Master’s Thesis Report

Federated learning approach for IoT network intrusion detection

NGUYEN VAN TUAN

Supervisor BEURAN, Razvan Florin

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

September, 2024

Abstract

The rise of the Internet of Things (IoT) has revolutionized various human life aspects by interconnecting numerous information devices, but this has also increased the cyber attack surface to more sophisticated intrusions. An intrusion is defined as any activity that attempts to compromise the CIA characteristics (confidentiality, integrity, and availability) or bypass the security of a computer or network. The intrusion detection system is one of the most popular and effective solutions to secure modern information systems. Host-based and network-based approaches are the main categories of intrusion detection systems. In which, a host-based method observes the information on individual computers in the system. Meanwhile, a network-based method captures and analyzes network traffic to protect multiple hosts in network segments. Leveraging the outbreak of machine learning, there are so many studies trying to improve the performance in IoT intrusion detection, which can be categorized as supervised, semi-supervised, and unsupervised approaches depending on the availability of training data. To tackle the existing challenges such as the evolution of attacking techniques, and the lack of labeled anomalous data, the semi-supervised and unsupervised approaches are adopted more popularly.

Besides developing an effective learning data model, addressing resource constraints and ensuring privacy preservation are critical concerns in modern IoT networks. Traditional machine learning methods operate within a centralized paradigm, where a single model is trained on a server using all the data collected from connected devices. This faces challenges related to computational resources, data latency, data transfer cost, and particularly privacy preservation when collecting all data to a single server.

This thesis addresses these problems of IoT intrusion detection by proposing a novel network-based approach based on federated learning that leverages the computational power of distributed IoT devices while training the local model itself and protecting the local data from being accessed directly. I propose a semi-supervised federated learning approach using the combination of the Shrink Autoencoder model and Centroid one-class classifier (SAE-CEN) to enhance IoT intrusion detection. The Shrink Autoencoder tries to represent the normal network data in a new optimal data space around the origin in the latent layer, then, the Centroid algorithm can detect the unseen data point based on its distance to the origin. This makes the detection task more effective and efficient. I develop a novel mean square error-based aggregation algorithm (MSEAvg) to improve global model performance when prioritizing the more accurate local model in aggregating the global model. Our approach aims to address issues such as data heterogeneity, unbalanced and noisy data, and the scarcity of labeled abnormal data, which are prevalent in IoT environments.

One of the most important aspects is applying the experimental study to real-world scenarios. Some existing research using federated learning creates experimental environments that do not accurately reflect real-world conditions. An IoT network is divided into multiple sub-networks, each having some IoT devices of different types and innovating over time. During the lifetime, some new devices are added or removed, changing seriously the network topology, and leading to a change in the data distribution. In this thesis, I construct the experimental scenarios to be more practical by including both IID (Independent and Identically Distributed) and non-IID settings using the N-BaIoT dataset and the Dirichlet distribution.

The experimental outcomes in this setup demonstrate that our SAE-CEN model, combined with the MSEAvg aggregation algorithm, significantly improves detection accuracy and robustness in heterogeneous IoT networks. I also conduct some investigations in different federated learning settings to examine the robustness of my approach. The results also expose that my approach not only can boost the performance but also reduce the learning costs of federated intrusion detection and adapt strongly in large-scale IoT networks.

This work contributes to the field by presenting a practical federated learning framework for IoT intrusion detection, highlighting the capability of tailored aggregation methods and the potential of semi-supervised learning techniques in addressing real-world cybersecurity challenges.

Keywords. IoT, Intrusion Detection, Machine Learning, Federated Learning.

Contents

This thesis was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and Le Quy Don Technical University.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The Internet of Things (IoT) has emerged as a trendy technology because of its enormous potential in a variety of industries, including transportation [1], healthcare [2], and smart cities [3]. IoT is the interconnected network of numerous physical objects, often known as things [4]. IoT systems are increasing dramatically with new, sophisticated, and modern devices being produced every day. The data generated by an enormous number of interconnected devices is heterogeneous, diverse, and complex. Result in increasing the cyber attack surface which is affected by several novel IoT anomalies [5], such as botnet, DoS, DDoS, and Spoofing. Thus, IoT anomaly detection is a very essential task in new-fashioned IoT networks.

To secure system networks and privacy in deployment against cyber risks, an intrusion detection system (IDS) by analyzing and monitoring network traffic is one of the most efficient solutions [4, 6], these approaches are also known as anomaly detection systems in some research. Modern IDSs that leverage machine learning (ML) techniques to detect unseen threats in IoT networks have been studied and adopted widely [4, 5, 7]. Besides developing effective data learning models, it is also very important to choose and define learning and deployment strategies to face new challenges when IoT networks scale up continuously. Traditional ML methods work in a centralized paradigm, in which, one model is trained in a single server using all data collected from connected agents [8]. In the modern IoT context, this approach depicts some challenges such as computational resource requirements for training and serving the models; data latency when the data transmission distance may be hundreds, even thousand miles to the server; data transfer cost; particularly data privacy preserving issues due to data leakage when sharing data among agents and server [9]. Therefore, Federated Learning (FL) has emerged recently as a cutting-edge approach to solving these problems. Rather than collecting data, training, and serving ML models in a single high-performance computer, FL takes advantage of the network client's computing capabilities in the distributed mechanism to train the local model itself and protect the local data from being accessed directly [9]. Then, the local models will be aggregated to a global model in the server for serving by some methodologies.

In recent years, many approaches have been released to leverage the power of federated learning to solve network anomaly detection [10, 6, 11, 12]. However, several issues have been making this task challenging, such as the experimental scenarios are not able to present practical problems well when the IoT intrusion is evolving day by day; the traffic collected from large-scale IoT networks with different application scenarios is unbalanced, sparse, diverse, high-dimensional, and noisy; the abnormal data is not sufficient for training process; the aggregation method of the global model is still not optimal to get a robust model to detect novel attacks effectively. In this thesis, I propose a semi-supervised federated learning approach based on the Autoencoder model. This is expected to increase the effectiveness of detecting IoT intrusion.

1.2 Problem definition

My thesis addresses the problem of improving the effectiveness of machine learning in IoT intrusion detection, where the IoT network is changed heterogeneously over time; the lack of anomalies, resource constraints, and privacy preservation are crucial.

An IoT network is typically divided into multiple network areas, each has a gateway, also known as an edge server, that manages various types of IoT devices and evolves heterogeneously over time, as described in Figure 1.1. These devices generate diverse amounts of data, depending on their functions. Therefore, the data statistical characteristics of each subnetwork are very different. Machine learning models are usually trained on data of a period from the initial state of the IoT network. However, after deployment, over the network's lifetime, some new devices are added or removed, changing significantly the network topology. This makes the machine learning model perform poorly in unseen data. Some research, such as [6] and [11], create IoT networks by randomly creating data partitions from existing datasets without control to evaluate the performance of their approaches. This may not ensure the practical characteristics of the scenarios. The goal of this research is to design and do experiments on dynamic scenarios to ensure that the experimental models can adapt to real-world conditions and maintain their effectiveness over acceptable time.

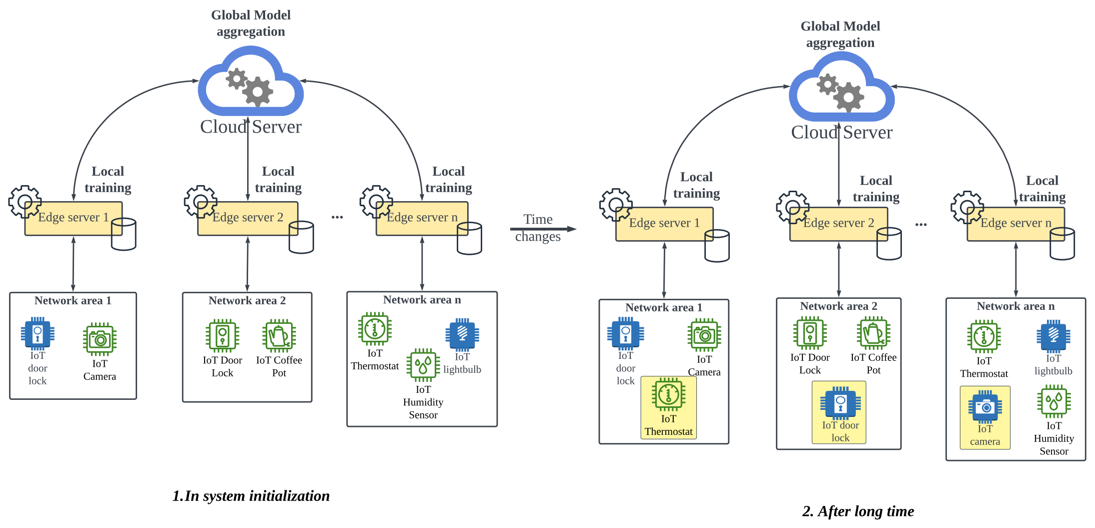


Figure 1.1: Changes in IoT networks over time

Another critical challenge is the lack of labeled abnormal data. Despite the vast amount of network traffic data generated by modern, complex IoT networks, there is a notable lack of labeled abnormal data due to expert knowledge limitations and labeling expenses. This absence makes it difficult to use supervised machine learning models to learn the characteristics of available data. To address this, the thesis leverages the power of semi-supervised learning techniques by learning from the available normal data, which helps detect anomalies in the network. I use a hybrid approach combining an Autoencoder-based model with a one-class classification model to enhance the detection of anomalies.

Resource constraints and privacy preservation are also significant concerns in modern IoT networks. When all devices are connected via the Internet, it is very essential to protect private data to prevent data leakage or any other threats. Additionally, IoT gateways often have insufficient computational power required to process the vast amounts of network data they collect. FL has emerged as a promising solution to these problems, enabling decentralized model training without sharing raw data, thereby preserving privacy and reducing the computational expense on individual edge devices. However, while current FL algorithms like FedAvg [13] and FedProx [14] weight contributions based on the number of data samples each participant has, they still face challenges in dynamic environments where participant availability and data characteristics can vary significantly. Both of these two algorithms update the global model based on the size of the data partition in each client. This makes them act poorly in the complex distributed network when the client that has a large number of data samples may not represent well for the whole network. This thesis aims to propose a new approach in FL that considers the role of each client in the training process, improving the performance of the global model. Furthermore, I also investigate the robustness of my proposed approach in different federated learning settings to pose outstanding effectiveness.

1.3 Research approach

This is the research approach I have taken in this thesis to address the above three problems.

Firstly, I do a literature review on some existing research related to intrusion detection and federated learning to pinpoint some challenges and draw a solution for these problems. This aims to get the background, recent trends, and existing approaches in IoT intrusion detection. I then identify remaining challenges and draw potential solutions.

To address the problem related to the experimental scenarios, I next construct distributed IoT networks using an existing IoT network traffic, called NBa-IoT [15], and control the heterogeneity using the Dirichlet distribution.

To address resting challenges, I deploy a hybrid approach combining the Shrink Autoencoder (SAE) with the Centroid algorithm (CEN) within a federated learning framework. I next design and develop a new federated learning algorithm, MSEAvg, based on mean square error, to enhance the performance of this hybrid model. My algorithm prioritizes the more accurate local models for updating. Various hyperparameters need to be set up in federated learning and the client selection ratio is one of the most important ones that affects both model accuracy and computational expense. In my experimental study, I investigate the model's effectiveness under different client ratios and evaluate its performance on large-scale IoT networks to find a setting that is good enough for this task.

1.4 Contributions

The main contributions of this thesis include:

- I do the literature review on network intrusion detection and IoT intrusion detection to show recent trends and pinpoint some challenges in these topics.

- Leverage a lightweight semi-supervised hybrid model of Shrink Autoencoder and Centroid model in the federated learning scheme.
- Propose a novel aggregation algorithm based on the mean square error to improve the accuracy of the above approach.
- Illustrate a practical scenario for studying federated learning in IoT intrusion detection
- Investigate the model's performance in different scenarios to find a suitable setting.

1.5 Thesis structure

The rest of this thesis is structured as follows:

- Chapter 2 provides some background knowledge on modern intrusion detection systems using machine learning and federated learning.
- Chapter 3 describes my approach to address the problems mentioned in Chapter 1.
- Chapter 4 describes the evaluation process and analyzes the results of experiments. Also points out some limitations of the research.
- Chapter 5 summarizes my research and points out some future directions to develop this study.

Chapter 2

Background and related work

2.1 Background

This section briefly introduces the background knowledge related to the intrusion detection system with machine learning, federated learning, the NBa-IoT network dataset, and some existing research in this domain.

2.1.1 Machine learning for intrusion detection systems

In information systems, an intrusion is defined as any activity that attempts to compromise the CIA characteristics (confidentiality, integrity, and availability) or bypass the security of a computer or network [16]. These activities can be caused by attackers or by inside users trying to gain additional unauthorized privileges.

To protect systems, Intrusion Detection Systems (IDS) have been developed as hardware or software products to automatically monitor and analyze the events inside a computer or network to recognize intrusions [16]. With the advancement of information systems, cyber threats have become increasingly sophisticated. Therefore, IDSs become more necessary for the information infrastructure of most organizations.

The IDS is often distinguished based on the target or the detection method. By the target, there are two types of IDS, host-based IDS (HIDS) and network-based IDS (NIDS) [16, 17]. HIDS inspects the information collected on individual machines in the system, such as resource metrics, and operating system audits. This makes them work accurately and reliably, determining exactly the behavior of attackers in a low lever, even the outcome of intrusion. However, HIDS has some downsides: When it resides on a computer targeted by attackers, the IDS itself may be compromised and disabled. Additionally, HIDS consumes host resources for its operations, which can negatively affect system performance. In contrast, NIDS detects intrusion by capturing and analyzing network packets. These IDS are placed beside a router or a switch to monitor a network segment, which enables it to protect multiple hosts in this segment. The network sniffer of this IDS type is often set up in the "stealth" mode, which makes them more secure against attackers and does not affect the host activities. However, the evolvement of network scale and attacking techniques pose challenges to NIDSs. The huge amount of data generated from large-scale networks makes it increasingly difficult to process all traffic.

Another way to categorize IDS is based on the detection method: signature-based IDS and anomaly-based IDS [17, 18]. Signature-based IDS, also known as rule-based IDS, relies on predefined rules and patterns derived from known threats. This method uses a database of known intrusion patterns to identify malicious activities. This approach is highly effective at detecting previously identified threats and providing detailed contextual information about the nature of the attack. Signature-based IDS is straightforward to implement and provides precise identification of known threats, but it struggles with detecting new, unknown attacks that are not defined in the database. On the other hand, anomaly-based IDS, aka. behavior-based IDS in some articles, focuses on detecting deviations from established norms of behavior within the network or system. Instead of relying on predefined rules, this method builds a model of normal activity and monitors for any behavior that significantly deviates from this baseline. Anomaly-based IDS can identify unknown or new types of attacks that do not match any existing signatures, but it requires careful tuning to reduce false positives and maintain accuracy. NIDS can not recognize the encrypted packets, and leave opportunities for sophisticated attacks. In this thesis, I focus on improving the NIDS abilities using flow-based analysis.

The challenges of traditional IDS highlight the requirement for more sophisticated and adaptive approaches. With its ability to learn from data, detect patterns, and make intelligent decisions, Machine Learning (ML) is a promising solution. An ML approach consists of three phases: Training, validation, and testing. In the training phase, some ML models are initialized and trained on an available dataset. Then, to decide which model is the most suitable one, we use a known sub-dataset, validation set, to validate the performance of these models. The model with the highest performance should be chosen to use in the testing phase [17].

In IDS development, there are three primary types of ML approaches: supervised, semi-supervised, and unsupervised learning [19]. In the supervised learning problems, there is an assumption that all available data is labeled for building the model. Any new data are fed into the trained model and classified as normal or abnormal. The big challenge of supervised methods is the lack of anomalous data, leading to the class imbalance problem. Additionally, this approach is ineffective in detecting novel anomalies that are not learned in the model. The authors in [20] evaluated some supervised learning models applied in IoT intrusion detection, such as Random Forest, Support Vector Machine (SVM), Naive Bayes, and Multi-Layer Perception. The semi-supervised learning methods suppose that there is only normal data available to construct the model. They infer the unseen data based on the deviation from normal data. By leveraging the huge amount of labeled normal data, semi-supervised learning can effectively detect unseen or unknown anomalies. Some well-known models of this type are One-class SVM [21], and Autoencoder and its variants [15, 22]. The last type of ML is unsupervised learning which works under the implicit assumption that the normal data points are far from anomalous ones [19]. Some famous models are Local Outlier Factor (LOF) [23], K-means [24].

In this thesis, I focus on improving the accuracy of network-based IoT intrusion detection with semi-supervised learning approaches.

2.1.2 Federated learning

This section introduces the background principle of FL and pinpoints some challenges in FL.

Overview

Federated Learning (FL) [13] is a novel approach to train machine learning models in a decentralized manner, aiming to address the growing concerns over data privacy and the increasing computational power available on edge devices such as smartphones and IoT devices. FL enables the development of machine learning models by leveraging data distributed across multiple devices, referred to clients, without transferring the data to a central server. The model on each client device is called the local model, while the aggregated model, obtained by combining updates from clients, is known as the global model.

Concept

The training process is conducted in several consecutive communication rounds until the global model reaches the desired accuracy or meets the specified training criteria [9]. Figure 2.1 depicts the steps of how a communication round occurs:

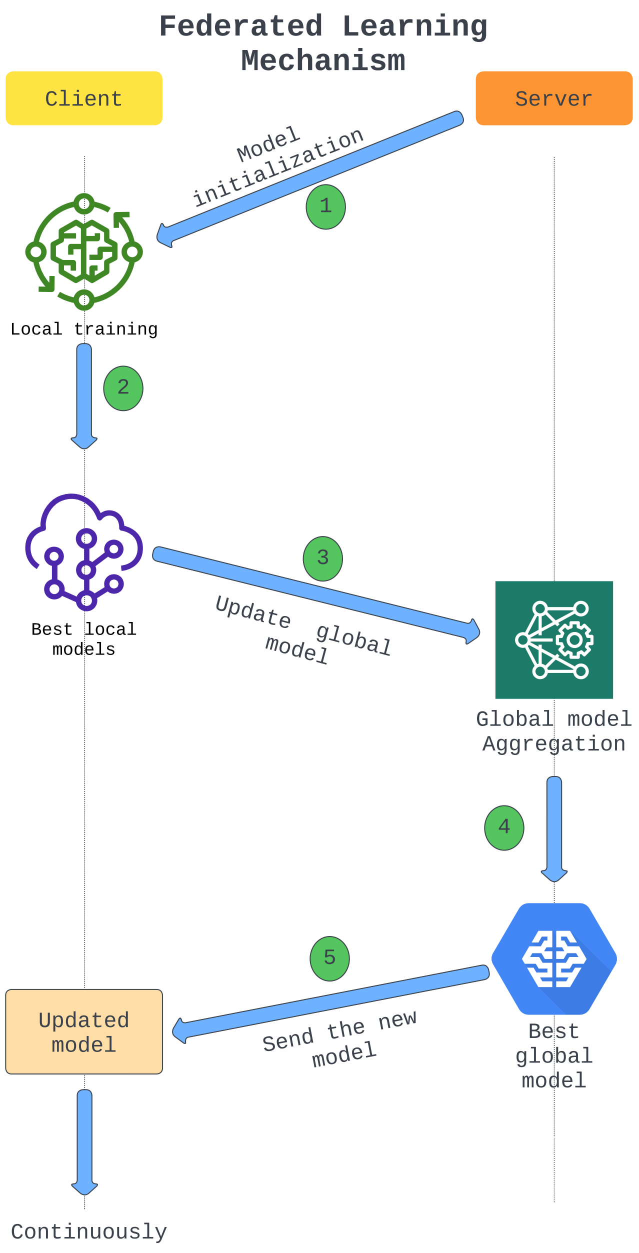


Figure 2.1: Federated Learning concept

1. *Model initialization*: Firstly, a subset of clients is selected in the global model with a client selection ratio. A global model is initialized and sent to all selected clients. This is the starting point for the training process.

2. *Local training*: Each client receives the global model and trains it using its local data. This results in a set of updated local models, each adapted to the data from its respective devices.

3. *Send optimized parameters*: Once all selected clients have done the local training process, optimized weights are sent to the global server.

4. *Global model aggregation*: The server aggregates the updates from all local models to form a new global model that has the knowledge of all participants.

5. *Send the new model*: The updated global model is then sent back to all clients in the network. This updated model now reflects a more comprehensive understanding based on the combined data from all gateways.

Challenges

One of the fundamental challenges in FL is the handling of non-IID (non-Independent and Identically Distributed) data [13]. In traditional centralized machine learning, data is typically assumed to be IID, meaning that each data point is independent of the others and drawn from the same distribution. However, in FL, the data on each client device is inherently linked to the user's behavior and usage patterns, which means that it can vary significantly from one device to another. This heterogeneity can lead to local models that are poorly representative of the overall population. Another significant challenge is the unbalanced nature of the data [13]. In a federated setting, some users will generate large amounts of data while others contribute very little. Addressing this issue requires careful consideration of how to weigh contributions from different clients to ensure that the global model remains fair and representative of all users. This is called the non-IID issue.

In the experimental environment, to better simulate practical scenarios, the Dirichlet distribution is often employed to construct the experiment evaluation. This statistical technique allows for the generation of data distributions that can reflect the non-IID (non-Independent and Identically Distributed) nature of real-world data [25, 26, 27].

Aggregation algorithms

To achieve the objectives of federated learning, the federated aggregation algorithm plays a crucial role. Researchers have studied and proposed numerous algorithms for various applications. Among these, FedAvg [13] and FedProx [14] are the most widely adopted and popular ones.

FedAvg is a straightforward algorithm that constructs a global model by calculating the weighted average of parameters from the client's models based on the number of data samples they hold. The process occurs until get the desired model. This algorithm is simple to implement and suitable for large-scale networks with a lot of participants [6].

FedProx is similar to FedAvg but introduces a regularization term to each client's loss function. This term, denoted as γ, acts as a proximal constraint, penalizing significant deviations of the client model parameters from the previous global model.

2.1.3 Machine learning IoT dataset

This section provides the details of an existing IoT dataset - N-BaIoT. The N-BaIoT dataset [15] is designed to aid in the detection of IoT botnet attacks. The dataset was collected from nine commercial IoT devices that were intentionally infected with two prevalent IoT-based botnets: Mirai and Gatfyt. The details of this dataset are described in Table 2.1.

Table 2.1: N-BaIoT network traffic dataset

ID	Device name	Normal	Gatfyt	Mirai
0	Danmini_Doorbell	49548	652100	316650
1	Ecobee_Thermostat	13113	512133	310630
2	Philips_B120N10_Baby_Monitor	175240	312273	610714
3	Provision_PT_737E_Security_Camera	62154	330096	
4	Provision_PT_838_Security_Camera	98514	309040	429337
5	Samsung_SNH_1011_N_Webcam	52150	323072	
6	SimpleHome_XCS7_1002_WHT_Security_Camera	46585	303223	513248
7	SimpleHome_XCS7_1003_WHT_Security_Camera	19528	316438	514860
8	Ennio_Doorbell	39100	316400	

The dataset comprises raw network traffic data captured in pcap format. Traffic data was collected using port mirroring on a network switch through which organizational traffic flows as described in Figure 2.2. This setup ensures that the dataset reflects realistic network conditions in an enterprise environment. The IoT devices used in the dataset include various types, each providing a diverse set of data representative of typical IoT device behavior.

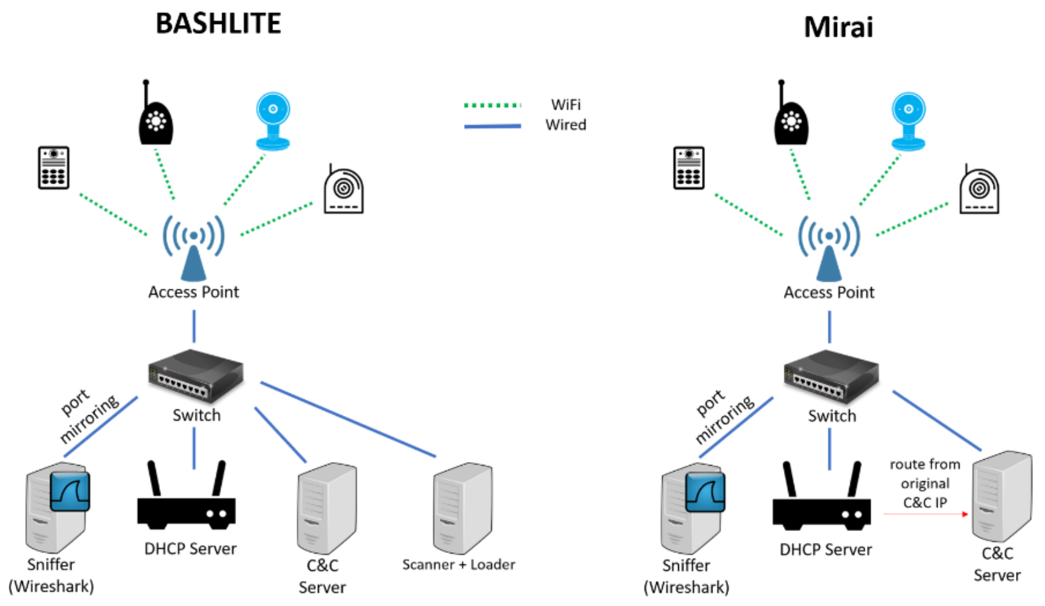


Figure 2.2: N-BaIoT network dataset topology [15]

The dataset includes 115 statistical features extracted over several temporal windows to capture the behavior of the IoT devices. These features are divided into four main categories based on the source of the traffic:

- **Source IP**: Traffic statistics for packets originating from the same IP address.
- **Source MAC-IP**: Traffic statistics for packets originating from the same MAC and IP address.
- **Channel**: Traffic statistics for packets sent between specific source and destination IP addresses.
- **Socket**: Traffic statistics for packets sent between specific TCP/UDP sockets.

The features (Table 2.2) are computed over five different time windows: 5 seconds, 3 seconds, 1 second, 0.1 seconds, and 0.01 seconds. This multi-scale approach enables the detection of both rapid and prolonged anomalous behaviors.

Table 2.2: Summary of NBa-IoT dataset features

Feature Name	Description
MI_dir_Lx_weight	Mutual Information (MI) directional data weight over a time window x. Indicates the importance or contribution of the MI feature.
MI_dir_Lx_mean	Mutual Information (MI) directional data mean value over time window x. Represents the average value of the MI feature.
MI_dir_Lx_variance	Mutual Information (MI) directional data variance over time window x. Indicates the variability or dispersion of the MI feature values.
H_Lx_weight	Entropy (H) weight over time window x. Indicates the importance or contribution of the entropy feature.
H_Lx_mean	Entropy (H) mean value over time window x. Represents the average value of the entropy feature.
H_Lx_variance	Entropy (H) variance over time window x. Indicates the variability or dispersion of the entropy feature values.
HH_Lx_weight	Hierarchical entropy (HH) weight over a time window x. Indicates the importance or contribution of the hierarchical entropy feature.
HH_Lx_mean	Hierarchical entropy (HH) mean value over time window x. Represents the average value of the hierarchical entropy feature.
HH_Lx_std	Hierarchical entropy (HH) standard deviation over time window x. Measures the dispersion or variability of the values.
HH_Lx_magnitude	Hierarchical entropy (HH) magnitude over time window x. Indicates the size or extent of the feature values.
HH_Lx_radius	Hierarchical entropy (HH) radius over time window x. Indicates the radius in the feature space.
HH_Lx_covariance	Hierarchical entropy (HH) covariance over time window x. Measures the joint variability of two random variables.
HH_Lx_pcc	Hierarchical entropy (HH) Pearson Correlation Coefficient (PCC) over a time window x. Measures the linear correlation between two variables.

Feature Name	Description
HH_jit_Lx_weight	Hierarchical entropy jitter (HH_jit) weight over time window x. Indicates the importance or contribution of the hierarchical entropy jitter feature.
HH_jit_Lx_mean	Hierarchical entropy jitter (HH_jit) mean value over time window x. Represents the average value of the hierarchical entropy jitter feature.
HH_jit_Lx_variance	Hierarchical entropy jitter (HH_jit) variance over time window x. Indicates the variability or dispersion of the hierarchical entropy jitter values.
HpHp_Lx_weight	Higher-order statistics (HpHp) weight over time window x. Indicates the importance or contribution of the higher-order statistics feature.
HpHp_Lx_mean	Higher-order statistics (HpHp) mean value over time window x. Represents the average value of the higher-order statistics feature.
HpHp_Lx_std	Higher-order statistics (HpHp) standard deviation over time window x. Measures the dispersion or variability of the values.
HpHp_Lx_magnitude	Higher-order statistics (HpHp) magnitude over time window x. Indicates the size or extent of the feature values.
HpHp_Lx_radius	Higher-order statistics (HpHp) radius over time window x. Indicates the radius in the feature space.
HpHp_Lx_covariance	Higher-order statistics (HpHp) covariance over time window x. Measures the joint variability of two random variables.
HpHp_Lx_pcc	Higher-order statistics (HpHp) Pearson Correlation Coefficient (PCC) over a time window x. Measures the linear correlation between two variables.

2.2 Related work

In this section, I review some recent trends and approaches for IoT intrusion detection. FL has been successful in various fields, particularly in intrusion detection. The first research applying FL in IoT intrusion detection is DIoT [10] which proposed a device-type-specific self-learning framework with a GRU network. It consists of two main components: The security gateway acts as an access point for IoT devices and the IoT security service maintains a set of device-type-specific models. The system autonomously learns and updates its models without requiring human intervention or labeled data. This allows DIoT to adapt to new device behaviors and emerging threats dynamically. The authors conducted extensive experiments using over 30 IoT devices with Mirai botnet in both laboratory and real-world smart home deployment settings. The evaluation showed that DIoT achieved a 95.6% detection rate with zero false alarms in real-world conditions. However, each device type in the system maintains its model, which can make system management challenging as the scale increases, and this research is limited to the Mirai botnet threats.

The author of [11] utilized LSTM and GRU with the ensemble learning technique to enable on-device learning. The predictions from local GRU models are combined using a Random Forest Classifier (RFC). Each GRU model provides probability values for each potential label (attack type) for a given input. The RFC aggregates the probability values from all GRU models. It uses these probabilities as votes to determine the final prediction. The label with the highest combined probability across all models is selected as the final prediction. Their evaluation demonstrated the performance of the FL approach compared to the non-FL approach. However, the ability of this approach to detect unknown intrusion is limited due to the supervised learning problem. Furthermore, the authors construct the experiments without considering the heterogeneity of the IoT network, making it not close to the practical cases.

Fed-ANIDS [6] is based on the Autoencoder model similar to my proposal, a semi-supervised learning method. Various autoencoder models, including simple autoencoders (AE), variational autoencoders (VAE), and adversarial autoencoders (AAE), are utilized for anomaly detection based on reconstruction errors of normal traffic. The FL setting was employed by using FedAvg and FedProx algorithms. The authors conducted several experiments on some well-known network traffic datasets such as USTC-TFC2016, CIC-IDS2017, and CSE-CIC-IDS2018. The results presented the performance of Fed-NIDS in detecting network intrusions with high accuracy and low false alarms while preserving privacy and also showed that FedProx has a slightly better accuracy than FedAvg. However, the experimental scenarios lack practicality as the authors did not apply any criteria for randomly partitioning the original dataset.

Chapter 3

Methodology

3.1 Approach overview

I proposed an IoT intrusion detection approach using federated learning and a hybrid model that is based on Shrink Autoencoder and Centroid algorithms. Figure 3.1 presents the overall architecture of my approach. It consists of two main components as described in the federated learning scheme: *(1) Hybrid intrusion detector*, *(2) MSEAvg aggregation*. The details of these two parts are explained in the rest of this chapter.

3.2 Hybrid intrusion detection approach

Figure 3.2 depicts the activities completed in the IoT gateways. By modeling normal network data, an Autoencoder-based model can leverage its latent layer to transform the original data to a new data space where data has a smaller number of dimensions and presents the most important characteristics. Hence, some common traditional one-class classifiers can work effectively on this new data to detect anomalies. In this part, a hybrid model is constructed by using Shrink Autoencoder as a data representation method and Centroid anomaly detection algorithm (CEN) as a detector. This approach is named as SAE-CEN model.

To feed to the machine learning model, firstly, the network data need to be preprocessed. I use the Standard Normalization method to normalize the data to reduce the complexity and ensure all features contribute equally to the training process. The formula for standard scaling (also known as Z-score normalization) is given by:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

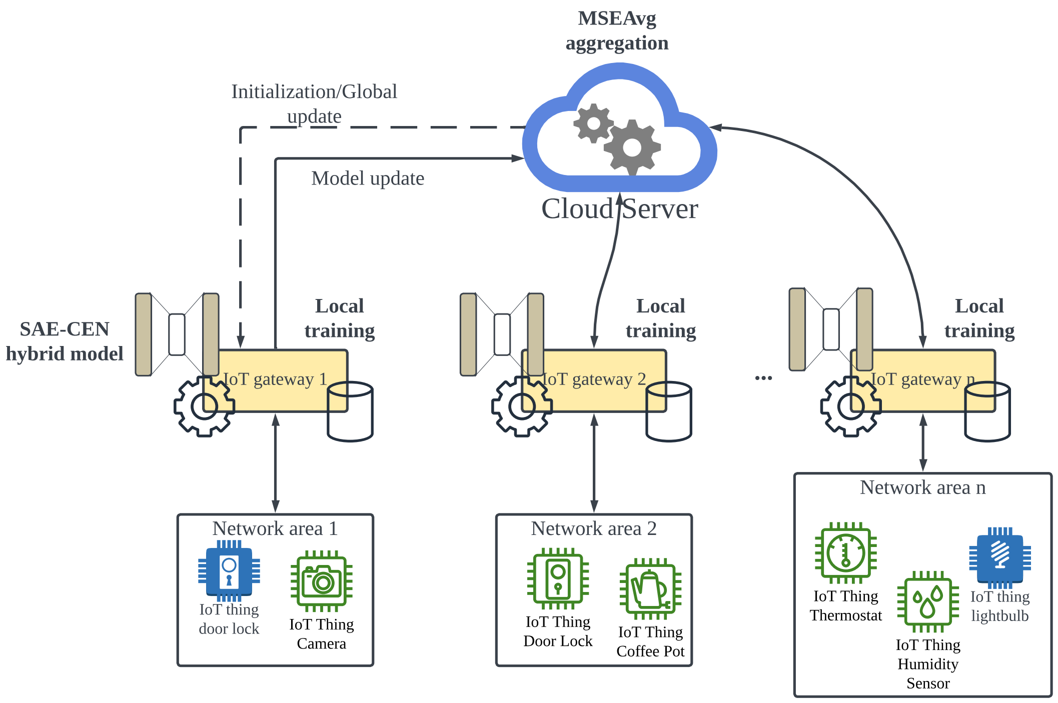


Figure 3.1: Proposed approach architecture

where x is the original feature value, μ is the mean of the feature, σ is the standard deviation of the feature.

3.2.1 Shrink Autoencoder (SAE)

Shrink Autoencoder (SAE) [22] is a powerful data representation model that helps common network intrusion detection algorithms deal with sparse and high-dimensional network data, even with a small amount of training data. This model is an Autoencoder variant since a new regularization term is added to the Autoencoder objective function (Equation 3.2). This makes it easier to construct the normal network data in the latent layer.

The objective function of the SAE training process is formulated as follows:

$$\mathcal{L}_{\text{SAE}}(\theta; x^i, z) = \frac{1}{n}\sum_{i=1} \|x^i - \hat{x}^i\|^2 + \lambda\frac{1}{n}\sum_{i=1}^{n} \|z^i\|^2 \tag{3.2}$$

where $\hat{x}^i$ and z^i are the reconstruction output and the latent vector of the input x_i, correspondingly. The first term is the reconstruction error (RE), and the sec-

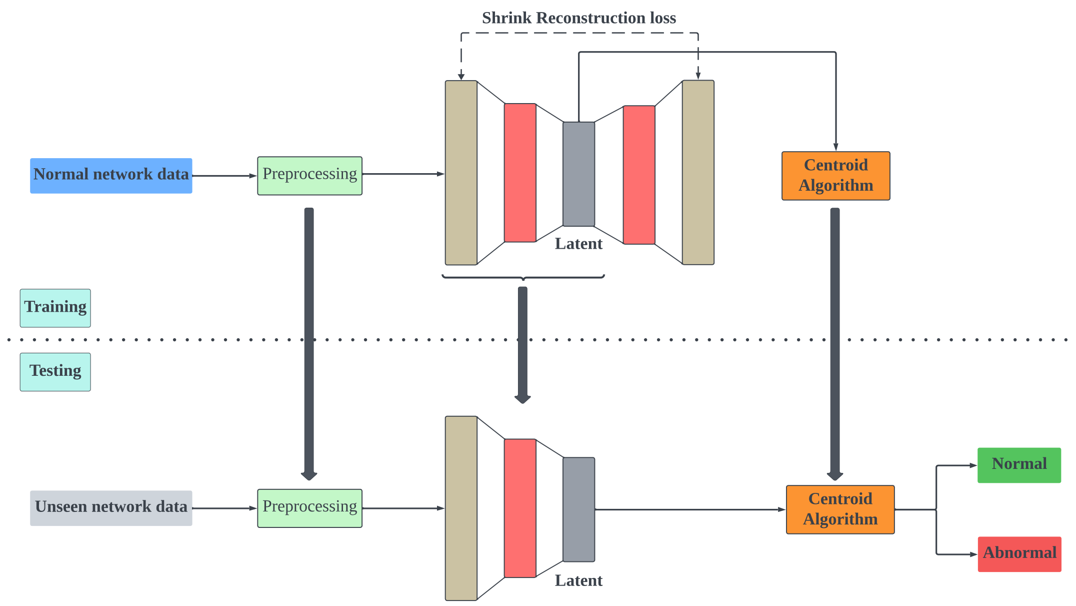
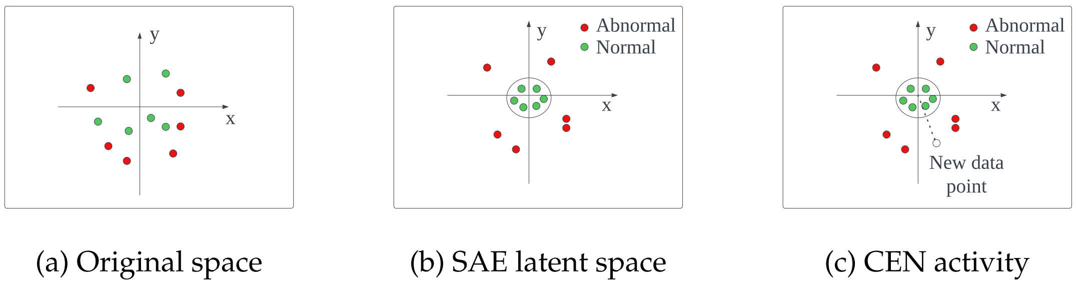


Figure 3.2: Local processing in hybrid federated learning approach

ond term is the new regularizer. The parameter λ controls the trade-off between the two terms in the equation. It is optimized using the Algorithm 1.

Figure 3.3b illustrates the SAE behavior. It will try to force the normal latent data closer to the origin.



(a) Original space (b) SAE latent space (c) CEN activity

Figure 3.3: Simulation of data representation and CEN activity

3.2.2 Centroid algorithm (CEN)

Given a normal network data set $X_0 = \{x_0^{(1)}, \ldots, x_0^{(n)}\} \subset \mathbb{R}^d$, the goal of intrusion detection is to determine whether a new data point $\mathbf{x}$ has the same

Algorithm 1 ClientUpdate: Optimize local models

```
Require: Local normal data $X = [x_0, ..., x_{N-1}] \sim p(x)$; Number of local epochs $I$; Set of mini-batches $\mathcal{B}$ each of size $N$; Learning rate $\eta$; Shrink regularizer factor $\lambda$;
Ensure: Trained Shrink Autoencoder
 1: Receive initial weights $\theta_e$ for encoder and $\theta_d$ for decoder from server
 2: for epoch = 1 to $I$ do
 3:     for each mini-batch $x_i$ in $X$ do
 4:         $z_i \leftarrow \text{Encoder}(x_i; \theta_e)$
 5:         $\hat{x}_i \leftarrow \text{Decoder}(z_i; \theta_d)$
 6:         $\mathcal{L}_{\text{SAE}} \leftarrow \frac{1}{N} \sum_{i=0}^{N-1} \|x_i - \hat{x}_i\|^2 + \lambda \frac{1}{N} \sum_{i=0}^{N-1} \|z_i\|^2$
 7:         Compute gradients $\nabla_{\theta_e} \mathcal{L}_{\text{SAE}}$ and $\nabla_{\theta_d} \mathcal{L}_{\text{SAE}}$
 8:         $\theta_e \leftarrow \theta_e - \eta \nabla_{\theta_e} \mathcal{L}_{\text{SAE}}$
 9:         $\theta_d \leftarrow \theta_d - \eta \nabla_{\theta_d} \mathcal{L}_{\text{SAE}}$
10:     end for
11: end for
12: Send $\theta_e$, $\theta_d$ to the global server
```

probability characteristics as the set X_0. A straightforward approach to anomaly detection involves estimating the probability density function (PDF) of the distribution from which the dataset X_0 is derived. If a new instance $\mathbf{x}$ is located in an area of the distribution where the density is low, it is flagged as anomalous. However, estimating the density of a distribution is a challenging task, particularly in high-dimensional spaces. Another simple way comes from clustering-based algorithms with the assumption that normal data instances lie close to their cluster centroid meanwhile abnormal data points are far from the centroid [19], such as the centroid (CEN) algorithm.

The central idea is leveraging the distance from an observation to the cluster centroid as the abnormality of the observation (Figure 3.3c). This distance is also known as the anomaly score. A higher score suggests that the data point has a higher probability of being an anomaly. By specifying a threshold, a query data point can be classified as either normal or anomalous. The CEN algorithm is a very simple algorithm with no hyper-parameters and its computational expense is small.

3.2.3 SAE-CEN combination

After receiving the best SAE model from the server, the decoder part will be dropped and the encoder will be used as a data manipulator that forms the data in a good shape to help CEN work more effectively (Figure 3.2). This also makes the time of incident response shorter. Therefore, the SAE-CEN combination can not only work powerfully in detecting anomalies but also respond to incidents rapidly.

The authors of [22] demonstrated that the CEN performed very accurately under the SAE representation data in centralized network anomaly detection. In this research, I conduct a federated learning scheme to evaluate the accuracy of this solution in the IoT intrusion detection task.

3.3 Mean Square Error-based model aggregation

The core component in all federated learning architectures is the global model aggregation. How effectively the global model is updated decides the power of the FL scheme. This section provides the design of a novel FL algorithm that boosts the performance of an Autoencoder-based model, especially SAE-CEN.

During training a machine learning model, it is necessary to control the convergence of the model on both local and global sides to prevent under-fitting and over-fitting problems. I assume that there has been a normal network dataset in the server to validate the global model convergence in each training round. This dataset contains data of all clients with uniform distribution. That means all clients have the same amount of normal data in this set and the representativeness of the whole IoT network will be ensured.

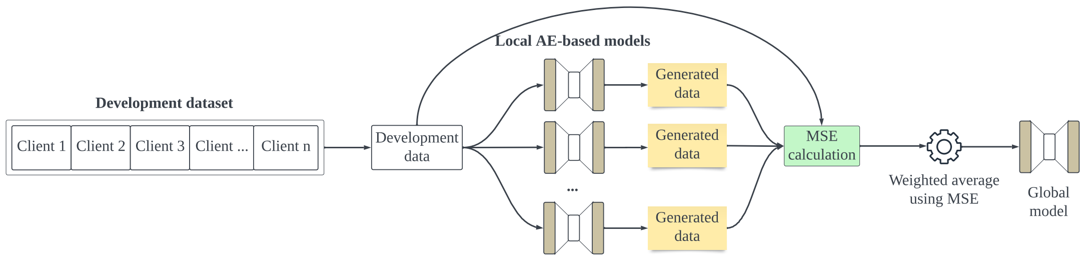


Figure 3.4: Global aggregation using MSEAvg

This research leverages as much as possible the strength of the Autoencoder-based model in modeling normal data. Therefore, I propose the **MSEAvg** algorithm to aggregate the global model by comparing the ability of each local model in reconstructing the above validation dataset. The model that works better will play a more important role in updating the global model.

Figure 3.4 and Algorithm 2 show the principle of MSEAvg and pseudo-code for the whole operation. Each local model takes development data as input, then returns the reconstruction error and is assigned a weight in the updating process. The smaller error implies a better model in learning normal data which means a higher weight. After that, the global model will be aggregated by using the Equation 3.3.

$$W_{global} = \frac{\sum_{i=1}^{n} \alpha_i \cdot W_i}{\sum_{i=1}^{n} \alpha_i} \tag{3.3}$$

Where α_i is the update weight based on MSE loss; W_i is the local model weights.

Algorithm 2 GlobalAgg: MSEAvg-based aggregation

```
Require: Number of global round $E$; Local models $L = \{L_1, L_2, \ldots, L_n\}$; Development dataset $D$;
Ensure: Updated global model $M$
 1: Initialize lists $update\_weights \leftarrow []$
 2: for each local model $L_i$ in $L$ do
 3:     Generate new development data: $\hat{D}_i \leftarrow L_i(D)$
 4:     Calculate similarity score between $D$ and $\hat{D}$: $sim\_score \leftarrow Mean\_Square\_Error(D, \hat{D})$
 5:     Compute update weight $w_i \leftarrow 1/sim\_score$
 6:     Append $(L_i, w_i)$ to $update\_weights$
 7: end for
 8: for each ($L_i$, $\alpha$) in $update\_weight$ do
 9:     Compute $M \leftarrow \frac{\sum L_i * \alpha}{\sum \alpha}$
10: end for
11: Send updated model $M$ to all clients
```

Chapter 4

Evaluation and discussion

4.1 Evaluation overview

This section overviews the background context motivating the further experimental implementation, which includes the scenarios and research questions.

4.1.1 Evaluation scenarios

Relying on the description of IID and Non-IID problems in Chapter 2, I conduct experiments on both these scenarios.

- For the IID case, I abstract an IoT network where the clients have independent and identically distributed data. In an IoT network, there are several IoT gateways, and each gateway manages many IoT devices that build a subnetwork. In this scenario, I assume all IoT subnetworks have the same network topology and device type. Therefore the data of each network will have the same distribution. Practically, this scenario describes static IoT networks with similarities between subnetworks, which do not change throughout their lifetime.

- In contrast, a non-IID IoT network will be illustrated in which the clients have non-independent and identically distributed data. This means there are differences among subnetworks in all topologies, IoT device types, and the number of data records for each device type.

4.1.2 Research questions

I mainly focus on resolving three research questions in each case of the experiments.

1. **RQ1**. How does the proposed approach work in IoT federated intrusion detection?
2. **RQ2**. How does the client selection ratio affect the federated learning intrusion detection?
3. **RQ3**. How does the proposed approach work on IoT networks of different scales?

4.2 Experiment settings

Detailed information on how I carry out the experiments will be provided in this section: IoT network construction and hyperparameters settings.

4.2.1 IoT network construction

To abstract the IoT network for this research, a small partition of the NBa-IoT dataset is used to make experiments close to practical scenarios. In this research, I consider a 10-client IoT network, which means there are 10 IoT gateways in the network. The client data is selected from NBa-IoT data using random algorithms and constraints to ensure the IID and non-IID characteristics of training data. Following the introduction in Chapter 2, I use the Dirichlet distribution $Dir_n(\alpha)$ with n is the number of clients and α as the concentration parameter and Jensen-Shannon measurement (JS) to control the non-IID characteristic of networks. To simulate a homogeneous setting, for each device type k, I sample $p_k \sim Dir_{10}(1000)$ and allocate the proportion data $p_{k,j}$ of device k for the client j. The measurement is $JS = 0.01$. I do the same process for the heterogeneous setting with the concentration parameter $\alpha = 0.1995$ and $JS = 0.83$.

Table 4.1 and Figure 4.1 show the information in the IID context. Each client contains all nine commercial IoT devices with specific data samples. The **training** and **testing** columns show dataset information of training and testing phases, respectively. The pair of numbers in **Normal data** and **Abnormal data** is formed as (IoT device id - number of data samples). For example, pair **531(5)** indicates that the `Samsung_SNH_1011_N_Webcam` device, index 5, has 531 data samples in **Client 1**.

Similarly, Table 4.2 and Figure 4.2 describe the data information in a non-IID scenario with 10 clients. Each client contains some of nine IoT devices and I depict IoT network innovation when new devices are added to the network in the testing phase.

To solve the **RQ3** when considering the performance of models in different network scales, I use the same methodology to select the data in both contexts for 20-client, 30-client, 40-client, and 50-client networks that have 15 clients, 20 clients, and 50 clients in the topology, correspondingly.

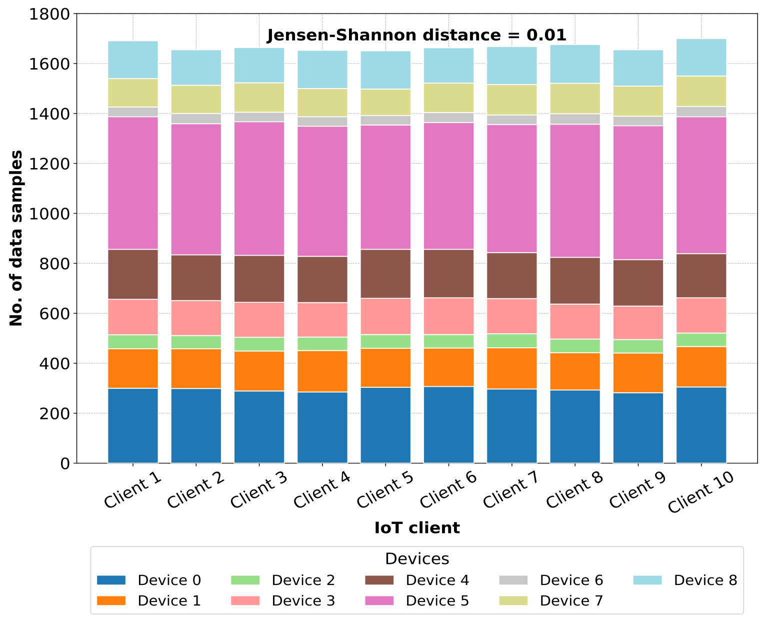


Figure 4.1: IID scenario

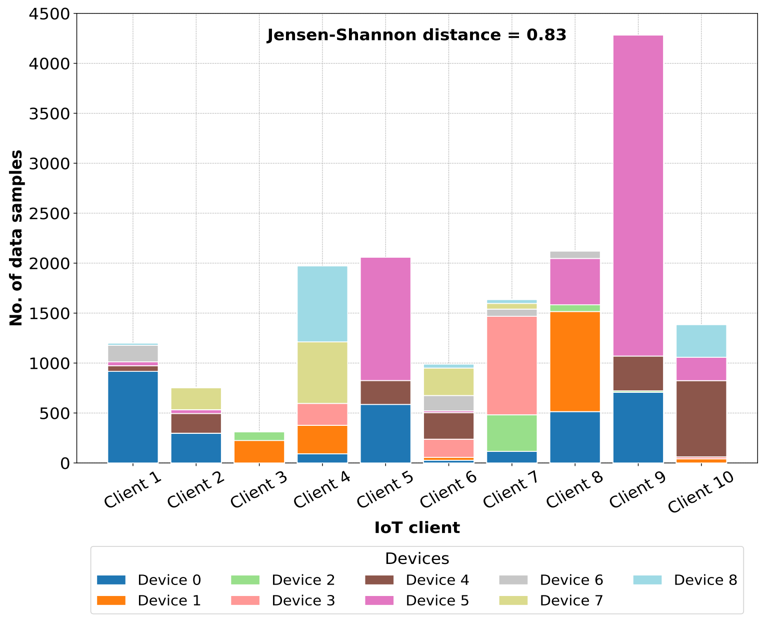


Figure 4.2: Non-IID scenario

Table 4.1: Data allocation for the IID scenario.
The pair of numbers in Normal data and Abnormal data columns is formed as (**number of data samples (IoT device id)**)

Agents	**Training**	**Testing**	
	Normal data	**Normal data**	**Abnormal data**
Client 1	531(5)	486(8)	351(5)
	300(0)	456(5)	196(0)
	200(4)	432(3)	101(1)
	158(1)	417(2)	122(4)
	151(8)	414(6)	92(3)
	142(3)	375(4)	78(7)
	114(7)	367(0)	99(8)
	56(2)	160(1)	43(1)
	39(6)	159(7)	26(6)
	1691	**3266**	**1108**
Client 2	525(5)	497(8)	351(5)
	299(0)	472(5)	196(0)
	183(4)	431(2)	101(1)
	159(1)	421(3)	122(4)
	142(8)	410(6)	92(3)
	140(3)	376(4)	78(7)
	113(7)	368(0)	99(8)
	53(2)	173(1)	43(1)
	41(6)	155(7)	26(6)
	1655	**3303**	**1108**
Client 3	535(5)	471(5)	351(5)
	289(0)	470(8)	196(0)
	188(4)	421(2)	101(1)
	160(1)	415(6)	122(4)
	141(8)	414(4)	92(3)
	140(3)	412(3)	78(7)
	118(7)	374(0)	99(8)
	55(2)	161(7)	43(1)
	38(6)	156(1)	26(6)
	1664	**3294**	**1108**

Agents	Training	Testing	
	Normal data	Normal data	Abnormal data
Client 4	521(5)	480(8)	351(5)
	285(0)	452(5)	196(0)
	185(4)	434(6)	101(1)
	166(1)	422(3)	122(4)
	153(8)	404(2)	92(3)
	138(3)	396(4)	78(7)
	113(7)	380(0)	99(8)
	54(2)	164(7)	43(1)
	38(6)	158(1)	26(6)
	1653	**3290**	**1108**
Client 5	498(5)	493(8)	351(5)
	304(0)	450(5)	196(0)
	196(4)	402(2)	101(1)
	156(1)	395(3)	122(4)
	153(8)	388(6)	92(3)
	145(3)	374(4)	78(7)
	106(7)	367(0)	99(8)
	55(2)	158(7)	43(1)
	38(6)	155(1)	26(6)
	1651	**3182**	**1108**
Client 6	508(5)	482(8)	351(5)
	307(0)	471(5)	196(0)
	194(4)	432(2)	101(1)
	154(1)	395(3)	122(4)
	147(3)	390(6)	92(3)
	141(8)	389(4)	78(7)
	118(7)	373(0)	99(8)
	54(2)	165(1)	43(1)
	40(6)	150(7)	26(6)
	1663	**3247**	**1108**
Client 7	513(5)	473(8)	351(5)
	297(0)	461(5)	196(0)
	184(4)	443(6)	101(1)
	165(1)	414(2)	122(4)
	152(8)	391(3)	92(3)
	141(3)	376(4)	78(7)
	122(7)	360(0)	99(8)
	56(2)	162(1)	43(1)
	38(6)	154(7)	26(6)

Agents	Training	Testing	
	Normal data	Normal data	Abnormal data
	1668	**3234**	**1108**
Client 8	533(5)	479(8)	351(5)
	293(0)	474(5)	196(0)
	187(4)	412(6)	101(1)
	155(8)	410(2)	122(4)
	149(1)	408(3)	92(3)
	140(3)	364(4)	78(7)
	122(7)	352(0)	99(8)
	55(2)	159(7)	43(1)
	42(6)	159(1)	26(6)
	1676	**3217**	**1108**
Client 9	536(5)	491(8)	351(5)
	282(0)	453(5)	196(0)
	186(4)	420(2)	101(1)
	159(1)	387(3)	122(4)
	145(8)	385(6)	92(3)
	134(3)	377(4)	78(7)
	121(7)	361(0)	99(8)
	54(2)	161(7)	43(1)
	38(6)	159(1)	26(6)
	1655	**3194**	**1108**
Client 10	548(5)	487(8)	351(5)
	305(0)	452(5)	196(0)
	177(4)	418(6)	101(1)
	162(1)	415(3)	122(4)
	150(8)	400(2)	92(3)
	141(3)	384(0)	78(7)
	122(7)	384(4)	99(8)
	54(2)	166(1)	43(1)
	41(6)	158(7)	26(6)
	1700	**3264**	**1108**

Table 4.2: Data allocation for the non-IID scenario.
The pair of numbers in Normal data and Abnormal data columns is formed as (**number of data samples (IoT device id)**)

Agents	**Training**	**Testing**	
	Normal data	**Normal data**	**Abnormal data**
Client 1	917(0)	1536(2)	98(0)
	166(6)	1312(0)	50(1)
	56(4)	405(3)	46(3)
	39(5)	71(1)	21(2)
	1199	**3324**	**215**
Client 2	298(8)	1096(8)	98(0)
	220(7)	941(0)	61(4)
	197(4)	766(7)	50(1)
	38(5)	173(1)	49(8)
		102(3)	39(7)
		90(6)	46(3)
		16(2)	21(2)
		13(4)	13(6)
	753	**3197**	**377**
Client 3	225(1)	732(7)	61(4)
	88(2)	326(8)	49(8)
		284(4)	39(7)
		174(6)	13(6)
	313	**1516**	**162**
Client 4	760(8)	3030(6)	98(0)
	616(7)	438(3)	50(1)
	285(1)	187(0)	46(3)
	219(3)	153(1)	61(4)
	92(0)	30(5)	175(5)
		22(2)	13(6)
			21(2)
	1972	**3860**	**403**
Client 5	1235(5)	3400(5)	175(5)
	586(0)	1364(3)	61(4)
	239(4)	468(2)	46(3)
		312(4)	21(2)
	2060	**5544**	**303**

Agents	Training	Testing	
	Normal data	**Normal data**	**Abnormal data**

Client 6	275(7)
	266(4)
	182(3)
	154(6)
	39(8)
	29(1)
	27(0)
	17(5)
		2628(4)
		1060(0)
		641(3)
		271(2)
			98(0)
			61(4)
			46(3)
			21(2)

	989	**4600**	**226**

Client 7	986(3)
	366(2)
	116(0)
	72(6)
	57(7)
	38(8)
		249(5)
		193(6)
		105(1)
		87(8)
		66(7)
		39(4)
		10(0)
			175(5)
			50(1)
			49(8)
			61(4)
			46(3)
			39(7)
			13(6)

	1635	**749**	**485**

Client 8	1002(1)
	514(0)
	464(5)
	67(2)
		920(5)
		804(3)
		418(4)
		379(8)
		313(2)
		75(6)
		13(7)
			175(5)
			61(4)
			49(8)
			46(3)
			39(7)
			21(2)
			13(6)

	2047	**2922**	**404**

Client 9	3213(5)
	708(0)
	348(4)
	14(2)
		2949(8)
		1524(2)
		334(6)
		321(3)
		44(4)
			49(8)
			61(4)
			46(3)
			21(2)
			13(6)

	4283	**5172**	**190**

Client 10	763(4)
	326(8)
	234(5)
	41(1)
	20(3)
		1109(1)
		213(6)
		171(0)
		86(4)
		12(5)
			175(5)
			98(0)
			61(4)
			46(3)
			50(1)
			13(6)

	1384	**1591**	**397**

4.2.2 Hyperparameter settings

This research studies the accuracy of Autoencoder and SAE-CEN models under FedAvg, FedProx, and MSEAvg federated learning algorithms. In intrusion detection, it is challenging to determine a threshold that accurately classifies a data point as either normal or anomalous. Thus, the Area Under the ROC Curve (AUC) metric is adopted to evaluate these models at different thresholds. I ran each experiment five times and calculated the mean accuracy and standard deviation of the AUC value.

Due to the lack of anomalies during training, the experiment hyperparameters can not be tuned, which is one of the considerable challenges of this work. I set up them using common values. The mini-batch size value is chosen as 12, learning rate $lr = 0.00001$, local epoch $I = 100$, number of global round $E = 20$, $\mu = 0.001$ for the FedProx proximal term. In each round, I select half of the clients to join the training process based on the majority rule. For the SAE shrink parameter, I test the values in the set $\{2, 5, 10, 15, 20\}$ and find value 10 is appropriate for balancing shrink loss and MSE loss. I employ the Adam optimizer [28] along with early stopping techniques [29] to train these local models. The early stopping techniques are also used in the global server to control the convergence of the global model on development data. For the number of neurons in the latent layer of the autoencoder-based model, I configure based on rules of thump as mentioned in [30] with the value $m = \left[1 + \sqrt{n}\right]$, where n is the original space dimension.

All experiments were implemented in Python language and run on one KAGAYAKI high-performance computing GPU server at Japan Advanced Institute of Science and Technology, which has an Intel Xeon GOLD 5320 52-core 2.2 GHz CPU, 512 GB DDR4/3200 SDRAM memory, and two NVIDIA A100 48 GB GPUs.

4.3 Experiment results

I do three corresponding experiments to solve research questions as mentioned before. To answer RQ1, I experiment on a 10-client IoT network to evaluate the model performance for all clients. With RQ2, I conduct the experiments in different settings of client selection ratio to examine the model robustness and find an optimal solution. Finally, I investigate the model accuracy in multiple scales of IoT networks.

4.3.1 Experiment 1: Federated intrusion detection performance

Tables 4.3 and 4.4 present the AUCs gotten by Autoencoder and SAE-CEN models under three aggregation algorithms. The results indicate that the non-IID setting poses greater challenges for machine learning models in detecting anomalies, leading to less consistency among clients compared to the IID scenario.

Table 4.3: AUCs for different agents in the IID setting

Agent	**Autoencoder**			**SAE-CEN**		
	FedAvg	**FedProx**	**MSE-Avg**	**FedAvg**	**FedProx**	**MSE-Avg**
Client 1	98.77±0.33	98.53±0.51	98.54±0.42	97.93±0.75	97.67±1.18	98.68±0.64
Client 2	97.81±0.50	97.53±0.59	97.51±0.73	97.73±0.75	97.76±0.73	97.83±0.82
Client 3	98.43±0.47	98.10±0.57	98.02±0.58	97.80±0.55	97.74±0.49	98.26±0.85
Client 4	99.42±0.20	99.39±0.25	99.32±0.30	99.22±1.00	98.52±0.82	99.31±0.70
Client 5	99.75±0.10	99.75±0.04	99.73±0.07	99.84±0.06	99.81±0.07	99.86±0.05
Client 6	99.23±0.26	99.16±0.31	99.09±0.35	98.92±0.92	98.22±0.69	99.16±0.41
Client 7	98.30±0.35	98.08±0.45	98.06±0.39	97.82±0.78	97.59±1.28	98.30±0.79
Client 8	100.00±0.00	100±0.00	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00
Client 9	99.03±0.17	98.94±0.17	98.93±0.20	98.38±0.76	98.06±1.22	98.72±0.54
Client 10	99.97±0.02	99.97±0.02	99.97±0.02	99.95±0.02	99.95±0.03	99.95±0.02
Average	***99.07±0.24***	*98.95±0.29*	*98.92±0.31*	*98.76±0.56*	*98.53±0.65*	*99.01±0.48*

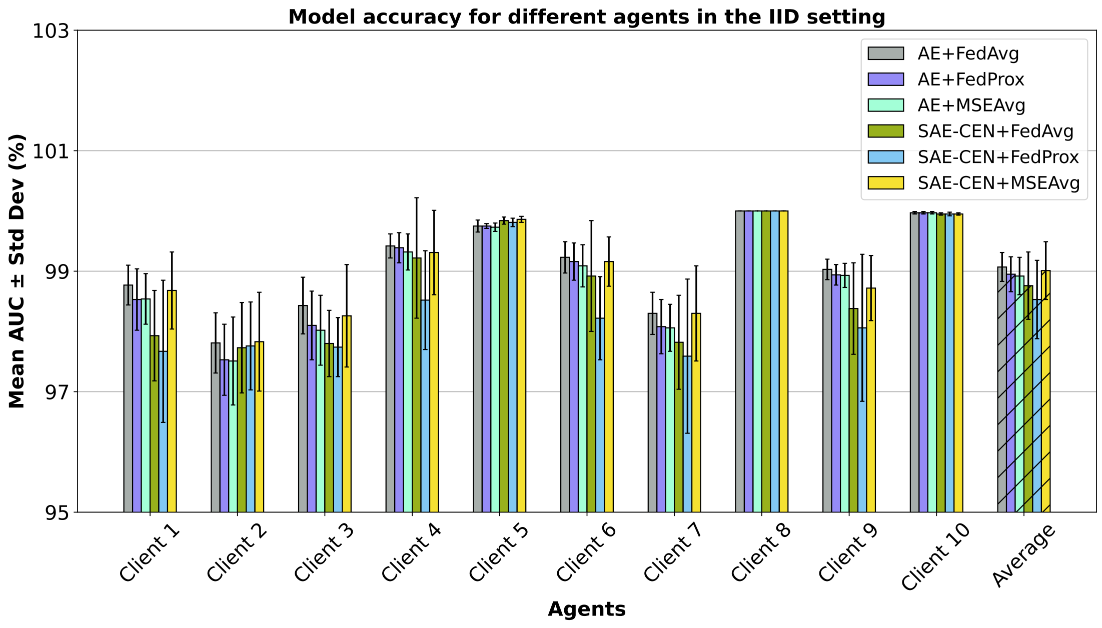


Figure 4.3: Accuracy for federated intrusion detection in the IID setting

In the IID context

In the IID case, both Autoencoder and SAE-CEN models have high accuracy for all clients under all federated learning algorithms, and the Autoencoder is slightly better than SAE-CEN in some algorithms. This suggests that the Autoencoder model can generalize effectively the whole data across clients in this case. The reason may be due to the variability of training data among clients is not much, all clients have a similar data distribution, which helps Autoencoder to easily model the data. However, SAE-CEN does not outperform Autoencoder in all algorithms. The root cause is related to the architecture of the SAE model. During training, the SAE model must balance two objectives: representing the latent data close to the origin and reconstructing normal data. As a result, the SAE-CEN model cannot show outstanding performance compared to Autoencoder in the IID scenario. This signifies that in a simple IID federated learning case, the Autoencoder model is good enough to detect anomalies accurately.

Table 4.4: AUCs for different agents in the non-IID setting

Agent	**Autoencoder**			**SAE-CEN**		
	FedAvg	**FedProx**	**MSE-Avg**	**FedAvg**	**FedProx**	**MSE-Avg**
Client 1	91.00±4.84	87.96±6.25	86.61±1.92	96.65±1.00	97.44±1.62	97.25±1.04
Client 2	99.58±0.03	99.58±0.03	99.56±0.00	99.62±0.05	99.61±0.09	99.64±0.09
Client 3	88.68±5.76	87.70±5.87	86.80±4.84	93.55±1.09	93.82±1.30	94.58±0.89
Client 4	89.42±5.14	88.72±4.93	86.14±2.48	94.40±1.11	94.41±1.66	94.97±1.03
Client 5	97.96±1.19	97.53±1.17	96.91±0.28	98.63±0.47	98.76±0.62	98.69±0.46
Client 6	92.70±3.52	91.75±3.90	89.47±1.13	93.84±2.03	95.38±0.87	94.43±0.25
Client 7	94.62±2.81	93.94±3.25	92.53±0.53	96.18±0.34	96.63±0.94	96.58±0.58
Client 8	95.84±2.14	95.47±2.20	94.18±0.94	97.51±0.55	97.87±0.50	97.73±0.07
Client 9	98.05±1.39	97.65±1.33	97.04±0.50	99.14±0.33	99.02±0.71	99.30±0.42
Client 10	99.52±0.09	99.52±0.09	99.48±0.00	99.77±0.05	99.81±0.10	99.77±0.05
Average	*94.74±2.69*	*93.98±2.90*	*92.87±1.26*	*96.93±0.70*	*97.28±0.84*	***97.30±0.49***

In the non-IID context

However, in the heterogeneous setting, the Autoencoder accuracy drops dramatically under all aggregation algorithms, meanwhile, SAE-CEN shows excellent effectiveness across all clients. This can be explained as follows: The training data distribution across clients is significantly different, and each local Autoencoder model converges in various directions. Thus, when accumulating all local models to a unique model and sending them to all gateways, the

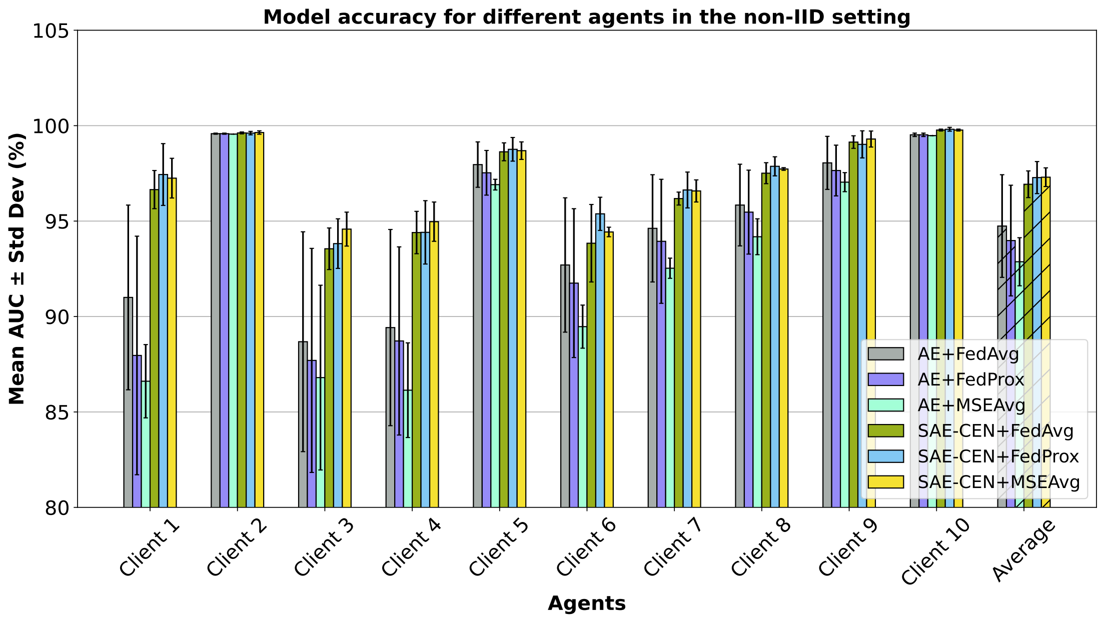


Figure 4.4: Accuracy for federated intrusion detection in the non-IID setting

new model performance will drop substantially. In contrast, each local SAE-CEN model aims to create a compact latent space for normal data when trying to force the normal latent data close to the origin. Therefore, the aggregated model will also represent the latent data approximately near the optimal area, leading to the ability to detect unseen data effectively.

The results also demonstrate the improvements brought by the MSEAvg algorithm on the SAE-CEN machine learning model, especially in the heterogeneous case. The consistency among clients also improves, as evidenced by the smaller standard deviation. FedAvg and FedProx update the global models based on the training data size of selected clients. In the non-IID setting, the client holding larger data may not ensure the representativeness of the whole data, for example, in this experiment setting (Table 4.2), Client 5 may not be better than Client 6. Instead, MSEAvg enhances this by prioritizing updates from models that better model the normal data. By giving more weight to accurate models, MSEAvg reduces the influence of noisy updates that might cause the abnormal data points to be mixed in the normal latent region. Another reason is the behavior inside the SAE model, the SAE model needs to control the trade-off between reconstruction error and shrink error. MSEAvg helps the SAE model form the normal data comprehensively while keeping the latent representation capability. Therefore SAE model can separate the normal data points better than FedAvg and FedProx.

Figure 4.5 visualizes the SAE's latent data on the testing dataset in 2D and 3D. The normal cluster that is constructed by MSEAvg is smaller, denser, and closer to the origin than that of FedAvg and FedProx algorithms. This helps the CEN model work more effectively in detecting abnormal data points, particularly on unseen data when IoT networks innovate.

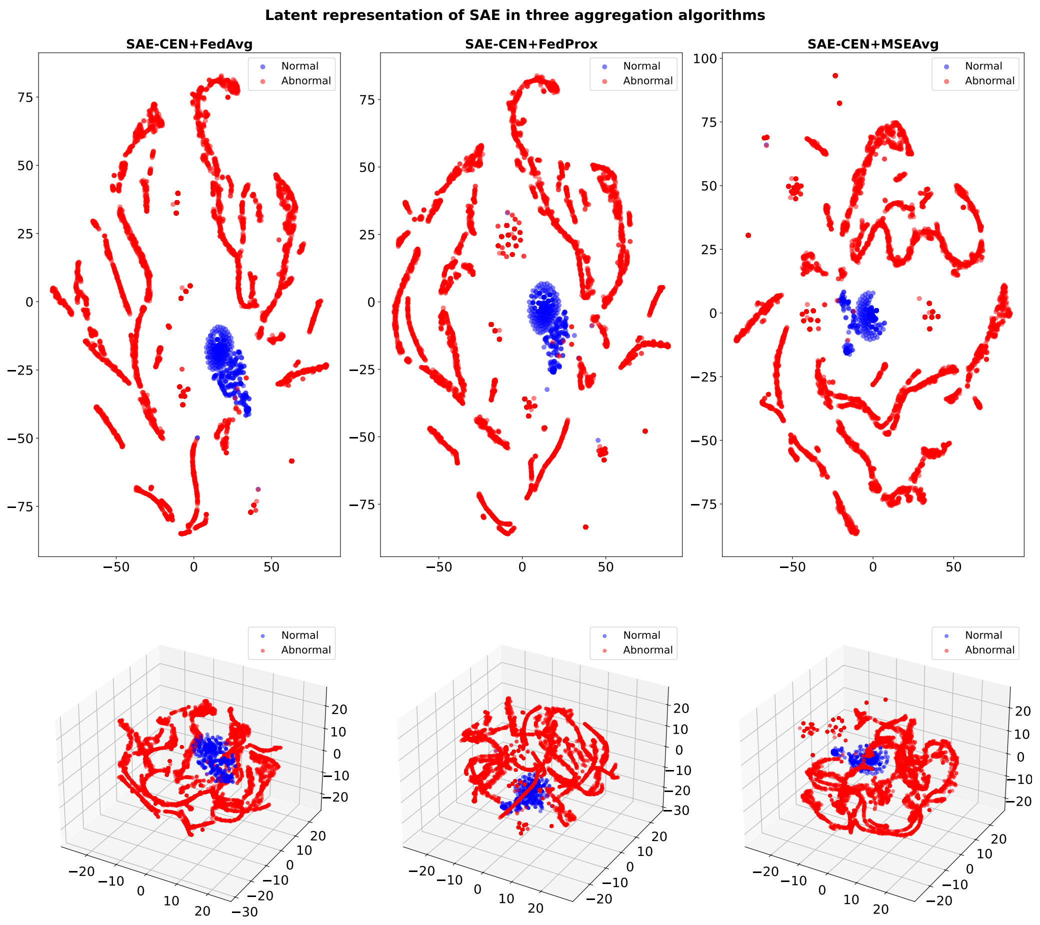


Figure 4.5: Latent representation of SAE in the federated learning scheme

4.3.2 Experiment 2: Effects of client selection ratio

In federated learning, the percentage of clients chosen to participate in the training process can significantly affect the model's performance, efficiency, and communication overhead. There is always a trade-off among these criteria in choosing an optimal client selection ratio.

Table 4.5 and Table 4.6 present the average accuracy of comparison candidates under different settings of the client selection ratio. The results indicate that the client selection ratio affects the consistency among clients and the combination of SAE-CEN and MSEAvg still shows outstanding power. This is evidenced by changes in the standard deviation and the complexity of algorithms although the mean accuracy fluctuates slightly and all FedAvg, FedProx, and MSEAvg help SAE-CEN have comparable accuracy.

Table 4.5: Performance comparison of SAE-CEN and Autoencoder based on client ratio in the IID scenario

Client ratio	**Autoencoder**			**SAE-CEN**		
	FedAvg	**FedProx**	**MSE-Avg**	**FedAvg**	**FedProx**	**MSE-Avg**
50%	**99.07±0.24**	98.95±0.29	98.92±0.31	98.76±0.56	98.53±0.65	99.01±0.48
60%	98.48±0.44	98.12±0.45	98.47±0.37	98.76±0.42	98.85±0.56	**98.96±0.68**
70%	98.13±0.70	98.17±0.56	98.04±0.54	98.51±0.74	**98.69±0.37**	98.44±0.67
80%	97.96±0.93	97.77±0.69	97.66±0.78	98.77±0.85	**98.79±0.54**	98.71±0.80
90%	97.24±0.86	97.27±0.85	97.06±0.89	98.24±1.39	**98.70±0.71**	98.60±0.56
100%	97.61±0.52	97.34±0.36	97.29±0.63	**98.72±0.56**	98.61±0.64	98.69±0.69

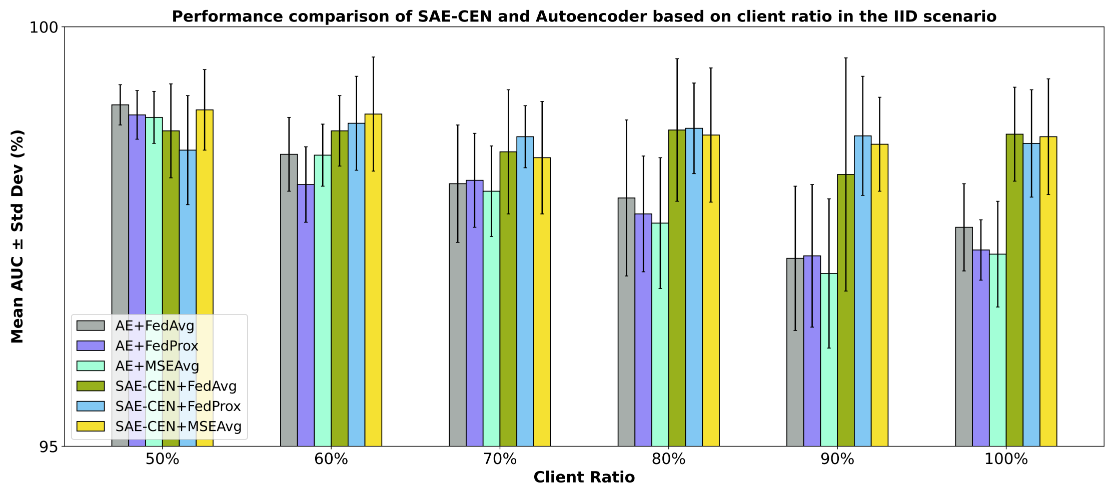


Figure 4.6: Accuracy for different client selection ratios in the IID scenario

In the IID context

Considering the IID scenario, data is uniformly distributed among clients. Therefore, each client has similar data characteristics, ensuring that even a subset of clients can provide a representative sample for model training. Therefore, the standard deviation is relatively low for all client ratios, reflecting consistent performance among clients. Autoencoder is sensitive to noise, hence, it tends to drop accuracy when updated by more clients. This is due to a bit of difference in the data distribution among clients, making more noise and variability in aggregation compared to a small ratio. The SAE-CEN model performs very accurately and consistently in all settings because of the representation ability as discussed in Experiment 1, even in different ratios.

In the non-IID context

It is more clear in the non-IID case that the standard deviation is higher, indicating more variability in performance due to the non-uniform data distribution. The model accuracy has a trend similar to the IID context, and the SAE-CEN model outperforms the Autoencoder model in all settings for both accuracy and consistency. The standard deviation of the Autoencoder model under the FedAvg and FedProx tends to be reduced when all clients participate in the updating process. This behavior occurs because including all clients ensures that the global model is updated with the complete data distribution, enhancing generalization and leading to smaller standard deviations. The MSE-Avg algorithm shows relatively lower standard deviations compared to FedAvg and FedProx, suggesting more consistent accuracy among clients. I currently select the clients randomly, which does not ensure the representativeness of clients for the population. Therefore, with smaller client selection ratios, there is a risk of over-fitting due to the bias toward the clients that are selected more times than others in the training process, which causes high variance across clients in the IoT network. However, effective aggregation strategies like MSEAvg can partially offset this by updating the global model based on a representative dataset - development dataset and prioritizing more accurate local models. This makes the global model act more consistently across clients and ensures its generalization in small client ratios.

The computational costs aspect also needs to be considered rigorously in the IoT network where IoT gateways do not have powerful resources. Higher client ratios mean more clients are participating in each training round, increasing the computational load, communication overhead, and training time. Thus, it is critical to select an efficient model and client ratio. In the IID case, it is good enough to choose Autoencoder and FedAvg as a solution for IoT feder-

Table 4.6: Performance comparison of SAE-CEN and Autoencoder based on client ratio in the non-IID scenario

Client ratio	Autoencoder			SAE-CEN		
	FedAvg	FedProx	MSE-Avg	FedAvg	FedProx	MSE-Avg
50%	94.74±2.69	93.98±2.90	92.87±1.26	96.93±0.70	97.28±0.84	**97.30±0.49**
60%	92.85±1.05	93.36±1.26	93.43±1.05	**97.31±0.68**	96.92±1.10	97.08±0.88
70%	95.28±1.45	94.39±1.36	94.09±1.97	**97.04±0.80**	97.03±0.74	97.00±1.09
80%	93.14±1.29	92.32±1.22	93.63±1.70	97.06±0.77	97.13±0.81	**97.21±0.75**
90%	92.83±1.18	92.61±0.97	92.70±1.17	**97.24±1.21**	97.15±0.73	97.11±0.88
100%	94.06±1.31	93.65±1.35	93.60±1.05	**97.30±0.71**	97.29±0.46	97.21±0.63

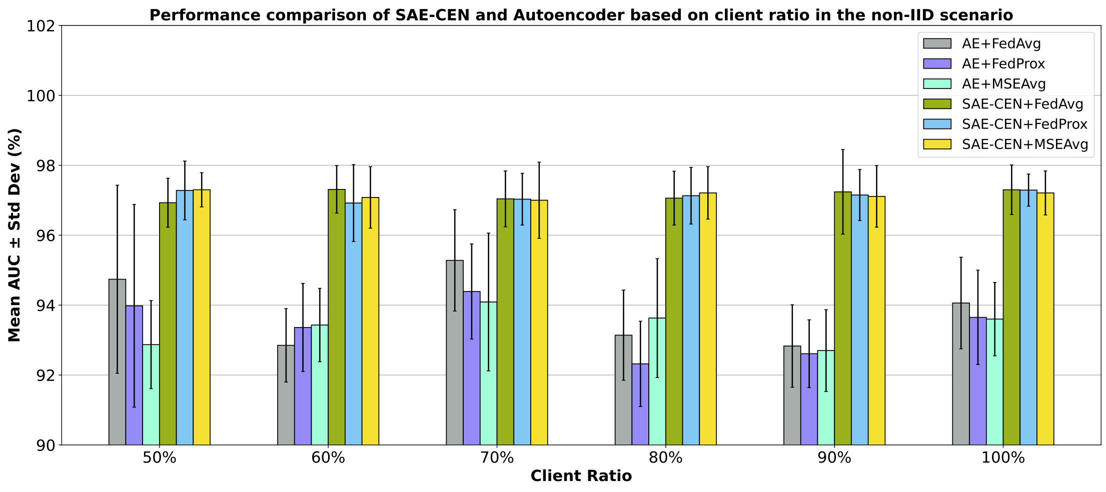


Figure 4.7: Accuracy for different client selection ratios in the non-IID scenario

ated intrusion detection for even a small client ratio setting as 50%. However, in more practically heterogeneous scenarios, the SAE-CEN model is the best selection. FedAvg and FedProx also make SAE-CEN have high accuracy in some settings because they can leverage both the information of the data size and the data characteristics to update the global model with more participants. Nevertheless, the combination of SAE-CEN and MSEAvg can get the highest accuracy and consistency even only with the smallest ratio, half of all clients. This achieves the goal of both performance and computation expenses. In both settings, it is enough to use a small ratio with 50% of all clients participating in the training process with suitable models. This improves federated intrusion detection in all mentioned criteria.

Overall, the combination of SAE-CEN and MSEAvg is the better solution for practical cases.

4.3.3 Experiment 3: Accuracy for different IoT network sizes

Table 4.7 and Table 4.8 show the models' average performance under different settings of IoT network scale with the client ratio of 50% and the similar hyperparameters to Experiment 1.

When IoT networks scale, federated learning requires careful adjustment to balance the variability ratio among clients and the ability to generalize knowledge effectively. In this research, I use an IoT device set that includes nine types of IoT appliances to illustrate practical IoT networks. The data partitions generated by the same IoT device, also known as the same distribution, may be similar. Thus, in the same client ratio setting and fixed device set, the selected data in a larger network provides a better representation of the entire network compared to a smaller network. However, the noise ratio in the aggregation process may be increased by more participants, leading to instability in federated learning.

Table 4.7: Performance comparison of SAE-CEN and Autoencoder based on network scales in the IID scenario

Network scale	**Autoencoder**			**SAE-CEN**		
	FedAvg	**FedProx**	**MSEAvg**	**FedAvg**	**FedProx**	**MSEAvg**
10-client	**99.07±0.24**	98.95±0.29	98.92±0.31	98.76±0.56	98.53±0.65	99.01±0.48
20-client	98.43±0.26	98.51±0.21	98.56±0.31	98.59±0.37	**99.02±0.39**	98.54±0.37
30-client	97.40±0.67	97.45±0.47	97.37±0.57	98.27±0.43	**98.34±0.50**	98.34±0.55
40-client	97.05±0.97	97.16±0.66	96.95±0.95	98.38±0.42	98.38±0.42	**98.45±0.39**
50-client	97.01±0.75	96.74±0.56	97.20±0.82	**98.33±0.63**	98.16±0.37	98.2±0.56

In the IID context

The Autoencoder performance tends to decrease with the expansion of the network. The cause may be because of a bit of difference in data distribution controlled by Dirichlet distribution as discussed in Experiment 2 and the sensitivity of the Autoencoder model to noise. When the network grows, more clients join in the training process, leading to a gradual increase in noise and variability. Meanwhile, it can not improve the generalization by the reason of the uniform distribution among clients. This implies that Autoencoder can not perform effectively in large-scale networks, even in IID settings. In contrast, with the powerful representation capability, SAE-CEN still maintains outstanding performance.

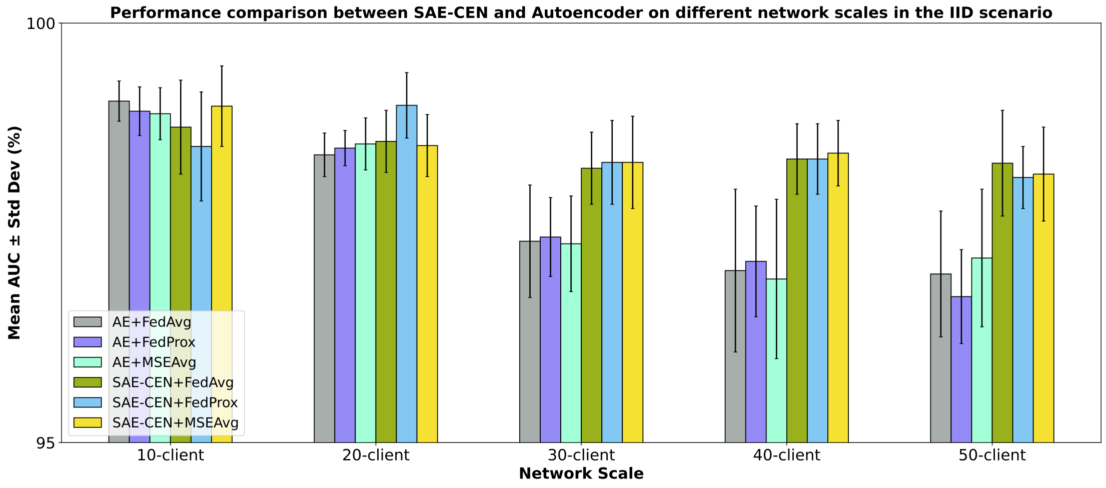


Figure 4.8: Accuracy of SAE-CEN and Autoencoder for different IoT network sizes in the IID scenario

Table 4.8: Performance comparison of SAE-CEN and Autoencoder based on network scales in the non-IID scenario

Network scale	**Autoencoder**			**SAE-CEN**		
	FedAvg	**FedProx**	**MSEAvg**	**FedAvg**	**FedProx**	**MSEAvg**
10-client	94.74±2.69	93.98±2.90	92.87±1.26	96.93±0.7	97.28±0.84	**97.30±0.49**
20-client	95.72±1.42	95.62±1.18	95.68±1.11	97.17±0.90	97.19±0.83	**97.29±0.96**
30-client	96.95±1.51	96.96±1.19	96.98±1.43	97.54±0.61	97.65±0.54	**97.73±0.56**
40-client	95.99±1.83	95.78±1.78	95.72±1.80	**97.81±0.94**	97.58±0.84	97.77±1.01
50-client	96.30±0.98	96.39±0.85	96.42±2.37	98.41±0.72	98.36±0.65	**98.52±0.51**

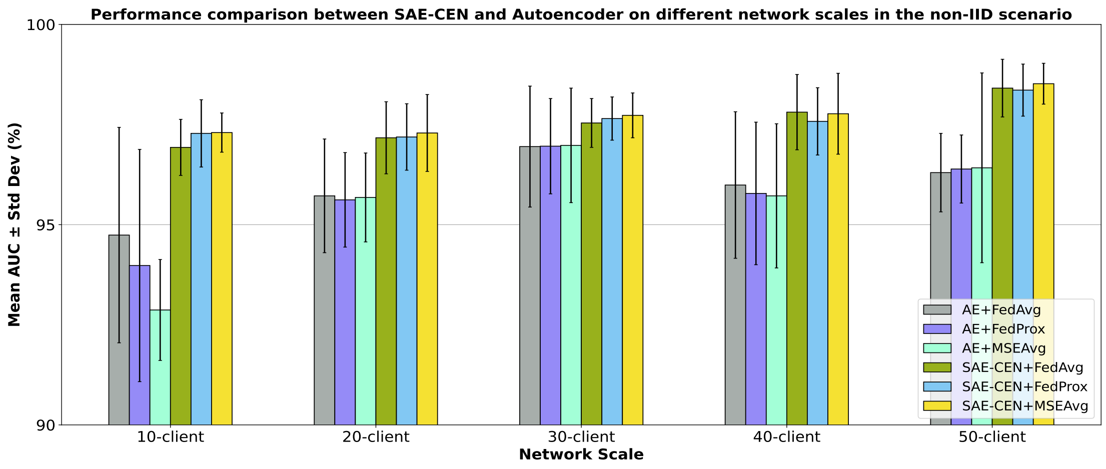


Figure 4.9: Accuracy of SAE-CEN and Autoencoder for different IoT network sizes in the non-IID scenario

In the non-IID context

The effectiveness of combining SAE-CEN and MSEAvg is more clear in the non-IID case when it shows the excellent and highest accuracy in most cases (Table 4.8). Furthermore, the accuracy of SAE-CEN demonstrates an upward trend as the network scale increases. Due to the heterogeneity of training data among clients, more joining clients bring more knowledge to the global model. This enables the SAE-CEN model to more effectively learn the normal data patterns across the entire network, especially under MSEAvg. Hence, the accuracy of SAE-CEN improves when the network size increases.

The results of this experiment indicate that the SAE-CEN model under MSEAvg still poses an outstanding power on large-scale networks, particularly in heterogeneous scenarios.

4.4 Limitations

The proposed approach demonstrates outstanding results in both effectiveness and efficiency; however, it has several limitations that need to be addressed.

The approach is sensitive to hyperparameters, which poses a significant challenge. Since there is no abnormal data available for tuning these hyperparameters, finding the optimal values for model parameters is difficult and may affect the overall performance. Especially in the scenario that the IoT network is changing during use. Furthermore, the MSEAvg algorithm requires additional server-side computations to assign weights to local models, increasing the training time and computational costs.

The current method of randomly selecting clients to update the global model is not optimal for federated learning, particularly in heterogeneous data distributions. An improved client selection strategy that ensures a more representative subset of clients could enhance the effectiveness of the global model updates.

Chapter 5

Conclusion

5.1 Summary

This thesis presents a novel approach to enhancing intrusion detection in IoT networks using federated learning. Traditional centralized machine learning methods are increasingly impractical due to privacy concerns, communication overheads, and the sheer volume of data generated by IoT devices. Federated learning (FL) emerges as a promising solution, enabling decentralized model training while keeping data localized on client devices.

We proposed a semi-supervised federated learning approach utilizing the Autoencoder model to improve IoT intrusion detection. The core contributions of this research include the development of a hybrid Shrink Autoencoder with the Centroid algorithm (SAE-CEN) and a novel mean square error-based aggregation algorithm (MSEAvg). These innovations address critical challenges such as data heterogeneity, unbalanced data distributions, resource constraints, and the lack of labeled abnormal data prevalent in IoT environments, especially the issue of the innovation of IoT networks changing in both topology and data characteristics.

Our experimental evaluation using the NBa-IoT dataset demonstrated that the proposed SAE-CEN model, combined with the MSEAvg aggregation algorithm, not only significantly enhances detection accuracy and robustness in heterogeneous IoT networks. These findings confirm the both effectiveness and efficiency of my approach in real-world scenarios, providing a practical framework for IoT intrusion detection.

5.2 Future work

While the results of this research are already valuable, there are several future work directions to further enhance the proposed approach and address the remaining challenges, as follows:

- It is necessary to implement an experimental framework to evaluate the prediction performance for malicious activities and bring the experiments to practical cases.
- My proposed aggregation algorithm still does not outperform the popular ones in all settings. Therefore, some techniques such as hierarchy learning could be integrated to boost the performance.
- The hyperparameters tunning is still a challenging task in semi-supervised learning, especially in this thesis. In the future, I will investigate and find the optimal set of hyperparameters for real-world cases.

In conclusion, this thesis has provided a solid foundation for using federated learning in IoT intrusion detection. By addressing the identified future work areas, the proposed framework can be further refined and adapted to meet the evolving security needs of IoT networks.

Bibliography

[1] Mohammad Derawi, Yaser Dalveren, and Faouzi Alaya Cheikh. “Internet-of-Things-Based Smart Transportation Systems for Safer Roads”. In: *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. 2020, pp. 1–4. DOI: 10.1109/WF-IoT48130.2020.9221208.

[2] Abderahman Rejeb et al. “The Internet of Things (IoT) in healthcare: Taking stock and moving forward”. In: *Internet of Things* 22 (2023), p. 100721. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2023.100721.

[3] Preeti Yadav and Sandeep Vishwakarma. “Application of Internet of Things and Big Data towards a Smart City”. In: *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*. 2018, pp. 1–5. DOI: 10.1109/IoT-SIU.2018.8519920.

[4] Bhawana Sharma et al. “Anomaly based network intrusion detection for IoT attacks using deep learning technique”. In: *Computers and Electrical Engineering* 107 (2023), p. 108626. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2023.108626. URL: https://www.sciencedirect.com/science/article/pii/S0045790623000514.

[5] Ly Vu et al. “Learning Latent Representation for IoT Anomaly Detection”. In: *IEEE Transactions on Cybernetics* 52.5 (2022), pp. 3769–3782. DOI: 10.1109/TCYB.2020.3013416.

[6] Meryem Janati Idrissi et al. “Fed-ANIDS: Federated learning for anomaly-based network intrusion detection systems”. In: *Expert Systems with Applications* 234 (2023), p. 121000. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.121000. URL: https://www.sciencedirect.com/science/article/pii/S0957417423015026.

[7] Xiaofeng Wang et al. “Federated deep learning for anomaly detection in the internet of things”. In: *Computers and Electrical Engineering* 108 (2023), p. 108651. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2023.108651. URL: https://www.sciencedirect.com/science/article/pii/S0045790623000769.

[8] Badra Souhila Guendouzi et al. "A systematic review of federated learning: Challenges, aggregation methods, and development tools". In: *Journal of Network and Computer Applications* 220 (2023), p. 103714. ISSN: 1084-8045. DOI: `https://doi.org/10.1016/j.jnca.2023.103714`. URL: `https://www.sciencedirect.com/science/article/pii/S1084804523001339`.

[9] Sawsan Abdulrahman et al. "A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond". In: *IEEE Internet of Things Journal* 8.7 (2021), pp. 5476–5497. DOI: `10.1109/JIOT.2020.3030072`.

[10] T. D. Nguyen et al. "DÏoT: A Federated Self-learning Anomaly Detection System for IoT". English. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. International Conference on Distributed Computing Systems. International Conference on Distributed Computing Systems , ICDCS ; Conference date: 07-07-2019 Through 10-07-2019. United States: IEEE, 2019, pp. 756–767. DOI: `10.1109/ICDCS.2019.00080`.

[11] Viraaji Mothukuri et al. "Federated-Learning-Based Anomaly Detection for IoT Security Attacks". In: *IEEE Internet of Things Journal* 9.4 (2022), pp. 2545–2554. DOI: `10.1109/JIOT.2021.3077803`.

[12] Xunzheng Zhang et al. "Federated Feature Selection for Horizontal Federated Learning in IoT Networks". In: *IEEE Internet of Things Journal* 10.11 (2023), pp. 10095–10112. DOI: `10.1109/JIOT.2023.3237032`.

[13] H. B. McMahan et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *International Conference on Artificial Intelligence and Statistics*. 2016. URL: `https://api.semanticscholar.org/CorpusID:14955348`.

[14] Tian Li et al. "Federated Optimization in Heterogeneous Networks". In: *Proceedings of Machine Learning and Systems*. Vol. 2. 2020, pp. 429–450.

[15] Yair Meidan et al. "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders". In: *IEEE Pervasive Computing* 17.3 (2018), pp. 12–22. DOI: `10.1109/MPRV.2018.03367731`.

[16] Rebecca Bace and Peter Mell. *Intrusion Detection Systems*. en. Nov. 2001.

[17] Anna L. Buczak and Erhan Guven. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". In: *IEEE Communications Surveys & Tutorials* 18.2 (2016), pp. 1153–1176. DOI: `10.1109/COMST.2015.2494502`.

[18] Shaashwat Agrawal et al. “Federated Learning for intrusion detection system: Concepts, challenges and future directions”. In: *Computer Communications* 195 (2022), pp. 346–361. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2022.09.012. URL: https://www.sciencedirect.com/science/article/pii/S0140366422003516.

[19] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: https://doi.org/10.1145/1541880.1541882.

[20] Eirini Anthi et al. “A Supervised Intrusion Detection System for Smart Home IoT Devices”. In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 9042–9053. DOI: 10.1109/JIOT.2019.2926365.

[21] Ming Zhang, Boyi Xu, and Jie Gong. “An Anomaly Detection Model Based on One-Class SVM to Detect Network Intrusions”. In: *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*. 2015, pp. 102–107. DOI: 10.1109/MSN.2015.40.

[22] Van Loi Cao, Miguel Nicolau, and James McDermott. “Learning Neural Representations for Network Anomaly Detection”. In: *IEEE Transactions on Cybernetics* 49.8 (2019), pp. 3074–3087. DOI: 10.1109/TCYB.2018.2838668.

[23] Markus M. Breunig et al. “LOF: identifying density-based local outliers”. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’00. New York, NY, USA: Association for Computing Machinery, 2000, 93–104. ISBN: 1581132174. DOI: 10.1145/342009.335388. URL: https://doi.org/10.1145/342009.335388.

[24] Abiodun M. Ikotun et al. “K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data”. In: *Information Sciences* 622 (2023), pp. 178–210. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2022.11.139. URL: https://www.sciencedirect.com/science/article/pii/S0020025522014633.

[25] Mikhail Yurochkin et al. “Bayesian Nonparametric Federated Learning of Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 7252–7261. URL: http://proceedings.mlr.press/v97/yurochkin19a.html.

[26] Durmus Alp Emre Acar et al. "Federated Learning Based on Dynamic Regularization". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=B7v4QMR6Z9w.

[27] Hongyi Wang et al. "Federated Learning with Matched Averaging". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=BkluqlSFDS.

[28] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.

[29] Lutz Prechelt. "Early Stopping — But When?" In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_5. URL: https://doi.org/10.1007/978-3-642-35289-8_5.

[30] Van Loi Cao, Miguel Nicolau, and James McDermott. "A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection". In: *Parallel Problem Solving from Nature – PPSN XIV*. Ed. by Julia Handl et al. Cham: Springer International Publishing, 2016, pp. 717–726. ISBN: 978-3-319-45823-6.