Fall 12-18-2020

A FEDERATED DEEP AUTOENCODER FOR DETECTING IOT CYBER ATTACKS

Christopher M. Regan
Kennesaw State University

A FEDERATED DEEP AUTOENCODER FOR DETECTING IOT CYBER ATTACKS

A Thesis Presented to

The Faculty of the Department of Computer Science

by

Christopher M. Regan

In Partial Fulfillment

of Requirements for the Degree

Master of Science in Computer Science

Kennesaw State University

December 2020

A FEDERATED DEEP AUTOENCODER FOR DETECTING IOT CYBER ATTACKS

Approved:

Dr. Reza M. Parizi

Your Advisor's Name

Dr. Jose Garrido

Second Advisor's Name

Dr. Mohammad Aledhari

Third Advisor's Name

Dr. Coskun Cetinkaya

Department Chairperson's Name

Dr. Jeff Chastain

Dean's Name

In presenting this thesis as partial fulfillment of the requirements for an advanced degree from Kennesaw State University, I agree that the university library shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from or to publish, this thesis may be granted by the professor under whose direction it was written, or, in his absence, by the dean of the appropriate school when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from our publication of, this thesis which involves potential financial gain will not be allowed without written permission.

Christopher M. Regan

_____

Notice to Borrowers

Unpublished theses deposited in the Library of Kennesaw State University must be used only in accordance with the stipulations prescribed by the author in the preceding statement.

The author of this thesis is:

Christopher M. Regan

1100 South Marietta Pkwy,
Marietta GA 30060
USA

The director of this thesis is:

Dr. Reza M. Parizi,

Department of Software Engineering and Game Development
Kennesaw State University
Building J, Office – 382
1100 South Marietta Pkwy,
Marietta GA 30060
USA

Users of this thesis not regularly enrolled as students at Kennesaw State University are required to attest acceptance of the preceding stipulations by signing below. Libraries borrowing this thesis for the use of their patrons are required to see that each user records here the information requested.

Name of user	Address	Date	Type of use (examination only or copying)

A FEDERATED DEEP AUTOENCODER FOR DETECTING IOT CYBER ATTACKS

An Abstract of

A Thesis Presented to

The Faculty of the Department of Computer Science

by

Christopher M. Regan

Bachelor of Science in Computer Game Design and Development,
Kennesaw State University, 2018

In Partial Fulfillment

of Requirements for the Degree

Master of Science in Computer Science

Kennesaw State University

December 2020

Acknowledgments

This research was funded by US SunTrust Fellow in Cybersecurity/Information Security Research Funding Program, No. ST20-01.

Abstract

Internet of Things (IoT) devices are mass-produced and rapidly released to the public in a rough state. IoT devices are produced by various companies satisfying various goals, such as monitoring the environment, senor trigger cameras, on-demand electrical switches. These IoT devices are produced by companies to meet a market demand quickly, producing a rough software solution that customers or other enterprises willingly buy with the expectation they will have software updates after production. These IoT devices are often heterogeneous in nature, only to receive updates at infrequently intervals, and can remain out of sight on a home or office network for extended periods. Security and privacy are two of the many ongoing research and operational challenges in IoT systems. Potential threats to IoT devices, such as botnets and malware-based attacks, have always been difficult for traditional detection systems. However, deep learning-based solutions have been utilized in recent years, and many challenges have yet to be addressed. In this thesis, we propose a federated-based approach, this will employ a deep autoencoder to detect botnet attacks using on-device decentralized traffic data. This suggested federated learning solution will be able to address the privacy and security of data by ensuring that the device's data is not transferred or moved off the network edge. Instead, the machine learning computation will be brought to where living data is born (e.g. the edge layer); thus, providing the sought-after results of a traditionally centralized machine learning technique, with the added benefit of data security. We demonstrate that our proposed model has achieved up to 98% accuracy rate in anomaly detection while using features such as source IP, MAC IP, and destination IP and socket channel for training. The comparative performance analysis between our proposed approach and a traditionally centralized format demonstrates that our approach achieves a significant improvement in the accuracy rate of attack detection.

A FEDERATED DEEP AUTOENCODER FOR DETECTING IOT CYBER ATTACKS

A Thesis Presented to

The Faculty of the Department of Computer Science

by

Christopher M. Regan

In Partial Fulfillment

of Requirements for the Degree

Master of Science in Computer Science

Advisor: Dr. Reza M. Parizi

Kennesaw State University

December 2020

Table of Contents

List of Figures

Chapter 1. Introduction

The Internet of Things (IoT) is broadly recognized as the integration of heterogeneous devices that can operate and communicate autonomously to enable people, devices, and services to interconnected and the exchange of information or data, in a wide range of applications, such as healthcare and industrial environments [1,2]. All of these devices are potential vectors of attack from malicious actors. For example, botnet and malware-based attacks pose a severe threat to the security of the devices and the network hosting such IoT devices [3–5]. As a result, there is a high demand to dynamically detect and identify compromised devices on time for mitigating risk to a network and the consequential downtime of the IoT devices. These key features are crucial for current and future network security standards. As such, there is a need to investigate the solutions to counter these attacks.

1.1 The Problem

Intrusion Detection System (IDS) has been a popular approach to tackle network security issues for decades. It is widely utilized to monitor network traffic and identify suspicious activities. In general, IDSs are categorized into signature-based or anomaly-based defense mechanisms. Signature-based IDSs recognize intrusions and guard against previously learned rules/signatures of known attacks. Anomaly-based IDSs monitor network traffic and compare the traffic with previously learned patterns to spot malicious activities. In comparison, anomaly detection methods have shown to be more effective to recognize known and new attacks but need frequent updates to detect new attacks [6]. In recent years, deep learning-based solutions [7–10] have been largely advocated in the IoT security space for cyber-attacks detection due to their intelligent learning power of patterns of the network's traffic, resulting in more effective detection of anomalies when behavior changes. However, such solutions are often unseemly designed based on the assumptions that there is ample(big) data widely available to train the model and such data can be collected by

a central authority without considering the possible privacy concerns. Furthermore, with 5G-enabled networks rolling out, the amount of (decentralized) data generated by IoT devices are dramatically growing which is also not factored into the current solutions' design. Those taking these approaches regarding the data need to look at the big picture and ask themselves about the security and application of the models. For an enterprise network of heterogeneous IoT devices, is it possible to maintain data security as well as centralized accuracy levels with low network overhead? Additionally, can the IoT devices self-produce current learning models? Can a federated approach be implemented for detecting compromised IoT devices with a reduction in security risk on the edge layer?

1.2 Traditional Solutions

To answer the big picture questions around data application, the security of data should be addressed first as a lack of security is a constant concern. In traditional centralized machine learning techniques, the risk is higher as the data is moved away from the source or local network. Moving the data to be worked on off-premise, increases the possibility of attacks. In this thesis, we propose a federated IoT attack detection approach at the edge layer using on-device decentralized data. This software solution uses Federated Learning (FL) [11] at its core and brings the computation to the local network, such as an office environment, rather than taking the data to the computation offsite. As FL does not require transporting data off the network the devices are located on, is an improvement because it does allow the computational performance of the machine learning on local devices within the network. This way is preferred to address the concerns around data security. The IoT data will have in response improved in security, because as of a result of FL's design that minimizes sensitive data movement and provides quick-acting services that react to the FL results [11,12].

1.2.1 Combining Federated Learning with A Deep Autoencoder

In combination with implementing FL, we propose utilizing a deep autoencoder model [13] as the main detector of anomalies associated with IoT security attacks [14]. The proposed anomaly detection model differentiates benign patterns of behavior from malicious activities from decentralized on-device data at the edge layer. In this collaborative manner, our federated approach is capable of accommodating different machine learning techniques, that can be distributed to the network security gateway on the edge network, where the machine learning models are developed based on the datasets distributed across different devices. In this way, the models are transferred to and from the FL server, rather than the data itself. The security gateways can perform independently to ensure single or multiples of the same device types are secured by a self-adapting machine learning model. These security gateways locally generate models and send these models to an external service called the FL sever. While the individual models are aggregated to the service, the FL server will perform a process to aggregate the models to produce an improved model, then a new improved global model will be sent back to the individual security gateway units. This federation of single independent security gateway units occurs as individually made models are aggregated and redistributed in a system called federated machine learning. The application of FL is privacy-preserving in nature [15,16] which addresses the concerns around data security.

1.2.2 Security Implementation and Machine Learning

It is clear the way forward for this form of network security implementation, using machine learning as an overtime implementation, as a way to push the limits of cybersecurity, will be using FL on-premise. Traditional centralized machine learning problems are avoided when FL is utilized. The data security is self-evident as the data is not being sent or moved off the network. Reducing the transfer of data also minimizes overall network traffic. Previously, the network security gateway system was not practical due to limitations in technology. The network security gateway solution

employs lower-powered hardware or uses locally present hardware (i.e. network switches) to perform the complex FL operation. This is possible as the software has improved in efficiency, while the hardware side has also seen an increased computational power. The network security gateway becomes a cornerstone of the FL process, being the primary computational unit for machine learning processes, rather than on another machine learning process away from the network edge. Now the network security gateway has become feasible as a result of improved machine learning techniques, computational power, and superior energy efficiency of physical hardware.

The rest of this thesis is structured as follows. Section 2 gives the related works in the area of IoT security and FL. Section 3 presents the proposed approach and discusses the design and implementation details. Section 4 explains the research methodology including dataset and evaluation metrics. Section 5 gives the evaluation results and provides a discussion. Finally, Section 6 concludes the thesis.

Chapter 2. Related Works

The smart home industry and smart office are growing every day by integrating different IoT devices such as environment monitoring, mobility aids, personal health care, smart lighting, and others to facilitate and support various tasks. A plethora of IoT devices are being deployed, effectively increasing their footprint across the globe. As these IoT devices become more available and personal, the consequences of attacks become greater [17]. As they are deployed, these devices may lack sufficient features to defend against various attacks, might leak sensitive data, and could result in unauthorized network access [18].

2.1 Issues with the Heterogeneous IoT Environment

Given the presence of this heterogeneous IoT environment, it is challenging to use classical cybersecurity approaches. To overcome this challenge, FL is a promising solution as it enables machine learning techniques on IoT devices to detect fewer known attacks. Detecting compromised IoT devices was proposed by building effectively on device-type-specific communication profiles [15]. This non-human intervening system with no labeled data for detecting anomalous deviations in device communications, called DIoT, uses a federated approach for aggregating behavior profiles. The DIoT was able to report no false alarms when evaluated in real-world smart networks while on deployment settings. Similarly, the IoTFLA approach [16] is an architecture for security and privacy and shows the use of FL to maintain data security. This combination of FL with secure data aggregation would reflect the modern trends of privacy and protection on dynamic living systems and networks. Having FL be more than a continuous learning network has revealed the possibility that its operations would induce an objectively more secure network [14]. This is since the IoT device data never leaves the network that the IoT devices operate on and demonstrated by the team at Opened Minded [19]. Another

approach shows the use of a recurrent neural network with ensemble learning. This approach was proposed and demonstrated by integrating a set of long-short-term memory modules into an ensemble of detectors [6]. The evaluation was done with a Modbus dataset of network traffic, where they detected cyberattacks against IoT devices. This proposal [20] only uses 4 types of attacks for training and 10 types of attacks for testing. They achieved a 99% detection of true positive rate, 100% in true negative rate, and near-zero false alarms. Considering the benefits of FL, it is also important to think of the associated costs of network communication. There is a proposal [21] that indicates that higher accuracy and reduced communication cost can be achieved even with a separate server performing global aggregation. There are numerous pros with minor cons presented in the suggestions for use of FL in both evidenced and speculated opportunities. The tested suggestions for FL open doors to new implementations. The techniques employed across the cybersecurity field have exemplified a wide variety of novel methods to address detection, classification, localized anomalies, and the occurrence of these anomalies. An example of a successful technique includes an FL neural net using a Bayesian non-parametric framework [22]. In this inference approach, the Bayesian framework synthesized a global network without additional supervision completing testing on a popular image classification dataset. The capability of using FL has been demonstrated and continues to open new avenues.

2.2 Current Works in Anomaly Detection

Much of the workaround anomaly detection and classification was unfortunately only used for a testbed lab set up, with a results-oriented publication by Median, et al. [13]. To compensate for their results, a separate thesis [23] created an anomaly detection simulation to accompany the anomaly classification system, using traditional machine learning techniques. In one of the recreations described in the thesis, the result saw 99% accuracy using less than 10 extracted features. This example showed that the testing simulation implementation using the

same data set, was able to reach near identical results as the original paper stated when using the deep autoencoder, even when using the same data set [24]. As FL is an evolving area, it is imperative for anyone interested in real-world applications to stay up to date with the latest deep learning models and architectures. In this work, we propose a decentralized model using the FL technique. Herein we detail how the anomaly detection success found in the related fieldwork described above, was likewise reflected in deep learning models applied in detecting IoT-derived botnet attacks and identifying malicious data. Given a network with a wide variety of IoT devices, this proposal would still be applicable even without isolating features for the training of the deep autoencoder.

Chapter 3. Proposed Approach and Architecture

In this section, the proposed approach is discussed in detail from architecture to the implementation.

3.1 Architecture

Figure 1 shows the architecture for an end-to-end process for our approach. The proposed architecture is a decentralized FL-based approach, with a deep autoencoder anomaly detection engine for botnet attacks detection. In the figure, as the network traffic travels to the designated IoT device, which includes UDP/TCP protocol-based network traffic, the IoT security gateways operate and monitor the through-put traffic. These IoT security gateways contain the FL process, and the anomaly detection process (discussed more in Section 3.4). As the network traffic is monitored with port mirroring, additionally the flow of the network traffic going to the IoT device(s) is not interrupted. The network traffic is monitored in both directions, going through the IoT security gateway for entry and exiting traffic, as botnets could have made their way to the IoT device, appearing as normal traffic. However, botnets that have infected any IoT device(s) on the network would be expected to produce abnormal behavior, such as sending signals to unusual destinations (I.e. port scan attacks). IoT security gateways, once deployed, will communicate with their specified FL server, the FL server will host a selection of models for specified devices. The security gateway will be set to protect a particular type of IoT device, then communicate to the FL server to select the deep autoencoder model for the specified IoT device(s) being protected. This process, while living on the same network as the IoT device(s) for this proposal, will retrieve an FL model from the global model FL sever, which may also live on the same network or be off premise, as the model is only meaningful for machine learning processes. Local models will be

requested by the IoT security gateways, then the FL server will send to the network security gateway's virtual workers a specified global model for the device being protected. The IoT security gateway will later send back a trained model for model aggregation with other locally worked on models, this process improves the overall quality of the FL over time. These processes will continue indefinitely without a manual stop.

As far as the security gateway's hardware, we have not specified the hardware, as the proposed approach can be made to work with various hardware, devices, and environments. Given that a security gateway can process network traffic flowing through the gateway itself, we claim that by using port mirroring we maintain a short-lived instance of the network traffic called a snapshot. Using a snapshot, we can use this as our input to our virtual worker for the process. In addition to having this snapshot, these gateways can host more than just one virtual worker (explained in Section 3.4). Each security gateway can truly host an *n* number of virtual workers. As the workers host their specified model for the devices that they keep secure. The relation between the virtual works and the security gateways is a many-to-one relation. As there can be many virtual workers on one gateway, but a gateway does not have to host more than one virtual worker. However, this thesis will treat the relation between security gateways and virtual workers as one-to-one, for simplicity.

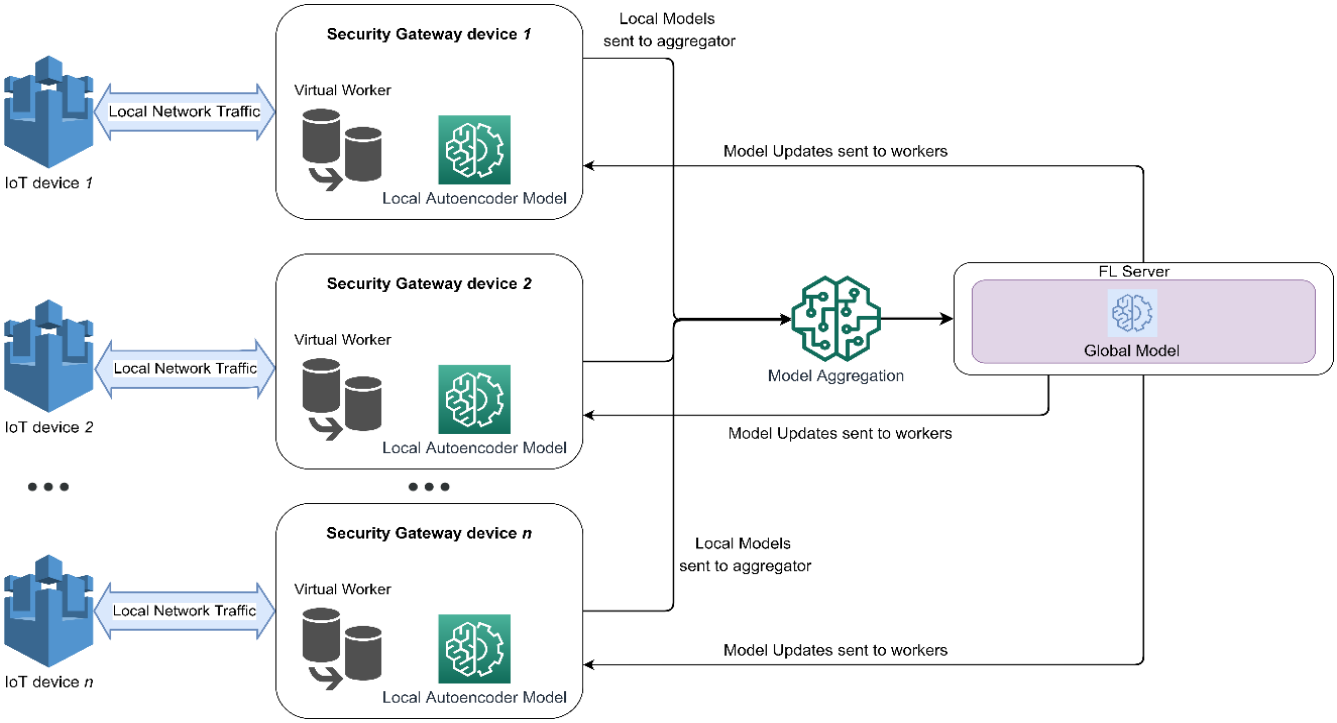


Figure 1: Overview of the proposed architecture

3.2 Deep Autoencoder Design

Implementation of the deep autoencoder has been inspired by the work of Meidan et al. [13] as demonstrated to work with high accuracy while in use for IoT intrusion detection systems in recent studies [25]. The deep autoencoder, as shown in Figure 2, works by receiving the input data representing the network traffic that is flowing to the individual IoT device. The data will then be encoded over the next 4 layers to an encoded state. Within each layer, the input features are multiplied with the layer's specified encoding value to compress the features further, addressing feature set size changes. Each layer from the encoding to the decoding is affected by the number of input features, the specified level of encoding is multiplied by the number of input features, and this is done for each layer, which impacts the performance of the deep autoencoder. After the

deepest specified encoding layer, acting as the encoded state, the reverse is the decoding layers. The decoding layers will reconstruct the data through linear layers from the lowest encoding layer. Following the final encoding layer, are 4 decoding layers, the decoding layers will recreate the dataset, this is done by expanding the encoded data incrementally over the next four layers, resulting in a recreation of the input, the same size as the original. As this process is sensitive to changes, when the output crosses the defined threshold set by the model, the resulting implementation can identify malicious behavior by the recreation alone. The steps are explained as follows:

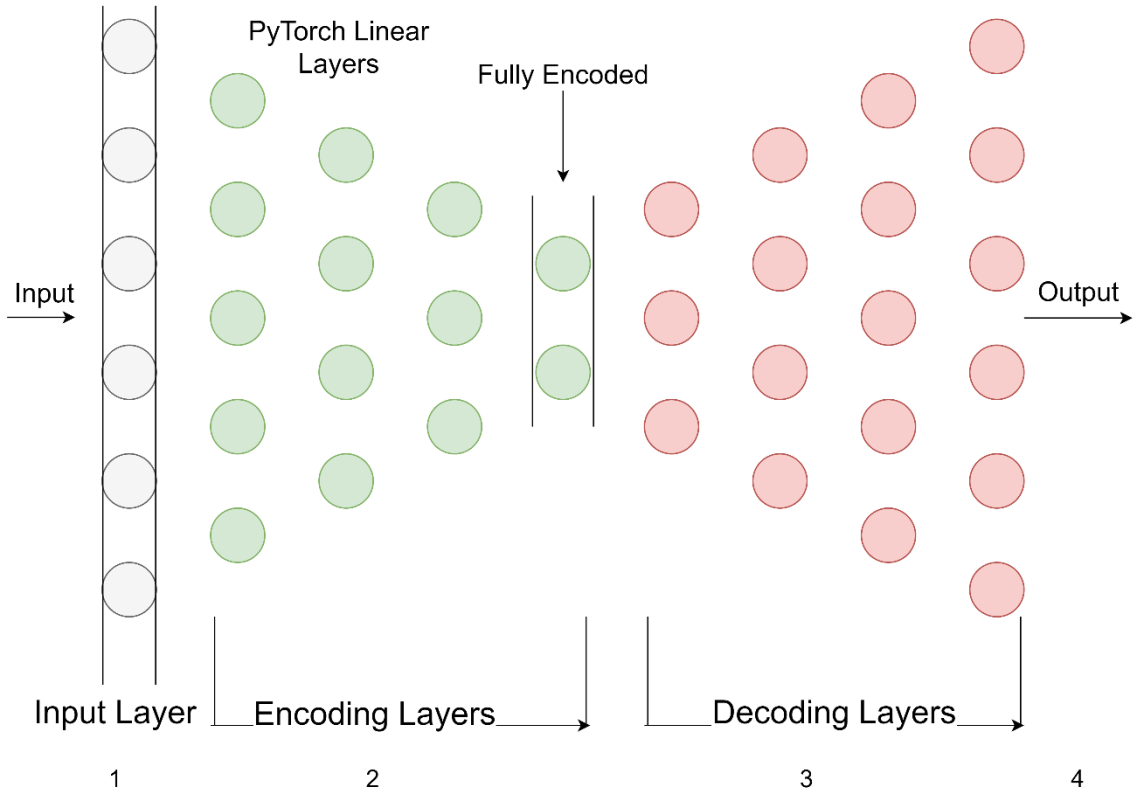


Figure 2: The autoencoder implementation

- In step one of the autoencoder, the input layer will be sent the input data to begin the ML process, using PyTorch linear layers for all steps of the deep autoencoder, each layer will constantly encode and decode accordingly. The first layer will take its input data then

encode the set of data to 75% of the original size, where the input of data is representing the source IP, destination IP, UDP/TCP socket details.

- Upon each sup sequential layer for the encoding will take in the previous layer's input. The next encoding layer will encode the previous layer's input to 50% of its size. The third layer, continuing with the encoding, again take the previous layer, encodes the input down to 33%of its size. The final encoding layer again will encode the input down to 25% of the previous step. This final encoding is the lowest compress.
- The decoding stage will reverse the effects of the encoding stage. Takes the input and then decodes the input to increase its size for the next layer. using the same values for decoding as encoding, and the same effect that input features have been used to aid the decompression. The opposing direction the decoder presents aids in making a decompressed data set that isn't 1-to-1 identical to the input and the expanding and zeroing out of some data points, aid in producing the threshold, done by bringing the values as close to the original as possible.
- The output layer will form a recreation of the encoding and decoding process. Where the behaviors of the network traffic get encoded, then decoded, followed by a comparison of the input to the output, resulting in a threshold being produced and used for testing.

3.3 Security Gateways

The proposed FL-based solution will be a software service, running on dedicated security gateways. These security gateways are placed on the edge of networks before the IoT device. As Botnets have been shown to use the UDP and TCP protocols, the security gateways will mainly monitor the UDP and TCP traffic as well as the connection that IoT devices use. The network traffic will pass through the gateway onto the IoT device, without interruption of the network traffic by

the security gateway. By using a network port mirroring technique to copy specific traffic data passing in and out of the IoT devices, an anomaly detection solution can work on the network traffic without interrupting the IoT device. If an anomaly is detected, then an alert system will notify a security service living on the network. This will result in the security service to deactivate the IoT device, blocking, or preventing the IoT device from communicating any further until corrected. To address the usage of security gateways in this architecture, we first assume that the native hardware and software of IoT devices are not able to run, nor will be able to operate our machine learning process. Thesis because the IoT embedded hardware is expected to not be computationally powerful enough to perform any operation other than its specified task. Besides, the software environment's ability to work with tools and data, in the way expected of off-the-shelf hardware and software.

3.4 Multi Virtual Workers Process

The virtual worker conceptually represents an edge device. In practice, it is a part of the toolset PySyft [19] provides for the simulation implementation of the FL process. The virtual worker will remain on the simulation machine for the duration of the experiment and does not leave the simulation space. The virtual worker is tasked with the computation of the specified model, the virtual worker will be sent a model to work on, then return the model when requested. The virtual worker process for training is shown in Figure 3.

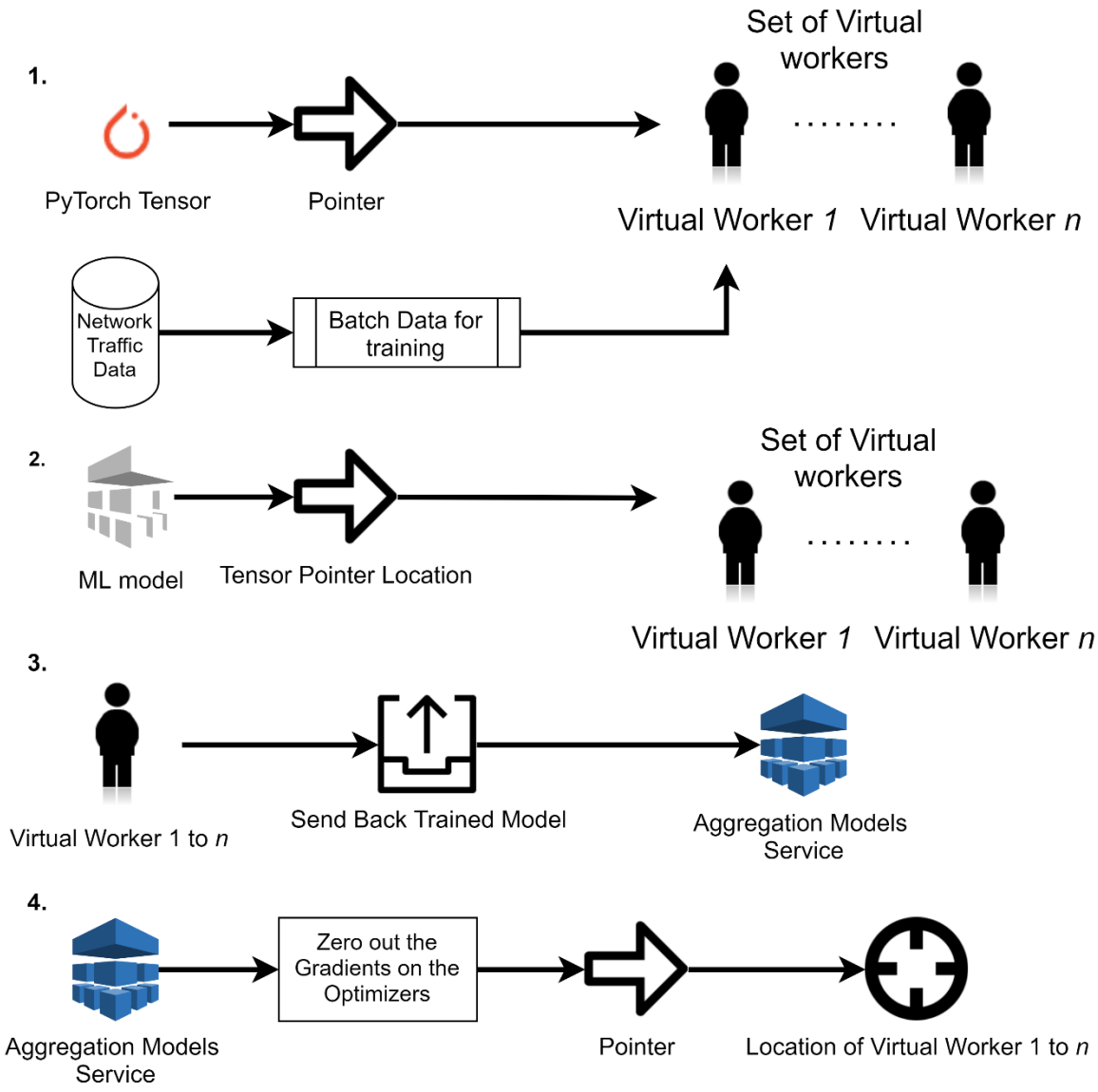


Figure 3: Virtual worker architecture in a federated learning environment

1. The virtual workers will first be assigned a PyTorch Tensor, this will store the virtual location and information about the tensor; after the data (see dataset in Section 4.1) is pre-processed, the virtual workers will be sent a batch of that data as seen in step 1.

2. A standard PyTorch Machine learning Model, the deep autoencoder, is created using a linear layer. The model will be sent to all virtual workers, this sending process is done via tensor pointer location, as shown in step 2.

3. As the virtual worker finishes the training step. That virtual worker will send their model to the aggregation service for processing. For the simulation, the model aggregator will wait for all virtual workers to send in their model before beginning the aggregation itself. Once all virtual worker's local models are aggregated, the process can continue.

4. Now that all of the trained models are in, the loss will be retrieved, the gradient optimizer will be zeroed out for the next step, as is standard practice on the optimizer, and prevents any residual effects that might perform on the bias and weights of the models. This will enable the next round of training to begin, as shown in step 4. These steps are the main actions taken by virtual workers in the training process.

3.5 Federated Learning Simulation and Implementation

The implementation uses the research-driven platform PyTorch, which enables the use of the FL-specific framework, PySyft [19]. PySyft is a Pythonic FL implementation for secure and private Deep Learning. PySyft separates private data from model training by employing FL for PyTorch machine learning. Using PyTorch to produce a traditional centralized implementation, a local simulation built on PySyft virtual workers can produce the model to be aggregated at the end of each epoch.

In this work, a model for the anomaly detection trains using the proposed deep autoencoder, using a similar method shown in [23, 13]. The autoencoder will be trained on statistical features from the benign traffic dataset (described in Section 4.1) from each device type. When malicious traffic is flowing through the autoencoder, anomaly detection is sensitive enough to indicate ab-

normal results. Then a testing phase uses a mix of benign and attack data, which is based on UDP/TCP traffic. With the assumption that the autoencoder will only train initially on benign traffic, the reconstruction of that data will be normal, but the failure to reconstruct that data will indicate an abnormality. The training of the anomaly detector is a key component, the model training and optimization is the main way to determine an anomaly. Though the possibility for the packet stream to be considered anomalous based on a single instance, which can enable very accurate detection of botnet attacks, still proves as a prime indicator of malicious traffic. The maintaining loop is built into several steps.

1. To begin, a profile request is made to the IoT security service and the initial global model. However, for this simulation, the training will be done only on benign data from our dataset to create a model for the testing phase. First, that required we create the virtual workers that are used for the training and testing. The optimizer will be sent to the virtual worker's location based on location id; this is set before any training begins as it is best practices to dose. With the virtual workers created we batch and send data from the dataset to the working hardware device, or in this case, the simulation of virtual workers.

2. The proposal is to use the data that is mirrored over the network specified port, this port mirroring is done by the security gateway itself as to not interrupt the signal going to the IoT device. In the simulation, for the training, we batched the benign data for each virtual worker evenly. Once the training data is batched and sent to each virtual worker in the training, an epoch of work begins, where each worker has its batch of data. Each worker once done, will signal the end of an epoch of training. The optimized gradients will be cleared with the zero-gradient function at the end of each epoch.

3. Next the local model is returned to the IoT global model aggregation system. The newly aggregated global model will be sent back to the security gateways of the specified device type

to continue training. During training, after each epoch, the local models are aggregated, and the output loss is collected from the new global model.

4. The application will continue to run indefinitely improving accuracy over time. In the simulation, after all, epochs are spent, the threshold will be produced for use by the testing process

Chapter 4. Research Methodology

This section describes the dataset used, its contents, and the details of the data processing for the learning tasks as well as evaluation metrics.

4.1 Dataset

The dataset used is an IoT botnet [24] that has been widely used by other research studies [23, 15, 13]. The dataset is comprised of two separate sets: one contains only benign traffic gathered when none of the devices on the testbed were infected, and the other contains malicious traffic when devices on the testbed were infected. The dataset consists of network traffic going in and out of 9 different devices, all devices had been infected with BASHLITE and 7 devices of the 9 devices had Mirai botnet infections. In the dataset, 115 statistical points were captured for each device for benign traffic. On the 7 devices, 5 types of attack data were captured for each device using Gafgyt attacks (e.g. TCP and UDP attacks), and 5 types of attack data were captured for Mirai attacks (e.g. Syn and Ack flooding).

1. Dataset preprocessing for training data: The dataset is imported using pandas, then is read in from CSV. Once the CSV for each benign traffic is imported, the fisher features are extracted, which features are extracted is determined by the input dimension. From the features, a feature set is extracted. After this, the dataset is split evenly into a training set, and a testing set for predicting benign traffic. After splitting the separate datasets are scaled using a scikit-learn standard scalar.

2. Training set usage: The dataset after being scaled will be made into PyTorch Tensor objects and passed to the training method. Here a neural net, composed of the autoencoder, the

dataset, complementing batch size, and the number of epochs is passed in as well through the hyperparameters.

3. Dataset preprocessing for testing data: The attack will be loaded using pandas from the CSV file. From this, a set of malicious data will be sampled using a feature set specified in the hyperparameters. Then this set will be appended to the testing set of benign data. The column marked malicious will be dropped, leaving the combined dataset to be a new mixed testing set.

4.2 Performance metrics

The following criteria are used in evaluating the utility of FL aided anomaly detection.

- True Positive (TP): When the malicious attack is detected.
- False Positive (FP): When benign data is detecting as malicious.
- True Negative (TN): When benign data is detected as a non-malicious activity.
- False Negative (FN): When malicious activating is not detected as a non-malicious activity.
- Threshold (TR): The discrimination between benign and malicious observations.
- Mean Squared Error (MSE)
- Standard Deviation (STD)

4.2.1 Training Metrics

Based on the described criteria above, the following metrics are introduced to quantify the efficiency of the given system. The optimization task is to minimize the loss function, which is the mean squared error between the original input and the reconstructed output.

- **Threshold** is defined as:

$$TR = MSE + STD$$

- **False positive** is defined as the sum of the predicted values greater than the threshold.

$$FP = sum(\hat{y} > TR)$$

4.2.2 Testing Metrics

For the testing phase, the model will be used to generate the accuracy, precision, recall, and confusion matrix on the testing set.

- **Accuracy** is defined as:

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

- **Precision** is the ratio of true detection of malicious attacks, over the sum of true malicious attacks and false detection of benign data.

$$PR = \frac{TP}{TP + FP}$$

- **Recall** is the ratio of true positives over the sum of true malicious detection and false detection of benign traffic:

$$Recall = \frac{TP}{TP + FN}$$

- **F-measure** is the weighted average of the precision and recall, defined as follows:

$$F - Measure = \frac{2 * TP}{2 * TP + FP + FN}$$

4.3 Testing and Training details

This section will discuss the process behind the training and testing of the deep autoencoder in the federated simulation.

4.3.1 Training goals

The training process is defined to produce a model in an FL simulation environment. The training section is not used in the comparison with previous works done, but its measurements of loss across each epoch aided in understanding the effectiveness of the model in an FL environment. The model training in the simulation is the same across all experiments, with only adjustments made for batch data delivery to multiple workers. As multiple workers would not necessarily be expected to work on identical data or all of the data directly. The training is only done on benign traffic data to measures the loss across epochs, the false positive rate and to produce a threshold for the testing. This simulation as mentioned in Section 3.5 is to show the process of training, the goals that to satisfy the key components are as follows:

1. **Model**: This model will be built using PyTorch linear layers with a hyperbolic tangent activation function. The data and model will be sent to virtual workers, simulating a model being sent to a network device, the data as to be sent on simulation as a part of the training as the virtual device, which operates as a software service on one machine and is not sent over a network.
2. **Loss**: The retrieval of the loss rate is a key identifier of the efficiency of the model is during training. As the loss decrease, the expectation of that accuracy will increase.

3. **False Positive Rate**: The FPR is produced by the difference between the mean squared error between the benign data and the predicted data, then the total number of items that are over the threshold will be the false positives without attack data.
4. **Threshold**: as mentioned in metrics, the threshold is used by the testing to indicate benign or malicious traffic.

4.3.2 Testing goals

Testing is to produce the accuracy, precision, recall, and F-Measure scores per each change of input dimension or feature count. The scores will be calculated using the functions provided by the sci-kit learn package. Using multi-label classification, this function aids in the return of the subset accuracy, precision, recall, and f1-score for benign and malicious traffic.

1. Test using the non-FL approach. The initial test is to run a traditional centralized approach (which has been used as the baseline for comparison in our experiments, see Section 5) that would perform as a non-FL version. With the loss, threshold, and false positives acting as the results from the training section, testing will use the threshold from the training to produce our accuracy, precision, recall, and F-Measure scores for the experiment run. As stated in Section 3.5, the centralized (i.e., non-FL) test will have four variances, where the only changed variable is the Input dimension.

2. Test using with FL (multi virtual workers). The second test is to extend the initial test by adding two virtual workers to the process. This will be where the results would indicate either a hindrance, maintaining, or improvement to the intended growth areas. Only these tests are run, as additional tests would be expecting a larger experiment out of the scope of this thesis.

Chapter 5. Evaluation and Results

In this section, we report the evaluation results of the proposed approach. To demonstrate the effectiveness of the FL approach, as an enterprise network environment solution for edge IoT device security, results will be compared against a baseline work. In this context, the baseline refers to a non-FL based approach that uses a centralized training and testing of a traditional PyTorch machine learning process. The non-FL solution does not incorporate any federated techniques or technologies. The experiments were processed on a Lambda Labs workstation hosting 128GB of RAM, 4 Nvidia RTX 2080 Ti's, operating with Ubuntu 18. All the data pre-processing was done within Python using public libraries. The deep learning platform was used as PyTorch. The FL was implemented using PySyft open-source library [19].

5.1 Training Results

Figure 4 provides the confusion matrices, representing the TP, TN, FN, FP rates across experiments for 15 and 115 input dimensions. These matrices are representative of the experiments of the lowest tested input features and highest tested input features for the non-FL baseline and the proposed FL approach. Within figure 4, the benign traffic is represented by 0 and malicious by 1. Consider Figure 4a, the false positives produced results as high as 43954 on the non-FL figure, juxtapose to the multi worker (i.e., the proposed FL approach) with the same parameters where the false positives produced results 43959 shown in Figure 4c. This reflects well on the model, maintaining performance even across multiple workers. This will be maintained across the higher input dimensions as well. Take Figure 4b the false positives produced was 31 for the non-FL, for comparison the false positives of the multi worker Figure4d resulting in a total of 36; again, keeping in the margin and show that the model can maintain performance across multiple workers.

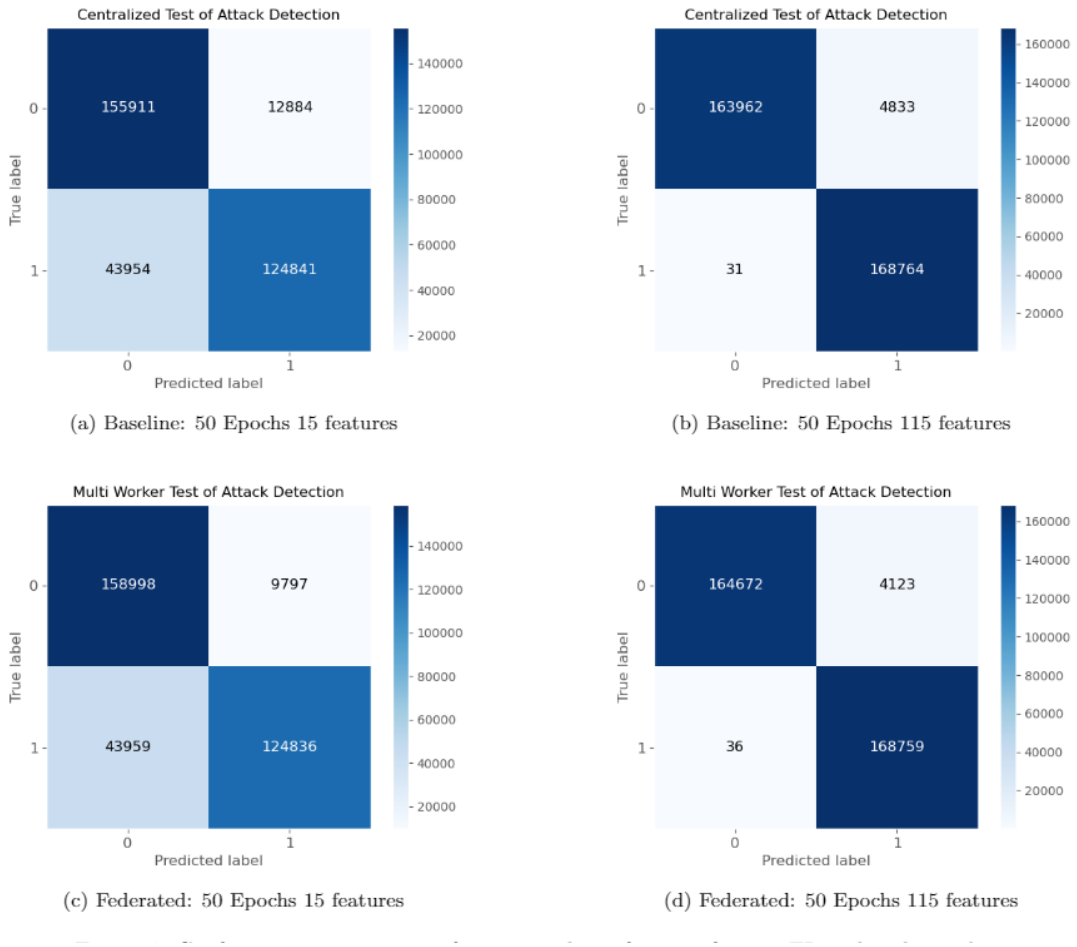
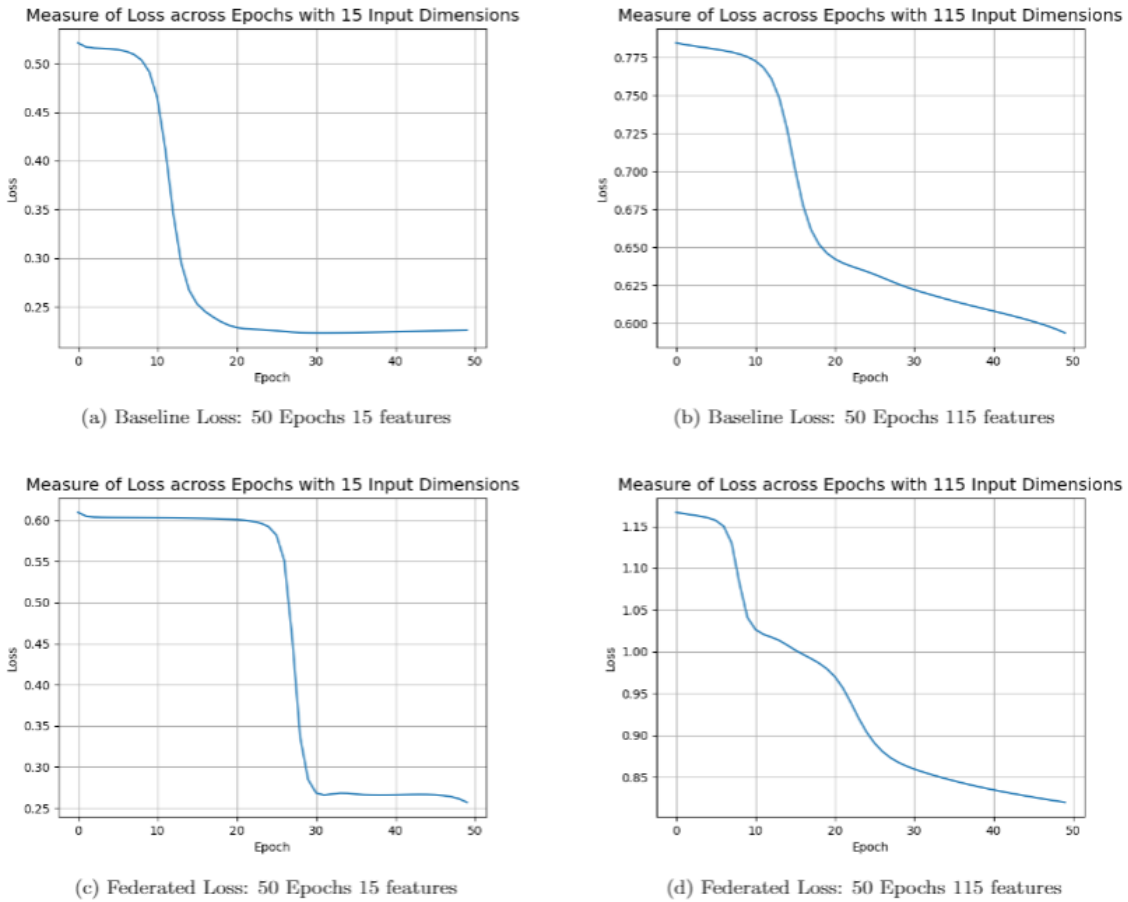


Figure 4: Confusion matrix using 15 features and 115 features for non-FL and multi worker

Figure 5 presents the loss of four separate experiments, the lowest possible experiments for then on-FL (baseline), and the proposed federated approach, along with the highest possible experiments. Using the same 50 epochs, 0.001 learn rate, batch size of 128, these experiments used 15and 115 input features, as lowest tested input features and highest possible to test input features, from left to right respectively. The non-FL loss as shown in Figure 5a begins as high as 0.5 in the early epochs. The loss will continue to decrease steadily till the 15 epochs, here the expected decrease begins to bottom out. The non-FL loss would continue to the downtrend, from 0.25 after25 epochs, presenting results that defined the non-FL model and the comparison begins to shape. This trend would not stay the same for the full feature set across all experiments.

Figure5b the loss would start with a score of 0.775 only to steadily decrease for the remaining experiments, allowing a bottom out to be just less than 0.6. In comparison the loss for the 15-feature experiment for FL, shown in Figure 5c, starts as high as 0.6 and remains at that peak until around 25 epochs where the loss rapidly decreases, only to bottom out at around 0.25 by the end of the experiment. In the multi worker experiment for 115 input dimension experiment, shown in Figure5d, the loss would be far above. The start of that experiment would see a score of around 1.15 for the loss. However, the stark decrease in the loss is more apparent than in other experiments. The end of the experiment score sat beneath 0.85. The run results would almost be double in comparison to the centralized equivalents. This may be contributed to the fact that there are at least 2 virtual workers that are known to affect performance in various ways.



(a) Baseline Loss: 50 Epochs 15 features

(b) Baseline Loss: 50 Epochs 115 features

(c) Federated Loss: 50 Epochs 15 features

(d) Federated Loss: 50 Epochs 115 features

Figure 5: Loss using 15 features and 115 features for non-FL and multi worker

5.2 Testing Results

The performance results in the form of a scatter plot are shown in Figures 6-9, where each dot represents an individual run. The figures represent the value of accuracy, precision, F-measure, and recall, experimented across the input dimensions or features, from 15 to 115. Each experiment used 50epochs, a learning rate of 0.001, with a batch size of 128. The orange represents the centralized non-FL (i.e., baseline) results, while the blue represents the proposed federated (i.e., multi virtual worker) results.

When considering the accuracy, the FL solution is meeting and exceeding expectations 6 over the non-FL baseline. The accuracy continues to trend upward for the FL results, being competitive and better in many cases when comparing to the non-FL baseline accuracy, even following the on-FL trend as features increase, rising to 99% after 55 features were used for training. The FL solution shows better-sustained accuracy with using the multi worker, as the accuracy would peak to 98% to 99%, unlike the non-FL solution where the higher accuracy would result to 97%. The constant maintaining of above 98% accuracy when using more than 50 input features, showing a promising FL solution for detection botnet attacks, Ack and syn flooding, UDP/TCP socket attacks, IP-spoofing, and source/destination attacks. The proposed approach being on-premise would shin in a production role as all these attacks and botnets would be detectable, given a starting model on a currently active network. The precision results are shown in Figure 7. The non-FL centralized precision score would peek at 0.98 as early as 30 epochs. This peak at a score of 0.98 would be unconventional from 40 input features to 90 features, going as low as 0.96 in some cases. When observing the proposed federated approach the precision saw improved stability after 30 features. The federated (multi worker) precision would only range from 0.97 to 0.98 for all remaining experiments. While evaluating the F-Measure of the non-FL approach and the federated approach (shown in Figure 8), both approaches would maintain a steady trend of

rising from 0.8 to 0.85 until around 50 epochs. However, the F-Measure would rise to around 0.98 once the past 55 epochs for both the FL and non-FL approaches, which would indicate ideal performance by both approaches. A similar situation would occur between the recall scores of the non-FL and the federated approaches, presented in Figure 9. The scores were below 0.75 up until the 50 to 55 epoch range where the score would jump up to 0.99 from 60 epochs and would remain at that high level while using the FL approach, for all experiments. As the testing was done with both benign and attack data. The testing phase using this data mix and maintaining high accuracy for most of the testing bodes well for future implementations in production. As we might expect IoT devices to have a high volume of mix traffic and a low volume of mix traffic. The FL and non-FL, ability to handle the large volume of packages, but to filter through the benign and attack data. Given these details, DDoS based attacks are to be manageable by the proposal. Even as the number of features increased the accuracy stayed constantly high, not showing the typical signs of overfitting. This being, the unusual behaviors of flooding, oddities in channel, socket, and the protocol forms of attacks, the approach would appear to be able to work in a testing environment so well, that the frequency of the attacks would only enhance the model further. Lastly, Figure 10 gives the average accuracy of the attack detection using the proposed approach versus the non-FL baseline. As it can be seen from the figure, the accuracy for the federated approach for all experiments came at 94.04% where the average accuracy of the non-FL baseline reached 93.73%.

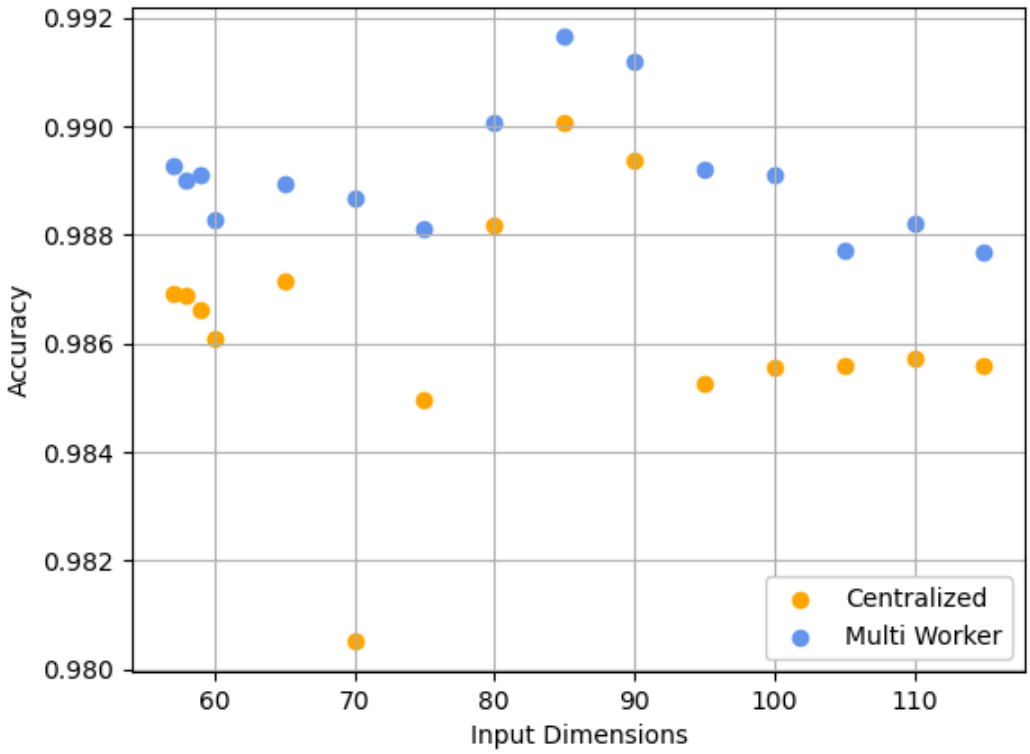


Figure 6: Accuracy per number of features

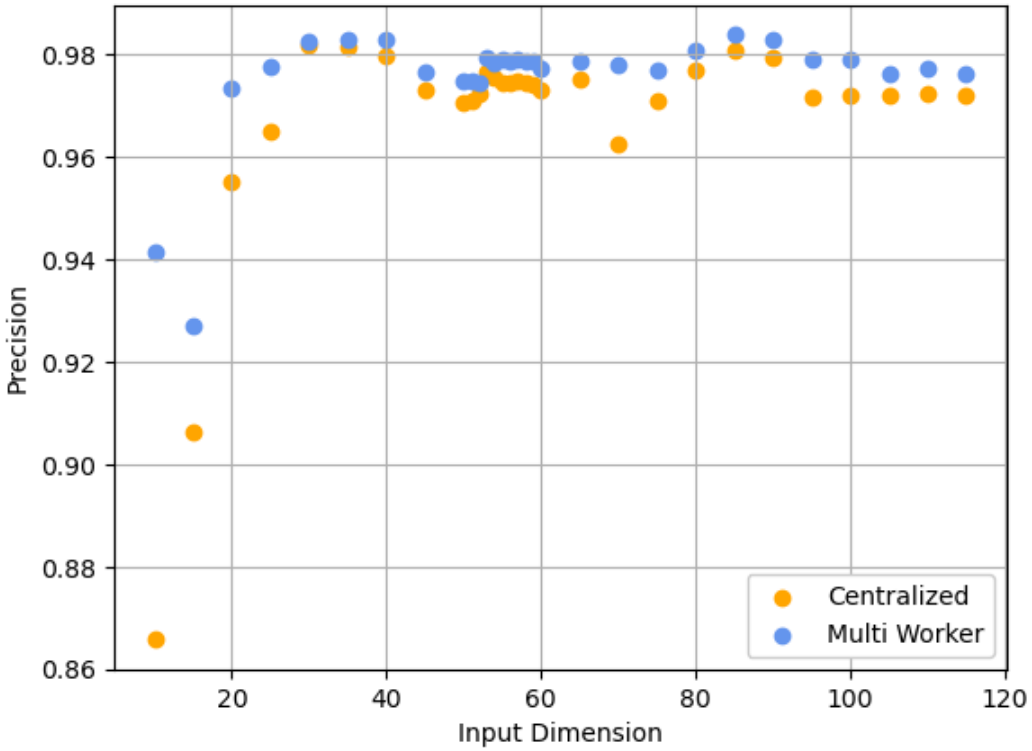


Figure 7: Precision per number of features

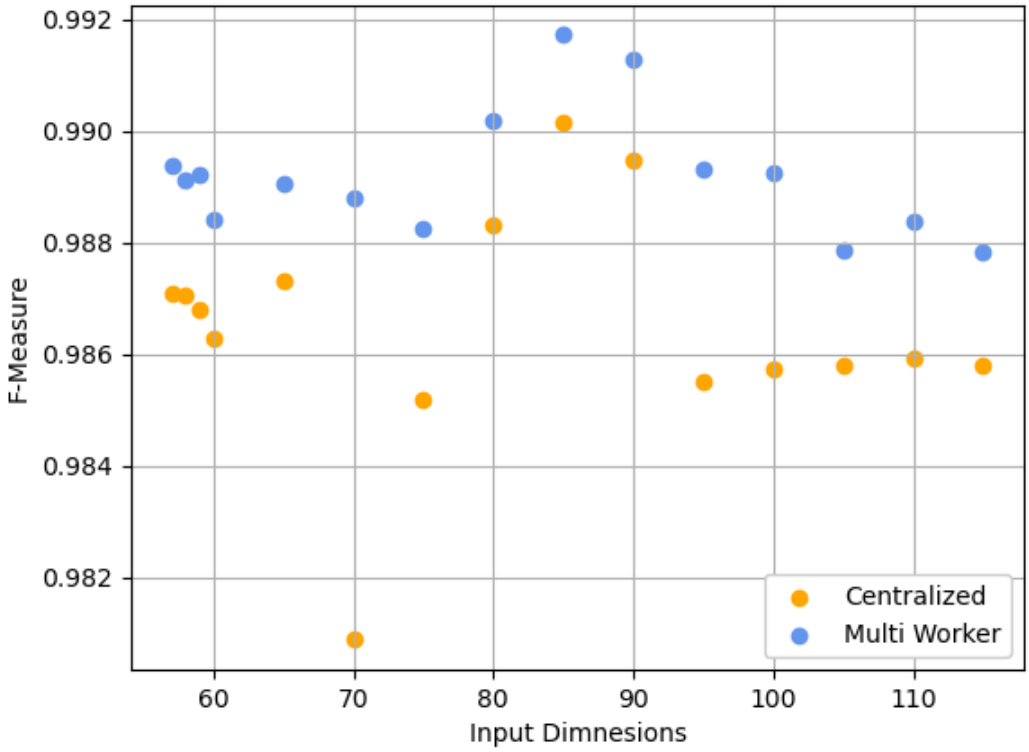


Figure 8: F-Measure per number of features

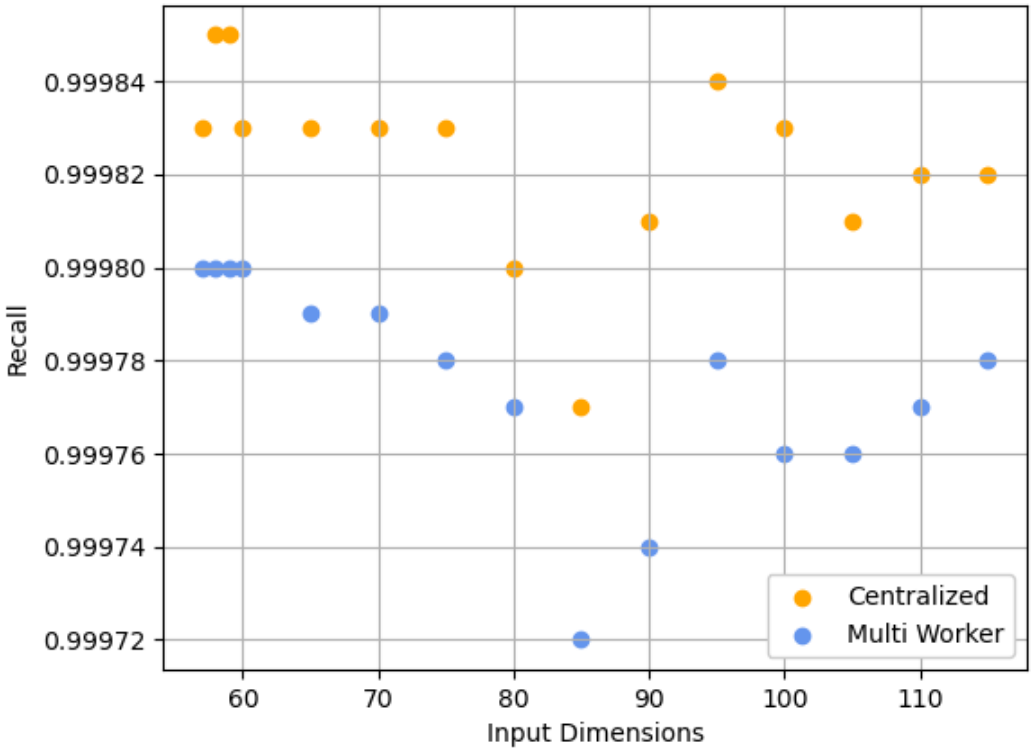


Figure 9: Recall per number of features

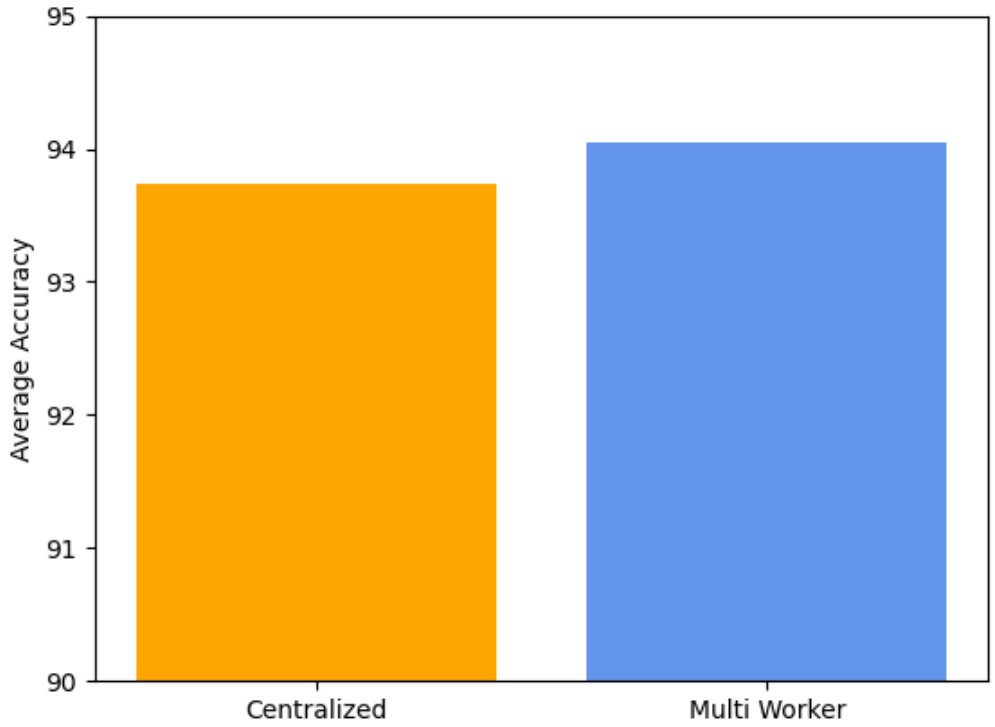


Figure 10: Average accuracy of proposed federated vs. non-federated baseline

Although the difference seems marginal in the simulation, it has to be said here practical application in a production environment, using our FL solution and given a pre-trained model to start with, would be an ideal place to grow this solution and evolve it into a full product pipeline. The true potential of the federated-based approach comes from decentralized data generated from many devices with continuous training to realize its full capability in detecting attacks based on the frequency of package movement in low detailed environments or low input features. The opposite is also true, using a highly detailed environment or high input feature solution would provide to accurately detect the attacks. All in all, even with the limitations subjected to simulating the FL environment the results attest that our FL solution could scale in its feature selection in practice.

Chapter 6. Conclusions

As industries and enterprises begin to expand the usage and diversify the IoT devices on their network, the risks of new attacks, botnets, and other forms of cyberattacks are present. Without hardened security, potential vulnerabilities yet to be discovered on new devices could leave networks exposed. The responsible and threatened parties are then forced to race to fix and patch devices; leaving new and old networks exposed for unknown periods. Herein, we presented a practical proof of concept as a proactive answer to identifying risks. FL has been evidenced as a solid solution for enterprise networks. This approach addresses sensitive data flowing to and from IoT devices, and complex machine learning model implementation. As well, it provides a self-updating attack detection system on edge layer networks and a foundation to monitor network security through on-premise gateways. The simulation shows that with enough features to train a model, peak levels of accuracy are maintained, and scores have found success within a minor margin. These findings demonstrate the approach's potential for securing IoT devices on an enterprise network, using any number of security gateways with any number of devices.

The next phase would involve deploying the proposed software solution to networks with living IoT devices and containerize the environment on security gateways. This will expand further upon our research, we would look deeper into selecting features that would have more impact on the learning process, and while in production be able to specifically select these higher impact features for monitoring. Additional research into feature selection would enable us to find what devices are most vulnerable to which set of features. Furthermore, we would be able to address this in the training of new models by adjusting the number of features to use, across the optimal input dimensions or features.

References

[1] S. Badotra, D. Nagpal, S. N. Panda, S. Tanwar, S. Bajaj, IoT-enabled healthcare network with sdn, in 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2020, pp. 38–42.

[2] I. Udrea, I. Gheorghe Viorel, A. Cartal Laurentiu, D. Duminica, S. Petrache, C. Aposto-lescu Tudor, F. Kraus Valeru, IoT solution for monitoring indoor climate parameters in open space offices., E3S Web of Conferences 180 (2020) 02012.

[3] T. Alladi, V. Chamola, B. Sikdar, K. R. Choo, Consumer IoT: Security vulnerability case studies and solutions, IEEE Consumer Electronics Magazine 9 (2) (2020) 17–25.

[4] S. Almutairi, S. Mahfoudh, S. Almutairi, J. S. Alowibdi, Hybrid botnet detection based on host and network analysis, Journal of Computer Networks and Communications 2020 (2020) 9024726.doi:10.1155/2020/9024726.

[5] H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, M. Aledhari, H. Karimipour, A survey on internet of things security: Requirements, challenges, and solutions, Internet of Things (2019)100129doi: https://doi.org/10.1016/j.iot.2019.100129.

[6] M. Saharkhizan, A. Azmoodeh, A. Dehghantanha, K. -K. R. Choo and R. M. Parizi, "An Ensemble of Deep Recurrent Neural Networks for Detecting IoT Cyber Attacks Using Network Traffic," in IEEE Internet of Things Journal, vol. 7, no. 9, pp. 8852-8859, Sept. 2020, doi: 10.1109/JIOT.2020.2996425.

[7] H. HaddadPajouh, R. Khayami, A. Dehghantanha, K.-K. R. Choo, R. M. Parizi, Ai4safe-iot: an ai-powered secure architecture for edge layer of internet of things, Neural Computing and Applications (2020) 1–15.

[8] H. HaddadPajouh, A. Azmoodeh, A. Dehghantanha, R. M. Parizi, Mvfcc: A multi-view fuzzy consensus clustering model for malware threat attribution, IEEE Access 8 (2020) 139188–139198.

[9] A. Al-Abassi, H. Karimipour, A. Dehghantanha, R. M. Parizi, An ensemble deep learning-based cyber-attack detection in industrial control system, IEEE Access 8 (2020) 83965–83973.

[10] A. Yazdinejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, M.-Y. Chen, Cryptocurrency malware hunting: A deep recurrent neural network approach, Applied Soft Computing 96 (2020) 106630.doi: https://doi.org/10.1016/j.asoc.2020.106630.17

[11] M. Aledhari, R. Razzak, R. M. Parizi, F. Saeed, Federated learning: A survey on enabling technologies, protocols, and applications, IEEE Access 8 (2020) 140699–140725.

[12] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Generation Computer Systems 115 (2021)619 – 640.

[13] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baiot—network-based detection of IoT botnet attacks using deep autoencoders, IEEE Pervasive Computing 17 (3) (2018) 12–22.doi:10.1109/MPRV.2018.03367731.

[14] K. Vandikas, S. Ickin, G. Dixit, M. Buisman, J. Akeson, Privacy-aware machine learning, Ericsson Technology Review 09 (2019) 02–10.

[15] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A. Sadeghi, D ̈Iot: A federated self-learning anomaly detection system for IoT, in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 756–767.doi:10.1109/ICDCS.2019.00080.

[16] U. M. A ̈ivodji, S. Gambs, A. Martin, Iotfla: A secured and privacy-preserving smart home architecture implementing federated learning, in: 2019 IEEE Security and Privacy Workshops (SPW), 2019, pp. 175–180.doi:10.1109/SPW.2019.00041.

[17] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, A. Zanella, Iot: Internet of threats? a survey of practical security vulnerabilities in real IoT devices, IEEE Internet of Things Journal6 (5) (2019) 8182–8201.

[18] H. U. Rahman, M. A. Habib, S. Sarwar, N. Mahmood, M. Ahmad, H. Ahmad, Fundamental issues of future internet of things, in: 2020 International Conference on Engineering and Emerging Technologies (ICEET), 2020, pp. 1–6.

[19] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, J. Passerat-Palmbach, A generic framework for privacy preserving deep learning, CoRR abs/1811.04017 (2018). arXiv:1811. 04017.URLhttp://arxiv.org/abs/1811.04017

[20] Hadeel Alazzam, Abdulsalam Alsmady, and Amaal Al Shorman. 2019. Supervised detection of IoT botnet attacks. In Proceedings of the Second International Conference on Data Science, E-Learning and Information Systems (DATA '19). Association for Computing Machinery, New York, NY, USA, Article 42, 1–6. DOI: https://doi.org/10.1145/3368691.3368733

[21] X. Yao, T. Huang, C. Wu, R. Zhang, L. Sun, Towards faster and better federated learning: A feature fusion approach, in: 2019 IEEE International Conference on Image Processing (ICIP),2019, pp. 175–179.

[22] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, T. N. Hoang, Y. Khazaeni, Bayesian nonparametric federated learning of neural networks (2019). arXiv:1905.12022.

[23] S. Tsimbalist, Detecting, classifying and explaining IoT botnet attacks using deep learning

methods based on network data, in: Bachelor's thesis, 2019, p. 48.18

[24] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, detection of IoT botnet attacks n-baiot data set (2018). URLhttps://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

[25] Y. Mirsky, T. Doitshman, Y. Elovici, A. Shabtai, Kitsune: An ensemble of autoencoders for online network intrusion detection (2018). arXiv:1802.09089.