Anomaly Detection for IoT: Another Look At Federated Learning

Ciprian Bica
University of Twente
The Netherlands
c.bica@student.utwente.nl

ABSTRACT

The increase in the amount of common, everyday objects that contain embedded devices capable of processing data and communicating over a network gave rise to the paradigm called Internet of Things (IoT). As these devices integrate into daily life, they become indispensable for many different purposes ranging from home automation to industrial control systems. However, their increasing presence also makes them prime targets for malicious actors seeking to exploit any vulnerabilities these devices may exhibit. From orchestrating large-scale botnet attacks to stealing sensitive data or disrupting critical services, the motivations behind targeting IoT devices are diverse and often financially driven. For the most part, these target devices are not equipped with the best security measures to stay resilient against attackers, thus making them easy targets. One way of improving the security aspect of this system would be to analyse the network traffic of these devices and scan for and identify malicious attacks targeting them. Utilising the IoT-23 dataset, comprised of network traffic captures from various IoT devices, alongside a Federated Learning approach, the objective is to spot any anomalous traffic between these devices, which usually indicates an attack is happening. The IoT-23 dataset is comprised of 20 malware captures for IoT devices, and 3 captures for benign traffic. This study compares two established federated learning algorithms, FedAvg and FedProx, to determine their effectiveness in anomaly detection for IoT devices. Two different setups were tested, and it was found that for this dataset and the anomaly detection task, FedAvg seems to perform better in terms of accuracy, precision, recall and f-score on one of the setups, while on the other setup, the performance of the two algorithms was more similar. These results are analysed and conclusions are drawn.

KEYWORDS

IoT, federated learning, IoT-23, anomaly detection, network traffic
Ciprian Bica. 2024. Anomaly Detection for IoT: Another Look At Federated Learning. In *Proceedings of Twente Student Conference on IT (41st Twente Student Conference on IT).*

1 INTRODUCTION

The Internet of Things can be defined as a network of different devices, ranging from household items to industrial machines, that are connected to a singular network and can both process data and exchange information with each other, forming a sort of ecosystem. The number of these devices is increasing at a rapid pace, as the applications of IoT are extensive and diverse. From agriculture, transportation and industrial manufacturing to household items and wearable technology like smart watches, the impact of IoT in our life cannot be underestimated. The IoT brings a large number of advantages to the table, such as more data and a potential decrease in costs and waste of resources, as well as more a accurate analysis, and an improvement to efficiency and decision-making.

Because of this, it is not hard to infer that the IoT ecosystem has become a victim of an increasing number of attacks in the last years by malicious parties, in order to damage or stop critical services, or to expose and get access to private information. Additionally, it looks like this ascending trend is not likely to stop or slow down in the near future. Consequently, the vital task of maintaining the security and integrity of these devices is becoming harder and harder, especially as their number keeps growing.

A way to make this environment safer comes with the help of artificial intelligence technology capturing the network traffic between these devices and performing a detailed network analysis on it. Artificial intelligence approaches can usually benefit from the large amount of data that these devices produce by learning how they communicate and detecting unusual patterns in the traffic of information, thus knowing if and when an edge device has become infected or is being attacked.

Federated learning (FL) [17], has the objective of training a machine learning algorithm on multiple local datasets that belong to local nodes, with no data samples being exchanged between them, thus making it secure. What is being exchanged are the resulting parameters of the models (weights and biases) after training them, so that a centralised entity, for example a server, can produce a global model. The main advantage of this new method is maintaining the privacy of the nodes, as the actual data never leaves the device. This also means that, if this approach was applied in a real-world application, the training would happen on real-world data, and not on some publicly available dataset, so the results would be more likely reflect reality. On the other hand, the main downside of this approach is that recurrent communication is needed during the training process, causing overhead [17]. This could of course be subject to the influence of network conditions as well, which may not always be ideal, especially in the case of IoT devices. Usually only a part of the available devices will actually be capable of training the model effectively. This relatively novel approach was first introduced by Google in 2016 [8] to predict the text input of users on thousands of Android devices while not transferring any data from them.

For the federation part of the project, the Flower framework was used. This framework already implements several established federated learning algorithms. In this project, a multilayer perceptron

41st Twente Student Conference on IT, July 2024, Enschede, The Netherlands

(MLP) with one hidden layer is used. Due to the relatively low number of features that each packet had, a MLP was a more appropriate choice for the model because it is less complex, shortening the running time of the training and also reducing the risk of overfitting. Two setups are created for the data, and a number of simulations are run with both FedProx and FedAvg on both setups. The results from these simulations are then analysed and interpreted.

This paper will first provide a short review of similar works in the field of federated learning, and some insight about the dataset that was used. Then the methodology of the research will be explained along with some details about the processing of the data. Finally, the results from the experiments are shown and conclusions are drawn from them.

2 RELATED WORK

In order to search for literature related to this subject Google Scholar was used.

The concept and potential applications of Federated Learning have been thoroughly explored in previous papers [6, 18], with many practical examples in diverse fields, such as industrial engineering (for example creating a environmental monitoring frame based on federated region learning [4]), healthcare, where this approach excels in regards to maintaining the privacy of the personal data usually involved in this field (using health records, a federated patient hashing framework that detects similar patients located in different hospitals without sharing any patient-specific data using FL was created [5]), as well as applications concerning mainly mobile devices (ranging from trying to predict the next emoji that a user will type [13], to predicting human behaviour using FL [16]). Another recent paper [1], also on the topic of comparing federated learning algorithms, came to the definite conclusion that no single algorithm is completely superior to the others, with six main algorithms (FedAvg, FedProx, FedYogi, FedAdam, SCAFFOLD, and FedDyn) put to the test. Also on the topic of comparing algorithms, [11] found that out of FedAvg, Federated Stochastic Variance Reduced Gradient, and CO-OP, FedAvg performed the best on the MNIST dataset, even when partitioning the data in a non-iid (independent and identically distributed) way. The same study showed that FedAvg was not able to surpass a centralised approach if the data was not iid.

For anomaly detection specifically, it has been shown before that machine learning is very adequate in multiple studies [2, 10, 15], with the former even comprising a review of a collection of techniques of machine learning dedicated solely to this application.

Combining the two fields, an interesting paper [9] showed that FL outperforms a centralised approach using the same deep learning algorithms in terms of accuracy when attempting to identify attacks in the network traffic of IoT devices. The research was performed on the ModBus dataset, using gated recurrent units (GRUs) models with an ensembler (which also aids in improving the accuracy rate by combining the predictions from different layers of GRUs). The same paper also showed that the training time was also greatly reduced when using the FL approach, which highlights another advantage of this approach. Another interesting proof of concept has been implemented for anomaly detection for smart buildings using federated learning [14], and its performance has been validated on three real-world datasets from the IoT production system at General Electric Current smart building. One other paper that comes to mind is a similar attempt on the IoT-23 dataset [12], that used TensorFlow as it's framework, and managed to reach 70% accuracy for its global model. It implemented the basic version of the FL algorithm (FedAvg) with 23 nodes (consequently each node had one sample) and also used a Multi-Layer Perceptron (MLP) as the model for each node. It is also worth mentioning that the paper used a multi-class classification model instead of a binary one. This paper plans to expand on those findings by comparing multiple algorithms, as well as compare the accuracy with changed parameters, such as number of nodes used.

In a previously mentioned paper [7], it is stated that FedProx is expected to produce better results in terms of accuracy and converging on non-iid data, which is the case for the second setup in which the tests were running, as each client had it's own "unique" dataset on which it learned, leading in theory to global convergence as each node can contribute meaningfully to improving the global model.

3 IOT-23 DATASET

The dataset used for this project was the publicly available IoT-23 dataset [3] containing network traffic captures from Internet of Things devices. As the name suggests, the dataset is made up of 20 malware captures and 3 benign captures. This dataset was published in January 2020, with the captures ranging from 2018 to 2019. This IoT network traffic was captured in the Stratosphere Laboratory, AIC group, FEL, CTU University, Czech Republic. For the malicious captures, a specific malware sample was executed on a Raspberry Pi, while for the benign captures, the normal traffic from three IoT devices (Amazon Echo, Soomfy smart Doorlock, Philips Hue lamp) was used. Both malicious and benign scenarios were run in a controlled environment with unrestricted internet access to help emulate the real-life conditions of these devices.

Every capture mainly consists of the following files, along with a README file:

- .pcap: this is the original .pcap file from the network traffic capture, for which Wireshark was used.
- conn.log.labeled: this is the Zeek conn.log file obtained by running the Zeek network analyser using the original .pcap file. This conn.log.labeled file has the flows of the capture network connection as a normal Zeek conn.log file with two new additional columns for the labels. The labels were added by a Python script according to a set of rules specific to each capture.

For this project the latter file was used, as it includes the labels for the packets, which are required when using this data for machine learning, as well as avoiding the difficulties encountered when processing and working with a .pcap file.

In total, the dataset contains 325,307,990 Zeek flows, out of which 30,858,735 were benign, leaving 294,449,255 to be classified as malicious, giving a ratio of approximately 9.5 malicious packets to every benign one. This shows a notable imbalance in the dataset.

Furthermore, the size of some data samples exceeded available working memory and contained millions of almost identical (malicious) packets alongside thousands of benign ones. Consequently, the dataset was divided into 15 smaller files: 12 files containing malicious samples, each corresponding to a distinct type of attack, and three benign samples, which were left intact due to their smaller size. Each file had a similar size of around 500 KB, or around 3000 packets which were manually selected, to create an optimal distribution that the model can learn the most from. This modification does not only considerably speed up the training process but it can also help the model generalise better, making the data less imbalanced, as the packets from each attack type were sampled to roughly resemble the distribution in the original scenario. The other approach would have been to include the entire samples, but training on millions of malicious samples while having only a few thousand benign ones, as was the case for the majority of samples, would likely have made the task of correctly identifying benign traffic even more difficult.

3.1 Features

The IoT-23 dataset packets contain 23 features that describe the network traffic flows. Each feature is listed in Table 1, along with a short description:

During the preprocessing phase, some of these features were removed to focus on the more relevant ones for the analysis, as is explained in more detail below.

3.2 Data Preprocessing

The inclusion of the full, unaltered dataset in the model training process would likely have negatively impacted the model's general accuracy. This is because highly imbalanced datasets can lead to biased learning, where the model becomes overly tuned to the majority class (in this case malicious flows) and underperforms on the minority class (benign flows). This imbalance can cause the model to exhibit poor generalisation on unseen data, resulting in higher false positive rates and potentially overlooking benign traffic entirely by erroneously classifying every packet as malicious.

The dataset originally had 23 columns for every flow, with 21 being features and the remaining two being the label and a more detailed description of the label, as it is explained in the previous subsection. Out of these 23 columns, 4 were removed when processing the files, as either they were weakly correlated to the label column, or there was so much data missing from them that keeping them would have likely impacted the accuracy negatively or not at all.

The columns that were removed are the following, along with the reasons for their removal.

(1) **ts** - The timestamp was removed because the exact time of the connection is not relevant to the classification task. The model should focus on the nature of the connections rather than when they occurred.

Field	Description
ts	Timestamp of the network connection, indicating the time when the connection was initiated.
uid	Unique identifier for each connection, used to distinguish between different network flows.
id.orig_h	Origin host IP address, representing the source of the network traffic.
id.orig_p	Origin host port, indicating the port number used by the source.
id.resp_h	Responding host IP address, representing the destination of the network traffic.
id.resp_p	Responding host port, indicating the port number used by the destination.
proto	Protocol used for the network connection (e.g., TCP, UDP).
service	Service used in the connection (e.g., HTTP, DNS).
duration	Duration of the connection, measured from initiation to termination.
orig_bytes	Number of bytes sent from the originator to the responder.
resp_bytes	Number of bytes sent from the responder to the originator.
conn_state	Connection state, representing the status of the network connection (e.g., established, closed).
local_orig	Indicates whether the originating host is within the local network.
local_resp	Indicates whether the responding host is within the local network.
missed_bytes	Number of bytes missed or dropped during the connection (packet loss).
history	History of the connection states and events, providing a sequence of state changes.
orig_pkts	Number of packets sent from the originator to the responder.
orig_ip_bytes	Number of IP bytes sent from the originator to the responder.
resp_pkts	Number of packets sent from the responder to the originator.
resp_ip_bytes	Number of IP bytes sent from the responder to the originator.
tunnel_parents	If tunnelled, is the connection UID value.
label	General label indicating whether the connection is benign or malicious.
detailed-label	Detailed label providing more specific information about the type of malicious activity, if applicable.

Table 1: Descriptions of the packet features.

(2) **uid** - The unique identifier for each connection was removed as it does not contribute to the classification of benign or malicious traffic.

(3) **tunnel_parents** - This feature was removed because it was missing from so many flows that no useful information could be extracted from it if it appeared.
(4) **detailed_label** - The detailed label was removed to simplify the classification task into a binary problem (benign or malicious). This helps to improve the model's performance by reducing the complexity of the labels.

The rest of the features were either encoded in categories if they were categorical (for example the protocol field), or normalised if numerical.

4 PROBLEM STATEMENT

Although there has been research done on this dataset and approach before, some possibilities were still left unexplored.

This paper will compare the results of the FedAvg [8] and FedProx [7] approaches for the anomaly detection problem and provide a broader perspective for this challenge on the IoT-23 dataset. The results of these approaches will be analysed in order to ascertain the effectiveness of each algorithm in comparison to the others when using the same machine learning model for the nodes.

4.1 Research Question

The research question that will be answered is:
How do different federated learning approaches (FedAvg, FedProx) compare to each other in terms of performance on the IoT-23 dataset?

The proposed metric for measuring performance for each of these algorithms will be the global accuracy. Other performance metrics such as precision, recall, F1-score may also be of interest.

5 METHODS OF RESEARCH

5.1 Implementation

This project relied upon the Flower framework, as this was used to simulate the clients and the server, as well as their communication. Python was chosen for the implementation, along with other machine learning libraries, among which flwr, torch, pandas, and sci_kit_learn are some of the most notable. The plots were produced using the matplotlib library. The MLP was the model used for the nodes in the implementation, with 18 input nodes, a hidden layer of 64 nodes, and 2 output nodes corresponding to the two types of labels. Various configurations for the hidden layer were experimented with, including configurations with 16, 32, 64, and 128 nodes. It was found that a hidden layer of 64 nodes yielded the best accuracy. The model by itself consistently achieved relatively good results of up to 99% accuracy, so it was deemed suitable to be used by the nodes in a federated setting. Simulations were run on the dataset in two ways: by giving each client one specific attack type so they can contribute to the global model, or by shuffling the data and dividing it evenly between the clients. The first setup allowed each node to learn from a "unique" dataset, leading, in theory, to an eventual global convergence, while the second was aimed at a more general distribution. To help with the unbalanced nature of the dataset, custom class weights were applied, making the model rank the packets that are part of the minority class as more important for tuning the loss function. For the experiments, all the nodes used a 80-20 split for the data, meaning that 80% of the data was assigned to the training set while the rest of 20% was used for testing. The experiment was conducted on a machine running Windows 10, equipped with 16 GB of RAM, 8 CPUs, and a 512 GB SSD.

The MLP itself could also be modified through the tweaking of its hyperparameters. These hyperparameters and their possible values are shown below:

Hyperparameter	Possible values
Hidden Size	8, 16, 32, 64, 128
Batch Size	16, 32, 64, 128
Learning Rate	0.01, 0.001, 0.0001, 0.00001

Table 2: Hyperparameters tried for the model

After some trial and error, as well as some heuristics, the values for the hyperparameters were chosen as 64 for the number of nodes in the hidden layer, a batch size of 128, along with a low learning rate of 0.00001, as these seemed to give the best and most stable results.

5.2 Methods

Two experimental setups were used. One where all the flows from the attacks and the benign samples were shuffled and evenly split between a number of clients, and a second one that aims to be closer to the real life, where some nodes are likely only exposed to one type of attack each, while some may only be exposed to benign traffic. The nodes exposed to benign traffic had to include both a benign sample and one random type of attack sample (otherwise there would not have been anything to classify). A more detailed justification for each setup can be found in the subsections below.

5.2.1 Setup 1: Shuffled and Evenly Split Flows. In this setup, all network packets from both attack and benign samples were shuffled and evenly split among a number of clients. This configuration is closer to what can be expected from a centralised model.

Justification:

- More Balanced Representation: By shuffling and evenly distributing all packets, each client receives a somewhat balanced mix of attack and benign traffic. This maximises the chance that the local model trained on each client has a good understanding of both types of traffic and of different types of attacks.
- Generalisation: This setup should test the model's ability to generalise across different types of attacks and benign traffic, providing a good baseline for classification.
- More Homogeneous Exposure: Since every client is exposed to a diverse set of attack types and benign samples, this setup helps in creating a more uniformly trained model, reducing the risk of bias towards specific types of traffic.

5.2.2 Setup 2: Simulation with Specialised Nodes. In this setup, some nodes were exposed to only one type of attack, while others were exposed to benign traffic. Additionally, nodes exposed to benign traffic included both benign samples and one random type of attack sample to help them differentiate the traffic.

Justification:

- Real-life Scenario Simulation: This setup more closely mimics real-world conditions where different devices or nodes might encounter specific types of attacks rather than a variety of them. This should test the model's practical applicability and resilience in real-life deployments.
- Specialisation: By exposing nodes to specific types of attacks, their local models can specialise in identifying those attacks. This can be particularly useful in environments where certain nodes are more likely to encounter specific threats. This also means that each node's contribution to the global model will be more valuable.
- Diverse Learning: This setup helps in understanding how well the global model can adapt to diverse conditions and specialise in detecting particular threats while still being able to identify benign traffic, giving some general insights into the model's flexibility and robustness.

These methods cover both a balanced and generalised scenario, as well as a more specialised and realistic scenario. They also ensure that the model is tested for classifying unseen data and specific attack detection capabilities. Future research can perhaps experiment more with the amount and the type of data available to each node and document how the learning changes.

6 RESULTS

The main results of this paper are the global accuracy comparisons between the FedAvg and FedProx federated learning algorithms on the IoT-23 dataset. First, the uniform setup is shown with both 10 and 100 clients, which achieves some respectable, yet expected results, proving the effectiveness of both algorithms in a controlled environment. Then the results from the specialised setup are shown, along with some interesting statistics, followed by a conclusion. A more detailed breakdown of the expected results can be found below.

6.1 Experimental Results

6.1.1 First setup. For the first setup, simulations were run with both 10 clients and 100 clients, and the results were fairly similar, with the accuracy usually spiking after the first few rounds of federation, and plateauing or growing very slowly after, as was expected. The plots of the accuracy for this setup with 10 clients can be seen in Figure 1.

A noticeable difference between the simulations with a different number of clients is that the accuracy seems to be negatively correlated with the number of clients, which was to be expected, as when there are fewer clients, each client has more data available to train on, making the model learn more. The global model always quickly converged at over 90% accuracy when using this setup with 10 clients regardless of the federation algorithm used.

With an increase in the number of clients a drop in accuracy is also expected, as can be seen from the graphs below, with the final accuracy after 200 rounds ranging from close to 60%, to around 80% in the better cases. The graphs show a strong indication of growth after the first spike, likely leading to an eventual accuracy convergence close to the simulation with fewer clients. This seems to show that convergence occurs faster when using fewer clients.

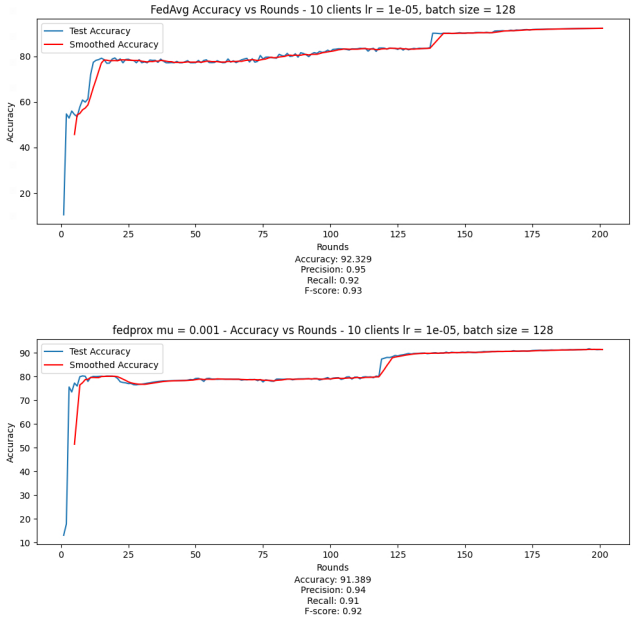

Figure 1: Accuracy over time for the first setup with 10 clients using FedAvg and FedProx, measured over 200 rounds of federation.

The proximal μ value for FedProx did not seem to make any meaningful difference to the results, probably because of the uniformity of the data of each client. The plots of the accuracy for the same setup, this time with 100 clients, can be seen in Figure 2.

What is also visible in all the runs, notably in the case with 100 clients, is that the precision of the model is a lot greater than the recall, in some extreme cases even by 30 percentage points. This indicates that the model model misses many of the positive cases. But, when it does classify a case as positive, the case is very likely to be a real positive case. This difference between precision and recall suggests that the model is conservative in its positive predictions. It is unlikely to classify a packet as malicious, but when it does, it must have been because of a good reason. This is not ideal, especially for the task of identifying malicious packets in network traffic for IoT devices, where missing any positive instances is highly undesirable.

6.1.2 Second setup. The experimental results for the second setup reveal interesting patterns in the accuracy progression of the federated learning model across different simulations. In all cases, the final accuracy of FedAvg did not drop below 70%, indicating a generally good performance of the model.

A μ value of 1 was used for FedProx for these tests, as it seemed to yield the best accuracy, but values of 0.001, 0.01 and 0.1 were all tried (50 simulations each), as per the recommendation of the paper introducing the FedProx algorithm [7].

In several simulations, the accuracy exhibited a steady increase, similar to the expected behaviour. The accuracy improved rapidly during the initial rounds, followed by a gradual stabilisation as the model continued to aggregate updates from multiple clients. This pattern suggests that the model effectively learned from the

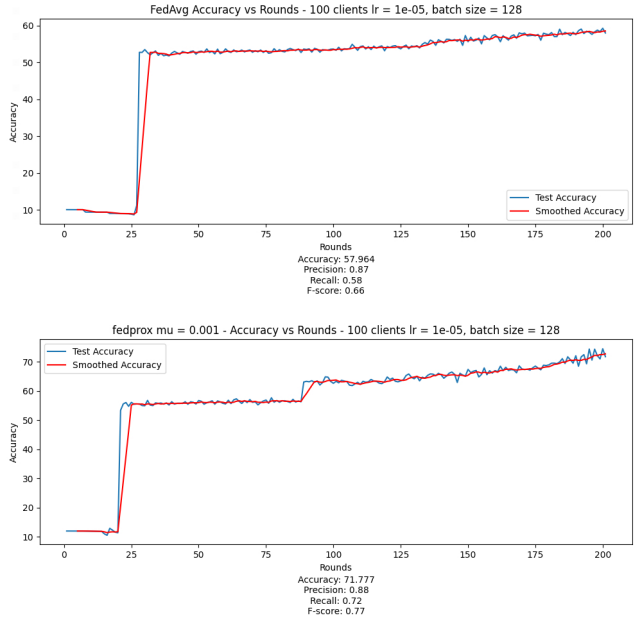

Figure 2: Accuracy over time for the first setup with 100 clients with FedAvg and FedProx, measured over 200 rounds of federation.

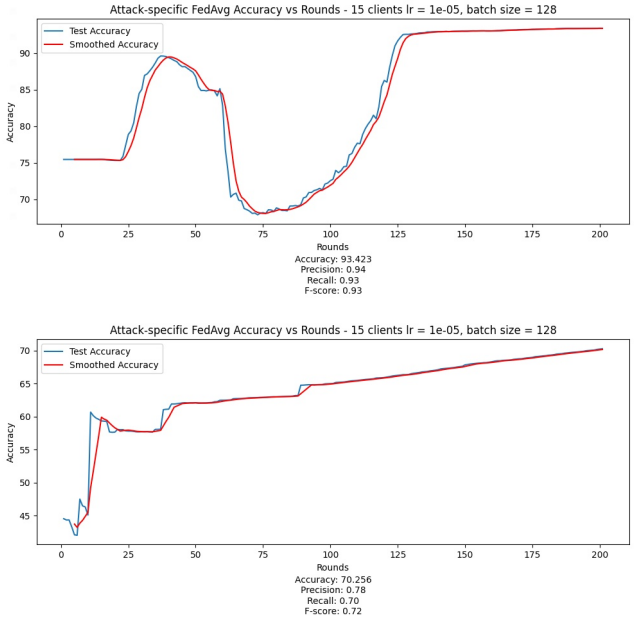

Figure 3: Plots of the highest and lowest accuracy runs achieved with FedAvg over 200 rounds

data, progressively improving its performance with each round of federation.

For some of the runs, the central model might receive parameters from a subset of clients that have performed exceptionally well on their local datasets. This can cause a temporary spike in accuracy. However, as more rounds progress and more client updates are aggregated, the central model parameters will start to reflect a broader range of client data distributions, which might include more challenging data, leading to a drop in accuracy. What can also happen is that some clients might have data that is easier for the model to learn from, resulting in a spike in accuracy when these clients' updates are incorporated. The spike could also indicate overfitting to certain clients' local data. As training progresses and more diverse data is incorporated, the model begins to generalise better, leading to a drop in accuracy before it stabilises and continues to grow again.

Both algorithms were allowed to run 50 times for 200 rounds in order to get a conclusive result. During the 50 runs of training with FedAvg, the model achieved varying levels of performance across accuracy, precision, recall, and F-score metrics. The average accuracy stood at 82.73%, with the peak reaching up to 93.42% and the lowest being 70.26%. Precision averaged at 0.848, with a maximum of 0.94 and a minimum of 0.77. Recall averaged 0.822, ranging from 0.93 at its highest, to 0.67 at its lowest. The F-score, which balances precision and recall, averaged 0.829, reached its maximum at 0.93 and its minimum at 0.7. The accuracy plots of the lowest and highest scoring runs of FedAvg can be seen in Figure 3.

With a μ value of 1 and 50 samples, FedProx showed a broader range of performance metrics. The average accuracy was 80.08%, with the highest accuracy reaching 93.21% and the lowest at 64.64%. The precision was similarly varied, with an average precision of 0.83, a maximum of 0.94, and a minimum of 0.74. The recall averaged 0.8, peaking at 0.93 and dropping to 0.65. The F-score averaged 0.8, with the highest at 0.93 and the lowest at 0.67. These results suggest that while FedProx can also achieve a good performance, its metrics are more variable and less consistent overall. The accuracy plots of the lowest and highest scoring runs of FedProx can be seen in Figure 4.

6.2 Interpretation and Analysis

With the first setup, the results were in line with what was expected, with the accuracy jumping rapidly at the start for both FedProx and FedAvg, and stabilising over the course of the 200 rounds. This likely happens because the data is more evenly distributed and each client can learn to generalise well at the start, since they probably have a very similar distribution each, along with a similar testing set. This demonstrates how both algorithms can work well if the clients are provided with iid data.

Also in the first setup, a higher number of clients generally led to a lower final accuracy, highlighting the importance of sufficient data per client for effective local training. This shows that the abundance of data of each client is a determining factor of the success of federated environments. The high precision but low recall observed in scenarios with more clients outlines a challenge in achieving a balanced model that minimises both false positives and especially false negatives, while maximising true positives. Improving the recall is vital for applications like malicious packet detection in IoT networks, where missing a malicious packet can

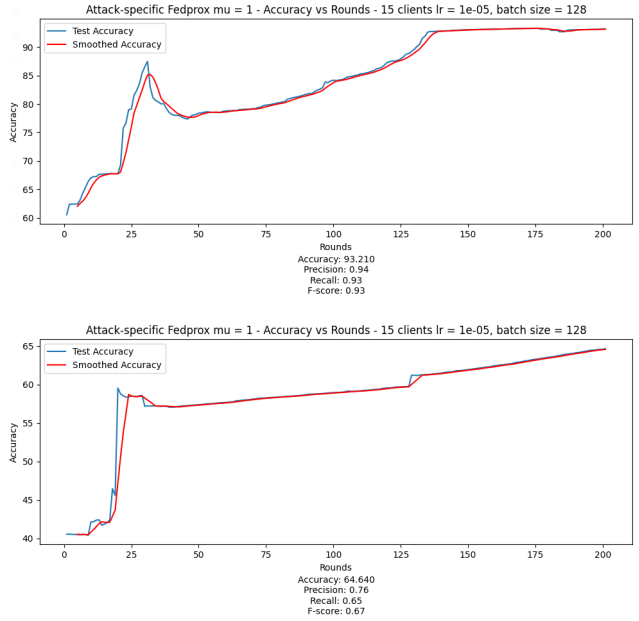

Figure 4: Plots of the highest and lowest accuracy runs achieved with FedProx over 200 rounds

have severe consequences, while classifying a benign packet as malicious is not nearly as much of a concern.

For the second setup there is a more detailed comparison below:

Comparison

- **Accuracy:**
  - **FedAvg:** Average accuracy of 82.38%, peak accuracy of 93.42%.
  - **FedProx:** ($\mu = 1$) Average accuracy of 80.08%, with a peak accuracy of 93.21%.
  - **Conclusion:** FedAvg provides higher average accuracy and more consistent performance compared to FedProx, while the peak accuracy of the algorithms is comparable.
- **Precision:**
  - **FedAvg:** Higher average precision at 0.848.
  - **FedProx:** ($\mu = 1$) Average precision at 0.83.
  - **Conclusion:** FedAvg has better average precision, indicating better minimisation of false positives.
- **Recall:**
  - **FedAvg:** Higher average recall at 0.822.
  - **FedProx:** ($\mu = 1$) Average recall at 0.8.
  - **Conclusion:** FedAvg demonstrates higher average recall, suggesting it is more effective at capturing true positives. This metric is very important when considering the task of anomaly detection, as it would be best to reduce the amount of false negatives as much as possible in this context.
- **F-score:**
  - **FedAvg:** Higher average F-score at 0.829.
  - **FedProx:** ($\mu = 1$) Average F-score at 0.8.
  - **Conclusion:** FedAvg shows a better overall balance between precision and recall.

While these results suggest that FedProx can achieve a similar performance to FedAvg, it is less consistent and exhibits greater variability. FedAvg consistently delivers slightly higher average metrics across accuracy, precision, recall, and F-score, showing it is a more reliable choice for achieving a good performance in this context. Further experimentation is needed to fully understand the behaviour of both algorithms under different conditions and datasets. Further experimentation with varying client distributions and data heterogeneity are needed to provide accurate insights into the optimal conditions for each algorithm.

Something to note is how the recall is considerably lower than the precision in the runs with poorer performance of both algorithms. This means that in addition to having a worse accuracy, the models also missed a lot of positive cases. Again, this behaviour is disadvantageous for the anomaly detection task.

There are several reasonable explanations as to why FedAvg seems to be doing better than FedProx listed below:

(1) **Simpler Aggregation Method:** FedAvg uses a simpler method of aggregating updates from different clients. This simplicity could have led to more a stable and faster convergence compared to FedProx, which uses an additional regularisation term that complicates the process.
(2) **Homogeneity of the Data:** If the data distribution across clients is relatively homogeneous, the advantages of FedProx's proximal term (designed to handle heterogeneous data) may not be as noticeable. In these cases, the simpler FedAvg method can perform equally well or even better due to its less complex update rules. Some statistical tests, such as the Kolmogorov-Smirnov test, could be performed, comparing the data of every client to the data of every other client to get a better idea of the distributions of the packets and their similarity. In this case, the data may have been more similar than it seemed.
(3) **Optimal Tuning of the Proximal Term:** The proximal term in FedProx should weaken the effects of data heterogeneity. However, this regularisation will sometimes hinder learning if the strength is not optimally tuned. If the proximal term is too strong or too weak, it may slow down the learning process, leading to lower accuracy compared to FedAvg. Perhaps an optimal value for this term is found somewhere between the values that were tried.

The exact reason of these performance figures is likely a combination of the explanations listed above. More experimenting with this setup is needed in order to come to a clear conclusion.

6.3 Conclusion

This study investigated the performance of two federated learning algorithms, FedAvg and FedProx, in the context of anomaly detection for IoT devices, using the IoT-23 dataset. The experiments compared these algorithms under two different data distribution setups: shuffled and evenly split flows, and unique attack types per client.

The first setup showed the importance of the initial conditions of each client and the amount of data they have, as well as the number of clients. Under favourable conditions, both algorithms have shown that they can perform this task relatively well. This, however, changed with the increase in the number of clients, as the central model became slower and slower to incorporate the updates from the clients, in addition to each client having less, and possibly more varied data.

The experiments from the specialised setup seemed to show that, although FedProx achieved a comparable accuracy, FedAvg was the slightly better choice for this task on the IoT-23 dataset, in terms of average accuracy, precision, f-score and recall, which is one of the most important traits of this type of system, as not flagging a true positive can have severe consequences in the real world. Part of the reason why this simpler algorithm proved more effective is that, although there are 12 types of attacks, most of them only contain a combination of 3 or 4 common types of packets, thus making even the attack-separated data more homogeneous than what was thought previously, favouring a simpler approach. The simplicity of FedAvg can produce a more stable and consistent performance across different scenarios as well, which is why the results appeared to be more consistent when using FedAvg.

The practical implications of these findings are significant when deploying anomaly detection systems in real-world networks. The first setup shows that a balanced and diverse training dataset will quickly lead to a good model, though the number of clients needs to be managed to ensure they all have a sufficient amount of data. The second setup's varied results shows the importance of understanding the specific context and type of traffic a node will encounter. Adjusting the training approach based on expected traffic patterns should improve the performance of the models, as under real-life conditions, the overwhelming majority of the traffic one random node will encounter will be completely benign, with the occasional presence of an anomalous packet.

Future work could explore additional federated learning algorithms and their applicability for the IoT anomaly detection task, not only on the IoT-23 dataset, but also on other, maybe larger, more encompassing datasets. Additionally, the experiment could have allowed the simulations to run for more rounds and more runs could have been conducted to further confirm the results are conclusive. An interesting modification would have allowed each node to run a few epochs for every federation round to see how the metrics change, although this would have serious consequences on the processing time of the simulations. Letting the algorithms run until convergence could have also been an option, but this would have taken a significant amount of time. In addition, extending the evaluation to include more complex models and real-world IoT deployment scenarios would provide further insights into the robustness and scalability of federated learning approaches in this domain.

REFERENCES

[1] Gustav A. Baumgart, Jaemin Shin, Ali Payani, Myungjin Lee, and Ramana Rao Kompella. 2024. Not All Federated Learning Algorithms Are Created Equal: A Performance Evaluation Study. arXiv:2403.17287 [cs.LG] https://arxiv.org/abs/2403.17287

[2] Dhruba Kumar Bhattacharyya and Jugal Kumar Kalita. 2013. *Network anomaly detection: A machine learning perspective*. Crc Press.

[3] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. 2020. IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0). https://doi.org/10.5281/zenodo.4743746

[4] Binxuan Hu, Yujia Gao, Liang Liu, and Huadong Ma. 2018. Federated Region-Learning: An Edge Computing Based Framework for Urban Environment Sensing. In *2018 IEEE Global Communications Conference (GLOBECOM)*. 1–7. https://doi.org/10.1109/GLOCOM.2018.8647649

[5] Junghye Lee, Jimeng Sun, Fei Wang, Shuang Wang, Chi-Hyuck Jun, and Xiaoqian Jiang. 2018. Privacy-Preserving Patient Similarity Learning in a Federated Environment: Development and Analysis. *JMIR Med Inform* 6, 2 (13 Apr 2018), e20. https://doi.org/10.2196/medinform.7744

[6] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. 2020. A review of applications in federated learning. *Computers Industrial Engineering* 149 (2020), 106854. https://doi.org/10.1016/j.cie.2020.106854

[7] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. arXiv:1812.06127 [cs.LG]

[8] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[9] Viraaji Mothukuri, Prachi Khare, Reza M. Parizi, Seyedamin Pouriyeh, Ali Dehghantanha, and Gautam Srivastava. 2022. Federated-Learning-Based Anomaly Detection for IoT Security Attacks. *IEEE Internet of Things Journal* 9, 4 (2022), 2545–2554. https://doi.org/10.1109/JIOT.2021.3077803

[10] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. 2021. Machine Learning for Anomaly Detection: A Systematic Review. *IEEE Access* 9 (2021), 78658–78700. https://doi.org/10.1109/ACCESS.2021.3083060

[11] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. 2018. A Performance Evaluation of Federated Learning Algorithms. In *Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning* (Rennes, France) *(DIDL '18)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3286490.3286559

[12] Mauricio Leonel Merchan Prado. 2024. Anomaly detection in IoT: Federated Learning approach on the IoT-23 Dataset - BSc Thesis.

[13] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. 2019. Federated Learning for Emoji Prediction in a Mobile Keyboard. *CoRR* abs/1906.04329 (2019). arXiv:1906.04329 http://arxiv.org/abs/1906.04329

[14] Raed Abdel Sater and A. Ben Hamza. 2021. A Federated Learning Approach to Anomaly Detection in Smart Buildings. *ACM Trans. Internet Things* 2, 4, Article 28 (aug 2021), 23 pages. https://doi.org/10.1145/3467981

[15] Taeshik Shon and Jongsub Moon. 2007. A hybrid machine learning approach to network anomaly detection. *Information Sciences* 177, 18 (2007), 3799–3821. https://doi.org/10.1016/j.ins.2007.03.025

[16] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. 2018. Human Activity Recognition Using Federated Learning. In *2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. 1103–1111. https://doi.org/10.1109/BDCloud.2018.00164

[17] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* 10, 2, Article 12 (jan 2019), 19 pages. https://doi.org/10.1145/3298981

[18] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. 2021. A survey on federated learning. *Knowledge-Based Systems* 216 (2021), 106775. https://doi.org/10.1016/j.knosys.2021.106775