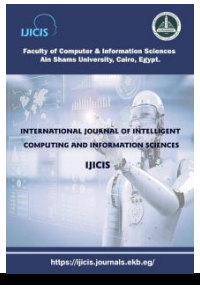
IJICIS, Vol.24, No.1, 13-28
DOI: 10.21608/ijicis.2024.262382.1315

International Journal of Intelligent Computing and Information Sciences

https://ijicis.journals.ekb.eg/



Federated Learning Enabled IDS for Internet of Things on non-IID Data

Omar Elnakib*

Department of Computer Systems,
Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
omar.elnakib@cis.asu.edu.eg

Eman Shaaban

Department of Computer Systems,
Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
eman.shaaban@cis.asu.edu.eg

Mohamed Mahmoud

Electrical and Computer Engineering,
Tennessee Technological University,
Cookeville, TN, USA
mmahmoud@tntech.edu

Karim Emara

Department of Computer Systems,
Faculty of Computer and Information Sciences, Ain Shams University,
Cairo, Egypt
karim.emara@cis.asu.edu.eg

Received 2024-01-13; Revised 2024-01-13; Accepted 2024-03-05

Abstract: *Critical applications in IoT systems are being targeted by attackers. Using a smart intrusion detection system (IDS) is crucial for protecting IoT systems. Centralized learning is commonly used to create smart IDS and has been successful in IoT networks. However, the Iot nodes in critical applications with highly sensitive information, are not willing to send their data through the network and share it with another party. To solve the problem of data privacy, researchers came up with distributed and federated learning. Both methods allow learning to happen within a local network, with data remaining inside the network and the learning process being done by the edge devices. In this research, a deep learning model is proposed to classify the types of behaviors provided in the CICIDS2017 dataset using the three learning approaches. The experiments were performed by splitting the dataset over ten simulated nodes. In the centralized learning approach, an F-Score of 98% can be achieved. In distributed learning, the F-Score achieved an average of 78% over the ten nodes. In the federated learning, the F-Score achieved an average of 89% over the ten nodes. A comparative study among the centralized, distributed, and federated approaches is done and the challenge that may arise from using each approach. Moreover, an evaluation of the effect of the data distribution, the number of local training rounds and the global communication rounds on federated learning's efficiency. The federated learning approach has shown promising improvements for both accuracy in addition to preserving data privacy.*

Keywords: *Federated Learning, Cyber Security, Intrusion Detection, Deep Learning, Distributed Learning.*

***Corresponding Author**: Omar Elnakib

Computer Systems Department, Faculty of Computer and Information Science, Ain Shams University, Cairo, Egypt

Email address: omar.elnakib@cis.asu.edu.eg

1. Introduction

Because IoT devices have limited resources, it is not possible to create Intrusion Detection Systems (IDS) on these devices using machine/deep learning models. Machine and deep learning models require a lot of computer power to be trained. This lack of ability has made IoT devices vulnerable to many dangers that can harm or control the data or the devices, causing big harm to the users. Researchers have successfully used the cloud to solve problems in the IoT field. They have been able to analyze data, share information, and training advanced machine learning programs by using the cloud. After models are trained, they can be used on devices that are near to the IoT nodes like gateways [22]. These gateways connect the IoT devices to the cloud. They are strong enough to run a smart IDS that uses machine learning. Using these in-between devices to connect IoT devices to the cloud creates a fog architecture [1]. This cloud-fog architecture lets IoT devices share their data with the cloud. This helps create an accurate and efficient model by learning from each other's data. This means that if a node has never experienced a particular kind of attack before, it can still recognize that attack by learning from the knowledge of other nodes. But even though this architecture is helpful, it causes a problem with the privacy of the data on the IoT node. The node has to share its network traffic with the cloud node to train the model, but this data is sensitive and should not be shared over the network because it might be accessed by unauthorized parties. So, new innovative ways need to be applied that benefit from the advantages of centralized learning but keep the nodes' information private.

There are three main approaches for learning: centralized, distributed, and federated learning. As mentioned earlier, the centralized learning approach has its benefits because the nodes share their data with one central device to take advantage of its strong capabilities. This lets the model train on a huge amount of data which will make it more accurate. However, using this method makes the parties reveal their private data and goes against their right to privacy. This cannot be used in critical applications. On the other hand, the distributed learning approach solves this problem of privacy by using the edge devices (gateways) in a fog network. The edge devices are used to train the model locally without sending the data outside of the local network. So, distributed learning has solved the data privacy issue, but the local model has only used the data from the local IoT nodes. This could make the local models less accurate and unable to identify new attacks that have not been seen before in the local networks.

Google has introduced a new approach called federated learning [2]. Federated learning combines fog computing and cloud computing to use edge devices for machine learning solutions. A central node collects and combines the contributions of many clients. Federated learning means training a shared model using lots of data from many devices. The goal is to teach the model while keeping the data on each device and handling it locally [3]. The only thing that can be shared is the weights of the trained local model of each device. This information is regularly sent to the central aggregator. As shown in Fig. 1. A basic federated learning scenario can be explained as follows. The IoT nodes, such as sensors, gather private data from their surroundings and send it to the nearby connected edge device on their network. Each edge device on the network receives data and uses it to train a local ML/DL model. After a certain number of rounds of training (i.e., epochs), the edge device communicates their model with a central device called the aggregator. In easier terms, the system doesn't break the privacy of the local data because it only shares the weights of the local model. The weights are sent to make changes to a global model without looking at the original data from each device. The central device can be reached from the edge nodes of all the local networks. It brings together all the local models in the entire network, updates the global model, and then sends it back to the edge devices. This means that all devices on the edge can join in the training and keep their information private. The aggregation of the

data is done using methods like Fed Averaging (FedAvg), Fed Proximal (FedProx), and Fed Plus (Fed+).

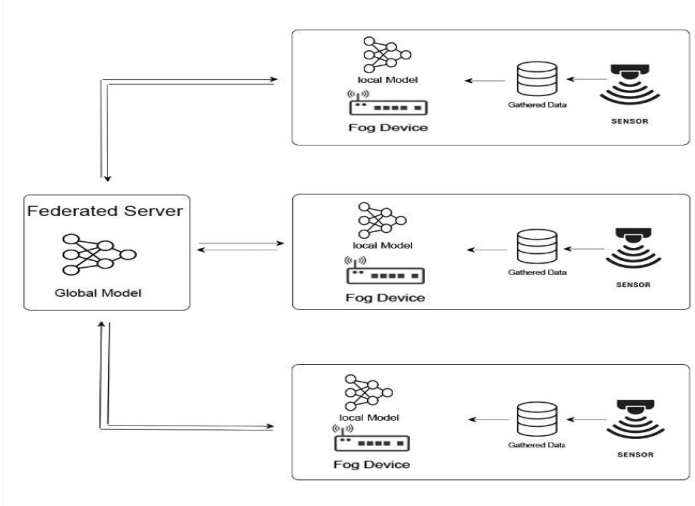


Figure 1. Federated learning-based system

1.1. Challenges

There are four challenges associated with the distributed optimization problem in federated learning [3].

- Challenge 1: Huge number of communications. In distributed and federated learning, millions of devices generating billions of data are joining the federated network, which makes communication become a critical issue in the network especially if they send raw data. So, it's necessary to find methods to minimize the communication between the devices on the network like sending smaller messages, sending only the local model updates or increasing the number of local trainings before sending the model to reach the optimal accuracy faster and decrease the number of communication rounds.
- Challenge 2: Systems Heterogeneity. The capabilities of the devices joined in the federated network may differ because of the variability of the hardware such as the CPU, RAM, Battery, and network. Many devices may disconnect or dropout at the middle of the training due to their limitations and constraints which leads to a very small fraction of devices that actually join the federated network [4]. So federated learning methods need to address these system characteristics and anticipate a low number of joining nodes, tolerate this heterogeneity, have higher fault tolerance and straggler mitigation.
- Challenge 3: Statistical Heterogeneity. In real life scenario, the data generated by the devices in each local network are non-identically distributed. This distribution violates the identically distributed data assumption which may lead to increasing the complexity of the model and evaluation. Moreover, the model can diverge and decrease the accuracy of it. New Aggregation methods are introduced that considers the quality of the data and distribution like fed+ to help the model to converge.
- Challenge 4: Privacy. Although federated learning has lowered the chance of exposing sensitive data by sending them through the network, it still needs the devices to send the updated local weights to the central device. Attacking methods have evolved that can extract useful information from the updates. Recent techniques introduced to reduce the risk of exposing the

data through the weights sent like dropping some weights before sending them or encrypting them before sending. But these solutions come at the cost of accuracy and time [5].

The main contributions of this paper can be summarized as follows:

- A DL model for IoT intrusion detection system is proposed.
- Apply centralized, distributed, and federated learning approaches with the proposed model to identify attack type in a state-of-the-art CICIDS2017 dataset .
- Define the challenges that may arise from using each approach .
- Comparatively evaluate the accuracy of these learning approaches to verify how much accuracy federated learning can attain while preserving data privacy.
- Evaluate the effect of the data distribution, the number of local training rounds and the communication rounds on federated learning's efficiency.

The remaining part of the paper is structured like this. Section II discusses previous research on using machine and deep learning in IoT. This includes centralized, distributed, and federated learning techniques. Section III explains the model we suggest and provides all the information about each learning method. Section IV talks about how the environment was prepared, the data used, and shows the results of the model on the three approaches. Section V finishes the paper and talks about what will be done in the future.

2. Related Work

In recent literature, a lot of studies that focus on transforming the traditional intrusion detection systems to smart IDSs, are targeting to apply federated learning solutions for their benefits in privacy. E. Mármol et al. [5] evaluate and perform analysis of the impact of non-iid data with different aggregation methods and training rounds using ToN_IoT dataset and IBMFL tool for simulation. They experimented 3 scenarios: basic scenario with unbalanced data, balanced scenario where all nodes have equal amounts of class and samples in each class and a mixed scenario where the data is unbalanced across all nodes but they choose certain nodes to enter the training based on their entropies by choosing the nodes which have entropy value that's higher than a certain threshold (0.2) and then enhancing their entropies by artificially resampling the data. all scenarios have been tested using fedAvg and fed+ aggregation methods. In the basic scenario, the results haven't been changed no matter how much rounds have been done or which aggregation method was used, due to the huge imbalance of the data which makes it very difficult to converge. In the balanced scenario, all parties have improved after the first round. But some of the nodes have faced convergence issues when the fedAvg method was used, but when they used the fed+ method, all nodes have converged successfully. The same case happened in the mixed scenario when using the fedAvg method, while using fed+ has led the model to converge to closer results as the balanced scenario. S. Rahman et al. [6] conducted experiments using Raspberry Pi devices to evaluate federated learning with different situations and compared it to a centralized model and a self-learning model. The first use case involves data being spread out based on different attack types. The experiment used 4 nodes, with each node only identifying one type of attack. Clients 3 and 4 have gained significant advantages from FL. In the final round, node 3 achieved a accuracy of 49.99% by learning on its own, but it was able to reach up to 76.06% starting from the second round by using federated learning. Node 4 has achieved 52.24% accuracy by self-learning, but it has achieved 82.23% accuracy using fl. In scenario #2, the data is divided evenly among 4 nodes. In self-learning, the accuracy reached was between 79.24% and 80.47%. For FL, the accuracy is between 80.45% and 81.48%. In use case #3, the data is spread randomly across the 4 devices. The accuracy of FL training is

between 76.84% and 77.79%. Zhao et al. [7] propose a new model called FL-LSTM for federated learning. FL-LSTM was tested using the SEA dataset and compared to the centralized version of the model and another FL model that uses CNN. The LSTM model that is centralized had an accuracy of 99.51%, while the fl model had an accuracy of 99.21%, which is very close to the centralized model and considered a good accuracy. The CNN has achieved a high accuracy rate of 95.48%. The FL-LSTM has achieved good performance and meets the privacy needs. Z. Chen et al. [8] propose an improved algorithm called FedAGRU that combines Federated Learning and Attention Gated Recurrent Unit. The FedAGRU uses federated architecture. It also has a special improvement using the attention mechanism. This mechanism gives devices different levels of importance by assigning them weights. The importance of a device is determined by how much it improves the classification performance of the overall system. The more important the device, the more weight it has. After determining the weight, the system prevents less important devices from sending their local updates during times when the available bandwidth is scarce. This improvement reduces calculation and communication costs. The new model was tested by using three datasets: KDD CUP99, CICIDS2017, and WSN-DS. FedAGRU was tested and compared to a similar model using FedAVG and CMFL. In every situation, FedAGRU performed better than all other models when it came to accuracy, time, and scalability. It was also tested with varying numbers of clients. The FedAGRU was able to stay very accurate, with about 99.8% success rate. However, the other methods became less accurate as more clients were added. Using the fedAVG method resulted in a success rate of around 69%, and using CMFL resulted in a success rate of around 89%. D. Attota et al. [9] proposed an IDS with multi-view information of IoT network data using federated learning. MQTT protocol dataset was used in the simulation. They simulated a network of ten devices connected with a central node acting as an aggregator. The average accuracy is 98% and it was compared to a non-fl system and the accuracy of it was 75%. Z. Tang et al. [10] propose a fl model that uses GRU as the detection model and CICIDS2017 dataset for the evaluation. The model was also evaluated by comparing it with the centralized approach. The research involved using two devices as clients and a device called an aggregator. Device 1 had an accuracy of 97.1% and Device 2 had an accuracy of 97.3% However, the centralized approach had an accuracy of 98.1%, which is a reasonable difference between the FL method and the non-FL method. P. Ruzafa et al. [11] uses fl model with unbalanced data distribution. The model is trained using the CIC-ToN_IoT dataset. The model was evaluated using the FedAVG and Fed+ aggregation methods and on 10 devices. The accuracies of the devices using fedAVG faced a convergence issue due to the unbalancing of the data. While in the fed+ case, the model has converged successfully and reached its optimal value by the 10th round.

3. Methodology

3.1. Learning Approaches

The evaluation took place among the aforementioned three approaches: centralized, distributed and federated learning, respectively. To apply centralized learning, a powerful device acting as a cloud is needed to train the model. IoT devices send their gathered raw data to the cloud to train the model on all traffic passing through the whole devices in the network. On the other hand, distributed learning relies on the concept of independence. Each device trains their own model on their local dataset but with the same model structure. In the experiment, the local dataset for each device is created by dividing the CICIDS2017 dataset into ten subsets of data with different levels of entropy and attacks.

The federated approach also performs the training phase on the devices without sending their raw data. But the difference between the federated and the previous approach, is that federated learning allows the devices to periodically share their local models' parameters with each other through the cloud. This setting has allowed the nodes to benefit from the others knowledge to improve their local model's accuracy. To perform this action, the central node first initiates a connection with devices and sends the original model to them. Each device then performs local training for a certain number of iterations using the local dataset. They send back the updated models to the cloud for two important tasks, the first task is to aggregate all the updated models into one updated global model. Secondly, it sends back the updated global model to the registered nodes. This whole process is repeated until reaching the desired accuracy. Distributed learning is expected to be the weakest regarding accuracy because each local model is trained on portions of data and doesn't contain all types of attacks. Centralized learning on the other hand, is expected to beat the other approach and achieves the best accuracy as it consists of one single global model trained on the whole dataset. The goal is to see if the federated approach can achieve the same accuracy as the centralized approach, or if it only falls somewhere in between the two approaches. Also, to check if the accuracy of each device will increase using federated learning, distributed learning was considered as benchmark to be used for evaluating federated learning. Moreover, federated learning is evaluated by running multiple scenarios with different number of communication rounds and local epochs. It's meant to evaluate the effect of these parameters on the federated learning efficiency. This research considers five case studies.

Case #1: each device performs 2 local training epochs before sharing their knowledge. This case is tested with different number of communication rounds to evaluate the convergence speed. It is tested with 1, 20, 40, 60, 80 and 100 rounds.

Case #2: each device performs 4 local training epochs before sharing their knowledge. This case is tested with different number of communication rounds to evaluate the convergence speed. It is tested with 1, 10, 20, 30, 40 and 50 rounds.

Case #3: each device performs 5 local training epochs before sharing their knowledge. This case is tested with different number of communication rounds to evaluate the convergence speed. It is tested with 1, 10, 20,30 and 40 rounds.

Case #4: each device performs 8 local training epochs before sharing their knowledge. This case is tested with different number of communication rounds to evaluate the convergence speed. It is tested with 1, 5, 10, 15, 20 and 25 rounds.

Case #5: each device performs 10 local training epochs before sharing their knowledge. This case is tested with different number of communication rounds to evaluate the convergence speed. It is tested with 1, 5, 10, 15 and 20 rounds.

All previous combinations were carefully chosen this way to sum up to exactly 200 epochs to perform a fair comparative study between the five cases.

3.2. IDS Deep Learning Model

The previously mentioned learning approaches are used to train a deep learning model utilized to detect the types of behaviors in the CICIDS2017 dataset. Fig. 2 shows the MLP model proposed, it consists of seven layers. The input layer has 78 neurons that represent the 78 features of the dataset samples. After that, there are 3 layers in the middle of the neural network. Each layer has a different number of neurons: 512 in the first layer, 256 in the second layer, and 32 in the third layer. All of these layers use the relu activation function. The last hidden layer is a special layer called dropout. Its purpose is to prevent overfitting by randomly removing or "dropping out" some of the weights during each update of the weights. The probability of a weight being dropped out is set at 20%. Finally, the last layer of the

network is made up of 4 neurons. These neurons use a special function called sigmoid to calculate their output. Then, the output from these neurons is passed through another layer called softmax, which helps classify the input into one of the 4 different classes.

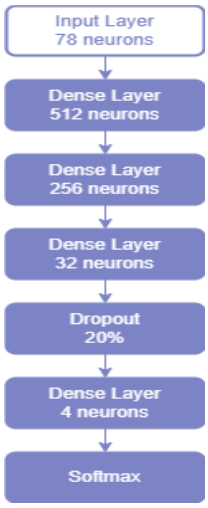


Figure 2. The used deep learning model.

3.3. Preprocessing

To prepare the traffic to be ready for training, some data had to be fixed because they were unlabeled corrupted data [13], to facilitate fixing the data, a cleaned version of CICIDS2017 was proposed by [14]. Since the experiments are meant to test a real scenario, the data needed to be kept unbalanced as it is, so no resampling was done. After cleaning the data, all semples are normalized over the features to be on a common scale to correctly model the data [17]. The final step is to convert the result labels to one hot encoded categories, each category represents a class of the behavior types.

3.4. Metrics

The performance of each local model was measured by calculating the precision, recall and F score. Precision describes the ratio between the correctly predicted positive samples to the total number of positive predictions whether they are correctly predicted or not. Recall describes the ratio between the correctly predicted positive samples to the total number of positive samples. Finally, the F-score is a measure of the test's accuracy and it is calculated from the precision and recall of the test.. All metrics were calculated by testing the local models' detection of the behavior types provided in the CICIDS2017 dataset. The precision, recall and F Score are measured using the following equations (1), (2) and (3), respectively, where TP, TN, FP, and FN stand for True Positive, True Negative, False Positive, And False Negative respectively.
The unbalancing of the data at each node is measured using Shannon Entropy [15], it's measured using equation (4) where the result is 0 if all classes are 0 except a single class, and is 1 if all ci = n/k , n is the number of samples, k is the number of the classes of behaviors, and ci is their size [5].

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \quad (1)$$

$$\boldsymbol{Recall} = \boldsymbol{TP}/(\boldsymbol{TP} + \boldsymbol{FN}) \quad (2)$$

$$Fscore = 2x\ PrecisionxRecall/(Precision + Recall) \quad (3)$$

$$Entropy = -\sum_{(i=1)}^{k} \frac{ci}{n} log \frac{ci}{n} / log\ k \quad (4)$$

3.5. Aggregation Methods

There are various types of aggregation methods introduced in federated learning. The commonly used algorithms are FedAvg, FedProx and Fed+. Noting that every one of them keeps their data locally for privacy preserving. FedAvg is a communication efficient algorithm for the distributed training with a large number of devices [18]. This method averages the updates incoming to the central device from the local devices. It has shown very good performance with the simple and non-convex problems, but it provides no guarantees to converge, it may diverge with the non iid data which is the real-life case. FedProx [19] was introduced as an extension of FedAvg. It has made a modification to the FedAvg to ensure the convergence in practice and overcome the statistical heterogeneity. Fed+ [20] is the most recent algorithm of them. It was also introduced as an extension of FedAvg. It unifies multiple federated learning algorithms to better deal with the real-life characteristics. It can address the non iid data and how to deal with stragglers. In this paper, Fed+ algorithm is used for its advantages to ensure the convergence of the model and focus on the main target and what needs to be evaluated.

4. Experimental Results

4.1. Dataset

The CICIDS2017 dataset [12], is one of the popular datasets for smart intrusion detection systems. The creators of the dataset have collected the network traffic for five days. They targeted specific behaviors each day. The creation of it has not put into consideration to be used with federated learning evaluation; however, it provides a huge number of samples and highly unbalanced classes which helps this article with evaluating how imbalanced and non-uniform data affects the model convergence. The purpose of this research is comparing the three learning approaches, not attempting to enhance the model. So, only four types of network behavior in the dataset are being considered. The chosen classes are Bengin, DDos, DoS Hulk and Portscan because the majority of samples found in the dataset are of these behaviors. Table 1 shows the distribution of the chosen data samples from the CICIDS2017 dataset. To make testing consistent across all training nodes, a separate set of data was created. This new dataset consists of 20% of the original full dataset and was given to each node to use for their testing. Table 2 shows the test samples.

Table 1. Data distribution of the training samples from CICIDS2017 dataset

Classes	No. of Samples
Benign	121,957,5
DDoS	128,027
DoS Hulk	231,073
PortScan	158,930

To get ready for the distributed and federated learning experiments, the dataset was divided randomly into 10 smaller datasets. To make sure that these smaller datasets were unbalanced, the entropy was

measured [15]. Entropy closer to 1 means better balance of the data. The ten subsets of data are changed until they have closer real data distribution. Table 3 displays how the training samples are spread out among ten nodes and their entropy. Moreover, to test the effect of changing the distribution of the data on federated learning, another distribution is protposed by changing the entropies of nodes 5, 6 and 9. Table 4 shows the new distribution of the training samples for specific nodes.

Table 2. Data distribution of the test samples

Classes	No. of Samples
Benign	295665
DDoS	25605
DoS Hulk	46025
PortScan	31761

Table 3. Distributed data samples across 10 nodes and their entropy

Device	Entropy	Benign	DDos	Dos Hulk	Port Scan
0	0.80192	72843	50532	12378	10428
1	0.76145	92915	0	45510	61374
2	0.40828	167686	7215	0	30429
3	0.36965	283074	0	23056	24650
4	0.1906	191886	0	15366	0
5	0.26683	88932	12300	0	0
6	0.27698	109028	3541	8492	0
7	0.97581	56653	31233	49094	32049
8	0.65819	99977	16135	67232	0
9	0.50327	56581	7071	9945	0

Table 4. Distributed data samples across specific nodes and their entropy for the second unbalanced scenario

Device	Entropy	Benign	DDos	Dos Hulk	Port Scan
2	0.37654	194817	7215	0	30429
5	0.18986	154511	12300	0	0
6	0.07517	200225	0	4421	0
9	0.35167	132540	7071	14016	0

4.2. Environment Setup

Experiments were performed on an Intel Core i7-8750H CPU running at 2.20GHz to 2.21 GHz, also has 16GB of RAM and a Nvidia GTX 1060 graphics card with 6GB of dedicated memory. The tool used during all experiments is Anaconda3 and python as the development language. The IBMFL framework [16] is a tool used to model and imitate a network that is spread out and apply the concept of federated learning. IBMFL is a tool made with python, it helps make virtual nodes and uses federated learning on them. IBMFL can use different ways to learn, so it can use any type of machine learning or deep

learning models. This means it can work with both supervised learning, where it is given labeled data to learn from, and unsupervised learning, where it learns patterns or structures in the data without any labels. The network has ten devices called edge devices. These nodes do the training of the model and there is a central node that collects all the trained models and shares the updated model with the nodes. The deep learning model that was used was made, trained, and tested using Keras [21]. Keras is an open-source python package, it provides a python interface for neural networks.

4.3. Results

Experiments were conducted to see how federated learning affects the accuracy of the proposed model compared to centralized learning when using the samples shown in table 1 for training. The model was also tried out using distributed learning where each node had some of the data samples and didn't share their model weights with other nodes. This was done to see if the nodes would gain any advantages from using federated learning. Finally, the model was tested in all previously mentioned case studies.

4.3.1. Centralized Learning

The model has been trained and tested using the traditional way of learning to achieve the highest accuracy possible. This result will be used as a reference for distributed and federated learning methods. All sub-datasets were combined and used to teach the main model. The model gets a fscore of 98.69% using the testing samples shown in table 2.

4.3.2. Distributed Learning

The model was taught and evaluated using distributed learning. This method involved dividing the data into ten parts and placing each part on a separate node, as shown in Table 3. Each individual node works on its own without needing to communicate with other nodes. Nodes from 0 to 9 have reached f-score of 85.3628%, 84.7%, 73%, 77.3%, 63.2%, 71.8%, 63.2%, 93.4%, 86.684% and 84.6%, respectively as shown in Fig. 3. The f-score was lowest when using nodes 4 and 6. These nodes had datasets that were not evenly balanced and had the lowest levels of entropy. Node 7 has the highest f-score, which is 93.41% This is because it has the most evenly distributed samples for all categories according to Table 3. None of the nodes has achieved the f-score of the centralized approach because they haven't been trained on the entire dataset and most nodes haven't even seen one or two classes.

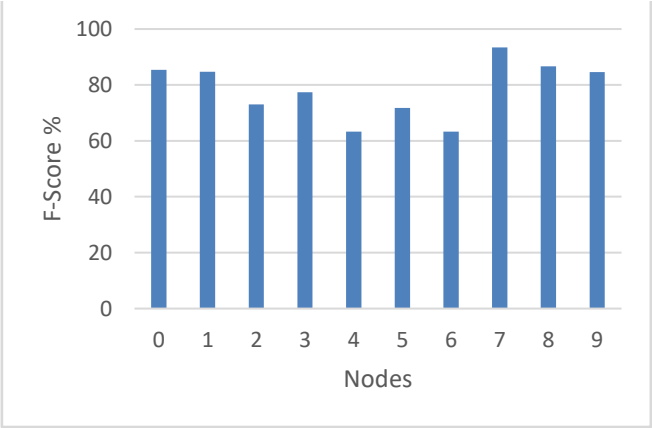


Figure 3. F-Score for the Distributed Learning Scenario.

4.3.3. Federated Learning

To simulate federated learning, IBMFL framework is used. IBMFL is a python framework used to create virtual nodes and apply federated learning on them. IBMFL provides a basic structure of FL, and a researcher may add more advanced features to it. It's not dependent on specific learning algorithm allowing it to apply any machine learning or deep learning models which means it supports supervised and unsupervised learning .The federated learning experiment has been conducted by creating ten virtual nodes acting as edge devices. Each node is connected through a central node acting as the aggregator using the Fed+ aggregation function.

a) Evaluation of FL against other approaches

The dataset is splitted across the nodes as shown in table 3. Two iterations of local training were done with 1, 20, 40, 60, 80 and 100 global rounds which is case #1. As shown in Fig. 4, most of the devices took around 100 complete global rounds to reach their best value. Federated learning has made most of the nodes reach a higher level of accuracy compared to distributed learning. The convergence happened because the connected nodes were exposed to a larger collection of data and different types of attacks that they had never encountered within their smaller sets of data. It is observed that even though all points receive the same updated overall model, they tend to have varying accuracies. This happens because the new local model is a combination of the old local model and the incoming global model. In this article, both local and global models are merged with the same percentage.

b) Evaluation of using different epochs and rounds

One of the important parameters that control the model convergence are how many local training epochs and communication rounds the model needs to reach the optimal accuracy. To test the effectiveness of increasing the local epochs, the experiment has been conducted by running the five different cases defined previously. Figure 4 shows the f-score of each case study. In case # 1, each node performs two local training epochs for their local model before sharing their knowledge. With this number of epochs, the model needs about 100 communication rounds to converge to its optimal accuracy. On the other hand, when the number of epochs increased, the nodes began to reach their optimal values earlier. In case #2 and case #3, the nodes needed from 10 to 20 rounds to reach their optimal accuracy. Moreover, by even more increasing the epochs, case #4 and case #5, the nodes were able to reach their optimal accuracy in only 5 communication rounds. By increasing the number of epochs, allows each local model to better understand and build better knowledge before sharing it with others. This increase has significantly improved the whole system's efficiency by greatly reducing the overhead on the network due to the amount of communication done between the nodes and the aggregator. In real life, thousands of devices may be connected through the IoT network, which will introduce huge overhead on the network.

c) Effect of the distribution on federated learning

The effect of the distribution on federated learning was studied by first categorizing the nodes that have similar types of classes to study their behavior. To more understand the effect of the distribution, the model was trained on the other proposed distribution of the data shown in table 4. The nodes are categorized as follows, Nodes 0 and 7 form category 1 that contains all classes. Nodes 1 and 3 form category 2 that don't have the Ddos class. Nodes 6, 8 and 9 form category 3 that don't have the portscan class. Finally, nodes 2, 4 and 5 don't have similar nodes, so they don't belong to any category. The nodes in category 1 as shown in fig. 4, were able to converge and almost reach the centralized accuracy,

since they have access to all types of classes from beginning and relatively balanced. The nodes in category 2 were also able to reach almost the same accuracy but with slight difference as node 1 has higher entropy than node 3. However, their accuracies were less than category 1 nodes as their local model weren't trained on Ddos locally, which in turn affected their ability to detect the missing class. The nodes in category 3 on the other hand, have the same classes, but have much different accuracies than each other. Even though node 6 has the same classes as nodes 8 and 9, but its accuracy is less than them by more than 10%. The reason for this difference is that node 6's entropy is very low compared to nodes 8 and 9 which means that the number of samples of each class is highly unbalanced. The majority of the samples are from the benign class, and the number of samples of the ddos and dos hulk classes are not enough for the model to correctly classify them. Nodes 4 and 5 have the lowest accuracies due to the miss of two classes out of four, even though node 5 has better entropy than node 6, but still node 6 was able to reach higher accuracy because of its access on a greater number of classes than node 5. Finally, it is noticed that node 2 has better entropy than nodes 3 and 6 and missing only the dos hulk class, but still had lower accuracy than them which had dos hulk samples. Apparently, the type of class and its feature values also affect the node's accuracy, according to how hard it can be differentiated from the others. The node's accuracy is affected by the number of classes, the number of samples of each class, the balancing of the classes and the type of classes. These constraints can put upper limit to the accuracy a node can achieve. To further study those constraints, the datasets were switched to the other dataset defined in table 4. Nodes 5, 6 and 9's entropies were decreased. As shown in fig. 5, node 5 was not really affected by the change because the change was only by increasing the benign samples but keeping the same number of samples of the ddos class, it was already having high number of samples of the benign class. So it was still able to classify between the classes with the same accuracy. It's noticed that even though node 6's entropy was decreased by completely removing the ddos class, it was able to achieve higher accuracy. It could be because when the ddos was removed, the local model was able to better classify the dos hulk. Another assumption is that this difference falls in the normal range when running the experiment multiple times, the accuracy may vary few percentages. Finally, node 9 decreased about 10% by this change because the entropy was much decreased. Although the number of samples of dos hulk increased, it still decreased because it may need a greater number of samples of ddos and dos hulk to differentiate between them.

In Fig. 5, the f-scores of the three learning approaches are being compared for each node. The centralized learning approach is the best because it trains the model using the entire dataset, resulting in the highest f-score for all nodes. The federated learning achieves higher f-scores compared to distributed learning for all nodes and can achieve the same values as centralized learning for nodes 0, 7, 8, and 9. Their datasets include samples from all types of attacks, except for the PortScan class. The data in node 6 is similar to nodes 8 and 9, but it has a lower entropy. This made it difficult for the model to learn accurately as in centralized learning. However, the f-score of node 6 in the federated learning is still much better than the f-score of the distributed learning. The same thing happens with node 1 and node 3, but their data does not include the DDoS class. It appears that the way the type of classes and their number of samples are distributed in the local sub-dataset sets a maximum limit on how much improvement federated learning can make. This conclusion is supported by the findings from nodes 4 and 5. These nodes have two missing classes in their smaller datasets and have low entropies, which in turn resulted in reduced accuracy in federated learning.

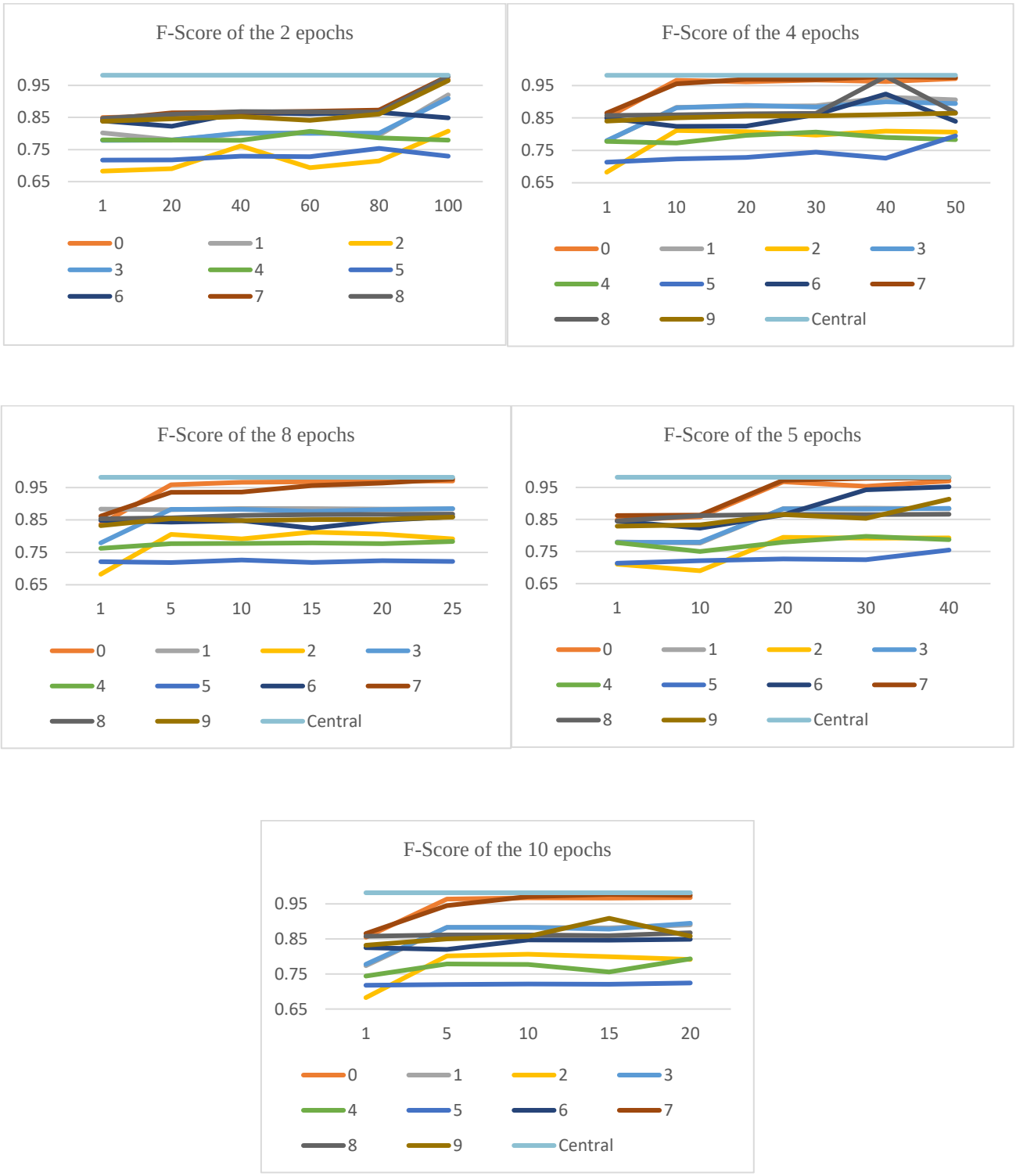


Figure 4. Fscore of each node for every case.

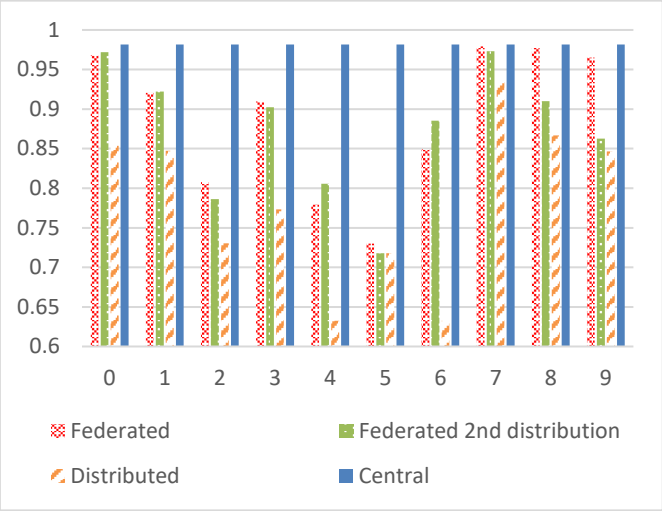


Figure 5. Centralized vs Distributed vs Federated Learning.

5. Conclusion

This paper proposed a deep learning model to detect threats in IoT networks. The model is used with three different learning approaches: centralized, distributed and federated learning. The model is trained using a dataset called CICIDS2017. This dataset is divided into ten smaller datasets. These smaller datasets are used to evaluate distributed and federated learning. The f-score is used to measure how accurate the model is. It has been found that using centralized learning resulted in the highest accuracy of all approaches, with an F-Score of 98%. However, this approach completely violates data privacy because the IoT nodes share their raw data in the cloud. In distributed learning, ten separate edge devices learned without exchanging their knowledge and got F-Scores ranging from 63% to 93%. It has protected the privacy of the data of the nodes, but the accuracy of the model is reduced because each individual model is trained separately. The federated learning approach has an F-Score ranging from 73% to 98% with an average of 89% across the ten nodes. The differences in accuracy came from how the data was distributed, how much uncertainty there was. So, the best way for IoT IDS is to use the federated approach. This means that the nodes can share their training with each other. This makes the model more accurate and brings it closer to the accuracy of centralized learning. Moreover, the effect of the number of local training rounds and the communication rounds on federated learning's efficiency was also evaluated. Five case studies were tested, each case study has different numbers of local epochs performed before sharing the parameters. Case #1 with the lowest epochs reached the optimal value after about 100 rounds. By increasing the epochs, the model kept converging faster than before, so by case #4 and 5, the model needed only about 5 rounds to reach its optimal value. Finally, the effect of the data distribution was tested by modifying the entropies of some sub-datasets. It's concluded that even when the entropy was lower for some nodes like node 6 doesn't necessarily mean that the accuracy will decrease, the type of behavior also controls their convergence according to its nature and feature values. It's also important to mention that if the data is highly unbalanced evenly between nodes, federated learning won't be able to achieve accurate results. This problem can be solved by leaving out poorly informed nodes. In the future, how to choose the best nodes in federated learning based on selection techniques to make the models more accurate will be studied.

References

1. S. Patel and R. Patel, “Fog Computing: A Comprehensive Analysis of Simulation Tools, Applications and Research Challenges with Use Cases,” J. Eng. Sci. Technol. Rev., vol. 15, no. 3, pp. 63–83, 2022, doi: 10.25103/jestr.153.08.
2. H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, “Communication-efficient learning of deep networks from decentralized data,” Proc. 20th Int. Conf. Artif. Intell. Stat. AISTATS 2017, vol. 54, 2017.
3. T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated Learning: Challenges, Methods, and Future Directions,” IEEE Signal Process. Mag., vol. 37, no. 3, pp. 50–60, 2020, doi: 10.1109/MSP.2020.2975749.
4. K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, and V. Ivanov, “T f l s : s d,” 2017.
5. E. Mármol et al., “Evaluating Federated Learning for intrusion detection in Internet of Things : Review and challenges,” vol. 203, no. July 2021, 2022.
6. S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, “Internet of Things Intrusion Detection : Centralized , On-Device , or Federated Learning ?,” pp. 1–8, 2020.
7. R. Zhao, Y. Yin, Y. Shi, and Z. Xue, “Intelligent intrusion detection based on federated learning aided long short-term memory,” Phys. Commun., vol. 42, p. 101157, 2020, doi: 10.1016/j.phycom.2020.101157.
8. Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, “Intrusion Detection for Wireless Edge Networks Based on Federated Learning,” IEEE Access, vol. 8, pp. 217463–217472, 2020, doi: 10.1109/ACCESS.2020.3041793.
9. D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, “An Ensemble Multi-View Federated Learning Intrusion Detection for IoT,” IEEE Access, vol. 9, pp. 117734–117745, 2021, doi: 10.1109/ACCESS.2021.3107337.
10. Z. Tang, H. Hu, and C. Xu, “A federated learning method for network intrusion detection,” Concurr. Comput. Pract. Exp., vol. 34, no. 10, pp. 1–16, 2022, doi: 10.1002/cpe.6812.
11. P. Ruzafa-Alcazar et al., “Intrusion Detection Based on Privacy-Preserving Federated Learning for the Industrial IoT,” IEEE Trans. Ind. Informatics, vol. 19, no. 2, pp. 1145–1154, 2023, doi: 10.1109/TII.2021.3126728.
12. I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,” no. Cic, pp. 108–116, 2018, doi: 10.5220/0006639801080116.
13. R. Panigrahi and S. Borah, “A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems,” no. January, 2018.
14. Koorosh Aslansefat, JelenaNikolicElfak, and Om Rastogi, “CICIDS2017 | Kaggle,” 2019. https://www.kaggle.com/datasets/cicdataset/cicids2017 (accessed Mar. 21, 2023).
15. J. A. Bonachela, H. Hinrichsen, and M. A. Mũoz, “Entropy estimates of small data sets,” J. Phys. A Math. Theor., vol. 41, no. 20, 2008, doi: 10.1088/1751-8113/41/20/202001.
16. H. Ludwig et al., “IBM Federated Learning: an Enterprise Framework White Paper V0.1,” pp. 1–17, 2020, [Online]. Available: http://arxiv.org/abs/2007.10987

17. Xiaoharper, “ML Studio (classic): Normalize Data - Azure | Microsoft Learn.” https://learn.microsoft.com/enus/previousversions/azure/machine-learning/studio-module-reference/normalize-data (accessed Mar. 22, 2023).
18. T. Sun, D. Li, and B. Wang, “Decentralized Federated Averaging,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 14, no. 8, pp. 1–13, 2022, doi: 10.1109/TPAMI.2022.3196503.
19. T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated Optimization in Heterogeneous Networks,” 2018, [Online]. Available: http://arxiv.org/abs/1812.06127
20. A. Kundu, P. Yu, L. Wynter, and S. H. Lim, “Robustness and Personalization in Federated Learning: A Unified Approach via Regularization,” Proc. – IEEE Int. Conf. Edge Comput., vol. 2022-July, pp. 1–11, 2022, doi: 10.1109/EDGE55608.2022.00014.
21. Ketkar, N. “Introduction to Keras. In: Deep Learning with Python” Apress, Berkeley, 2017 CA. https://doi.org/10.1007/978-1-4842-2766-4_7
22. O. Elnakib, E. Shaaban, M. Mahmoud, and K. Emara, “EIDM: deep learning model for IoT intrusion detection systems,” J. Supercomput., vol. 79, no. 12, pp. 13241–13261, 2023, doi: 10.1007/s11227-023-05197-0.