Anomaly Detection using Advanced Machine / Deep Learning

Final Report

THE UNIVERSITY OF SYDNEY

Data Science Capstone Project

DATA5703

Project Owner: Dr. Basem Suleiman

Group Members

1. Wenbo Yan (480410991)
2. Ling Nga Meric Tong (500272956)
3. Xia Wei (490519372)
4. Ruijue Zou (500709979)
5. Sophie Zou (500276312)

CONTRIBUTION STATEMENT

Our group, taking project CS40-1, with group members Wenbo Yan, Ling Nga Meric Tong, Xia Wei, Ruijue Zou and Sophie Zou, would like to state the contributions each group member has made for this project during semester 1 2021:

- Wenbo Yan:
  - Research related sutdies about Federated Learning
  - Coding: To implement Federated Learning based on Pysyft(Early attempts); Implementation of other local model such as One-Class SVM; Run different FL method combination to get results; Run different training devices combination to get results.
  - Proposal Report- Abstract; Research Question
  - Progress Report- Progress
  - Final Report Section - Research Question;Results(FL vs Non-FL);Discussion
- Ling Nga Meric Tong:
  - Coding:
    Prepare datasets for Device #5, #8, #9;
    Worked on EDA for partial devices;
    Worked with Jane on building DAE for partial devices;
    Tune different hyper-parameter for local model on device #9;
    Run different devices combinations of training model for experiments comparison;
    Initial implementation of FedAvgM aggregation
  - Research and summarize related studies of DAE and FL models at the starting stage
  - Proposal Report: Introduction section, Methodology-GAN section
  - Progress Report: Obstacles section
  - Final Report: Introduction section, Methodology - DAE model architecture, FedAvg with momentum, client selection and retraining process
  - Client meetings: Took minutes in week 6; Present the architectures and training process of models to clients
- Xia Wei:
  - Research related studies about anomaly detection and Federated Learning

  - Research about multi-classification
  - coding:Work on the PCA distribution of cameral devices; Work on the on initial implementation of FedAvg with Wenbo; Work on initial implementation of FedAvgM aggregation with Meric; Run different FL method combination to get results; Run different training devices combination to get results.
  - Proposal Report: Methodology- CNN/Multi-class; Implications
  - Progress Report:Deviation to Timeline;Part of milestone table
  - Final Report:Abstract; Results

- Ruijue Zou:
  - Research about anomaly detection and federated learning
  - Set up GitHub Repositories and Google Drive shared folder for the team
  - Proposal Report: Methodologies(Deep Auto-encoder Method, Data Collection and Data Analysis), Milestone Gantt Chart
  - Progress Report: Milestone table update
  - Final Report: Methodology(Data Collection, Data Analysis, Data Pre-processing and Federated Model with Deep Auto-encoder)
  - Coding:
    * Datasets initial exploration (data distribution and plot related charts)
    * Prepare datasets for Device #1, #2, #3, #4 and #7
    * Build Deep Auto-encoder model for Device #1 (use Keras)
    * Build Deep Auto-encoder model for local anomaly detection (use Pytorch)
      · pre-processing function
      · threshold calculation function
      · evaluation functions(F1-score, TPR, FPR, confusion matrix)
    * Build Federated Deep Auto-encoder model for anomaly detection.
      · pre-processing function
      · random selection function
      · retraining mechanism
      · FedAvg and FedAvgM functions
      · evaluation functions(F1-score, TPR, FPR, confusion matrix)
    * Experiments
      · different combination of devices

        - different combination of techniques
        - organise experiment records

- Sophie Zou:
    - Literature Review: Summarizing and documenting all the past papers
    - Coding:
        * Prepare data for Device 6;
        * Tried out Federated Learning with PySyft;
        * Implementation of other local models such as the Neural Network Model with Federated Learning;
        * Run different combinations of training devices datasets to get results
    - Proposal Report: Related Literature
    - Progress Report: Achievements
    - Final Report: Related Literature, Methodologies(FedAvg, Evaluation Methods and Others), Limitations and Future Work.

All group members agreed on the contributions listed on this statement by each group member.

Signatures:

ABSTRACT

With the increase in mobile devices and the development of IoT in recent years, more attention has been applied to the issues of network traffic attacks. In the traditional anomaly detection method, all the data need to be collected into an agency to train the model, which is likely to lead to data leakage. Federal learning is a good solution to this problem.The focus of this study is to improve the security and generalizability of traditional anomaly detection model by Federated Learning and then solve the problems related to data leakage, Non-IID data, model training efficiency and performance. Basic techniques in this project ranging from Deep Auto-encoder model to different Federated Learning methods were implemented. Due to the different characteristic of each client, different equipment can affect the overall model performance, therefore the FL learning is important which could improve the whole efficiency. In addition, FL learning could protect the privacy of clients, clients only need to share the parameters of the training model and do not need to share their actual data. In order to deal with the issue caused by Non-IID data property, the retraining methods is used to product more stable federated anomaly detection model, this study also implements different model methods and training devices, performs different experimental combinations. According to all experiments combinations, it is found that the FedAvgM has a better performance compared to other FL learning aggregation,which proves that FedAvgM is much stability on non-IID dataset. For different method combination, it is found that the MNP(Federated Deep Auto-encoder model with FedAvgM with No Retraining and Partial Selection)has the best performance. Moreover, compared to the Non-FL model, FL model has a better performance,including the higher average TPR, the higher F1 score, the efficient running time and the excellent security.

Index Terms: Anomaly detection, Deep Auto-encoder, Federated Learning, Network Traffic Intrusion,IoT.

TABLE OF CONTENTS

1 Introduction

Due to the growing usage of Internet of Things (IoT) devices, intrusion events in network traffic have also risen dramatically. The network traffic attacks not only become threats in cyber-security and fraud detection but also occur in every business company that contains essential information. Nowadays, the need of detecting abnormal events or behaviours has become the main challenge and focus to prevent and protect the risks against anomalous activities. To detect anomalous activities, the most common security solution is using a network intrusion detection system (NIDS) to monitor all the activities [1]. In general, a NIDS is a tool or software placed at the network gateways, proposed to detect if the traffic is most likely an attack or not, to protect the information systems for servers. There are different approaches combined with NIDS, such as anomaly-based systems and signature-based systems.

Signature-based and anomaly-based have one main different conceptual operation that the first is aimed to seek the specific, known attack, however, it is not suitable to detect unknown, unfamiliar malicious activities [2]. For instances, using the multi-classification model to apply on labelled traffic data, which is implying that an understanding of the attack is already applied. It will not able to recognise any new anomalous event easily leading to data leakage. On the other hand, the latter system has a great capacity for detecting any unfamiliar attacks with higher performance in intrusion detection analysis [2]. This paper focuses on anomaly-based detection approaches. By classifying the system acting as either normal or anomalous, anomaly detection is based on finding an unusual point or pattern in the given dataset.

In recent decades, many advanced anomaly-based approaches have been offered to enhance the efficiency of anomaly detection. Principally, all different anomaly-based approaches contain the following three stages, which are parameterisation, training stage, and detection stage. From establishing the observed instances, characterising the baseline behaviour and training the built model to analyse the testing values respectively, those stages are providing great effects [3]. For instance, the well-known algorithms are the Deep Auto-encoder and KitNET algorithm. Both algorithms extract important features in well-structured architecture as described in Meidan et al [4] and Mirsky et al.[1] where the auto-encoder techniques show their merits in nonlinear and complex problems. Both provide advantages in classifying the given observations as an anomaly by how they set up the auto-encoder to train the model.

In this study, the main aim is to design and implement the deep learning models for detecting anomalous events in network traffic through the IoT devices, such as webcam, doorbell, baby monitor, etc. By using provided datasets of interest, our models are trained to evaluate the results in the confusion matrix and F1-score for detecting

anomalies activities. We are motivated to detect the anomalous data with the existing implementation with a decentralised approach, where detection of the anomalous activities from the benign instances combines the deep auto-encoder model with Federated Learning (FL). In general, federated learning is a type of privacy-preserving method which aims to train a deep learning model on multiple devices (which also are called clients) to process personal data, without explicitly exchanging or storing data samples. In order to experiment with the best approach for anomaly detection, the federated deep auto-encoder model is trained with two different aggregations which are federated averaging and federated averaging with momentum. Moreover, our out-performed proposed approach improves both the efficiency and robustness of IoT devices anomaly detection compared with the non-FL model.

2 Related Literature

Many studies from past literature have attempted to solve the anomaly detection task. Anomaly Detection has been an important issue which raised lots of attention for experts in the application domain, such as computer intrusion attacks and credit cards fraudulent activities. To detect such anomaly behaviour, action is eagerly required.

2.1 Literature Review

2.1.1 Traditional ways of anomaly detection:

A survey conducted by [5] in 2015 provided several techniques for network anomaly detection, includes statistical anomaly detection, clustering-based anomaly detection, classification-based anomaly detection and information theory. This literature review will first introduce the common methods based on the classification.

The main idea of classification-based anomaly detection is to prepare a knowledge base that contains previous benign records and attacks formerly. Once a similar attack is launched to the system, the attack will be identified immediately. One downside of the classification-based approach is that anomaly detection attack is only effective if the attack has been classified by the model before. This may result in the classification model treating the new benign data that were not introduced to the knowledge base as false alarms. Extensive training and time is required to create an all-encompassing knowledge base to solve the problem.

In 2002, an article written by [6] applied an unsupervised approach of One-Class Support Vector Machine (SVM) to explore the feature space of data. The model can detect abnormal incident by measuring the feature space to see whether the data is benign. Nevertheless, the results show that not all attacks can be detected accurately. This is because the abnormal data could lays in the same feature space as benign data. One downside of unsupervised learning is the pure benign dataset can not be guaranteed. There is a supervised method similar to One-Class SVM proposed by [7] in 2003, which improves the detection performance.

Bayesian Network is another technique based on classification. In 2004, [8] presented an approach that applied Bayesian Network to detect anomalous data while solving the high false alarms problem that mentions above.

One widespread solution for Network Intrusion Detection Systems (NIDS) is Neural Networks (NN). Since it could learn non-linear complicated patterns and events. There is a Replicator Neural Network (RNN) proposed by [9]. They use RNN for anomaly

detection. However, there are some disadvantages of NN, such as the big storage space and long time required, expensive, high-velocity traffic handling complexity, supervised learning and offline processing.

Although classification algorithms can handle datasets with a fixed number of supervised samples, detecting anomalous events in IoT devices by using classification models might not perform powerfully because of the discontinuous and unbalanced datasets. There are some literature proposed approaches to handle anomaly detection for devices in IoT environment.

2.1.2 Anomaly Detection using Deep Auto-encoders:

Deep auto-encoder was first implemented by [4] to detect IoT Botnet Attacks on real traffic data. The N-BaIoT dataset would also be used in this project. The deep auto-encoder model is trained on benign data only. When a significant reconstruction loss is detected, then the given samples can be classified as the anomaly. In terms of anomaly detection, the false alarm rate of deep auto-encoder is remarkably lower than other commonly used algorithms – Isolation Forest, PCA and SVM [10]. Since the devices in the IoT environment have various functions, the researcher deploys separate auto-encoder models for each device. Hence, the advantages of this approach to detect IoT attacks are able to detect unseen attacks, efficiency to perform semi-online and heterogeneity tolerance.

Additionally, Kitsune [1] proposed a novel ANN-based NIDS approach that applies a group of small deep auto-encoder models instead of one deep auto-encoder model to process unsupervised, efficiently and online. The literature states that it is more efficient to train multiple layers than to train a single but huge transformation with dimensional reduction. The dataset they used is the N-BaIoT dataset. The model raises the processing rate by five times and performs better than other models such as isolation forest. In this study, we will prioritise apply multiple deep auto-encoder models for devices to detect anomalous data due to its outstanding performance when the problem is complicated and non-linear.

Similarly, [11] uses bagging deep auto-encoder with the dynamic threshold for semi-supervised anomaly detection. The dynamic threshold is found by setting a confidence level randomly to see which one brings up the highest accuracy during validation. Comparing to other methods of only using one deep auto-encoder, this method of combining multiple deep auto-encoders enhances the model's robustness. They extract samples from the original training set and obtain multiple sub-training sets for learning by multiple deep auto-encoders. These learners' predictions are combined to integrate the final prediction.

2.1.3 Anomaly Detection with Federated Learning:

Due to information security, the problem of training on decentralised data from IoT devices has become an important research direction. For the first time, academics proposed the concept of Federated Learning in 2016 [12]. The main purpose of federated learning is to prevent leakage of private information. Federated Learning allows each device to perform training on locally and send the results to the central server to ensure privacy. The paper introduces the Federated Averaging algorithm – a fraction of clients is selected in each round, the average gradient on each device's local data is calculated, the local update is iterated multiple times before the average, and the weighted average of model results for all these clients is then sent to the server. This algorithm is computationally efficient but requires immense number of rounds of training to generate well results. From their results, the FedAvg model reaches an accuracy of 99.44% after 300 rounds of training. In addition, FedAvg is effective at optimising the training loss, thus is robust to unbalanced and non-IID data distributions.

A recent study of federated learning on anomaly detection has been carried out by [13]. This paper used federated learning on a LSTM model to learn time series IoT data. The federated learning approach is efficient for demonstrations on simulated datasets such that real data from a general electric current smart building can be distribute evenly. This paper also uses the federated averaging algorithm to train their model and repeated iteratively until the model convergence or the maximum number of training rounds is reached. The training process involved inputting data from multiple heterogeneous data sources across several sensors to address the common heterogeneous challenges of IoT devices.

[14] integrated federated learning with block-chains to create a distributed and immutable audit trail. A deep auto-encoder with three hidden layers was chosen as the anomaly detection method in this paper, with a total of 3000 weights. The paper also uses the FedAvg algorithm for the aggregation. Based on the performance of deep auto-encoder, our project will also choose the deep auto-encoder model as the local model and will follow similar data pre-processing and setup of the loss function and metrics. A trade-off is identified between the local rate of convergence on the clients and the audit frequency of the global model updates.

From all the studies above, there still exist many challenges of federated learning in the IoT environment. The main challenges are the heterogeneity that are from the device (high communication cost), statistical (non-IID data will result in weight divergence), and model (different types of device may have different models). The concept of Federated Transfer Learning (FTL) was introduced in [15] in attempt to solve these issues. FTL takes inspiration from transfer learning, so that further personalisation can be used

to transfer the globally shared model to distributed IoT devices to resolve statistical heterogeneity. The performance of transfer learning is strongly dependent on the relation between different domains. As devices grouped in Federated Learning typically come from the same type of industry, it is suitable to apply transfer learning in the federated learning framework [16][17].

[18] delivered some other strategies to deal with the Non-IID data, such as data augmentation, creating a small dataset which can be shared globally, tuning the capability to run training on the local data on each device and adding an assumption on data dissimilarities. [19] created an imbalanced data to simulate a real dataset which is used to compare the local model with the FL model and to find whether the FL model can improve performance without sufficient data.

3 Research Problems

3.1 Research Aims and Objectives

With the development of information technology and the progress of the digital age, we are generating huge amounts of data every day. The foundation of artificial intelligence, deep learning, and other advanced data driven technologies lies in data. In many industries, organisations are unable to share data due to data privacy, trade secrets, and government regulations. This can lead to data island issues and hinder research institutions and companies from integrating data to drive industry development. Traditionally, businesses and government agencies use their own data to conduct research. For example, hospitals use their own patient data to conduct medical research to prevent leakage of patient privacy, which is not good for hospitals to build smart medicine businesses. However, when all the data are collected into a specific agent for data mining and analysis, it is easy to lead to data leakage and security risks. Federated learning can make full use of the data between different clients and build digital models such as artificial intelligence by uploading parameters of local models without sharing data. Thus, the problem of data island can be solved with ensuring data security. This study is mainly divided into two parts. The first part is to find the optimal federated learning model by analysing and comparing different federated learning models and the combination of different training devices. The second part illustrates the usability and advantages of federated learning by comparing the performance, security, and stability of the model between federated learning and non-federated learning.

3.2 Research Questions

Compared with the traditional machine learning approaches, the main advantage of federated learning is that it can realise the function of data security while guaranteeing the performance of the model. The fundamental question of this research is whether the anomaly detection model based on federated learning is superior to the anomaly detection model based on non-federated learning, and what are the specific advantages of federated learning. To answer this big question, we divided our research into the following small questions:

- **What's the best federated learning method for our study?**

There are different aggregation algorithms and some advanced training methods in federated learning. What impact will these different algorithms have on our dataset and

local model? What is the best method? We will try different federated learning algorithms to get the best performance and security of our model.

• **What's the best devices combinations in our training model?**

In this experiment, due to the property of the data, trying different training devices will have different results, and we want to find out an optimal devices combination to train our federated learning anomaly detection model.

• **How is federated anomaly detection superior to non-federated anomaly detection?**

In this experiment, we will compare the evaluation metric and running time of the two methods to illustrate the advantages of the FL model in this experiment, in addition to safety.

3.3 Research Scope

- Training clients typically have a large number of sets in the federated learning dataset, rather than nine in the current study. Therefore, we do not emphasise the influence of the amount of training clients in each round on the model performance, but more emphasis is placed on the influence of the combination of different specific devices on the model results.
- The local model adopted in this study is the state-of-art Deep Auto-encoder technology, so even a few rounds of training will have a very high performance. Therefore, this study no longer emphasises the tuning parameters of Epochs, Batchsise and Learning rate, but the influence of different federated learning methods on the model performance.

4 Methodologies

In this section, the two proposed approaches are discussed in details with data pre-processing, non-FL and FL models architecture, and implementation. By following our central idea, we experiment with the best practice on anomaly detection with a privacy-preserving approach. The deep auto-encoder model is used not only for non-FL and FL model comparison but also is used as the local model as a part of the FL model. The key idea is that using the DAE model can enhance the capacity of dealing with non-linear representations and project them into compressed layers for model learning. Eventually, the reconstruction error is used for detecting whether the observation is anomalous or benign events. Moreover, there are three major achievements after implementing the local model into the federated learning model: privacy-preserving, efficiency, robustness.

4.1 Data Collection

In this project, the N-BaIoT dataset is provided by the Kaggle website[20], which can also be collected from the UCI website[21]. This dataset contains 89 CSV files that consist of traffic data of nine devices. There are 7,062,606 traffic data in total, and each traffic data has 115 statistical features. The following screenshot is a preview of the one dataset. According to the explanation of the data provider, the names of features stand for the specific statistics of the traffic data during a time frame. Each feature name consists of three parts, and the three parts are connected by an underscore. The first part includes some abbreviation, such as "MI_dir", "HH" and "HpHp", to represent a stream aggregation. The second part, like "L5" and "L3", means a time frame, to represent the stream in the most recent 500 ms or 1.5 sec is captured. And the last part includes the specific statistics information, such as "weight", "mean" and "variance". A more detailed explanation of the feature names can be found on the UCI website.

	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mean	MI_dir_L3_variance
0	1.000000	60.000000	0.000000	1.000000	60.000000	0.000000
1	1.000000	354.000000	0.000000	1.000000	354.000000	0.000000
2	1.857879	360.458980	35.789338	1.912127	360.275733	35.923972
3	1.000000	337.000000	0.000000	1.000000	337.000000	0.000000
4	1.680223	172.140917	18487.448750	1.793580	182.560279	18928.175300

Figure 1: Original Data Preview

4.2 Data Analysis

There are nine devices involves in the dataset and these nine devices belong to five types of IoT devices, including thermostat, doorbell, baby monitor, webcam and security camera. Normally, when a device is functioning, it will generate normal traffic data, which is benign. However, the nine devices are attacked by ten types of Botnet attacks respectively, which will lead to each device generating a large amount of anomalous traffic data. The following table shows the number of benign traffic data and attacked traffic data for each IoT device. It can be seen that the Botnet attack has ten types.

Devices	Type	Benign	Botnet										Total
			Mirai Attacks					Gafgyt Attacks					
			Scan	Ack	Syn	UDP	UDPplain	Scan	Junk	UDP	TCP	Combo	
#1 Danmini	Doorbell	49,548	107,685	102,195	122,573	237,665	81,982	29,849	29,068	105,874	92,141	59,718	**1,018,298**
#2 Ecobee	Thermostat	13,113	43,192	113,285	116,807	151,481	87,368	27,494	30,312	104,791	95,021	53,012	**835,876**
#3 Ennio	Doorbell	39,100	-	-	-	-	-	28,120	29,797	103,933	101,536	53,014	**355,500**
#4 Philips B120N/10	Baby monitor	175,240	103,621	91,123	118,128	217,034	80,808	27,859	28,349	105,782	92,581	58,152	**1,098,677**
#5 Provision PT-737E	Security camera	62,154	96,781	60,554	65,746	156,248	56,681	29,297	30,898	104,011	104,510	61,380	**828,260**
#6 Provision PT-838	Security camera	98,514	97,096	57,997	61,851	158,608	53,785	28,397	29,068	104,658	89,387	57,530	**836,891**
#7 Samsung SNH 1011 N	Webcam	52,150	-	-	-	-	-	27,698	28,305	110,617	97,783	58,669	**375,222**
#8 SimpleHome XCS7-1002-WHT	Security camera	46,585	45,930	111,480	125,715	151,879	78,244	27,825	28,579	103,720	88,816	54,283	**863,056**
#9 SimpleHome XCS7-1003-WHT	Security camera	19,528	43,674	107,187	122,479	157,084	84,436	28,572	27,413	102,980	98,075	59,398	**850,826**
Total		**555,932**	**537,979**	**643,821**	**733,299**	**1,229,999**	**523,304**	**255,111**	**261,789**	**946,366**	**859,850**	**515,156**	**7,062,606**

Figure 2: Number of Traffic Data of Each Device

From the pie chart (Figure 3), the number of benign traffic data counted for only 8% while there are 92% of the traffic data is anomalous. Also, according to the bar chart (Figure 4) below, it is clear to see that, for each device, the number of anomalous data is much more than benign traffic data. Therefore, the data is highly unbalanced.

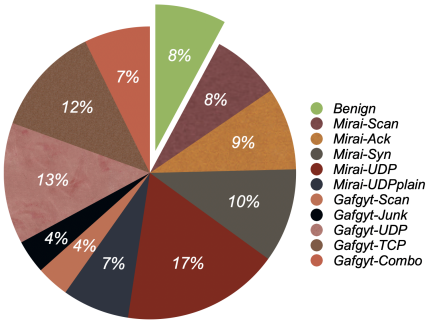

Figure 3: Overall Distribution of Benign Data and Anomalous Data

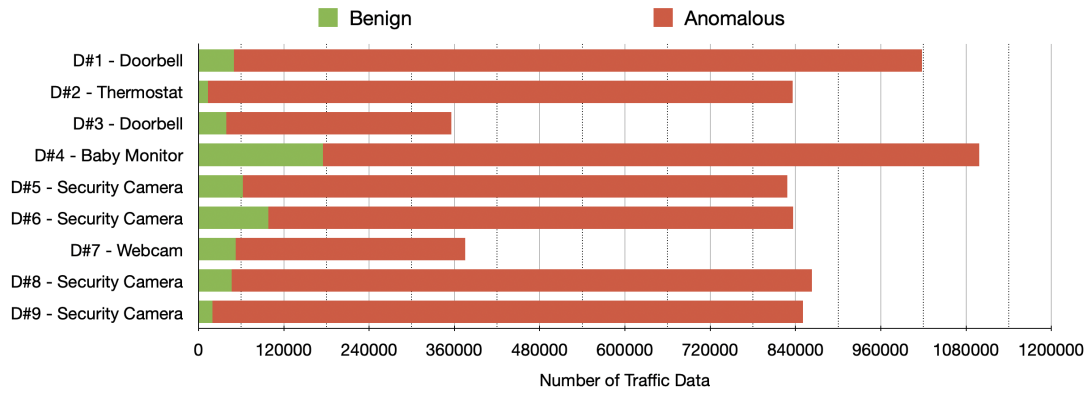

Figure 4: Distribution of Benign Data and Anomalous Data among Nine Devices

Additionally, to better understand the dataset, the statistical information of each feature has been generated. The table(Figure 5) shows a part of the outputs. It is noticed that the different features have a quite different range. As it is difficult to which features are more important than others, all the 115 features would be used in this project.

	MI_dir_L5_weight	MI_dir_L5_mean	MI_dir_L5_variance	MI_dir_L3_weight	MI_dir_L3_mean	MI_dir_L3_variance
count	1.018298e+06	1.018298e+06	1.018298e+06	1.018298e+06	1.018298e+06	1.018298e+06
mean	9.479704e+01	1.927187e+02	1.970133e+04	1.502972e+02	1.934838e+02	2.200175e+04
std	6.380090e+01	1.631548e+02	2.517558e+04	1.010881e+02	1.565037e+02	2.676519e+04
min	1.000000e+00	6.000000e+01	0.000000e+00	1.000000e+00	6.000000e+01	0.000000e+00
25%	1.124271e+01	6.000000e+01	3.755417e-09	2.661462e+01	6.000010e+01	3.010483e-05
50%	1.108011e+02	7.407356e+01	3.869695e+01	1.870080e+02	7.408601e+01	4.342837e+01
75%	1.471055e+02	3.308499e+02	4.867314e+04	2.311400e+02	3.470927e+02	5.360581e+04
max	3.416813e+02	8.861669e+02	1.738886e+05	4.704900e+02	8.463203e+02	1.731848e+05

Figure 5: Preview of Feature Summary

Besides, we have double-checked whether there is any missing value for each CSV file, and the results reflect that this dataset is complete.

4.3 Data Pre-Processing

There are 89 CSV files in total and more than six separated CSV files for each IoT device. To complete the following methodologies for anomaly detection, it is required to load all CSV files at the beginning, which is time-consuming and potential for making mistakes. Therefore, to prevent the possible risks and make the future detection process more easily, for each device, the separated CSV files that belong to the same IoT

device have been combined into a single CSV file. There is a new feature added at the end called "type", to distinguish the traffic data is benign or attacked by which Botnet attack. For instance, the type "benign" means the data is benign, whereas "m_scan" indicates the data is attacked by "Scan" of the Mirai Attacks and "g_combo" represents the data is attacked by "Combo" of the Gafgyt Attacks. Before the combination, the feature names and their order have been checked to ensure every CSV files are consistent. After the processing, the nine devices have a single CSV file with 116 features respectively.

4.4 Method for Anomaly Detection Locally

As we mentioned in the related literature, there exist many different kinds of neural network of choice for anomaly detection. In this project, we proposed the deep auto-encoder model for our anomaly detection application, both as the non-FL model and the local model for Federated Learning. This is because the deep auto-encoder is capable for learning the feature and representation of the observation through many layers and neurons. In the following, the architecture of the deep auto-encoder model is discussed in details, from data pre-processing to anomaly detection.

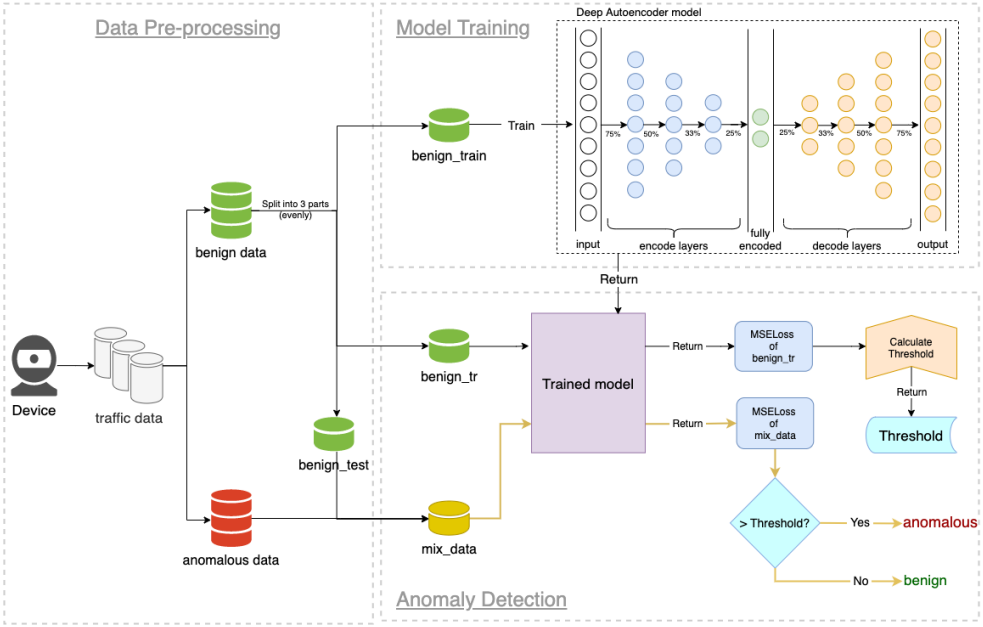

Figure 6: Anomaly Detection with Deep Auto-encoder model

Figure 6 provides an overview of the whole process of anomaly detection by using the deep auto-encoder model locally, starting from data pre-processing. After the datasets of different kinds of attacks and benign are combined and labelled as traffic data, the dataset if split into two kinds - benign and anomalous data. For data pre-processing,

the benign data was split into three parts evenly, which include benign_train, benign_tr, and benign_test.

Firstly, benign_train set is only used for training the DAE model. Since DAE is a special type of NN model that makes target values the same as the input values by forcing the auto-encoder model to learn the input representation, the proposed DAE model only uses benign_train to train and to extract the essential information of benign data. Through the DAE model, benign_train data are treated as the input values which are encoded and decoded from the input layer to hidden layers and to output layers. During this training process, the hidden layers learn the most patterns of the benign data and ignore the "noises". The loss function we used for this DAE model is the Mean Squared Error Loss (MSELoss). Once the model is trained, the benign_tr dataset is specifically used as input into the trained model with outputting the MSELoss for calculating the threshold as the reconstruction error for detecting anomalous. It is important to compute the reconstruction error by using the loss values from the output and input. As if the loss value is low, then it could be assumed that the model is recognising the instance is inside of the known distribution.

Eventually, after the model is trained and reconstruction error is defined, benign_test is combined with anomalous data as mix_data in this practice. The mix_data values will be imputed into the trained model for evaluation. With the return of the loss values from mix_data values, they will be compared with the threshold values. If the value is higher than the threshold values, the input is recognised as anomalous data. Or else, the input is recognised as benign data. In this practice, all nine devices are applied with the same local model structure as non-FL model results.

4.4.1 Architecture of Deep Auto-encoder Model

The initial auto-encoder model is published by Hinton and Zemel, it is an unsupervised model that uses a set of recognition weights to compress input vectors as code vectors, then converts back to reconstructed input vector as output vectors by using a set of generative weights [22]. Compared with the classical dimensionality reduction method, principal component analysis (PCA), deep auto-encoder (DAE) model can handle non-linear feature compression [23]. The deep auto-encoder we proposed for detecting the anomalous activities of the IoT devices, is inspired by the work of Median et al.[4] as the aim of this study is to compare the performance of the FL model and the non-FL model.

The proposed network structure of the Deep Auto-encoder (DAE) is shown in Figure 7. It contains an encoder layer, hidden layer and decoder layer. Using the DAE model, the size of the target values is as same as the input value, as it forces the DAE to learn

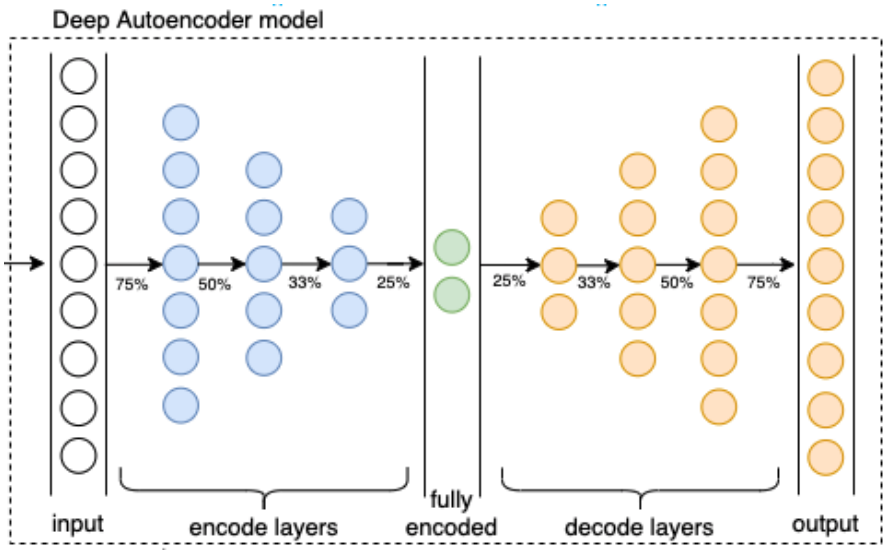

Figure 7: Deep Auto-encoder model Architecture

the representation of the input data. The deep auto-encoder model is only trained with benign traffic data to learn the essential characteristics, then uses the proposed reconstructed error from Median et al.[4] to recognised if the new observation is anomalous data or not.

Theoretically, the output of the encoder layer is the input of the hidden layer, then the output of the hidden layer is the input of the decoder layer. During the training process of the DAE model, the benign traffic data is firstly encoded into four convolutional neural network layers which are 75%, 50%, 33% and 25% of the set of the input of data as following the work of Median et al.[4]. As DAE is a symmetrical network structure, the encoded vector is then decoded layer by layer with 25%, 33%, 50%, and 75% respectively after the final encoding is fully compressed. The decompression in this model is followed through linear layers from the lowest encoding layer to reconstruct the data.

4.4.2 Detection Threshold Calculation

After the model is trained and its mean square error (MSE) is extracted, as demonstrated by Median et al.[4], the threshold (tr) is used as equation 1 shown for determining normal and abnormal observations. The threshold equation is the sum of the sample's MSE mean and the sample's MSE standard deviation over the benign train data. When the value of the instance is above the threshold value, it is considered anomalous data, otherwise, it is considered benign data.

$$tr = \overline{MSE}_{benign\_tr} + std(MSE_{benign\_tr})$$

4.5 Method for Anomaly Detection with Federated Learning

To achieve the project goal, the Federated Learning technique has been applied to the deep auto-encoder model to detect anomalous data. According to the literature review, Federated Learning could help training the deep learning model on various devices to process data while keeping the preservation privately. One of the advantages of applying Federated Learning is it does not require storing and exchanging any personal data, which could ensure the privacy problem.

The Figure 8 shows the model architecture of the anomaly detection by using the federated deep auto-encoder model. For each device involved in the training process, also called clients, the data pre-processing is the same as the way of the local detection method. When starting the federated detection process, the training process would happen simultaneously on all devices for training. Also, the weights of each deep auto-encoder model of each device are sent to the global server, and the aggregation function, such as FedAvg or FedAvgM, process the weights. Each deep auto-encoder model of each device would receive the aggregated weights and continues on the training process.

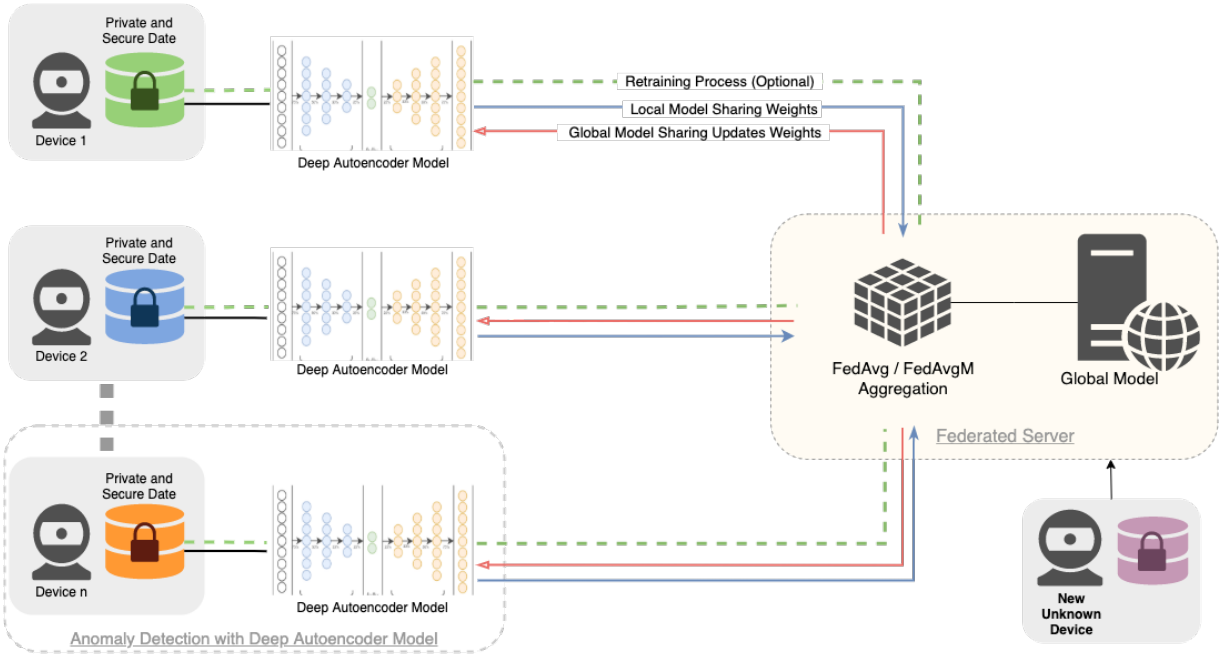

Figure 8: Anomaly Detection with Federated Deep Auto-encoder model

The main step of the entire process includes the forward process, sends weight to the global server for the update and the back propagation process. In federated learning, this whole process is called one communication round. There are a number of communication rounds have been applied to update the weights and more rounds tend to improve the performance of the federated model.

Furthermore, in the federated model architecture, there is a retraining process has been added to make the model handling non-IID datasets, which also ensures the model to be more suitable for the real world. The retraining process means the clients' model would be trained additionally before the weights aggregation on the global server. The data for this process is randomly selected from the corresponding training data.

Moreover, to improve the federated model efficiency and robustness, a partial selection mechanism has been involved in the federated model architecture. This means the partial client devices would be randomly chosen for the real training process in each communication round instead of using all of the client devices.

After the federated model is trained, the anomaly detection can be directly applied on both the client devices and any new devices.

4.5.1 Federated Learning with Client Selection

The datasets used in this study are heterogeneity and non-Identical Independent Distribution (non-IID). However, in the FL model, the whole configuration is sensitive to the distribution of the client's data classes and feature. It may affect the whole training time and accuracy. Driven by the mentioned acknowledgment, we also propose two approaches, which are client selection and retraining process in the FL with DAE model to improve the performance of the model. As shown by the Results section, the comparison of using/not using client selection and retraining approach has different effects on the FL model's efficiency and robustness.

Instead of using the communication rounds to update the weights and improve the performance of the whole FL model, client selection is also used after to minimising the non-IID dataset's impact on the model. During the client selection process, the model is randomly choosing partial training devices to be involved for training the model. For instance, five devices are chosen to train the model as clients, only two clients are chosen for training the global model in each communication round from total five number of clients. The selected clients are updated with the global weights in each communication round.

4.5.2 Federated Learning with Retraining Process

In addition, combined with the client selection process, the retraining process is involved optionally for addition improvement on the FL model. It also aims to resolve the issue of using Non-IID datasets and make the model more suitable for the real world. There will be randomly selected 1000 instances from the benign_train data for

the retraining process. The client's model will be retrained before the aggregation of weights on the global server, so that the weights from both communication round and retrained round are used for the aggregation of weights on the global server. The local models on each client will be updated with the global weights before the next round starts for further training. Therefore, when the global model is done with updating and training, it is used to test the model as anomaly detection.

4.5.3 Aggregation in FL Model - Federated Averaging Algorithm

Federated optimisation is built from stochastic gradient descent (SGD, and has the following properties):

- applied naively to the federated optimisation problem, where a single batch gradient calculation is done per round of communication
- computationally efficient
- requires very large numbers of rounds of training to produce good models (even with batch normalisation)

As mentioned previously, the Federated Learning approach first selects C-fraction of clients on each round and computes the gradient of the loss over all the data held by these clients, where C is the fraction of the global batch size.

A typical implementation of FedSGD is with $C = 1$ and a fixed learning rate η. each client k will compute:

$$g_k = \nabla F_k(w_t) \tag{1}$$

The average gradient on its local data at the current model w_t, and the central server aggregates these gradients and applies the update:

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k \tag{2}$$

Since $\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t)$, an equivalent update is given by $\forall k, w_{t+1}^k \leftarrow w_t - \eta g_k$,

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{3}$$

Each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. More

computation can be added to each client by iterating the local update:

$$w^k \leftarrow w^k - \eta = F_k(w^k) \tag{4}$$

Federated Averaging (or FedAvg) is computed by adding computation to each client by iterating the local update multiple times before the averaging step:

$$w^k \leftarrow w^k - \eta \nabla F_k(w^k) \tag{5}$$

The amount of computation is controlled by three key parameters:

- C, the fraction of clients that perform computation on each round;
- E, then number of training passes each client makes over its local dataset on each round;
- and B, the local minibatch size used for the client updates

From a statistical perspective, FedAvg has been shown to diverge empirically in settings where the data is non-identically distributed across devices.

The pseudocode for FedAvg is given in Algorithm 1:

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

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

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

4.5.4 Aggregation in FL Model - Federated Averaging Algorithm with Momentum

In our second approach on FL model aggregation, is inspired by Hsu et al., as demonstrated that using momentum on top of the FedAvg algorithm at the sever has improved the performance from 35% to 75% and stayed relatively constant [24]. Therefore, we are also expecting to achieve better performance by using FedAvgM than using FedAvg aggregation in this study.

Federated averaging with momentum (FedAvgM) is an algorithm that shares mostly the same theory and algorithm as we introduced in the FedAvg section, with adding the momentum on top of the SGD at the server instead. However, instead of updating the weights by averaging the weights from all client model at the server as algorithm 1 implies, the momentum vector is added for updating the weights to the updated model at the server. The algorithm of updating the weights changes from

$$w \leftarrow w - \Delta w$$

where $\Delta w_k$is the weighted updated from k th client, to

$$w \leftarrow w - v$$

as v is the momentum vector computed as $v \leftarrow \beta v + \Delta w$. By using momentum on top of the SGD at the server, it allows us to accelerate the convergence of the optimisation techniques by speeding up the learning in the directions of curvatures. In this experiment, we have set the momentum values β as 0.9, as demonstrated from related studies that when momentum is 0.9, the convergence rates on both loss function and accuracy also steadily increase [25], [26]. However, setting the value of momentum values to 0.9 is not the best behave for universal use, it is a relatively general use that not have to attempt for the further tune, especially we aim to compare the performance of non-FL and FL model in this practice.

4.6 Evaluation Methods

Confusion Matrix:

		True Class	
		Abnormal	**Benign**
Predicted Class	**Abnormal**	True Positive (TP)	False Positive (FP)
	Benign	False Negative (FN)	True Negative (TN)

The confusion matrix for anomaly detection plots the output of the predicted class against the actual class, where positive represents abnormal data and negative represents benign data. The following terms can be extracted from the confusion matrix:

- True Positive (TP) is when the model correctly predicts the positive class.
- True Negative (TN) is when the model correctly predicts the negative class.
- False Positive (FP) is when the model incorrectly predicts the positive class.
- False Negative (FN) is when the model incorrectly predicts the negative class.

The **Evaluation Metrics** used to compare the performance of different models are given by the equations below:

Evaluation Metric	**Formula**
Recall or True Positive Rate	$TPR = \frac{TP}{TP+FN}$
Fall Out or False Positive Rate	$FPR = \frac{FP}{FP+TN}$
F1 Score	$F_1 score = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$
Overall Accuracy	$Acc = \frac{TP+TN}{(TP+FP+FN+TN)}$

4.7 Other Related Techniques

MSE Loss measures the average squared difference between the estimated values and the actual value, as given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{6}$$

5 Resources

5.1 Hardware & Software

All the experiments were implemented on a virtual cloud environment Google Colab, the specific setting as follow:

Hardware	**Configuration**
CPU	IIntel(R) Xeon(R) CPU @ 2.30GHz
Hard Disk	69G
Memory	12.7G
GPU	Tesla K80

In this project,the Numpy and Pandas libraries were be used all the time to do the data manipulation. We mainly use scikit- learn libraries to conduct data preprocessing and build the evaluation metrics. Pytorch was used to build the local models and Federated Learning models.

Software	**Version**
Python	3.7.10
Pytorch	1.8.1
Numpy	1.19.5
Pandas	1.1.5
Scikit-learn	0.22.2.post1

5.2 Materials

Description	**Link**
Kaggle dataset	https://www.kaggle.com/mkashifn/nbaiot-dataset
Federated Learning tutorial course	https://classroom.udacity.com/courses/ud185
Pytorch tutorial	https://pytorch.org/tutorials/
Federated Learing implementation tutorial	https://towardsdatascience.com/preserving-data-privacy-in-deep-learning-part-3-ae2103c40c22
GitHub	https://github.com

5.3 Roles & Responsibilities

Name	Roles	Responsibility
Wenbo Yan	Group Leader	- Literature Review - Data Exploration - Training other local models - Training different FL method and devices combination for performance comparison - Worked and communicated with members actively - Worked on Final presentation and Proposal/Progress/Final Reports with member cooperatively
Sophie Zou	Spokesperson	- Documentation of Literature Review - Data Exploration - Helped out in coding e.g. data pre-processing and training other local models with Federated Learning - Training different FL method and devices combination for performance comparison - Improvement of Presentation Script and Report - Worked and communicated with members actively - Worked on Final presentation and Proposal/Progress/Final Reports with member cooperatively
Ruijue Zou	Coder	- Literature Review - Data Exploration - Data Pre-Processing - Built DAE and FL model - Finalized the FL aggregation - Training different FL method and devices combination for performance comparison - Worked and communicated with members actively - Worked on Final presentation and Proposal/Progress/Final Reports with member cooperatively
Ling Nga Meric Tong	Scribe	- Literature Review - Data Exploration - Partial Support on building models - Initial implementation of FedAvgM aggregation with members - Training different FL method and devices combination for performance comparison - Weekly presentation on models' architecture and training process - Worked and communicated with members actively - Worked on Final presentation and Proposal/Progress/Final Reports with member cooperatively
Xia Wei	Reporter	- Literature Review - Data Exploration - Training other local models - Training different FL method and devices combination for performance comparison - PCA of the datasets - Worked and communicated with members actively - Worked on Final presentation and Proposal/Progress/Final Reports with member cooperatively

6 Milestones / Schedule

Week	Tasks	Reporting	Date
1	Form Groups Expression of interest for capstone topics	Topic Chosen	04/03/2021
2	Analysis and design stage, explore datasets	First Client Meeting to review the project	11/03/2021
3	Literature Review of anomaly detection Simple machine learning model implementation	Client Meeting to review the work plan	18/03/2021
4	Literature Review of Federated Learning Decide models to implement on	Client Meeting to review proposed models	25/03/2021
5	Proposal Report Due	Client Meeting to review proposal ideas	03/04/2021
6	Summarising the difference of attacks Train local models for four camera devices	Client Meeting to review training results	15/04/2021
7	Train local models for four camera devices.	Client Meeting to review further results	22/04/2021
8	Connect FL with local models Testing Models on-camera devices	Client Meeting to review model results.	03/05/2021
9	Mid Semester Project Status Checking Due	Discuss the results with the client	10/05/2021
10	Finalise the federated detection models for camera devices.	Client Meeting to report the results.	17/05/2021
11	Apply the final federated model for other types of devices.	Client Meeting to report updated results.	24/05/2021
12	Experiments - Try different devices combination	Client Meeting to discuss final project	01/06/2021
13	Experiments - Try different FL methods Experiments - Compare FL with Non-FL	Client Meeting to review final experiments results	08/06/2021
14	Experiments Finalization Final Presentation	Final Presentation	11/06/2021
15	Q&A Session Final Report	Final Report	20/06/2021
16	Organise Final Project Artifacts	Final Project Artifacts Submission	25/06/2021

7 Results

We have set different experiment combinations with the methods mentioned in part 4 to test the FL model's performance. The components of the experiment combinations include FedAvgM, FedAvg, Retraining, No Retraining, Partial Selection, No Partial Selection. Since a lot of experiment combinations in our study, there is a need to use some series numbers to distinguish them. The abbreviation of each experiment combination is given by the letter of each component. Take the method of MRP as an example, it is the combination of FedAvgM, Retraining and Partial Selection.In more detail, for FL learning, the first letter "F" and "M" on behalf of FedAvg and FedAvgM. For the retraining process and no retraining process, the second letter "R" and "N" represent their processes respectively. During the training,the third letter "P" and "N" on behalf of partial selection and no partial selection. The figure 9 shows a summary of all the combinations. In order to find the best experimental combination, we compare the performance of every experiment combination using the average TPR, average FPR, average F1 score and Running time.

	MRP	MRN	MNP	MNN	FRP	FRN	FNP	FNN
FedAvgM	•	•	•	•				
FedAvg					•	•	•	•
Retraining	•	•			•	•		
No Retraining			•	•			•	•
Partial Selection	•		•		•		•	
No Partial Selection		•		•		•		•

Figure 9: All experiment combinations

The following parameters need to be adjusted based on each experiment combination, which are shown in the following table.

Table 1: The parameters used in our experiments

Parameters	**Value**	**Meaning**
lr	0.012	Learning rate for Deep Auto-encoder
num_clients	9	Number of clients
num_selected	9	Number of clients we choose for train
batch_size	128	Defines the dataset size in each training iteration
baseline_num	1000	Choose some data from the train set to retrain the data from trained model
num_rounds	100	Total number of communication rounds for the global model to train.
epochs	5	For train client model
retrain_epochs	5	Total number for retrian the global server after receiving the model weights
local_epochs	500	Only for the local deep auto-encoder training

7.1 FedAvg & Retraining VS FedAvgM & Retraining

To evaluate the performance of the FL aggregations, the combinations of FRN and MRN are compared first. Given that the condition of No Partial Select Clients is same, the parameters of these two experiments are same as shown in table 1. All devices are trained, and the averaged TPR, FPR and F1 score value from the nine devices are calculated. From the table below, the average FPR of MRN (FedAvgM,Retraining,No partial selection) is lower, and the average F1 score of MRN is higher, which means the performance of FRN is better. Thus, FedAvgM is shown to have better performance.

Table 2: FRN VS MRN

Method(Train 1-9)	**Avg TPR**	**Avg FPR**	**Avg F1 score**	**Time(mins)**
FRN	0.90090	0.03498	93.073	41.7
MRN	0.90090	0.02189	93.096	48

7.2 FedAvgM & Retraining VS FedAvgM & No Retraining

To evaluate the effects of retraining on the performance of FL model, it is necessary to compare FL learning with retraining and FL learning without retraining given that the condition of No Partial Select Clients is same. Previous experiments have shown that the performance of FedAvgM is better, so the FedAvgM will be used for comparison. The methods used to compare are MRN and MNN. Except for the retrain_epochs, all parameters in these two experiments are same as in the table1. For MNN, the retrain_epochs is not used. From the table below, there is almost no difference between the metrics results of these two experiment combinations. However, FedAvgM with no retraining does takes less time, thus making it a better approach.

Table 3: MRN VS MNN

Method(Train 1-9)	**Avg TPR**	**Avg FPR**	**Avg F1 score**	**Time(mins)**
MRN	0.90090	0.02189	93.096	48
MNN	0.90090	0.02189	93.096	36.4

7.3 FedAvgM & No Retraining & No partial Select Clients VS FedAvgM & No Retraining & Partial Select Clients

Intuitively, FL learning with partial Select Clients would take less time. The performance of FL learning with No Partial Select Clients are further investigated. From

previous experiments, it is known that the FedAvgM with No Retraining performs better. Therefore, we can compare MNN and MNP to evaluate the importance of the component of Partial Selection. For the parameters of these two experiments, there is no need to use the parameter retrain_epochs for both experiments. The other parameters for MNN are the same as in table 1. For MNP, the num_selected is 3 which means in each round, the global model randomly select 3 devices from all devices to train; the other parameters are kept the same as in table 1. From the table below, there is almost no difference between MNN and MNP, but the running time of MNP is efficient which is about 13.1 mins. Therefore, according to the experiment combinations previously, the MNP is the best method, where Federated Deep Auto-encoder model is combined with FedAvgM, No Retraining and Partial Selection.

Table 4: MNN VS MNP

Method(Train 1-9)	**Avg TPR**	**Avg FPR**	**Avg F1 score**	**Time(mins)**
MNN	0.90090	0.02189	93.096	36.4
MNP	0.90090	0.02189	93.096	13.1

7.4 Different Methods Combination in Nine Devices Results

To show the results of different federated learning methods in more detail, the average TPR, FPR, F1 score and Training time of the eight methods are shown in Table5. It can be seen that the MNP method has the best performance in FPR, F1 score and Training time. The MRP method's training time is only about 18 seconds slower than MNP, which indicates that FedAvgM,No Retraining and Partial Selection methods spend less time and have better performance. Tables 6, 7 and 8 respectively show the results of FPR, TPR and F1 score on nine devices by different methods in detail. It can be seen from the three tables that although the TPR and F1 score of devices 3 and 9 is relatively low, they still have relatively low FPR compared with other devices.

Table 5: Average results for all methods combination

Method\Devices	**Avg TPR**	**Avg FPR**	**Avg F1 score**	**Train Time**
FNN	**0.9009044444**	0.03499666667	93.07311111	43.3
FNP	0.9009033333	0.03525	93.07255556	52.6
FRN	0.9009044444	0.03498777778	93.07311111	41.7
FRP	**0.9009033333**	0.03500444444	93.07277778	29.32
MNN	0.9009022222	**0.02189777778**	**93.09566667**	36.4
MNP	0.9009022222	**0.02189777778**	**93.09566667**	**13.1**
MRN	0.9009022222	**0.02189777778**	**93.09566667**	48
MRP	0.9009022222	**0.02189777778**	**93.09566667**	13.4

Table 6: TPR for different methods in nine devices

Method	**Device Number**								
	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**#9**
FNN	**0.99999**	0.99999	0.35083	**0.99987**	0.99992	0.99988	**0.99961**	0.99991	**0.75814**
FNP	**0.99999**	**1**	0.35083	0.99986	0.99992	0.99988	0.9996	0.99991	**0.75814**
FRN	**0.99999**	0.99999	0.35083	**0.99987**	0.99992	0.99988	**0.99961**	0.99991	**0.75814**
FRP	**0.99999**	**1**	0.35083	0.99986	0.99992	0.99988	0.9996	0.99991	**0.75814**
MNN	0.99998	**1**	**0.35081**	0.99986	**0.99994**	**0.99989**	0.9996	0.99991	0.75813
MNP	0.99998	**1**	**0.35081**	0.99986	**0.99994**	**0.99989**	0.9996	0.99991	0.75813
MRN	0.99998	**1**	**0.35081**	0.99986	**0.99994**	**0.99989**	0.9996	0.99991	0.75813
MRP	0.99998	**1**	**0.35081**	0.99986	**0.99994**	**0.99989**	0.9996	0.99991	0.75813

Table 7: FPR for different methods in nine devices

Method	**Device Number**								
	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**#9**
FNN	0.00702	0.03478	0.00568	0.02415	**0.10706**	0.06767	0.01553	0.04939	0.00369
FNP	0.00702	0.03409	0.00568	0.02412	0.10826	0.06928	0.01553	0.04958	0.00369
FRN	0.00702	0.03478	0.00568	0.02415	0.10711	0.06754	0.01553	0.04939	0.00369
FRP	0.00702	**0.0334**	0.00568	0.0241	0.10731	0.06879	0.01553	0.04952	0.00369
MNN	**0.00702**	0.04118	**0.00476**	**0.02282**	0.02698	**0.02805**	**0.01547**	**0.04727**	**0.00353**
MNP	**0.00702**	0.04118	**0.00476**	**0.02282**	0.02698	**0.02805**	**0.01547**	**0.04727**	**0.00353**
MRN	**0.00702**	0.04118	**0.00476**	**0.02282**	0.02698	**0.02805**	**0.01547**	**0.04727**	**0.00353**
MRP	**0.00702**	0.04118	**0.00476**	**0.02282**	0.02698	**0.02805**	**0.01547**	**0.04727**	**0.00353**

Table 8: F1 score for different methods in nine devices

Method	**Device Number**								
	#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**#9**
FNN	99.993	99.99	**51.934**	99.917	99.851	**99.844**	**99.939**	99.948	86.242
FNP	99.993	**99.991**	**51.934**	99.917	99.85	**99.84**	99.938	99.948	86.242
FRN	99.993	99.99	**51.934**	99.917	99.851	**99.844**	**99.939**	99.948	86.242
FRP	99.993	**99.991**	**51.934**	99.917	99.851	99.841	99.938	99.948	86.242
MNN	99.993	99.989	51.933	**99.921**	**99.961**	99.932	**99.939**	**99.951**	86.242
MNP	99.993	99.989	51.933	**99.921**	**99.961**	99.932	**99.939**	**99.951**	86.242
MRN	99.993	99.989	51.933	**99.921**	**99.961**	99.932	**99.939**	**99.951**	86.242
MRP	99.993	99.989	51.933	**99.921**	**99.961**	99.932	**99.939**	**99.951**	86.242

7.5 Devices Combinations with MNP

From the previous experiments, the performance of device 3 and device 9 are much lower than that of the other devices. The following two figures show the performance comparison for the nine devices. The TPR and F1 Score of device 3 and 9 using all methods are lower than the others.

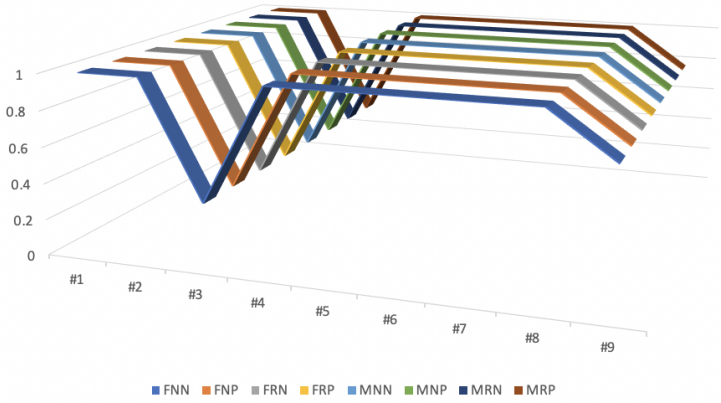

Figure 10: TPR comparison for nine devices

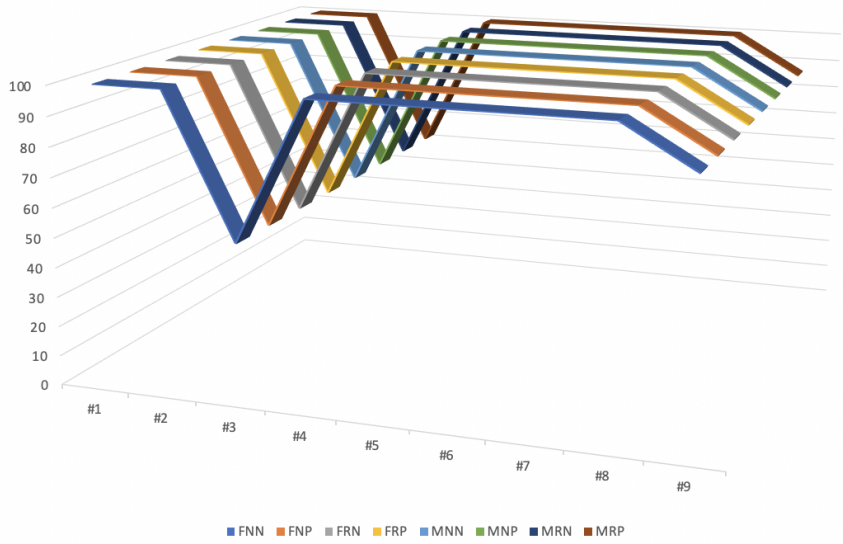

Figure 11: F1 score comparison for nine devices

Therefore, in order to reduce the impact of 3 and 9 devices on overall performance, it is necessary to test the performance with the device combinations for train. For the parameters of MNP experiments, there is no need to use the parameter retrain_epochs, and the num_clients, num_selected changed according to the train set of each experiment combination, and other parameters keep same as shown in Table1. From the table below, the performance of the device combinations with device1247 for train is

shown to be better, as it has a higher TPR, lower FPR, higher F1 score and relatively efficient running time. From the results, the best model is the MNP(Federated Deep Auto-encoder model with FedAvgM with No Retraining and No Partial Selection) with device1247 for train.

Table 9: Experiment combinations about MNP

Device Combinations for Train	**Avg TPR**	**Avg FPR**	**Avg F1 score**	**Time(mins)**
Train1-9 (num_clients=9,num_selected =3)	0.90090	0.02189	93.096	13.1
Train5689 (num_clients=4,num_selected =1)	0.97303	0.02833	98.445	3.6
Train568 (num_clients=3,num_selected =1)	0.99988	0.03288	99.957	4.9
Train1247 (num_clients=4,num_selected =1)	0.99988	0.02836	99.966	8.6
Train12347 (num_clients=5,num_selected =2)	0.92777	0.02393	94.632	9.2

7.6 Deep Auto-encoder Model VS Federated Deep Auto-encoder Model (best one)

After screening the best anomaly detection deep auto-encoder model based on federated learning, we compared the performance of federated learning model and non-federated learning model. The figure 12 depicts a comparison of TPR between FL and Non-FL in nine devices, where the TPR of Device3 and Device9 in Non-FL model performs extremely poor compared to the FL model. Besides, as we can see from figure 13 and figure 14, the FL model can achieve more stable and higher performance in average F1 score and TPR metrics. Moreover, the table 10 shows that the FL model training process can be much faster the Non-FL model, which is a huge superiority for building instant anomaly detection model. However, the Non-FL model have the lower mean of FPR which is about 0.006 less than the FL model.

Table 10: FL model VS Non-FL model

Model type	**Avg TPR**	**Avg FPR**	**Avg F1 score**	**Time(mins)**
FL model	**0.99988**	0.02836	**99.966**	**8.6**
Non-FL model	0.90076	**0.02232**	93.095	3109

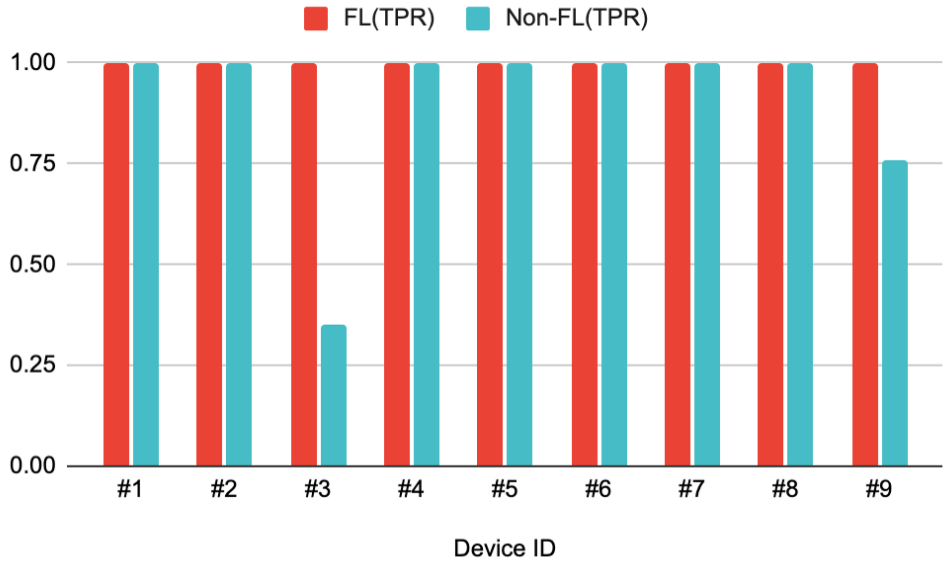

Figure 12: FL vs Non-FL TPR comparison for nine devices

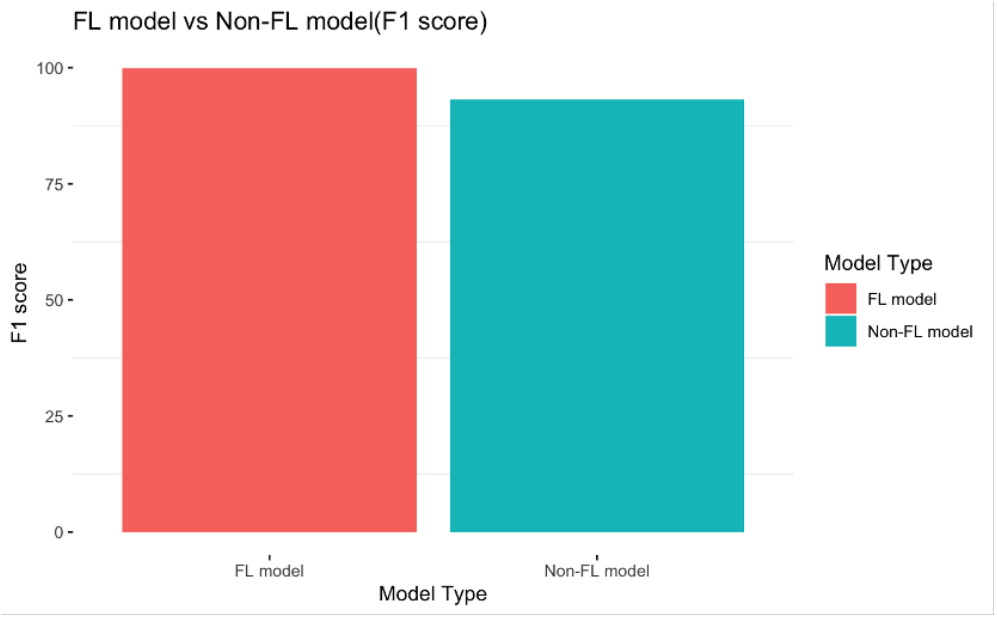

Figure 13: FL vs Non-FL Average F1 score comparison

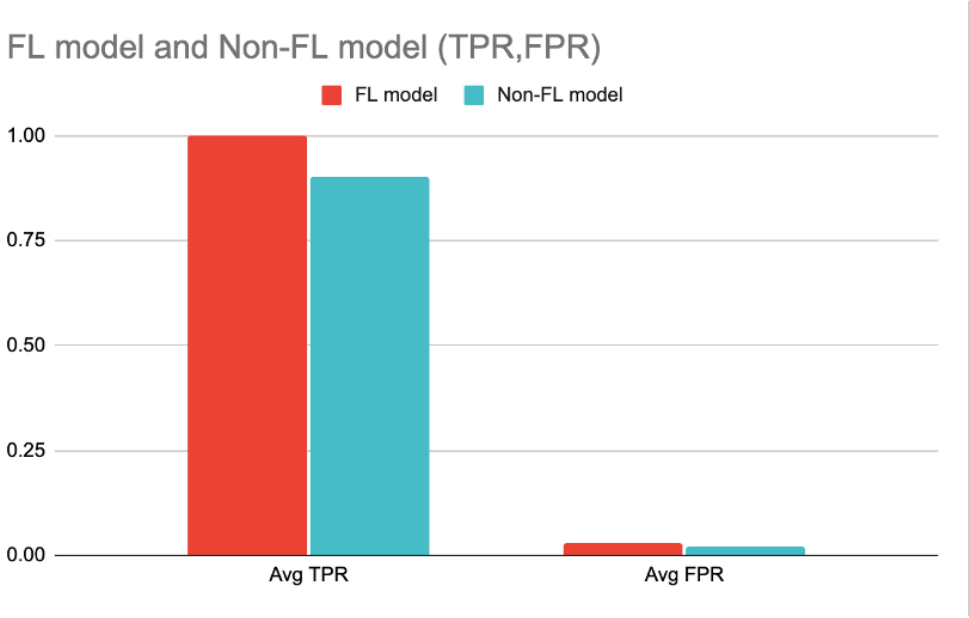

Figure 14: FL vs Non-FL Average TPR,FPR comparison

8 Discussion

8.1 FL Methods Combination Discussion

In this project, we have tested different federated aggregation algorithm, training model and client selection model respectively. By comparing different evaluation metrics, we can finally conclude that the combination of FedAvgM, No Retraining and Partial Selection is the best Federated Learning model. In addition, the dataset used in this study is Non-IID data, and the categories of data are very uneven. According to section7.1, we can find that the FedAvgM aggregation algorithm has better performance than FedAvg in this paper. This further verifies that the FedAvgM algorithm proposed by T.-M. H. Hsu et al. has better performance and higher stability on non-IID datasets [24]. Surprisingly, retraining and no retraining have the same evaluation scores, but the training time of retraining is longer because there is an extra training session for retraining. The Retraining process itself is used to deal with the instability in model training caused by Non-IID data, and the likely reason for this result in section7.2 is that the data sets used this time are all traffic data of Internet of Things. Although there exists different types of devices, these devices are likely to have overall similar features. As a result, retraining method may not play a good role in generalization. Partial selection can speed up the training of the model for it can decrease the number of clients in each communication round, but the FL model can still learn the characteristics of each device through multiple rounds of training.

Moreover, in traditional federated learning studies, the number of clients is very large. For example, in the vanilla FedAvg algorithms proposed by Google, the number of clients is 100, and each client only have 600 samples [12]. Therefore, the parameters tuning process of a federated learning technology have a significant impact on the results. As shown in table 11,12, adjusting parameters such as Batch size or Epoch has no effect on the results of the data set of this study, because there are few training devices and the data volume of each device is very large. Thus, by combining different federated learning techniques, we can implement the process of techniques tuning for a small number of clients data sets to select the best FL methods.

8.2 Different Devices Combination Discussion

Through the previous study, we found that when we trained with device #3 and #9 data, the models performed very poorly on both the test datasets of device #3 and #9, which was probably caused by the different data characteristics of them. As can be seen from figure 4, the amount of data of device #3 is small compared with other data, and the

proportion of normal data is large, which may lead to over-fitting, thus resulting in poor test result of the model. Device #5, #6, #8 and #9 are all security camera type devices, but the combination of Device #5, #6 and #8 performs well, so we guess that Device #9 may have a special distribution of data features. We performed PCA decomposition of the 115 features of these four devices, and drew the first two principal components as shown in figure15. Device #5 and #6 have similar pattern distribution, while device #8 and #9 are generally similar, but the abnormal data pattern in device #9 is not the same with device #8. This probably resulted in device #9 not performing as well as device #8.

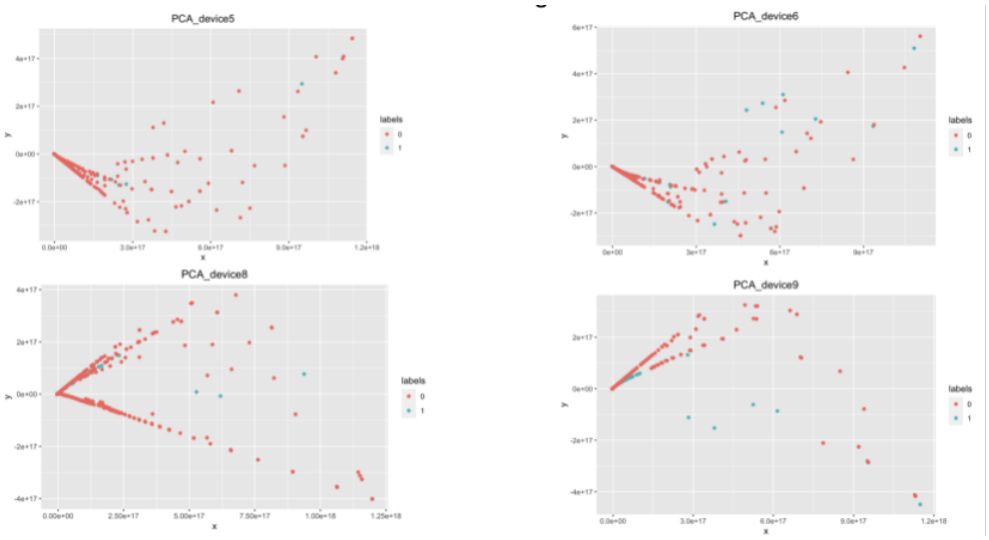

Figure 15: PCA plots of abnormal and benign data in Device #5, #6, #8 and #9(0:Abnormal, 1:Benign)

Device #5, #6 and #8 is a combination of the same device type (all security cameras), while Device#1, #2, #4 and #7 is a combination of different types of devices. Device#1, #2, #4 and #7 has the better performance, as the FPR is around 0.005 smaller than Device #5, #6 and #8. It is likely that this combination learns from the data characteristics in different devices, thus improving the generalization ability and stability of the model.

Finally, we get the optimal device combination for Federated Learning to achieve the best model performance. And we also found that when a single device, such as device #3 crashes, we can discard it, and federated learning can still train well-performing models on a small number of devices without put all devices into the training process.

8.3 FL vs Non-FL Comparison Discussion

To be fair and credible to compare the Federated Deep auto-encoder model and Deep auto-encoder model performance, we adopted the same model architecture and training

data partition way with this study [4]. However, they did not give the specific way of how to partition test data, so we have no way to directly compare our results with theirs, and we finally choose to build our own test set. In section 7.6, it can be found that the performance of FL model is better than the Non-FL model, especially in Device #3 and #9, where FL model has a high F1 score and FPR. This shows that FL model has a very good generalization ability, and the characteristics of different devices can be learned by aggregating weights and other parameters of different local models. In addition, compared with Non-FL model, FL has an extremely fast training speed, because FL model requires less training devices in each round, so it requires less training data. In industrial applications, it is important to build a fast network anomaly detection system, because the earlier the anomaly is detected, the lower the loss of business and personal property.

Furthermore, anomaly detection of federated learning has a very important advantage that it has very strong and robust security. In the research of this article, we used home IoT data. In reality, security companies usually collect IoT devices from multiple homes to train a model to achieve excellent performance, but the premise is that every home needs to upload all the data to the company, and many of these data involve personal privacy, such as camera device. But if a large amount of data is not gathered into one model, then the company cannot train an anomaly detection model with excellent performance and generalizability, and federated learning can solve this problem well. Federated learning uploads only the model parameters to the global model each communication rounds, and this easily avoids the direct data leakage problem, and ultimately guarantees the excellent performance of the global model.

Therefore, the FL model is superior to the non-FL model, especially in terms of data safety, model generalization and model training speed. Another advantage of the FL model is that it can train a small number of devices to obtain a global model that is applicable to all devices. It also ensures that in practical applications, IoT security companies can obtain a general model from only a small number of devices which is a huge promotion on IoT intrusion detection.

9 Limitation and Future Work

9.1 Limitations

There exists some limitations in our study which we intend to improve in the future, of which most limitations were due to time constraints of our capstone project.

- **Sample Size** – Previous literature on Federated Learning used a large number of devices for training and testing, however, the N-BaIoT dataset used in this report contained only nine devices. Thus, the sample size may have been too small to gain a throughout understanding of the performance of Federated Learning.
- **Quality of Dataset** – The distribution of the benign and malicious data was extremely uneven. Most devices had a vast number of anomalous traffic data and a lack of benign data. This uneven distribution may have resulted in the low accuracy of device #3 and #9 when using them as the training dataset.
- **Use of previous literature for local Auto-encoder model** – We used the data selection procedure and threshold calculation methods from previous literature as their results showed the best performance. The use of methods from previous literature may limit us as our dataset may require different percentages of training and testing data to calculate its best threshold for the auto-encoder model.
- **Limited Access to Methodologies** - The Federated Learning model is not very stable as it is sensitive to the Non-IID data properties of the local client. We used the simple approach of FedAvg and FedAvgM as the aggregation algorithm. There exists other literature which focus on resolving the issue of Non-IID in Federated Learning. Further explanation will be provided in the Future Work section.
- **Hyper-parameter Tuning** – has been performed on batch set however the accuracy and F1 score difference from hyper-parameter tuning was not obvious enough for interpretation.
- **Time Constraints** – This project was completed as a capstone project for Master of Data Science which had a time frame of one semester only. Due to such time constraints, we could not experiment with more advanced methodologies and algorithms.
- **Platform Constraints** – We mainly used Google Colab to run our codes. The FedAvg required many training rounds (at least 300 rounds) to generate a good model. Unfortunately, as Google Colab has a limit of only 12G of RAM for each

user, the approach of Federated Learning using PySyft resulted in Colab to crash. This limitation caused us to disregard PySyft in our code.

9.2 Future Work

9.2.1 Increase Number of Devices

As mentioned previously, the small sample size of only nine devices is a limitation in our study. We can introduce more devices by splitting our data from each device into 100 sets, each representing one device. Thus, we will generate a total of 900 devices for evaluating our federated model. This will allow us to model real world situations of federated learning and be able to gain a better understanding.

9.2.2 Block chain

Federated learning can be integrated with blockchains to support the auditing of machine learning models without the necessity of centralizing the training data. The basic methodology of setting up the auto-encoder model for anomaly detection will be the same as our proposed methods, where a metric used to identify whether a test sample belongs to benign or not. The weight updates and models are then stored with each epoch on the blockchain. To leverage the blockchain, a parameter server will then process the weight updates that have been stored and confirmed on the blockchain. As Federated Learning often involves numerous weight updates, public blockchains such as Bitcoin would be too slow to generate the new blocks on the ledger. An open source private blockchain called MultiChain supports multiple assets and mining without proof-of-work. However, the blockchain can only verify the data stored inside it but cannot assure accuracy overall. From the paper [14], the block-chained federated learning approach increased their performance of around 5~15%. By applying this approach to our data, we hope to increase the performance of federated learning overall when dealing with many devices.

9.2.3 Federated Transfer Learning

A major source of our limitation was the non-IID characteristic of the N-BaIoT dataset. Traditional Federated Learning assumes that the training data from different devices share the same feature space. This is often not applicable in real world circumstances. Our dataset consists of devices ranging from security cameras, doorbells, and thermostats. Transfer Learning allows models to be trained on a large dataset for one do-

main is applied to a different but related domain. This technique can be combined with Federated Learning to mitigate the statistical heterogeneity. Federated Transfer learning (FTL) transfers the globally shared model to distributed IoT devices. A typical architecture of federated transfer learning is shown in Figure 16. Consider two datasets A and B, where there exists only a small overlap in feature space and sample space. The model learned from B is transferred to A by leveraging the small overlapping data features. FTL transfers knowledge from non-overlapping features from source domain to the new samples in the target domain, the region in right upper corner of the Figure 16.

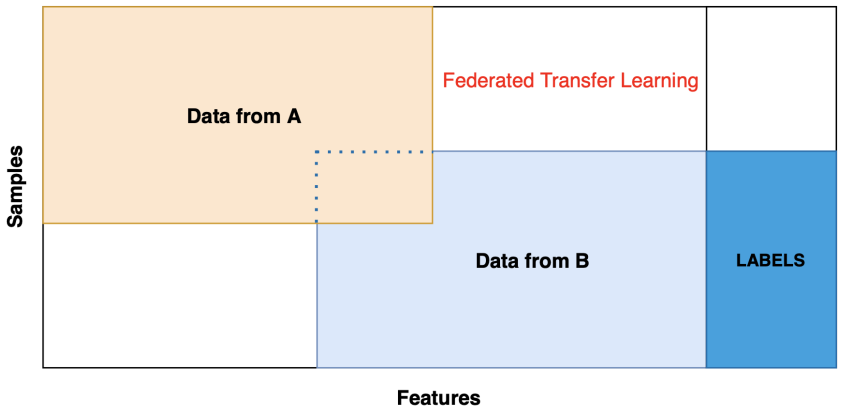

Figure 16: Federated Transfer Learning

9.2.4 Advanced Federated Aggregation Algorithm

In this project, we used use only two simple aggregation techniques for Federated Learning, which are FedAvg and FedAvgM respectively. Further Improvement can be achieved by changing the local model and the aggregation algorithm. The Federated Matched Averaging (FedMA) algorithm is a layer-wise federated learning algorithm designed for CNNs and LSTMs architectures. This aggregation algorithm accounts for permutation invariance of the neurons and permits global model size adaptation. The psuedocode for Federated Matched Averaging is given by:

Algorithm 1: Federated Matched Averaging (FedMA)

Input : local weights of N-layer architectures $\{W_{j,1}, \ldots, W_{j,N}\}_{j=1}^{J}$ from J clients
Output: global weights $\{W_1, \ldots, W_N\}$
$n = 1$;
while $n \leq N$ **do**
  if $n < N$ **then**
    $\{\Pi_j\}_{j=1}^{J}$ = BBP-MAP$\big(\{W_{j,n}\}_{j=1}^{J}\big)$; // call BBP-MAP to solve Eq. 2
    $W_n = \frac{1}{J} \sum_j W_{j,n} \Pi_j^T$;
  else
    $W_n = \sum_{k=1}^{K} \sum_j p_{jk} W_{jl,n}$ where p_k is fraction of data points with label k on worker j;
  end
  for $j \in \{1, \ldots, J\}$ **do**
    $W_{j,n+1} \leftarrow \Pi_j W_{j,n+1}$; // permutate the next-layer weights
    Train $\{W_{j,n+1}, \ldots, W_{j,L}\}$ with W_n frozen;
  end
  $n = n + 1$;
end

Figure 17: Federated Matched Averaging

The paper [27], which proposed the FedMA algorithm, illustrated that FedMA outperforms prior federated learning algorithms and can efficiently utilize well-trained local modals. The number of local training epochs E can affect the performance of FedAvg and sometimes lead to divergence. It is also observed that longer training benefits FedMA, which means that FedMA performs best on local models with higher quality. In contrast to FedAvg algorithm, longer local training leads to worsening overall accuracy. Thus, FedMA allows local clients to train their model as long as required. In addition, extensions of FedMA to improve Federated learning can be done by additional deep learning building blocks, such as residual connections and batch normalization layers.

References

[1] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018.

[2] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[3] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo, "Anomaly detection methods in wired networks: a survey and taxonomy," *Computer Communications*, vol. 27, no. 16, pp. 1569–1584, 2004.

[4] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, p. 12–22, Jul 2018. [Online]. Available: http://dx.doi.org/10.1109/MPRV.2018.03367731

[5] M. Ahmed, A. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 11 2015.

[6] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data," *Applications of Data Mining in Computer Security*, vol. 6, 02 2002.

[7] K. Heller, K. Svore, A. Keromytis, and S. Stolfo, "One class support vector machines for detecting anomalous windows registry accesses," 12 2003.

[8] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," 07 2004.

[9] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *Data Warehousing and Knowledge Discovery*, Y. Kambayashi, W. Winiwarter, and M. Arikawa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 170–180.

[10] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," 2017.

[11] B. Guo, L. Song, T. Zheng, H. Liang, and H. Wang, "Bagging deep autoencoders with dynamic threshold for semi-supervised anomaly detection," in *2019 International Conference on Image and Video Processing, and Artificial Intelligence*, vol. 11321. International Society for Optics and Photonics, 2019, p. 113211Z.

[12] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2017.

[13] R. A. Sater and A. B. Hamza, "A federated learning approach to anomaly detection in smart buildings," 2020.

[14] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Applied Sciences*, vol. 8, no. 12, 2018. [Online]. Available: https://www.mdpi.com/2076-3417/8/12/2663

[15] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent iot applications: A cloud-edge based framework," *IEEE Computer Graphics and Applications*, vol. PP, pp. 1–1, 05 2020.

[16] S. Saha and T. Ahmad, "Federated transfer learning: concept and applications," 2021.

[17] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[18] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," 2021.

[19] Y. Chen, J. Zhang, and C. K. Yeo, "Network anomaly detection using federated deep autoencoding gaussian mixture model," in *International Conference on Machine Learning for Networking*. Springer, 2019, pp. 1–14.

[20] K. Naveed, "N-baiot dataset to detect iot botnet attacks," Jan 2020. [Online]. Available: https://www.kaggle.com/mkashifn/nbaiot-dataset

[21] "Uci mchine learning repository: detection_of_iot_botnet_attacks_n_baiot data set." [Online]. Available: http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

[22] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length, and helmholtz free energy," *Advances in neural information processing systems*, vol. 6, pp. 3–10, 1994.

[23] Y. Ke and R. Sukthankar, "Pca-sift: A more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2. IEEE, 2004, pp. II–II.

[24] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[25] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[26] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*. PMLR, 2013, pp. 1139–1147.

[27] H. Wang, M. Yurochkin, Y. Sun, D. S. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *CoRR*, vol. abs/2002.06440, 2020. [Online]. Available: https://arxiv.org/abs/2002.06440

Appendices

A Federated Learning Hyperparameter Tuning

Table 11: Hyperparameter Tunning for MNP methods, Ecpoh=1

Epochs	Batchsize	Evaluation Metrics	Device								
			#1	#2	#3	#4	#5	#6	#7	#8	#9
1	16	F1 score	99.993	99.989	99.937	99.921	99.976	99.98	99.939	99.97	99.987
		TPR	0.99998	1	0.99984	0.99986	0.99995	0.99988	0.9996	0.9999	0.99994
		FPR	0.00702	0.04118	0.04463	0.02282	0.02582	0.01056	0.01547	0.04315	0.04455
1	32	F1 score	99.993	99.989	99.937	99.921	99.976	99.98	99.939	99.97	99.987
		TPR	0.99998	1	0.99984	0.99986	0.99995	0.99988	0.9996	0.9999	0.99994
		FPR	0.00702	0.04118	0.04463	0.02282	0.02582	0.01056	0.01547	0.04315	0.04455
1	64	F1 score	99.993	99.989	99.937	99.921	99.976	99.98	99.939	99.97	99.987
		TPR	0.99998	1	0.99984	0.99986	0.99995	0.99988	0.9996	0.9999	0.99994
		FPR	0.00702	0.04118	0.04463	0.02282	0.02582	0.01056	0.01547	0.04315	0.04455
1	128	F1 score	99.993	99.989	99.937	99.921	99.976	99.98	99.939	99.97	99.987
		TPR	0.99998	1	0.99984	0.99986	0.99995	0.99988	0.9996	0.9999	0.99994
		FPR	0.00702	0.04118	0.04463	0.02282	0.02582	0.01056	0.01547	0.04315	0.04455

Table 12: Hyperparameter Tunning for MNP methods, Ecpoh=5

Epochs	**Batchsize**	**Evaluation Metrics**	**Device**								
			#1	**#2**	**#3**	**#4**	**#5**	**#6**	**#7**	**#8**	**#9**
5	16	F1 score	99.993	99.989	99.937	99.921	99.976	99.98	99.939	99.97	99.987
		TPR	0.99998	1	0.99984	0.99986	0.99995	0.99988	0.9996	0.9999	0.99994
		FPR	0.00702	0.04118	0.04463	0.02282	0.02582	0.01056	0.01547	0.04315	0.04455
5	32	F1 score	99.993	99.989	99.937	99.921	99.976	99.98	99.939	99.97	99.987
		TPR	0.99998	1	0.99984	0.99986	0.99995	0.99988	0.9996	0.9999	0.99994
		FPR	0.00702	0.04118	0.04463	0.02282	0.02582	0.01056	0.01547	0.04315	0.04455
5	64	F1 score	99.993	99.989	99.937	99.921	99.976	99.98	99.939	99.97	99.987
		TPR	0.99998	1	0.99984	0.99986	0.99995	0.99988	0.9996	0.9999	0.99994
		FPR	0.00702	0.04118	0.04463	0.02282	0.02582	0.01056	0.01547	0.04315	0.04455
5	128	F1 score	99.993	99.989	99.937	99.921	99.976	99.98	99.939	99.97	99.987
		TPR	0.99998	1	0.99984	0.99986	0.99995	0.99988	0.9996	0.9999	0.99994
		FPR	0.00702	0.04118	0.04463	0.02282	0.02582	0.01056	0.01547	0.04315	0.04455