AI

MDPI

Article

Federated Learning for IoT Intrusion Detection

Riccardo Lazzarini [1,*], Huaglory Tianfield [1,*] and Vassilis Charissis [2]

[1] School of Computing, Engineering and Built Environment, Glasgow Caledonian University (GCU), Glasgow G4 0BA, UK
[2] School of Arts and Creative Industries, Edinburgh Napier University, Edinburgh EH10 5DT, UK ; v.charissis@napier.ac.uk
* Correspondence: rlazza200@caledonian.ac.uk (R.L.); h.tianfield@gcu.ac.uk (H.T.)

Abstract: The number of Internet of Things (IoT) devices has increased considerably in the past few years, resulting in a large growth of cyber attacks on IoT infrastructure. As part of a defense in depth approach to cybersecurity, intrusion detection systems (IDSs) have acquired a key role in attempting to detect malicious activities efficiently. Most modern approaches to IDS in IoT are based on machine learning (ML) techniques. The majority of these are centralized, which implies the sharing of data from source devices to a central server for classification. This presents potentially crucial issues related to privacy of user data as well as challenges in data transfers due to their volumes. In this article, we evaluate the use of federated learning (FL) as a method to implement intrusion detection in IoT environments. FL is an alternative, distributed method to centralized ML models, which has seen a surge of interest in IoT intrusion detection recently. In our implementation, we evaluate FL using a shallow artificial neural network (ANN) as the shared model and federated averaging (FedAvg) as the aggregation algorithm. The experiments are completed on the ToN_IoT and CICIDS2017 datasets in binary and multiclass classification. Classification is performed by the distributed devices using their own data. No sharing of data occurs among participants, maintaining data privacy. When compared against a centralized approach, results have shown that a collaborative FL IDS can be an efficient alternative, in terms of accuracy, precision, recall and F1-score, making it a viable option as an IoT IDS. Additionally, with these results as baseline, we have evaluated alternative aggregation algorithms, namely FedAvgM, FedAdam and FedAdagrad, in the same setting by using the Flower FL framework. The results from the evaluation show that, in our scenario, FedAvg and FedAvgM tend to perform better compared to the two adaptive algorithms, FedAdam and FedAdagrad.

Keywords: Internet of Things; intrusion detection systems; federated learning; deep learning

Citation: Lazzarini, R.; Tianfield, H.; Charissis, V. Federated Learning for IoT Intrusion Detection. *AI* **2023**, *4*, 509–530. https://doi.org/10.3390/ai4030028

Academic Editor: Giovanni Diraco

Received: 23 May 2023
Revised: 26 June 2023
Accepted: 11 July 2023
Published: 24 July 2023

1. Introduction

The Internet of Things (IoT) is a network of interconnected smart devices that contribute towards generating and gathering enormous amounts of data [1]. IoT devices are now used in every area of our daily life. Examples span from our home, with devices such as smart appliances and entertainments systems, to smart city and its invisible infrastructure such as pedestrian and road sensors. The smart grid, autonomous automobile systems, smart medical devices, industrial control systems and robotics are just some of the areas in which IoT devices are being used on a daily basis. Data gathered by all of these devices requires storage and analysis. To obtain insights into this data and enable intelligent applications, techniques such as machine learning (ML) have been widely deployed [2]. Detection of cyber attacks is part of these intelligent applications. Given the continuous increase in numbers of cyber attacks in IoT infrastructure [3], monitoring this high volume of data for the detection of cyber attacks is critical. However, it can be achieved only through the use of automated methods based on ML and deep learning (DL) [4]. DL is a branch of ML that has become widely popular in many fields, including science, finance, medicine and engineering [5]. DL for intrusion detection has also become increasingly

popular as it allows for a more sophisticated analysis of network traffic and more precise detection of anomalies compared to traditional ML methods [6]. DL models tend to achieve better performance and accuracy over ML models in highly complex environments, where large volumes of data exist [7], in exchange for more computational power. While different, ML and DL methods share a similar procedure. They both require data and model to be available at a central location. In other words, these systems are centralized. Data is captured in remote locations and transferred to a central repository where it is processed in preparation for classification. This works well in environments limited to a single organization, with sites located in the same geographic area. In contrast, in organizations with multiple sites or in the event of collaboration between organizations, data transfers could compromise privacy [8]. Moreover, data volumes represent another challenge. Captures from IDS sensors tend to be quite large. Transferring these large volumes of data to a central location for classification could represent a serious bottleneck for the network [9]. Furthermore, given that some IoT applications are latency critical [10,11], transfers of data to centralized location could compromise their correct functionality.

Federated learning (FL) is one of the latest paradigms in the area of ML that can be used to address these challenges. FL was introduced by Google in 2017 [12] with the aim of addressing issues related to data privacy. It uses a distributed environment where participating clients complete analysis of their own data with no need of transfers. Instead, clients share a model used for training their data. Only parameter updates are exchanged with a server that takes the role of the aggregator. The server coordinates clients until training is completed and performs aggregation of their weights and results through the use of an averaging algorithm such as federated averaging (FedAvg) [12].

Given that IoT networks are fundamentally distributed, FL can be applied to address the limitations of a centralized approach in IoT intrusion detection, as it can analyze traffic and identify attacks as close to the source of data as possible. However, FL approaches to IoT IDS are still at their infancy and require further evaluation before they can be deployed in real-world scenarios [13].

In this article, we evaluate how FL performs in a scenario where four distributed clients collaborate to classify attacks in the ToN_IoT [14] and the CICIDS2017 [15] datasets. The objective is to evaluate FL as an alternative to a typical DL method where data is stored and analyzed at a single location. The federated system created here uses horizontal data partitioning, where each client participating in the process owns different data samples but with the same dimensional space as every other client. Data from the two datasets is randomly divided so that each client has access to its own portion. No data sharing occurs between clients. On the other hand, the model, which is a shallow artificial neural network (ANN), is shared amongst clients, with parameter updates exchanged with the server for aggregation, as previously explained. Results are then compared against a centralized approach, using the same ANN model.

Key contributions of this article can be summarized as follows:

- We propose a method for the detection of attacks in IoT network environments based on a FL framework that uses FedAvg as the aggregation function. The distributed framework is composed of four clients, sharing a shallow ANN, and a server acting as the aggregator. The primary objective is the evaluation of FL as an approach to the detection and classification of attacks in an IoT network environments.
- We evaluate the framework on two open-source datasets, namely ToN_IoT and CICIDS2017, on both binary and multiclass classification. Our method offers a high level of accuracy with a low False Positive (FP) rate in both types of classification for both datasets.
- We compare results from our experiments against a centralized approach based on the same model, showing that performance of our FL framework is comparable to its centralized counterpart.
- In this scenario, we evaluate three alternative aggregation methods, namely FedAvgM, FedAdam and FedAdagrad, and compare their performances against FedAvg.

The remainder of this article is organized as follows: Section 2 provides a review of the related work in the area of IDS in the IoT environment using FL. Section 3 proposes the FL method for intrusion detection in IoT environments. Section 4 describes the datasets and performance metrics used. Results are discussed in Section 5. The conclusion is drawn in Section 6.

2. Literature Review

The term federated learning (FL) was firstly introduced in 2017 in a paper published at Google [12]. In their work the authors developed a decentralized approach, namely federated learning, with the objective of ensuring data privacy of participating clients. In FL, a server or aggregator takes the role in coordinating several clients into analyzing data using a shared model. The data owned by the clients remains with the data owner and it is never transferred between devices. The model is shared amongst clients and only parameter updates are exchanged. As explained by [12], when implementing FL, several key constraints need taking into consideration in order to optimize a solution to the problem. These constraints exist as FL must be able to train data with the following characteristics [12,16]:

- **Non-IID**—Data stored locally in a device is not a representation of the entire population distribution.
- **Unbalanced**—Local data has a large variation in size. In other words, some devices will train on larger datasets compared to others.
- **Massively Distributed**—Large number of clients.
- **Limited Communication**—Communication amongst clients is not guaranteed as some may be offline. Training may be completed with a smaller number of devices or asynchronously.

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

2.1. Federated Learning in IoT Intrusion Detection

Being a technology that often requires devices to connect to a central location remotely, the IoT fits well with the FL paradigm. Similarly, IDSs are normally structured as a distributed environment, making the use of FL in IoT intrusion detection even more appropriate. In fact, FL for IoT IDSs has seen a surge of interests in recent years. Research work in this area covers many aspects of this new technology. In this section, we discuss the current literature in the area of IoT intrusion detection using FL.

Sarhan et al. [17] presented a Cyber Threat Intelligence sharing scheme based on federated learning. The idea presented in their work is to create a framework to allow independent organizations to share their knowledge of cyber threats. Each organization can use a common global model, provided by an orchestrating server, to analyze their data. A federated averaging algorithm is used to aggregate results and update parameters to allow the model to adapt continuously to achieve better performance in the detection of threats. The framework requires each organization to maintain their local data using a common logging format and feature set. They compared results obtained from their framework with a centralized model, where data from all organization is stored at a single location, and a localized model where each organization complete their own analysis. The FL model achieved results generally on par with the other models, demonstrating that FL can be used efficiently in a collaborative intrusion detection system. However, the assumption of having a common feature set amongst different organization could be a limitation of their work, particularly in relation to IoT environments, where devices produce a large

amount of heterogeneous traffic, which can create difficulties in creating a commonly structured dataset.

Another work based on FL is given in [18]. They have used long short-term memory (LSTM) as the basis for their FL model, which was tested against a modified dataset of system calls created by AT&T several years ago. Results were compared against standard models based on LSTM and convolutional neural networks (CNNs). While the results presented were positive, their value is undermined by the use of an old dataset, which may not represent the type of command set commonly used in devices nowadays.

A FL method based on LSTM is given by [19] to identify false data injection attacks in solar farms. They have used a traffic generator to build their own dataset to test their method. Results show that their model offers efficient attack detection, improving over a standard centralized method based on the same LSTM model.

FL is also used in [20] as a method to detect attacks in IoT environments. Their work combines FL with ensemble learning. Each node uses a gated recurrent unit (GRU)-based model to perform the classification. Weights are updated globally using federated averaging. The outcome from classification by each node is used as input to an ensemble model based on a random forest algorithm. Using a dataset based on Modbus traffic, the authors have achieved promising results.

Zhang et al. [21] developed a platform named FeDIoT that uses FL on real devices to detect anomalies in IoT traffic. Using the N-BaIoT [22] dataset and the LANDER dataset [23] they employed a model based on Auto-Encoder run by the clients in their FL network. Results from their experiments demonstrate that their method can be an efficient technique in detecting attacks.

An interesting approach is proposed by [24]. They built a model using FL and an ensemble stacking approach during aggregation of results from clients. Their idea is to collect parameters from participating clients and concatenate them into a matrix. This is subsequently used with some test data by the aggregator to obtain a final result. They named their aggregation method FedStacking and tested it with multilayer perceptron (MLP) models running on several clients.

The authors in [25] proposed a FL framework in support of fog-based resource-constrained IoT devices. They named their approach Fog-FL and they have used an interesting approach of using local fog nodes as aggregators for the FL network. Rather than having a central aggregator communicating directly with distributed nodes, they added an additional layer of aggregators selected based on geospatial location. The approach selects one of these nodes as the global aggregator at each FL round. According to the authors, this process increases the efficiency of the system in terms of power consumption and communication delays considerably.

An interesting work is given by Chen et al. in [26]. Their work proposes a novel method for intrusion detection in wireless edge networks, named Federated Learning-based Attention Gated Recurrent Unit (FedAGRU). Their method demonstrated an improved communication in exchanging model updates compared to the FedAvg aggregation model. At the same time they achieved superior accuracy in detection of attacks when compared to a centralized CNN model.

An anomaly-based IDS using FL in Industrial IoT (IIoT) networks is proposed by Zhang et al. [27]. They adopted an instance-based transfer learning approach using ensemble techniques and proposed a novel aggregation algorithm based on a weighted voting approach. Their method achieved a superior detection performance when compared with a centralized model in multiclass classification.

Campos et al. [13] proposed an evaluation of an FL-enabled IDS approach, where they used three different settings with the ToN_IoT dataset. Using the IBMFL library they also tested different aggregation functions in the same scenarios with excellent results.

Several other relevant works have been presented in the area of IoT intrusion detection including [28–32]. Table 1 presents a summary of work applying FL for IoT intrusion detection.

Table 1. Summary of FL being applied for IoT intrusion detection.

Author	Dataset	Shared Model	Aggregation Function	No. of Clients	FL Library
Sarhan et al. [17]	NF-UNSW-NB15 NF-BoT-IoT	LSTM DNN	FedAvg	-	-
Zhao et al. [19]	Proprietary	LSTM	-	4	Flower
Mothukuri et al. [20]	MODBUS Network Data	GRU Random Forest	FlAverage	-	Pysyft
Zhang et al. [27]	CICIDS2017 CICIDS2018	Adaboost and RF	Weighed Voting FedAvg	5	-
Zhang et al. [21]	N-BaIoT USC LANDER IoT	Auto-Encoder		9	
Chatterjee et al. [24]	NSL-KDD DS2OS Traffic Gas Pipeline Data Water Tank Data	MLP Stacking Ensemble	FedStacking	4	-
Saha et al. [25]	MNIST	MLP	FogFL	6	-
Zhao et al. [18]	SEA Dataset	LSTM	FedAvg	4	Tensorflow
Campos et al. [13]	ToN_IoT	Logistic Regression	FedAvg Fed+	10	IBMFL
Chen et al. [26]	KDD CUP 99 CICIDS2017 WSN-DS	GRU-SVM	FedAGRU FedAvg	up to 50	Pysyft

2.2. Averaging Algorithms

FedAvg is an algorithm based on a federated version of stochastic gradient descent (SGD), namely FedSGD, which was also proposed on the original FL paper from Google [12] as a baseline for FedAvg. FedSGD uses a randomly selected client to complete a single batch gradient calculation for every round of communication. The average gradient on its local data is sent back to the server which, in turn, aggregates them and applies the update to the model. FedAvg is a generalization of FedSGD, where the client updates the weights, rather than the gradient, multiple times before it is sent to the server for aggregation. FedAvg makes it possible for a network of clients to train ML and DL models collectively but still using their local data. This is the basis for a successful FL network as it removes the need for clients to upload data to a centralized server, hence allowing the main requirements of privacy to be met. The pseudocodes of FedAvg are given in Algorithm 1.

FedAvg offers good performance in non-heterogeneous data. However, it is now established that the more heterogeneous the data the longer FedAvg takes to converge [33–35]. As a consequence research has been carried out to offer alternative solutions to FedAvg, to improve on it or to be used in specific scenarios. Several alternative methods have been proposed in the literature [34–40] to address limitations of FedAvg.

FedAdam and FedAdagrad have been proposed together in [40] as server-side methods to improve on FedAvg in situations where the noise distribution is high. The pseudocodes for both algorithms are presented in Algorithm 2. In Lines 15 and 16 of the pseudocodes, either FedAdagrad or FedAdam is to be selected as the rest of the algorithm is the same for both.

Algorithm 1 The FedAvg Algorithm. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs and η is the learning rate

Server executes:
initialize w_0
for each round $t = 1, 2 \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $St \leftarrow$ (random set of m clients)
  for each client $k \in St$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  end for
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

ClientUpdate(k, w) : *//run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in B$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
  end for
end for
Return w to Server

Algorithm 2 The FedAdam and FedAdagrad Algorithms

Input:$x_0, v_{-1} \geq r^2$, optional $\beta_1, \beta_2 \in (0, 1)$ for FedAdam
for $t = 0, \dots, T-1$ **do**
  Sample a subset $\mathcal{S}$ of clients
  $x_i^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $e = 1, \dots, E$ **do**
      for $b \in \mathcal{B}_i$ **do**
        $x_i^t = x_i^t - \eta_t \nabla f_i(x_i^t; b)$
      end for
    end for
    $\Delta_i^t = x_i^t - x_t$
  end for
  $n = \sum_{i \in \mathcal{S}} n_i, \Delta_t = \sum_{i \in \mathcal{S}} \frac{n_i}{n} \Delta_i^t$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\Delta_t$
  $v_t = v_{t-1} + \Delta_t^2$ **(FedAdagrad)**
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2)\Delta_t^2$ **(FedAdam)**
  $x_{t+1} = x_t + \eta \frac{m_t}{\sqrt{v_t} + r}$
end for

Another alternative algorithm is Federated averaging with Momentum or FedAvgM [35]. The pseudocodes for this are presented in Algorithm 3. Notice that the algorithm is practically the same as FedAvg at the server side. However, it does change how the clients calculate the weights. Using Nesterov accelerated gradient [41], a momentum is added to improve the calculation of weights when data contains too much noise. A momentum is an improvement to standard SGD accelerating the process of finding the best minimum when calculating the gradient [42]. Given that SGD has a limitation that can make it stagnant in flat areas in noisy environments, a momentum can be used as an approach to accelerate the progress of the search of the minimum without getting stuck. Nesterov accelerated gradient is a further improvement of the standard momentum as it updates parameters according to the previous momentum and then corrects the gradient to achieve the parameter updating [43].

Algorithm 3 The FedAvgM Algorithm. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs and η is the learning rate

Server executes:
initialize w_0
for each round $t = 1, 2 \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $St \leftarrow$ (random set of m clients)
  for each client $k \in St$ **in parallel do**
    $w_{t+1}^k \leftarrow$ Client Update (k, w_t)
  end for
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

Client Update(k, w) : *//run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in B$ **do**
    $v = \beta v + \Delta w$
    $w \leftarrow w - v$
  end for
end for
Return w to Server

2.3. Federated Learning Frameworks

While the use of FL is quite recent, several Python libraries exist for the development of its applications. For instance, as a part of TensorFlow, Google created TensorFlow Federated or TFF [44], which is an open-source framework for ML methods applied to decentralized data. According to their website, TFF was created to facilitate open research and experimentation using FL. Another popular library for FL is PySyft [45], recently renamed Syft. This was created by OpenMined and it's an open-source stack that focuses on providing FL with secure and private communication. IBM also created their own FL framework [46] which they named IBM Federated Learning. This is a library designed to support an easy implementation of ML in a federated environment. Flower, the library of choice for this work, is a stable high-level library for Python. Flower helps transitioning rapidly from existing ML implementations into a FL setup. This allows a quick way for the evaluation of existing models in a federated environment [47] and it was the main reasoning behind its choice.

3. Proposed Model

The experiments were carried out using a workstation with an Intel© Core™ i7-5960X CPU and 32 GB of RAM, running Linux Mint 20.3 Cinnamon as the main operating system (OS). The testbed for experiments was created using the Python library, Flower, at version 1.0.0.

3.1. Overall Architecture

The proposed model is composed of four virtual clients and one server acting as the aggregator. Figure 1 illustrates the topology and the steps taken by the FL model at each round. Before training can start, clients connect to the server. The training process begins when the server sends initial parameters to the clients. Upon receiving these, clients undertake training on their own data by updating weights locally. At the end of their training, each client sends their updates to the server. Using FedAvg, as described in Algorithm 1, the server aggregates these updates into a global update, which is then sent back to clients for a new training round. This process is repeated until all rounds have completed.

Overall, the process for a training round consists of the following steps:

- The server starts and accepts connections from a number of clients based on a specific scenario.

- The server sends initial parameters of the global model to clients.
- Each client completes training on their local data, calculates their local parameters and sends an update to the server.
- The server updates parameters for the global model and aggregates results.

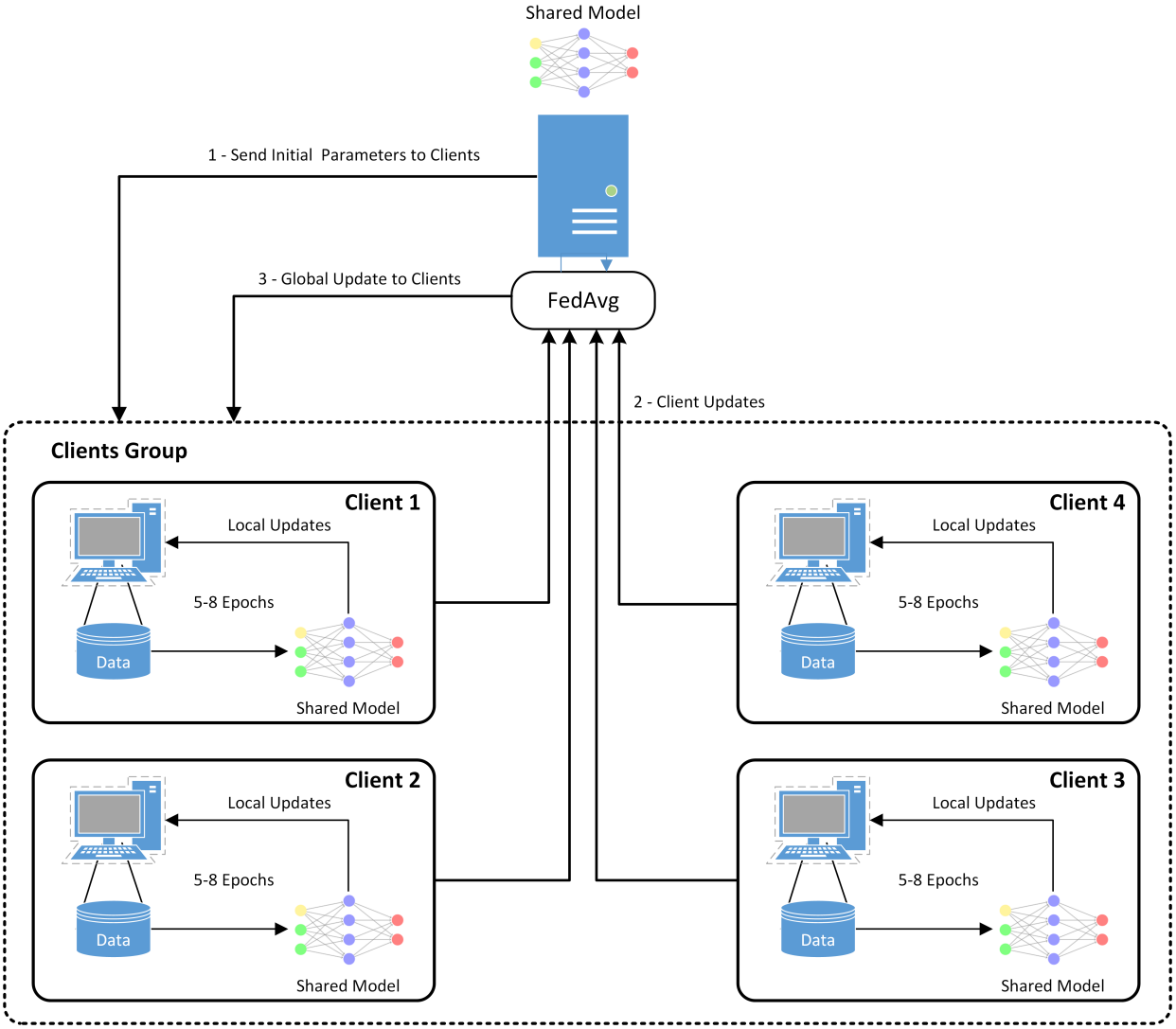


Figure 1. Federated learning topology.

A flowchart of the overall FL process using the Flower library is shown in Figure 2. High-level pseudocodes for the FL process with Flower are given in Algorithm 4.

Hyperparameters, as shown in Table 2, were set at the server side, using a method defined by Flower as a strategy, and sent to the clients at the start of the process. These are: learning rate (LR) set to 0.01, number of rounds set to 5 and epochs set to 5 for the first three rounds and then to 8 for the last two rounds. All of these settings were chosen following an empirical evaluation, where different values of LR, epochs and FL rounds were used. The values selected were those offering the best outcome in this scenario.

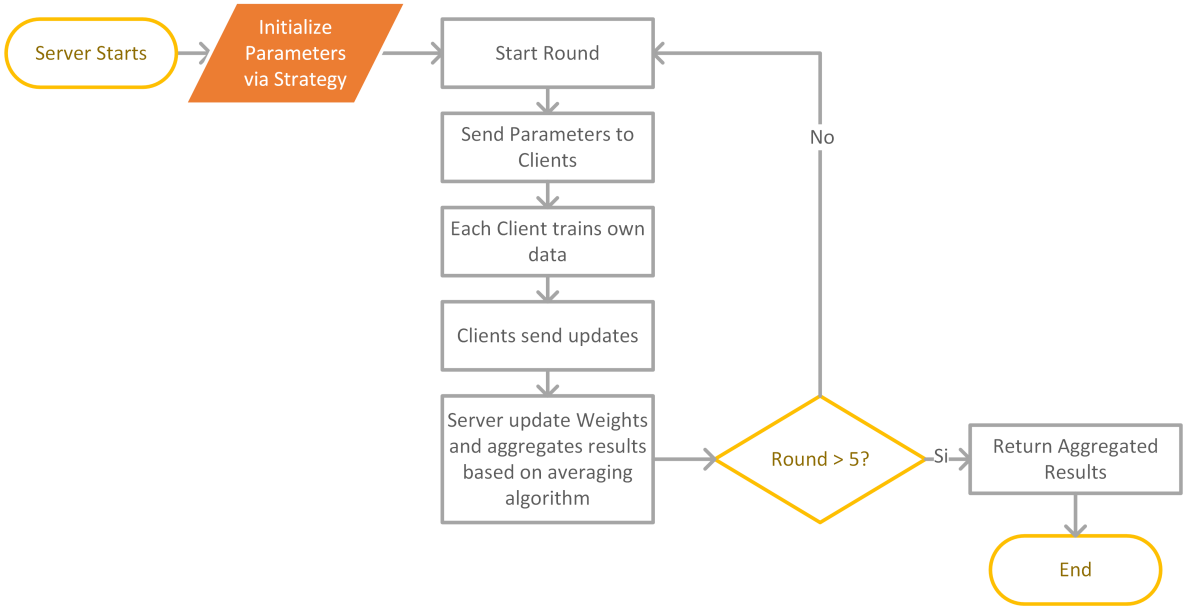


Figure 2. Flowchart of FL process using Flower.

Algorithm 4 High-level pseudocodes for the FL algorithm. C_n are the clients; S is the server, D_c is the local client's data and R is the aggregated results

Server S **starts**:
initialize parameters: p
Clients: C_n
Client's Data: D_c
for each round $r = 1 \rightarrow 5$ **do**
    $C_n \leftarrow p$
    for each Client C_n **in parallel do**
        Classify D_c
        $S \leftarrow \text{ClientUpdate}(p)$
    end for
    $S \rightarrow \text{ParameterUpdate}(p)$
    $S \rightarrow \text{AggregateResult}(R)$
end for
Return R

Table 2. Hyperparameters.

Hyperparameter	Value
Learning Rate	0.01
Epoch	5 in first 3 rounds 8 in last 2 rounds
FL Rounds	5

Flower uses the concept of strategies as a way to configure several options, including the type of averaging algorithm that is used to aggregate parameters during training. In fact, a strategy can be used to define several other customizable settings. For instance, the minimum number of total clients in the FL system, the minimum number of clients required to be available for training and the minimum number of clients required for validation are configurable directly via a strategy.

3.2. Shared Model

The model used for classification is a shallow ANN with a dense input layer formed by 24 neurons, a dense hidden layer formed by 16 neurons and an output layer. The

loss function is Adam. The activation function is ReLU for input and hidden layer, while sigmoid or softmax was used for the output layer depending whether the classification was binary or multiclass. The choice of the model, its activation and loss function was made to ensure the shared model was fast in training on the data, so that focus could be given on selecting the best options for the FL framework and its aggregation methods. Figure 3 illustrates the layers of the ANN model showing an example of multiclass output with 10 outcomes. The number 43 in the input layers indicates the number of data points or features fed into the input layer. This configuration is used on all clients. With this model, each client performs a classification of their portion of data and sends back weights to the server for aggregation and update. On the server side, the shared ANN model is also used, with the same configuration, at the beginning of the training process with a small portion of local data. A round of training is completed by the server to provide clients with initial weights that can be somewhat meaningful to the type of data used. This is to avoid using completely random weights as the initial weights for the global model.

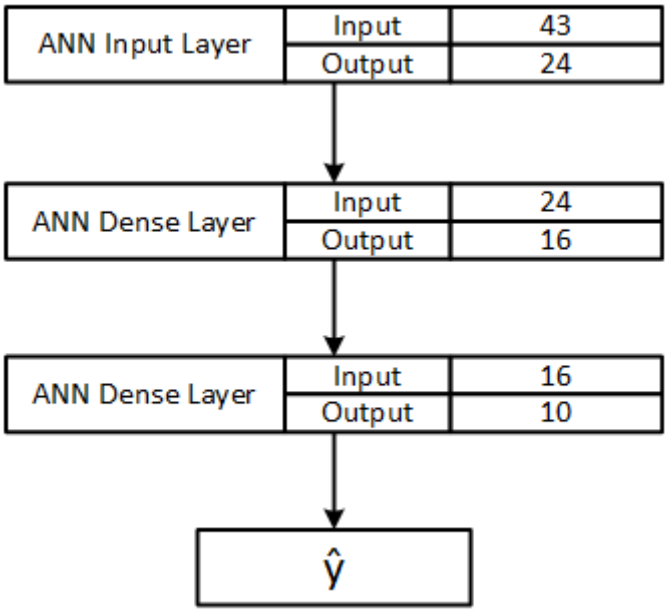


Figure 3. Representation of the shared model.

3.3. Comparison of Averaging Algorithms

As a part of the experiments, several aggregation functions were tested in this scenario. The results from the experiments above using the FedAvg algorithm were used as a baseline for evaluating the other aggregation methods including FedAvgM, FedAdam and FedAdagrad. All parameters and the shared models remain the same for each scenario.

4. Datasets, Pre-Processing and Performance Metrics

4.1. Datasets

The experiments were carried out using two open-source datasets: ToN_IoT and CICIDS2017. The first is data obtained from a large IoT network, while the other is purely based on a typical network environment. Both datasets are widely used in intrusion detection and present different characteristics which can be of value for testing the proposed model.

4.1.1. ToN_IoT Dataset

The ToN_IoT dataset [14] was collected using a large-scale network created by the University of New South Wales (UNSW) at the Australian Defence Force Academy (ADFA). This network included physical systems, virtual devices, cloud platforms and IoT sensors offering a large number of heterogeneous sources. The data include several captures from

devices with different perspective of the network: IoT/IIoT, Network, Linux and Windows. For this set of experiments, the network data was used for the model training. Preference was given to the train_test_network data as it provides a sample of the network data, as a single file in CSV format, specifically created with the intent of evaluating the efficiency of ML applications. The data contains 43 features in total and includes a large sample of normal traffic plus nine different types of attacks. These are listed in Table 3. The Numerical ID represents the value used by the algorithm to classify the samples during multiclass classification. This dataset represent actual IoT data, making it one of the most relevant for this work among those publicly available.

Table 3. ToN_IoT traffic type.

Numerical ID	Traffic Type	No. of Samples
0	Backdoor	20,000
1	DDoS	20,000
2	DoS	20,000
3	Injection	20,000
4	MITM	1043
5	Normal	300,000
6	Password	20,000
7	Ransomware	20,000
8	Scanning	20,000
9	XSS	20,000

4.1.2. CICIDS2017 Dataset

The CICIDS2017 [15] was created by the Canadian Institute for Cybersecurity and was specifically designed to help developing solutions to anomaly detection. The dataset contains traffic generated from a network captured over several days and includes a diverse range of attack scenarios. This is a larger dataset compared to the ToN_IoT in numbers of samples, features and classes. The diversity of data is one of the reasons behind its choice as it offers a more complex environment for network traffic analysis. In total, the dataset contains 79 features with each data sample labeled as either normal or as a specific attack type. A list of all types of attacks is presented in Table 4.

Table 4. CICIDS2017 traffic type.

Numerical ID	Traffic Type	Number of Samples
0	Benign	2,273,097
1	Bot	1966
2	DDoS	128,027
3	DoS GoldenEye	10,293
4	DoS Hulk	230,124
5	DoS slowhttptest	5499
6	DoS slowloris	5796
7	FTP-Patator	7938
8	Heartbleed	11
9	Infiltrator	36
10	PortScan	158,930
11	SSH-Patator	5897
12	Web attacks—brute force	1507
13	Web attack—SQL Inj	21
14	Web attack XSS	652

4.2. Data Pre-Processing

In order to simulate a realistic FL environment, each client has to obtain its own portion of the data. Therefore, both the ToN_IoT and the CICIDS2017 dataset were pre-divided into several parts randomly. However, to ensure horizontal FL could be achieved, each portion of the data maintained the same dimensionality. The same distribution of classes

in the labels was also maintained for all clients to ensure consistency during training. Before training, each client pre-processed their own data. This was achieved using the Scikit-learn library. Firstly the data was checked for null values. The rows containing these were removed as they represented an insignificant portion of data samples in both datasets. Categorical objects were also identified and encoded into numerical form to ensure data could be inputted into the DL models. The next step was the normalization of the data (i.e., scaling values between 0 and 1). This is an important step to ensure that no outliers exist in the data that could otherwise bias the outcome of the model training. Again, the Scikit-learn library with its MinMaxScaler class was used to complete this task. Mathematically, normalization was carried out by Equation (1).

$$x_i = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

To conclude the pre-processing of the data, at each client, the dataset was divided into train and test data using a 70:30 ratio, where 30% of the data was kept aside for testing the model with previously unseen data. This is a standard process for ML, as it allows validating results obtained from training using previously unseen data. This step ensures that ML models used in operational environment, with live data, can achieve similar results to their performance during training.

4.3. Performance Metrics

Evaluation of ML and DL models for classification problems such as the one presented in this work is mostly based on metrics obtained from a confusion matrix (CM). This is a cross table that reports how often a model is capable of correctly classifying a data sample with its real label. The model attempts to discover the correct type of data sample. This prediction is recorded and compared against the real type. The CM is used to calculate the number of occurrences the model correctly or incorrectly classifies data. In the context of anomaly or intrusion detection, a CM can be used to verify the rate at which a model manages to:

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

A CM is often displayed in a tabular format similar to Figure 4. On the right-hand side the numbers indicate the matching color code (e.g., dark blue indicates numbers in the order 80 K, in this case, but this value changes according to the number of samples classified).

An ideal model would identify all TP and TN correctly and never confuse one class of traffic for the other. Of course, this is not realistically achievable. However, the rates of FP and FN should be kept to a minimum. A CM allows for certain important metrics to be calculated. These are:

- **Accuracy**—This is the ratio of correctly classified instances among the total number as shown in Equation (2).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{2}$$

- **Precision**—This provides the rate of elements that have been classified as positive and that are actually positive. It is obtained by dividing correctly classified anomalies (TP) by the total number of positive instances (FP + TP) as shown in Equation (3).

$$\text{Precision} = \frac{\text{TP}}{\text{FP} + \text{TP}} \tag{3}$$

- **Recall**—Also defined as sensitivity or true positive rate (TPR), it is obtained from the correctly classified attacks (TP) divided by the total number of attacks (TP) + (FN) and

measures the model's ability to identify all positive instances (i.e., attacks) in the data. Recall is calculated by Equation (4).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

- **F1-score**—This uses both precision and recall to calculate their harmonic mean as shown in Equation (5). The higher the score the better the model.

$$\text{F1-Score} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \tag{5}$$

In multiclass classification an averaging technique is used to obtain an overall score for each metric. Several exist as explained by [48]. In this case a weighted averaging score is used where class imbalance is considered according to the number of samples of each class in the data.

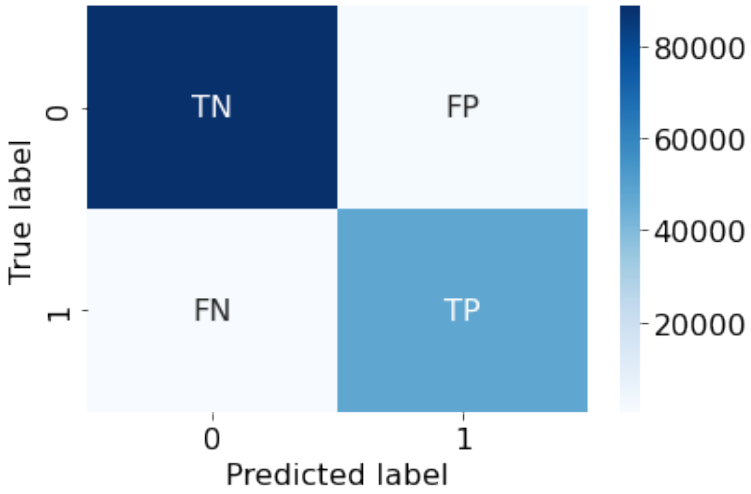


Figure 4. Confusion matrix.

5. Results and Discussion

In this section, we present results from the experiments on the ToN_IoT and CI-CIDS2017 datasets using the Flower FL environment. To evaluate the performance of the FL model, experiments were completed on both datasets for both binary and multiclass classification. Each is presented and discussed as a standalone scenario. Results from a traditional centralized approach, using the same ANN model on both datasets, are used as a baseline for evaluation of the FL model. Accuracy, precision, recall and F1-score are used as the metrics to compare performance between the FL model and its centralized counterpart. FedAvg is used as the averaging algorithm for aggregation of parameters. Results are also given of further experiments completed to evaluate FedAvgM, FedAdam and FedAdagrad as alternative methods to FedAvg.

5.1. Binary Classification

Binary classification aims at identifying anomalies from the given data. Table 5 and Figure 5 show results from the binary classification on the ToN_IoT dataset. The table presents results from each participating client, the actual aggregated results from the server and the centralized model. Results from each client are purely informational. In typical FL models the number of clients can be quite large; therefore, keeping track of scores from each client would be impractical and unnecessary. In this case, given that only four clients are available, seeing how they perform on their own data and comparing their results with the aggregated model can be useful particularly to identify possible substantial differences in results. From the results it is evident that the federated model offers scores that are slightly

lower than the centralized model. However, the difference is not substantial, indicating the horizontal FL system can perform well in binary classification using the FedAvg as the aggregating algorithm.

Table 5. Performance of FL in binary classification on ToN_IoT dataset.

Model	Accuracy	Precision	Recall	F1-Score
FL Model—Client 1	0.9758	0.9449	0.9883	0.9661
FL Model—Client 2	0.9755	0.9436	0.9892	0.9658
FL Model—Client 3	0.9748	0.9440	0.9864	0.9647
FL Model—Client 4	0.9760	0.9458	0.9878	0.9664
FL Model—Aggr.	0.9759	0.9487	0.9842	0.9661
Centralized Model	0.9840	0.9729	0.9817	0.9772

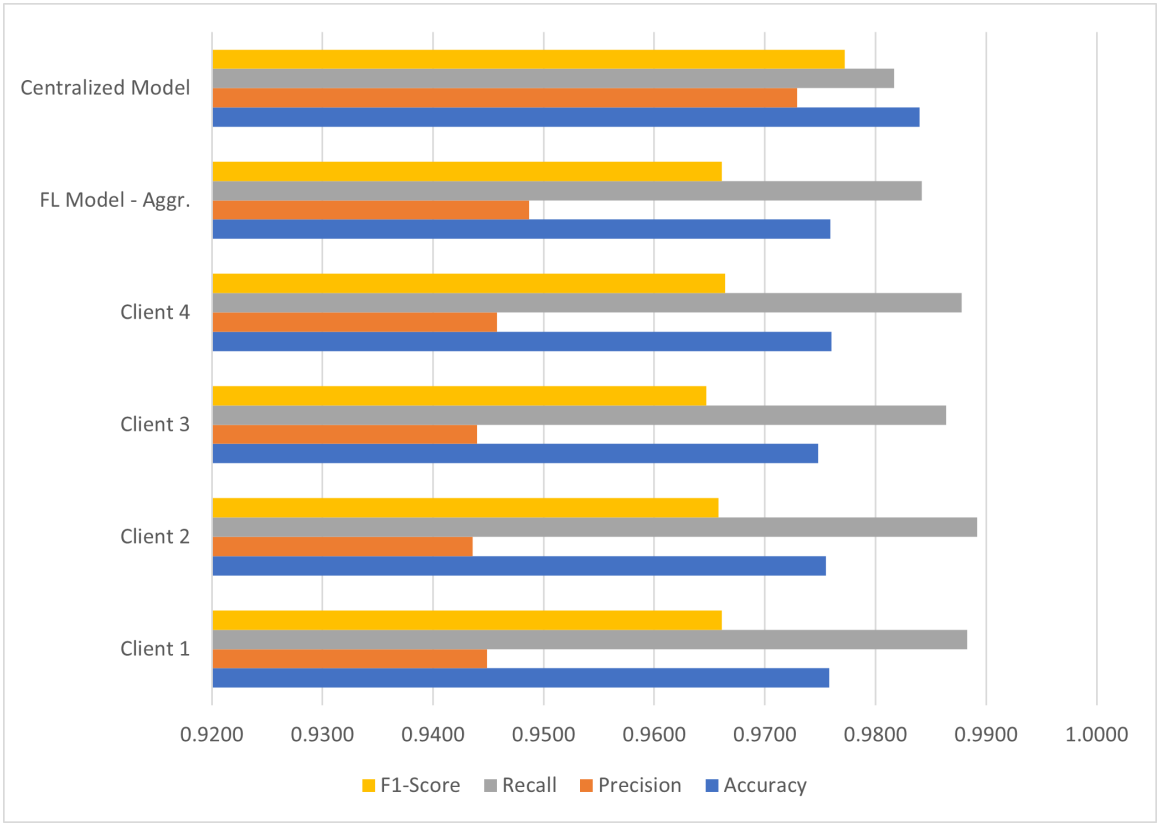


Figure 5. Comparison of FL model vs. centralized model in binary classification on ToN_IoT.

As to the performances of alternative averaging algorithms, results from the binary classification on the ToN_IoT dataset are given in Table 6.

From the results it is clear that, in this context, the FedAvgM algorithm seems to offer the best metrics. It offers superior accuracy, precision, recall and F1-score compared to any of the other methods. Only FedAvg seems to perform closely. FedAdam and FedAdagrad perform poorly compared to the others, with FedAdagrad in particular being the most unreliable.

As a way to validate the classification on the ToN_IoT dataset, the CICIDS2017 was used for a similar experiment. The results of the binary classification on the CICIDS2017 dataset are presented in Table 7 and Figure 6. The results are very similar to the previous scenario, indicating consistency. Moreover, even in this scenario, the centralized model offers a better performance compared to the FL system, confirming the results from the binary classification on the ToN_IoT.

Table 6. Performance of averaging algorithms in binary classification on ToN_IoT dataset.

Model	Accuracy	Precision	Recall	F1-Score
FedAvgM	0.9772	0.9512	0.9853	0.9679
FedAdam	0.8767	0.7580	0.9505	0.8434
FedAdagrad	0.8176	0.6635	0.9697	0.7879
FedAvg	0.9759	0.9487	0.9842	0.9661

Table 7. Performance of FL in binary classification on CICIDS2017 dataset.

Model	Accuracy	Precision	Recall	F1-Score
FL Model—Client 1	0.9851	0.9481	0.9778	0.9627
FL Model—Client 2	0.9841	0.9507	0.9694	0.9600
FL Model—Client 3	0.9821	0.9292	0.9841	0.9559
FL Model—Client 4	0.9821	0.9312	0.9816	0.9557
FL Model—Aggr.	0.9820	0.9326	0.9793	0.9554
Centralized Model	0.9840	0.9490	0.9720	0.9610

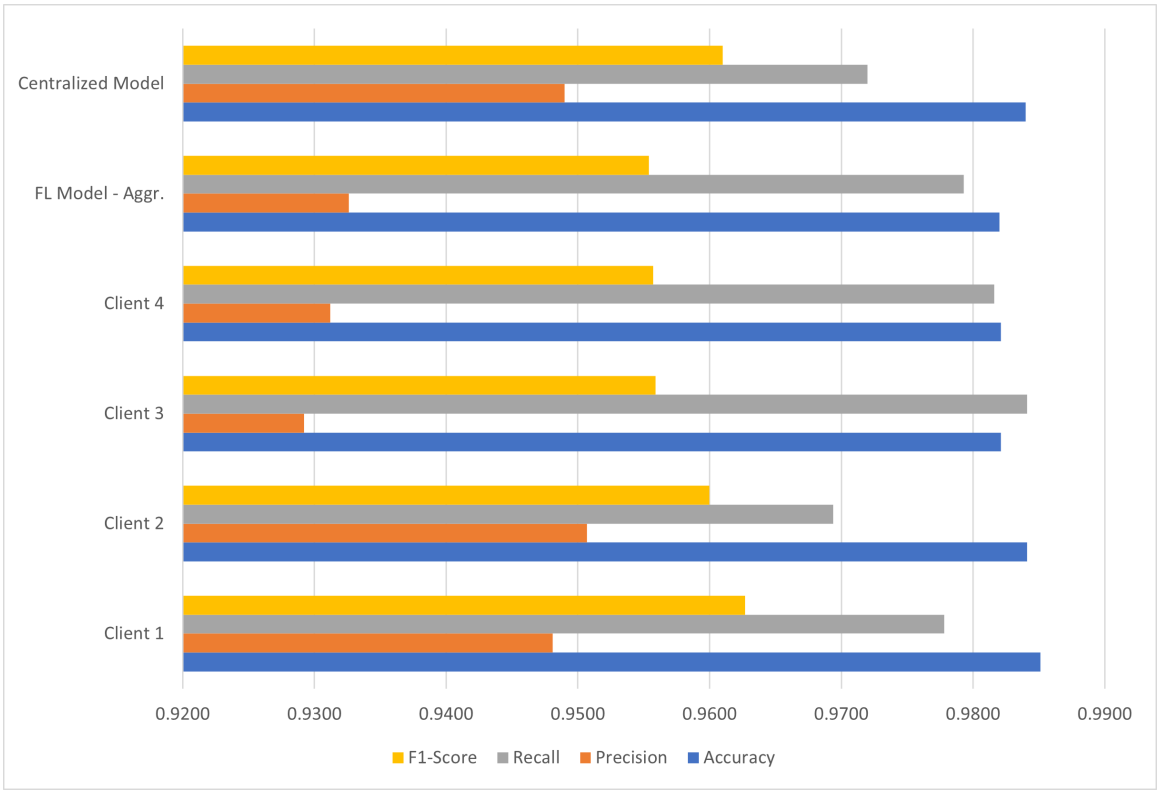


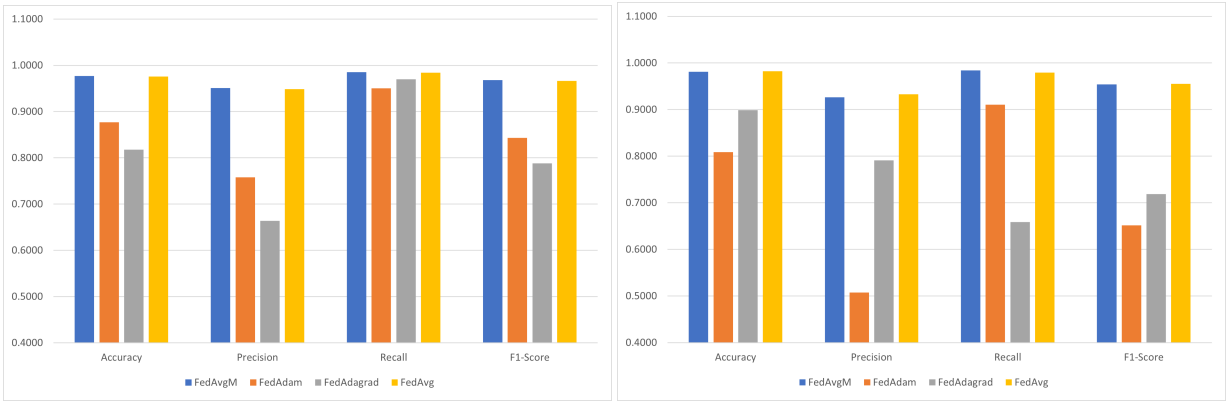
Figure 6. Comparison of FL model vs. centralized model in binary classification on CICIDS2017.

As to the performances of alternative averaging algorithms, binary classification on CICIDS2017 has given some different results. With a larger and more heterogeneous dataset, FedAvg performed better than the rest. FedAvgM is very close in most metrics and even offered better scores in Recall. Again, FedAdagrad and FedAdam have given the poorest results. This time FedAdam performed worse than FedAdagrad. However, performance from both methods is well behind the other two algorithms. Results are presented in Table 8.

Table 8. Performance of averaging algorithms in binary classification on CICIDS2017 dataset.

Model	Accuracy	Precision	Recall	F1-Score
FedAvgM	0.9814	0.9263	0.9839	0.9542
FedAdam	0.8085	0.5075	0.9104	0.6517
FedAdagrad	0.8986	0.7908	0.6589	0.7189
FedAvg	0.9820	0.9326	0.9793	0.9554

The performances of alternative averaging algorithms for the binary classification on the ToN_IoT dataset and the CICIDS2017 dataset are further illustrated in Figure 7.



(**a**) ToN_IoT (**b**) CICIDS2017

Figure 7. Comparison of different aggregation algorithms for binary classification on ToN_IoT and CICIDS2017.

5.2. Multiclass Classification

In multiclass classification the objective is to identify the actual attack from the target labels. The ToN_IoT dataset contains ten different classes of data: nine are attacks, while the remaining class is normal traffic. Table 9 and Figure 8 show the results obtained in multiclass classification. In this scenario, the centralized model clearly outperforms in all metrics its federated counterpart. However, scores from the FL system are still quite high, demonstrating the soundness of the approach even in multiclass classification.

Table 9. Performance of FL in multiclass classification on ToN_IoT dataset.

Model	Accuracy	Precision	Recall	F1-Score
FL Model—Client 1	0.9813	0.9816	0.9813	0.9812
FL Model—Client 2	0.9806	0.9810	0.9806	0.9806
FL Model—Client 3	0.9792	0.9794	0.9792	0.9791
FL Model—Client 4	0.9800	0.9803	0.9800	0.9800
FL Model—Aggr.	0.9786	0.9789	0.9786	0.9785
Centralized Model	0.9940	0.9940	0.9940	0.9940

As to the performances of alternative averaging algorithms, multiclass classification on the ToN_IoT dataset returned similar results to the binary classification. FedAvg and FedAvgM achieved better scores compared to the other two methods. FedAdagrad in particular performed quite poorly in this scenario with very low scores in all metrics. FedAdam achieved better scores but still well below the performance of FedAvg and FedAvgM. Table 10 provides the results for all algorithms.

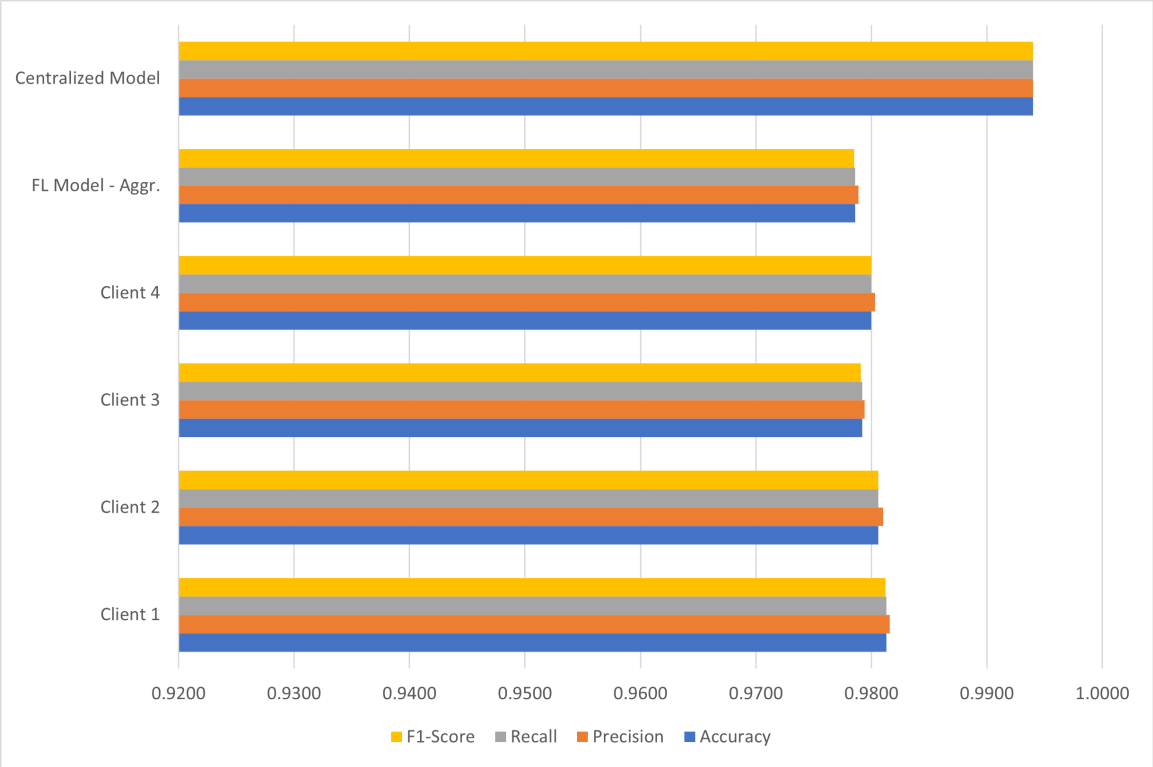


Figure 8. Comparison of FL model vs. centralized model in multiclass classification on ToN_IoT.

Table 10. Performance of averaging algorithms in multiclass classification on ToN_IoT dataset.

Model	Accuracy	Precision	Recall	F1-Score
FedAvgM	0.9757	0.9766	0.9757	0.9758
FedAdam	0.8340	0.8694	0.8340	0.8299
FedAdagrad	0.6983	0.6687	0.6983	0.6612
FedAvg	0.9786	0.9789	0.9786	0.9785

The multiclass classification on the CICIDS2017 confirms the soundness of results from the federated system. Moreover, it apparently shows that, in a more complex dataset such as this, a smaller discrepancy of results exists between the centralized model and the federated version. The centralized model has performed better once more but the performance from the FL system is closer. The results are presented in Table 11 and Figure 9.

Table 11. Performance of FL in multiclass classification on CICIDS2017 dataset.

Model	Accuracy	Precision	Recall	F1-Score
FL Model—Client 1	0.9823	0.9841	0.9823	0.9823
FL Model—Client 2	0.9727	0.9725	0.9727	0.9704
FL Model—Client 3	0.9849	0.9856	0.9849	0.9846
FL Model—Client 4	0.9812	0.9834	0.9812	0.9813
FL Model—Aggr.	0.9815	0.9829	0.9815	0.9816
Centralized Model	0.9820	0.9840	0.9820	0.9820

As to the performances of alternative averaging algorithms, in the multiclass classification on the CICIDS2017, the improved performance of FedAdagrad is noticeable as well as the very poor performance of FedAdam. FedAvg and FedAvgM performed well in all metrics. Again, all results are available in tabular format in Table 12.

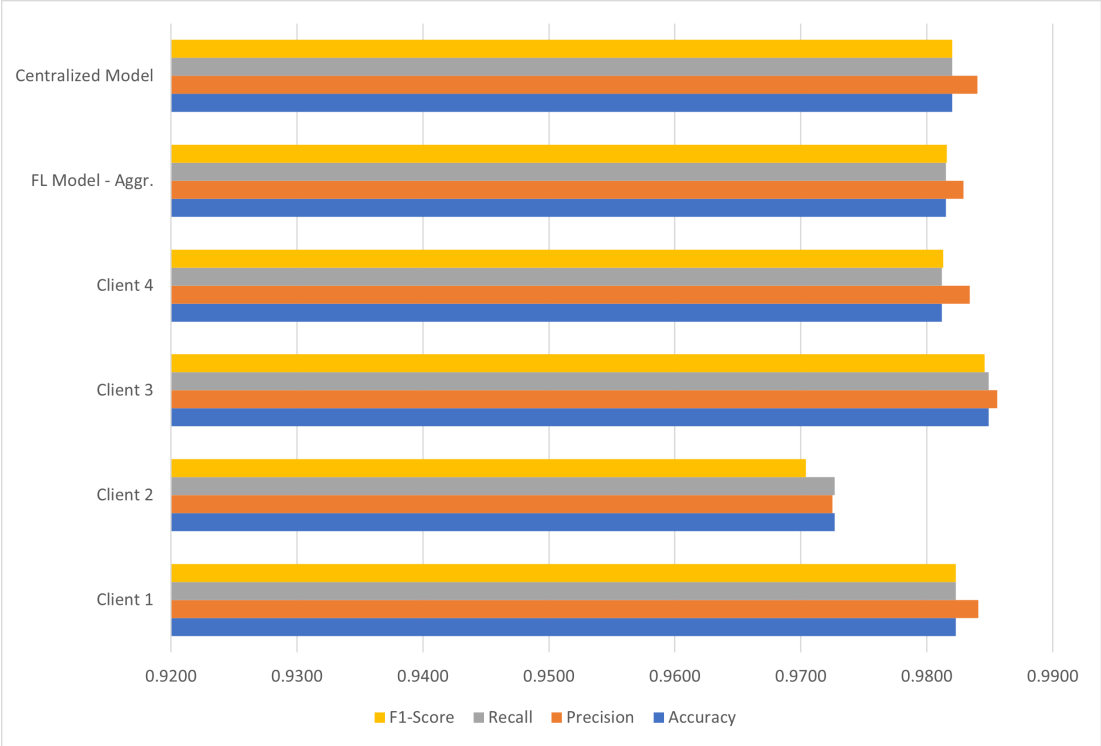
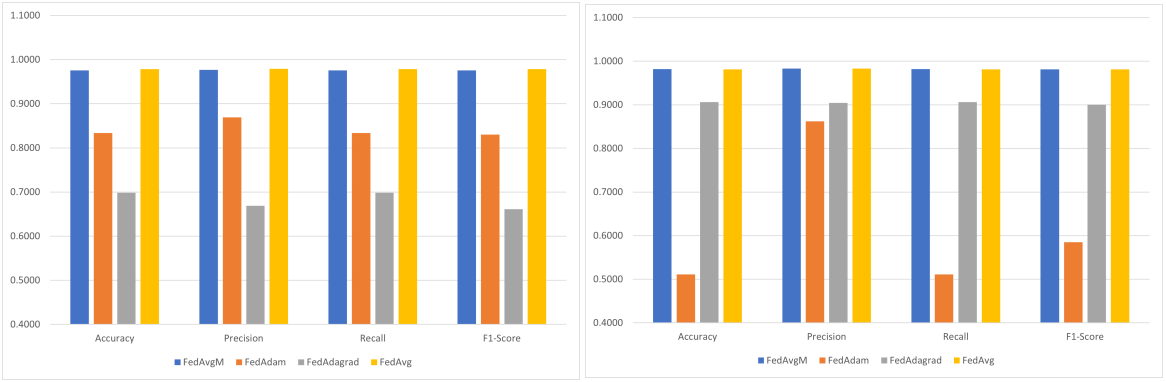


Figure 9. Comparison of FL model vs. centralized model in multiclass classification on CICIDS2017.

Table 12. Performance of averaging algorithms in multiclass classification on CICIDS2017 dataset.

Model	Accuracy	Precision	Recall	F1-Score
FedAvgM	0.9817	0.9831	0.9817	0.9816
FedAdam	0.5111	0.8624	0.5111	0.5847
FedAdagrad	0.9065	0.9046	0.9065	0.9005
FedAvg	0.9815	0.9829	0.9815	0.9816

The performances of alternative averaging algorithms for the multiclass classification on the ToN_IoT dataset and the CICIDS2017 dataset are further illustrated in Figure 10.

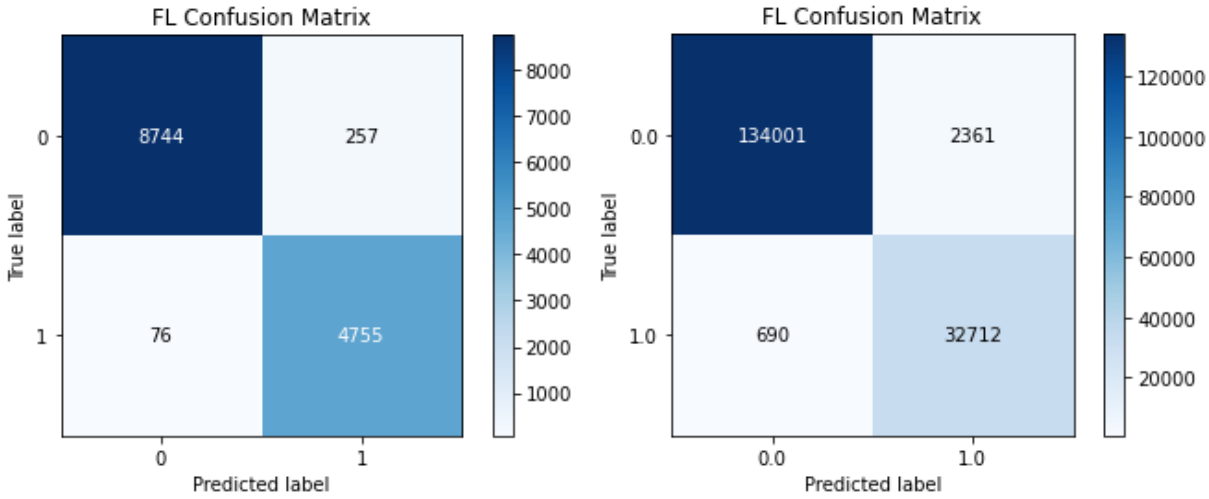


(**a**) ToN_IoT (**b**) CICIDS2017

Figure 10. Comparison of different aggregation algorithms for multiclass classification on ToN_IoT and CICIDS2017.

5.3. Confusion Matrices

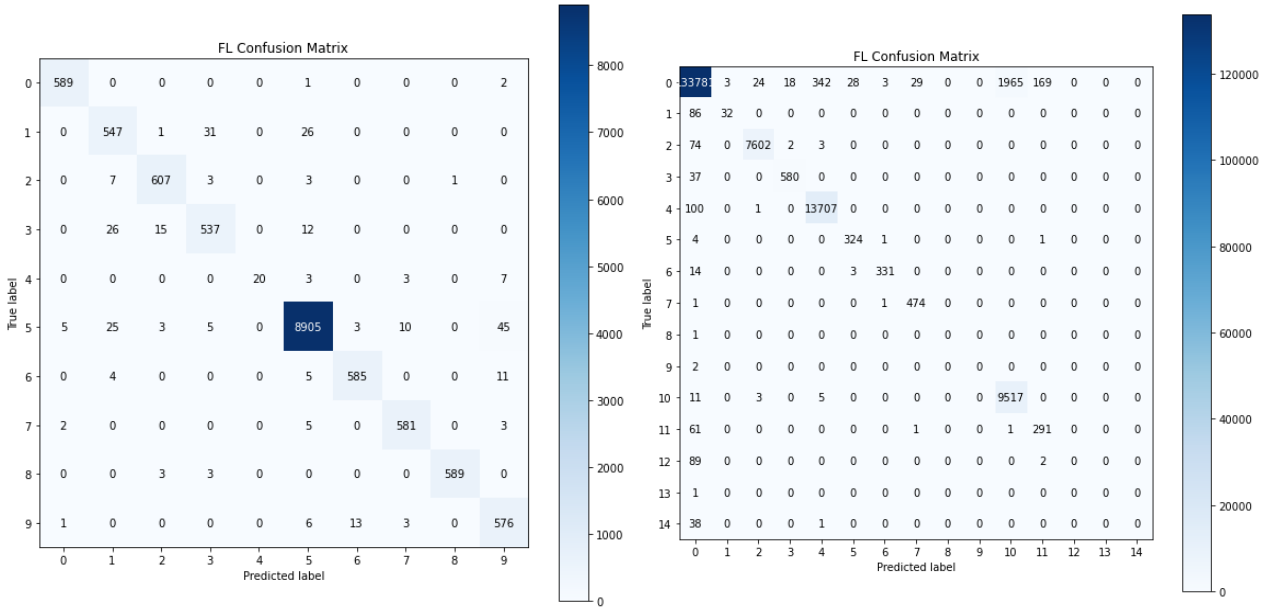
The CMs of the FL model for binary classification on the ToN_IoT dataset and the CICIDS2017 dataset are illustrated for reference in Figure 11.



(**a**) Results for ToN_IoT in binary (**b**) Results for CICIDS2017 in binary

Figure 11. Confusion matrices obtained from the FL binary classification.

The CMs of the FL model for multiclass classification on the ToN_IoT dataset and the CICIDS2017 dataset are illustrated for reference in Figure 12.



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

Figure 12. Confusion matrices obtained from the FL multiclass classification.

6. Conclusions and Future Works

User data privacy has become of paramount importance for organizations in recent years, particularly so, since the official publication of the European General Data Protection Regulation (GDPR) in May 2018, which intensified pressure on organizations to ensure privacy of users' data is maintained at all times. High fines are expected for those who do not comply with GDPR. Large data storage for ML analysis could potentially lead to privacy issues. In IoT environments, data needs to be transferred from devices to a central location for analysis and this has the potential to cause privacy issues. FL can be used as a method to ensure privacy is maintained while training ML models. In this article we have performed several experiments to evaluate FL as an alternative method to a centralized model in detecting attacks in IoT environments, while maintaining data privacy. The results

from the experiments demonstrated how a collaborative federated system using horizontal data partitioning can have a close performance to a centralized model. The FL model was built out of four clients and one server. Data analysis was performed at the client side, each using their own portion of the dataset. No data sharing between participants occurred. The role of the server was to coordinate the overall process. FedAvg was used as the algorithm for parameter aggregation. The overall process was completed over five rounds. At each round, the clients trained the DL model with their local data. The DL model used in this set of experiments is a shallow ANN with three layers. Results have shown that a FL system can provide an excellent alternative to a centralized model as it is capable of achieving comparable results in accuracy, precision, recall and F1-score in both binary and multiclass classification.

FedAvg is known to work well in networks where the data distribution is balanced. However, it has been found to have issues of convergence in situations where data distribution is highly variable. This tends to be a common situation in federated networks where each client uses its own data. Algorithms such as FedAdam, FedAdagrad and FedAvgM have been proposed to deal with these issues. In this article, we have presented an evaluation of these algorithms. Using the same testbed as the other experiments, each algorithm was evaluated in both binary and multiclass classification. Results from experiments have shown that FedAvg and FedAvgM tend to perform better in both scenarios compared to the two adaptive algorithms, FedAdam and FedAdagrad. In particular, FedAvg achieved a better score in binary classification on the CICIDS2017 dataset and multiclass classification on the ToN_IoT. In contrast, results from experiments with FedAdam and FedAdagrad, were generally negative. Only in multiclass classification with CICIDS2017 was FedAdagrad able to achieve scores in the order of 90% in all metrics. Other than that, the performances of both algorithms in this context was fairly poor. It is difficult to establish the exact reasons for this performance. However, given that these algorithms were designed to improve on the performance of FedAvg in a cross-device scenario [40], where a large number of clients is assumed, an empirical evaluation of these methods in more complex scenarios should be considered as a part of future works. Moreover, in order to improve on both binary and multiclass classification, the use of a more complex shared model should be considered. For instance, a possible approach that could be interesting to explore is the use of an ensemble of diverse models applied to a FL infrastructure where results are aggregated at a central location.

Author Contributions: Conceptualization, R.L. and H.T.; methodology, R.L. and H.T.; software, R.L.; validation, R.L.; formal analysis, R.L.; investigation, R.L.; resources, R.L.; data curation, R.L.; writing—original draft preparation, R.L.; writing—review and editing, H.T. and V.C.; visualization, R.L.; supervision, H.T. and V.C.; project administration, R.L.; funding acquisition, R.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Laghari, A.A.; Wu, K.; Laghari, R.A.; Ali, M.; Khan, A.A. A Review and State of Art of Internet of Things (IoT). *Arch. Comput. Methods Eng.* **2022**, *29*, 1395–1413. [CrossRef]
2. Nguyen, T.D.; Rieger, P.; Miettinen, M.; Sadeghi, A.R. Poisoning Attacks on Federated Learning-based IoT Intrusion Detection System. In Proceedings of the Workshop on Decentralized IoT Systems and Security (DISS) 2020, San Diego, CA, USA, 23–26 February 2020; Internet Society: Reston, VA, USA, 2021; Volume 8. [CrossRef]

3. Kagita, M.K.; Thilakarathne, N.; Gadekallu, T.R.; Maddikunta, P.K.R.; Singh, S. A Review on Cyber Crimes on the Internet of Things. In *Signals and Communication Technology*; Springer Science and Business Media GmbH: Berlin, Germany, 2021; pp. 83–98. [CrossRef]
4. Kuzlu, M.; Fair, C.; Guler, O. Role of Artificial Intelligence in the Internet of Things (IoT) cybersecurity. *Discov. Internet Things* **2021**, *1*, 7. [CrossRef]
5. Sengupta, S.; Basak, S.; Saikia, P.; Paul, S.; Tsalavoutis, V.; Atiah, F.; Ravi, V.; Peters, A. A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowl.-Based Syst.* **2020**, *194*, 105596. [CrossRef]
6. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. [CrossRef]
7. Tsimenidis, S.; Lagkas, T.; Rantos, K. Deep Learning in IoT Intrusion Detection. *J. Netw. Syst. Manag.* **2022**, *30*, 8. [CrossRef]
8. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 19. [CrossRef]
9. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
10. Ferrari, P.; Sisinni, E.; Brandão, D.; Rocha, M. Evaluation of communication latency in industrial IoT applications. In Proceedings of the 2017 IEEE International Workshop on Measurement and Networking, M and N 2017—Proceedings, Naples, Italy, 27–29 September 2017; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2017. [CrossRef]
11. Schulz, P.; Matthe, M.; Klessig, H.; Simsek, M.; Fettweis, G.; Ansari, J.; Ashraf, S.A.; Almeroth, B.; Voigt, J.; Riedel, I.; et al. Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture. *IEEE Commun. Mag.* **2017**, *55*, 70–78. [CrossRef]
12. McMahan, B.; 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, Ft. Lauderdale, FL, USA, 20–22 April 2017.
13. Campos, E.M.; Saura, P.F.; González-Vidal, A.; Hernández-Ramos, J.L.; Bernabé, J.B.; Baldini, G.; Skarmeta, A. Evaluating Federated Learning for intrusion detection in Internet of Things: Review and challenges. *Comput. Netw.* **2022**, *203*, 108661. [CrossRef]
14. Booij, T.M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; Hartog, F.T. ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets. *IEEE Internet Things J.* **2022**, *9*, 485–496. [CrossRef]
15. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the ICISSP 2018—4th International Conference on Information Systems Security and Privacy, Funchal, Portugal, 22–24 January 2018; SciTePress: Setúbal, Portugal, 2018; Volume 2018, pp. 108–116. [CrossRef]
16. Konečný, J.; McMahan, B.; Ramage, D. Federated Optimization:Distributed Optimization Beyond the Datacenter. *arXiv* **2015**, arXiv:1511.03575.
17. Sarhan, M.; Layeghy, S.; Moustafa, N.; Portmann, M. A Cyber Threat Intelligence Sharing Scheme based on Federated Learning for Network Intrusion Detection. *J. Netw. Syst. Manag.* **2023**, *31*, 3. [CrossRef]
18. Zhao, R.; Yin, Y.; Shi, Y.; Xue, Z. Intelligent intrusion detection based on federated learning aided long short-term memory. *Phys. Commun.* **2020**, *42*, 101157. [CrossRef]
19. Zhao, L.; Li, J.; Li, Q.; Li, F. A Federated Learning Framework for Detecting False Data Injection Attacks in Solar Farms. *IEEE Trans. Power Electron.* **2022**, *37*, 2496–2501. [CrossRef]
20. Mothukuri, V.; Khare, P.; Parizi, R.M.; Pouriyeh, S.; Dehghantanha, A.; Srivastava, G. Federated-Learning-Based Anomaly Detection for IoT Security Attacks. *IEEE Internet Things J.* **2022**, *9*, 2545–2554. [CrossRef]
21. Zhang, T.; He, C.; Ma, T.; Gao, L.; Ma, M.; Avestimehr, S. Federated Learning for Internet of Things: A Federated Learning Framework for On-device Anomaly Data Detection. *arXiv* **2021**, arXiv:2106.07976.
22. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
23. ANT. The ANT Lab: Analysis of Network Traffic. 2017. Available online: https://ant.isi.edu/ (accessed on 16 May 2022).
24. Chatterjee, S.; Hanawal, M.K. Federated Learning for Intrusion Detection in IoT Security: A Hybrid Ensemble Approach. *Int. J. Internet Things-Cyber-Assur.* **2022**, *2*, 62–86. [CrossRef]
25. Saha, R.; Misra, S.; Deb, P.K. FogFL: Fog-Assisted Federated Learning for Resource-Constrained IoT Devices. *IEEE Internet Things J.* **2021**, *8*, 8456–8463. [CrossRef]
26. Chen, Z.; Lv, N.; Liu, P.; Fang, Y.; Chen, K.; Pan, W. Intrusion Detection for Wireless Edge Networks Based on Federated Learning. *IEEE Access* **2020**, *8*, 217463–217472. [CrossRef]
27. Zhang, J.; Luo, C.; Carpenter, M.; Min, G. Federated Learning for Distributed IIoT Intrusion Detection using Transfer Approaches. *IEEE Trans. Ind. Inform.* **2023**, *19*, 8159–8169. [CrossRef]

28. Thonglek, K.; Takahashi, K.; Ichikawa, K.; Iida, H.; Nakasan, C. Federated Learning of Neural Network Models with Heterogeneous Structures. In Proceedings of the 19th IEEE International Conference on Machine Learning and Applications, ICMLA 2020, Miami, FL, USA, 14–17 December 2020; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2020; Volume 12, pp. 735–740. [CrossRef]
29. Qin, Q.; Poularakis, K.; Leung, K.K.; Tassiulas, L. Line-Speed and Scalable Intrusion Detection at the Network Edge via Federated Learning. In Proceedings of the IFIP Networking 2020 Conference and Workshops—Networking 2020, Paris, France, 22–25 June 2020; pp. 352–360.
30. Otoum, Y.; Nayak, A. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. *J. Netw. Syst. Manag.* **2021**, *29*, 23. [CrossRef]
31. Man, D.; Zeng, F.; Yang, W.; Yu, M.; Lv, J.; Wang, Y. Intelligent Intrusion Detection Based on Federated Learning for Edge-Assisted Internet of Things. *Secur. Commun. Netw.* **2021**, *2021*, 9361348. [CrossRef]
32. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A.; Lloret, J. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. *Sensors* **2017**, *17*, 1967. [CrossRef]
33. Li, X.; Huang, K.; Yang, W.; Wang, S.; Zhang, Z. On the Convergence of FedAvg on Non-IID Data. *arXiv* **2019**, arXiv:1907.02189.
34. Karimireddy, S.P.; Kale, S.; Mohri, M.; Reddi, S.J.; Stich, S.U.; Suresh, A.T. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Online, 13–18 July 2020; Volume PartF16814, pp. 5088–5099.
35. Hsu, T.M.H.; Qi, H.; Brown, M. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. *arXiv* **2019**, arXiv:1909.06335.
36. Sun, T.; Li, D.; Wang, B. Decentralized Federated Averaging. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 4289–4301. [CrossRef]
37. Muhammad, K.; Wang, Q.; O'Reilly-Morgan, D.; Tragos, E.; Smyth, B.; Hurley, N.; Geraci, J.; Lawlor, A. FedFast: Going beyond Average for Faster Training of Federated Recommender Systems. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Online, 23–27 August 2020; Volume 20, pp. 1234–1242. [CrossRef]
38. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated Optimization in Heterogeneous Networks. *Proc. Mach. Learn. Syst.* **2020**, *2*, 429–450.
39. Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; Khazaeni, Y. Federated Learning with Matched Averaging. *arXiv* **2020**, arXiv:2002.06440.
40. Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečný, J.; Kumar, S.; McMahan, H.B. Adaptive Federated Optimization. *arXiv* **2020**, arXiv:2003.00295.
41. Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence. *Dokl. Akad. Nauk. SSSR* **1983**, *269*, 543.
42. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
43. Su, W.; Chen, L.; Wu, M.; Zhou, M.; Liu, Z.; Cao, W. Nesterov accelerated gradient descent-based convolution neural network with dropout for facial expression recognition. In Proceedings of the 2017 Asian Control Conference, ASCC 2017, Gold Coast, QLD, Australia, 17–20 December 2017; IEEE: Piscataway, NJ, USA, 2018; pp. 1063–1068. [CrossRef]
44. Google Inc. TensorFlow Federated. 2022. Available online: https://www.tensorflow.org/federated (accessed on 7 February 2022).
45. Ziller, A.; Trask, A.; Lopardo, A.; Szymkow, B.; Wagner, B.; Bluemke, E.; Nounahon, J.M.; Passerat-Palmbach, J.; Prakash, K.; Rose, N.; et al. PySyft: A Library for Easy Federated Learning. In *Studies in Computational Intelligence*; Springer Science and Business Media GmbH: Berlin, Germany, 2021; Volume 965, pp. 111–139. [CrossRef]
46. Ludwig, H.; Baracaldo, N.; Thomas, G.; Zhou, Y.; Anwar, A.; Rajamoni, S.; Ong, Y.; Radhakrishnan, J.; Verma, A.; Sinn, M.; et al. IBM Federated Learning: An Enterprise Framework White Paper V0.1. *arXiv* **2020**, arXiv:2007.10987.
47. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A Friendly Federated Learning Research Framework. *arXiv* **2020**, arXiv:2007.14390.
48. Grandini, M.; Bagli, E.; Visani, G. Metrics for Multi-Class Classification: An Overview. *arXiv* **2020**, arXiv:2008.05756.