[112] SHEN, S.; TOPLE, S.; SAXENA, P. Auror: Defending against poisoning attacks in collaborative deep learning systems. In: *Proceedings of the 32nd Annual Conference on Computer Security Applications*. New York, NY, USA: Association for Computing Machinery, 2016. (ACSAC '16), p. 508–519. ISBN 9781450347716.

[113] XIE, C.; CHEN, M.; CHEN, P.-Y.; LI, B. Crfl: Certifiably robust federated learning against backdoor attacks. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2021. p. 11372–11382.

[114] COHEN, J.; ROSENFELD, E.; KOLTER, Z. Certified adversarial robustness via randomized smoothing. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2019. p. 1310–1320.

[115] PILLUTLA, K.; KAKADE, S. M.; HARCHAOUI, Z. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, v. 70, p. 1142–1154, 2022.

[116] YIN, D.; CHEN, Y.; KANNAN, R.; BARTLETT, P. Byzantine-robust distributed learning: Towards optimal statistical rates. In: DY, J.; KRAUSE, A. (Ed.). *Proceedings of the 35th International Conference on Machine Learning*. [S.l.]: PMLR, 2018. (Proceedings of Machine Learning Research, v. 80), p. 5650–5659.

[117] MHAMDI, E. M. E.; GUERRAOUI, R.; ROUAULT, S. The hidden vulnerability of distributed learning in Byzantium. In: DY, J.; KRAUSE, A. (Ed.). *Proceedings of the 35th International Conference on Machine Learning*. [S.l.]: PMLR, 2018. (Proceedings of Machine Learning Research, v. 80), p. 3521–3530.

[118] KIM, H.; PARK, J.; BENNIS, M.; KIM, S.-L. On-device federated learning via blockchain and its latency analysis. *arXiv preprint arXiv:1808.03949*, Aug, 2018.

[119] Weng, J.; Weng, J.; Zhang, J.; Li, M.; Zhang, Y.; Luo, W. Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive. *IEEE Transactions on Dependable and Secure Computing*, p. 1–1, 2019.

[120] RICHARDSON, A.; FILOS-RATSIKAS, A.; FALTINGS, B. Budget-bounded incentives for federated learning. In: _____. *Federated Learning: Privacy and Incentive*. Cham: Springer International Publishing, 2020. p. 176–188. ISBN 978-3-030-63076-8.

[121] HUANG, C.; ZHANG, H.; LIU, X. Incentivizing data contribution in cross-silo federated learning. *arXiv preprint arXiv:2203.03885*, 2022.

[122] FREDRIKSON, M.; JHA, S.; RISTENPART, T. Model inversion attacks that exploit confidence information and basic countermeasures. In: *CCS '15*. New York, NY, USA: Association for Computing Machinery, 2015. p. 1322–1333. ISBN 9781450338325.

[123] Triastcyn, A.; Faltings, B. Federated generative privacy. *IEEE Intelligent Systems*, v. 35, n. 4, p. 50–57, 2020.

[124] DEAN, J.; CORRADO, G.; MONGA, R.; CHEN, K.; DEVIN, M.; MAO, M.; RANZATO, M.; SENIOR, A.; TUCKER, P.; YANG, K. Large scale distributed deep networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1223–1231.

[125] TITCOMBE, T.; HALL, A. J.; PAPADOPOULOS, P.; ROMANINI, D. Practical defences against model inversion attacks for split neural networks. *arXiv preprint arXiv:2104.05743*, 2021.

[126] QI, T.; WU, F.; WU, C.; HUANG, Y.; XIE, X. Privacy-preserving news recommendation model learning. *arXiv preprint arXiv:2003.09592*, 2020.

[127] HITAJ, B.; ATENIESE, G.; PEREZ-CRUZ, F. Deep models under the gan: Information leakage from collaborative deep learning. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2017. (CCS '17), p. 603–618. ISBN 9781450349468. Disponível em: <https://doi.org/10.1145/3133956.3134012>.

[128] CHEN, Z.; FU, A.; ZHANG, Y.; LIU, Z.; ZENG, F.; DENG, R. H. Secure collaborative deep learning against gan attacks in the internet of things. *IEEE Internet of Things Journal*, v. 8, n. 7, p. 5839–5849, 2021.

[129] RIAZI, M. S.; WEINERT, C.; TKACHENKO, O.; SONGHORI, E. M.; SCHNEIDER, T.; KOUSHANFAR, F. Chameleon: A hybrid secure computation framework for machine learning applications. In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2018. (ASIACCS '18), p. 707–721. ISBN 9781450355766.

[130] YAN, X.; CUI, B.; XU, Y.; SHI, P.; WANG, Z. A method of information protection for collaborative deep learning under gan model attack. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 18, n. 3, p. 871–881, 2021.

[131] GAREY, M. R.; JOHNSON, D. S. Computers and intractability. *A Guide to the*, 1979.

[132] LAARHOVEN, P. J. V.; AARTS, E. H. Simulated annealing. In: *Simulated annealing: Theory and applications*. [S.l.]: Springer, 1987. p. 7–15.

[133] KIRKPATRICK, S.; JR, C. D. G.; VECCHI, M. P. Optimization by simulated annealing. *science*, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983.

[134] INGBER, L. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, Elsevier, v. 18, n. 11, p. 29–57, 1993.

[135] Sharafaldin, I.; Lashkari, A. H.; Hakak, S.; Ghorbani, A. A. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: *2019 International Carnahan Conference on Security Technology (ICCST)*. [S.l.: s.n.], 2019. p. 1–8.

[136] JACOBS, R. A. Increased rates of convergence through learning rate adaptation. *Neural networks*, Elsevier, v. 1, n. 4, p. 295–307, 1988.

[137] YU, X.-H.; CHEN, G.-A.; CHENG, S.-X. Dynamic learning rate optimization of the backpropagation algorithm. *IEEE Transactions on Neural Networks*, IEEE, v. 6, n. 3, p. 669–677, 1995.

[138] MAGOULAS, G. D.; VRAHATIS, M. N.; ANDROULAKIS, G. S. Improving the Convergence of the Backpropagation Algorithm Using Learning Rate Adaptation Methods. *Neural Computation*, v. 11, n. 7, p. 1769–1796, 10 1999. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/089976699300016223>.

[139] YEDIDA, R.; SAHA, S.; PRASHANTH, T. Lipschitzlr: Using theoretically computed adaptive learning rates for fast convergence. *Applied Intelligence*, Springer, v. 51, n. 3, p. 1460–1478, 2021.

[140] SMITH, S. L.; KINDERMANS, P.-J.; YING, C.; LE, Q. V. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[141] LI, Q.; HE, B.; SONG, D. Model-contrastive federated learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2021. p. 10713–10722.

[142] YOON, T.; SHIN, S.; HWANG, S. J.; YANG, E. Fedmix: Approximation of mixup under mean augmented federated learning. *arXiv preprint arXiv:2107.00233*, 2021.

[143] LI, T.; SAHU, A. K.; ZAHEER, M.; SANJABI, M.; TALWALKAR, A.; SMITHY, V. Feddane: A federated newton-type method. In: IEEE. *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. [S.l.], 2019. p. 1227–1231.

[144] KARIMIREDDY, S. P.; KALE, S.; MOHRI, M.; REDDI, S.; STICH, S.; SURESH, A. T. Scaffold: Stochastic controlled averaging for federated learning. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2020. p. 5132–5143.

[145] LIU, W.; CHEN, L.; CHEN, Y.; ZHANG, W. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 31, n. 8, p. 1754–1766, 2020.

[146] AZHAGUSUNDARI, B.; THANAMANI, A. S. Feature selection based on information gain. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, v. 2, n. 2, p. 18–21, 2013.

[147] MOMMA, M.; BENNETT, K. P. A pattern search method for model selection of support vector regression. In: SIAM. *Proceedings of the 2002 SIAM International Conference on Data Mining*. [S.l.], 2002. p. 261–274.

[148] HSU, T.-M. H.; QI, H.; BROWN, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[149] YUROCHKIN, M.; AGARWAL, M.; GHOSH, S.; GREENEWALD, K.; HOANG, N.; KHAZAENI, Y. Bayesian nonparametric federated learning of neural networks. In: PMLR. *International conference on machine learning*. [S.l.], 2019. p. 7252–7261.

[150] ACAR, D. A. E.; ZHAO, Y.; NAVARRO, R. M.; MATTINA, M.; WHATMOUGH, P. N.; SALIGRAMA, V. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

[151] WANG, J.; TANTIA, V.; BALLAS, N.; RABBAT, M. Slowmo: Improving communication-efficient distributed sgd with slow momentum. In: *International Conference on Learning Representations (ICLR)*. [S.l.: s.n.], 2020.

[152] WANG, Y.; KANTARCI, B. A novel reputation-aware client selection scheme for federated learning within mobile environments. In: IEEE. *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. [S.l.], 2020. p. 1–6.