

# Robust Federated Learning Models for Intrusion Detection Systems

Thesis submitted  
in partial fulfillment of the requirements for the award of the

**Master of Technology**  
in  
Computer Science and Engineering  
(under the Dual-Degree Programme)

by

**Sai Prasath S**

17CS02002

Under the supervision of  
**Dr. Padmalochan Bera**



SCHOOL OF ELECTRICAL SCIENCES  
INDIAN INSTITUTE OF TECHNOLOGY BHUBANESWAR

May 2022

©2022 Sai Prasath S. All rights reserved.

## APPROVAL OF THE VIVA-VOCE BOARD

May 9, 2022

Certified that the report entitled “**Robust Federated Learning Models for Intrusion Detection Systems**” submitted by **Sai Prasath S** (17CS02002) to the Indian Institute of Technology Bhubaneswar in partial fulfillment of the requirements for the award of the Master of Technology in Computer Science and Engineering under the Dual-Degree Programme has been accepted by the examiners during the viva-voce examination held today.

*Rajat Subhra Chakraborty*

(Supervisor)

(External Examiner)

(Internal Examiner 1)

(Internal Examiner 2)

## CERTIFICATE

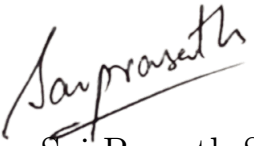
This is to certify that the report entitled "**Robust Federated Learning Models for Intrusion Detection Systems**" submitted by **Sai Prasath S** (17CS02002) to the Indian Institute of Technology Bhubaneswar is a record of bonafide research work under my supervision and the report is submitted for end-semester evaluation of the Dual Degree thesis work.

(Supervisor)

## DECLARATION

I certify that

1. The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.
2. The work has not been submitted to any other Institute for any degree or diploma.
3. I have followed the guidelines provided by the Institute in writing the thesis.
4. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
5. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
6. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.



Sai Prasath S  
17CS02002

# Abstract

Many industries today are increasingly adopting computerized systems to optimize their performance and increase their efficiency. The epicentre of this transformation has been the ability to leverage AI to analyze the large amounts of data generated by various systems for detecting attacks, forecasting demands, monitoring and controlling the system performance. With the ever-growing ability of AI models to perform at very high levels, many industries have been actively integrating their technologies with AI. One major drawback of AI systems is their need for large amounts of data, without which their performance is usually compromised. However, with growing concerns on the security and privacy of users, many companies worldwide are wary of sharing their system or customer information with any untrusted third party. Therefore, Traditional IDS systems that rely on centralized pooling of data and model training have suffered significantly from these security concerns. This research work proposes a Federated Learning-based IDS system capable of distributed training that ensures the security of the companies while at the same time being able to perform at high levels. We analyze the robustness of federated learning algorithms against Non-IID data, compromised clients, privacy invasions and model deployment attacks by empirically evaluating their accuracy levels on the NSLKDD dataset. Moreover, we study the advantages of using federated learning by comparing its performance with local self-training and traditional centralized training approaches. Further, we investigate various defense strategies including DQN based client selection, differential privacy, defensive distillation and adversarial training that will help secure the federated learning model against various attacks. Our results prove that FL models can achieve high-accuracy levels and robustness with minimal privacy concerns against most attacks with the help of the proposed defense strategies.

**Keywords:** Federated Learning, Intrusion Detection Systems, IID, NSL KDD, privacy invasion, differential privacy, DQN

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Introduction . . . . .	1
1.2	Related Works . . . . .	4
1.2.1	Traditional IDS Models . . . . .	4
1.2.2	Federated Learning based IDS . . . . .	5
1.2.3	Attacks on FL Models . . . . .	6
1.2.4	Defenses for FL Models . . . . .	7
<b>2</b>	<b>Background</b>	<b>10</b>
2.1	Types of Learning Processes . . . . .	10
2.2	Federated Learning Model . . . . .	11
2.2.1	Learning Algorithm . . . . .	11
2.2.2	Model Aggregation . . . . .	13
2.3	Reinforcement Learning Based DQN . . . . .	14
<b>3</b>	<b>Methodology</b>	<b>16</b>
3.1	Dataset Description . . . . .	16
3.1.1	Dataset Analysis . . . . .	16
3.1.2	Dataset Preprocessing . . . . .	17
3.2	Experiment Settings . . . . .	18
3.2.1	Neural Networks . . . . .	18
3.2.2	Federated Learning Parameters . . . . .	18
3.2.3	Entities Compared . . . . .	18
<b>4</b>	<b>Attacks in Federated Learning</b>	<b>20</b>
4.1	Local Data Attacks . . . . .	20
4.1.1	Varying Data Distribution . . . . .	20
4.1.2	Compromised Clients . . . . .	22
4.2	Privacy Invasion: Membership Inference . . . . .	25
4.3	Model Deployment Adversarial Attacks . . . . .	27

<b>5</b>	<b>Defenses in Federated Learning</b>	<b>29</b>
5.1	Deep Q Network for Client Selection . . . . .	29
5.2	Differential Privacy . . . . .	31
5.3	Adversarial Defenses . . . . .	33
5.3.1	Defensive Distillation . . . . .	33
5.3.2	Adversarial Training . . . . .	34
<b>6</b>	<b>Results and Discussion</b>	<b>36</b>
6.1	Performance on Local Data Attacks . . . . .	36
6.1.1	Varying Data Distribution . . . . .	36
6.1.2	Compromised Clients . . . . .	41
6.2	Defense against Data Attacks: DQN based Client Selection . . . . .	43
6.3	Membership Inference Attacks and Differential Privacy Defense . . . . .	45
6.4	Adversarial Attacks and Defense on Model Deployment . . . . .	47
<b>7</b>	<b>Conclusion</b>	<b>49</b>
<b>8</b>	<b>Future Work</b>	<b>51</b>
	<b>Bibliography</b>	<b>51</b>

# List of Figures

2.1	Different Types of Learning Process . . . . .	10
2.2	Federated Learning Process . . . . .	12
4.1	Varying Data Distribution . . . . .	21
4.2	Data Poisoning Attack . . . . .	23
4.3	Adversarial Attack: Fast Gradient Sign Method . . . . .	27
5.1	Deep Q Learning for Client Selection . . . . .	29
5.2	Defensive Distillation against Adversarial Attacks . . . . .	34
5.3	Gradient Masking: Adversarial Training and Defensive Distillation . . . . .	35
6.1	Results of IID . . . . .	37
6.2	Results of Non IID . . . . .	39
6.3	Performance on various attacks . . . . .	41
6.4	Weight Prediction by RL-DQN . . . . .	43
6.5	Membership Inference and Differential Privacy . . . . .	45
6.6	Adversarial Attacks and Defenses . . . . .	47



# Chapter 1

## Introduction

### 1.1 Introduction

Computerized systems have become the norm across multiple industries today. In general, computers monitor the data generated, resources consumed, and communications between various entities to detect patterns that can help forecast future demands, optimize the system performance and identify any inconsistencies in the systems. For example, smart grids use the SCADA system to monitor the grid performance, forecast future power demands, and detect any intrusions or attacks on the grid. In this research work, we study the ability of systems to proactively detect and report attacks on the systems to the system administrator; these systems are termed Intrusion Detection Systems.

Traditionally most of these IDS relied on statistical and mathematical models to track system behaviour and detect inconsistencies. Although these models were successful to an extent, they could not keep up with the exponential increase in the size and the complexities of industrial systems. Fortunately, the growth of computerization in industries coincided with the rise in AI/ML models that can analyze large and complex systems efficiently and understand the underlying relationship between the various system features. These AI models heavily rely on large datasets to tune their models to perform efficiently and detect attacks with high accuracies and low false alarm rates. Without access to enough data, the performance of the AI models will be severely compromised, and they

will not be able to operate at high-performance levels.

With growing concerns over the security and privacy of users and system information, many companies have been hesitant to share their data with untrusted third parties. For example, electric power companies worry that publicly sharing information of attacks on their power grids might reveal the vulnerabilities of their systems and expose themselves as potential targets in the future. On the other hand, these companies also need state-of-the-art IDS to protect their infrastructure against future attacks. Hence, AI-based systems that rely on centralized training by pooling data from multiple parties might not be sustainable due to privacy and security concerns.

Therefore, to efficiently address this issue, we introduce Federated Learning-Based IDS (FLIDS), which is capable of distributed training that will not compromise the privacy or the security of the systems while maintaining high-performance levels. Utilizing the ability of FL to train the models distributedly not only voids the privacy and security issues but also improves the quality of service provided by the IDS by improving their speed and optimizing all available resources. Unlike centralized training systems, FL models do not suffer from a single point failure and also have the potential to evolve and learn from their peers continuously. The federated learning process and its advantages are elaborately discussed in section [2.2](#).

This research work utilizes the famous NSL-KDD dataset to analyze the performance of the FL models. We compare the performance of three different learning models, namely: Centralized training, distributed training and federated learning. We also analyze the performance of FL models on non-IID data, compromised clients and privacy invasions, which are relatively common and can significantly impact the system performance. To handle these issues, we develop low cost defense strategies such as DQN based client selection, differential privacy and adversarial training that help minimize the effects of the attacks while also maintaining the integrity and performance level of the FLIDS. In

summary, the following are the contributions of this research work:

- **FL for IDS:** We present an Federated Learning based Intrusion Detection System that can distributedly train without compromising the privacy and security of the system. We compare the performance of our FL models with centralised training and self learning approaches.
- **Practical Study:** We study the FL model by analyzing its performance on Non-IID data, attacks initiated by compromised clients, privacy invasion during gradient transfer and model deployment attacks to determine their robustness by empirically evaluating their accuracy on the NSL-KDD dataset.
- **Low Cost Defenses:** In order to maintain the performance levels of the FLIDS against various attacks, we design multiple low cost defense techniques such as DQN based client selection, differential privacy, adversarial training and defensive distillation. We analyze the effectiveness of these techniques by comparing the model performance with and without these defense against various attacks.

## 1.2 Related Works

This section summarizes various traditional statistical and latest AI/ML models used as IDS. We also present recent works where federated learning models are used in the context of IDS. Moreover, we also discuss the various attack and defense strategies that have been use against FL models and highlight their strengths and weaknesses.

### 1.2.1 Traditional IDS Models

Most rule-based classifiers fail to detect complex attack patterns and intricate dependencies due to their relatively simple detection process; therefore, implementing such classifiers on large scale security systems is not efficient. For instance, Samaneh Rastegari et al. [1] tested multiple ruleset-based models, such as JRip, J48, MPLCS etc., on the NSL-KDD dataset and only achieved accuracies around 80%. Whereas ML models have gained popularity because of their ability to analyze vast amounts of data and detect attacks efficiently. To this extent, Decision Trees(DT), Support Vector Machines(SVM), Clustering algorithms, Random Forests(RF), Artificial Neural Networks(ANN) etc., [2] [3] have been extensively examined as potential solutions for IDS. All these models achieve very high accuracies and low false-positive rates, which are within the acceptable threshold, thus qualifying them as potential options for real-world implementations. Other strategies such as ensemble learning [4], feature extraction [5] and feature selection [2] have also been successfully utilized to improve the performance of the models.

Recently, despite the success of ML models, Deep Learning based solutions have been gaining traction primarily due to their ability to efficiently comprehend vast amounts of data and reliably understand the underlying representations. Deep Neural Networks(DNN) [6], Convolutional Neural Networks(CNN), and Recurrent Neural Networks(RNN - GRU and LSTM) [7] [8] have all been extensively analyzed and have been shown to perform better than ML models. Combining these architectures with denoising autoencoders [6], residual blocks [9] and attention mechanisms [6] has also been shown to increase the

model’s performance.

While ML and DL models have achieved high accuracy rates and low false positive values, they don’t preserve the security and privacy of user data which is of paramount importance. Whereas, Federated Learning models have the ability to protect user privacy through the use of distributed training algorithms.

### 1.2.2 Federated Learning based IDS

Most research on IDS focuses on improving the models’ performance metrics and does not often focus on preserving the security and privacy of the users and systems. In this subsection, we discuss some works that use federated learning models for IDS. Dapeng Man et al. [10] designed an FL based IDS for detecting intrusions for edge assisted IoT by utilizing attention-based CNN models and an intelligent model aggregation function using the normalized euclidean distance between the client model parameters and the server parameters as the attention weights for model aggregation. Rahman et al. [11] developed and compared the performance of centralized, local and federated training processes on the NSL KDD dataset and illustrated that the FL models were able to achieve centralized learning performances. Agarwal et al. [12] analyzes the various real-world problems and challenges faced while developing FL based IDS models including non iid dataset, high false alarm rates, and communication overheads. Zhao et al. [13] evaluated LSTM and CNN models in a federated setting on the SEA dataset and concluded that FL based LSTM model achieved accuracy and f1-score close to the centralized learning based LSTM model.

The above research works focus on the performance of the FL models, whereas in our research we focus on the robustness, privacy, security of the FL models and the performance trade-off involved while trying to achieve them.

### 1.2.3 Attacks on FL Models

There are multiple ways through which FL models are attacked, in this research work, we focus on non iid data distribution, adversarial clients, privacy invasion, and model deployment attacks.

Rahman et al. [11] compared the performance of FL models on iid and non-iid data and demonstrated that FL models fail to generalize when trained on non-iid data. Zhao et al. [14] studied the impact of non-IID data on the federated learning process by comparing various training parameters such as batch size and epochs on the overall model performance on the MNIST, CIFAR-10 and KWS dataset. They also explain the concept of weight divergence due to the skewed client data and propose a sharing technique to improve the overall model performance. Li et al. [15] study the effects of different aggregation algorithms(adding regularization to reduce the distance between client and server model, different number of local epochs for each client) and different data distribution(label skew, feature skew) by empirically comparing them on ten different datasets such as MNIST, CIFAR, SVHN, and FEMNIST.

Apart from non-IID data, the federated learning process also suffers from adversarial clients manipulating their shared model to affect the training process. Rey et al. [16] elaborate on various attack strategies including label flipping and model poisoning attacks that can compromise FL based IDS systems and also practically studied their effects on the malware detection capabilities of the DL model on the N-BaIoT dataset. Eugene et al. [17] exploited the non-iid distribution in a distributed setting using a constrain and scale strategy to perform model poisoning attacks where the actual server model is replaced by a poisoned model. The gradient updates sent to the server are crafted such that the aggregated model converges to the poisoned model. Zhang et al. [18] used two GANs, one for generating the local client distribution by using the model weights and the other to generate poisoned data for attacking the client model by generating samples that the first GAN model wrongly classifies.

Privacy invasion attacks focus on exploiting the privacy and security of the client models by trying to extract details about their local distribution. Zari et al. [19] monitored the changes in the class probability distribution across multiple update rounds in the FL training process and utilized a LSTM to differentiate between the members and non-members in the dataset. Nasr et al. [20] directly used the gradient norms from the model updates for performing membership inference attacks. They also analyse the impact of number of layers, gradient norm, training size and prediction uncertainty on the membership inference task. Ali et al. [21] used gradient matching loss and autoencoder to generate data samples belonging to the local client distribution by minimizing the distance between the gradient updates of the true samples and the generated sample.

Model deployment attack on federated learning models is similar to general adversarial evasion attacks. Goodfellow et al. [22] proposed an iterative gradient based adversarial sample generation algorithm when the input samples are updated to increase the loss of the neural network. Similarly, Papernot et al. [23] proposed a clever pixel-by-pixel based adversarial sample generation algorithm that'll wrongly classify the input sample with minimal changes to the actual input by selecting the pixel with the largest gradient first. Other than these attack, Jere et al. [24] and Bouacida et al. [25] proposed a robust taxonomy for classifying the various attacks and vulnerabilities of federated learning models and also discussed real-world issues faced by the model.

Most of the attacks discussed above are implemented for computer vision tasks, whereas in our research work, we extend a subset of the above discussed attacks to the context of IDS and empirically evaluate their performance on the NSLKDD dataset.

#### 1.2.4 Defenses for FL Models

Protecting the privacy and security of the clients while maintain the high performance levels of the server model is essential for any successful FL system. In this subsection,

we discuss various defense strategies through which the above described attacks can be neutralized.

One of the significant steps in the federated learning process is model aggregation, where the client models are combined to create the server model for the next update round. Choosing the right clients can help defend the FL models against non-iid data and adversarial clients. Sannara Ek et al. [26] compared various model aggregation strategies such as the use of personalized layers in the local client model and independent training of model layers and evaluated their performance on the REALWORLD human activity recognition dataset. Further, Wang et al. [27] proposed an RL based FL aggregation technique that uses a DQN model to select the top-k clients using their model weights for the model aggregation process by rewarding validation accuracy and penalizing communication rounds. They demonstrate a significant improvement in the model’s performance and reduction in the communication rounds on the FashionMNIST and CIFAR datasets. In our research work, we develop a low cost DQN inspired by [27] for selecting our clients for model aggregation.

Privacy invasion attacks can be defended through the use of homomorphic encryption and secure multi-party computation protocols. Graepel et al. [28] proposed “ML Confidential” where machine learning algorithms are implemented over encrypted data such that the privacy and security of the local client is intact. Moreover, Badavi et al. [29] used a fully homomorphic encryption scheme to train an NLP model on encrypted data that matched the performance of state of the art models. However, their model took almost 5 days to be fully trained. In our research work, to develop low cost defenses we utilize the differential privacy technique proposed by Abadi et al. [30]. They modify the gradient used for model updates by clipping and introducing noise to remove any client data information.

Goodfellow et al. [22] introduced the adversarial training technique where adversarial



samples generated from the dataset are used in the model training with their correct labels. Moreover, Papernot et al. [31] proposed the defensive distillation technique where the models are trained using the soft labels instead of hard labels. Both the above technique leads to gradient masking where the gradients across the decision boundaries are reduced making it difficult to perform adversarial attacks.

In our research work, we extend these defense strategies to federated learning models in large scale security systems. We evaluate the performance of our FL models before and after these defenses and empirically evaluate their capabilities by using the NSLKDD dataset.

# Chapter 2

## Background

### 2.1 Types of Learning Processes

In this research work, we study three different types of learning processes, namely: Centralized learning, distributed learning(on device/self learning), and federated learning [11].

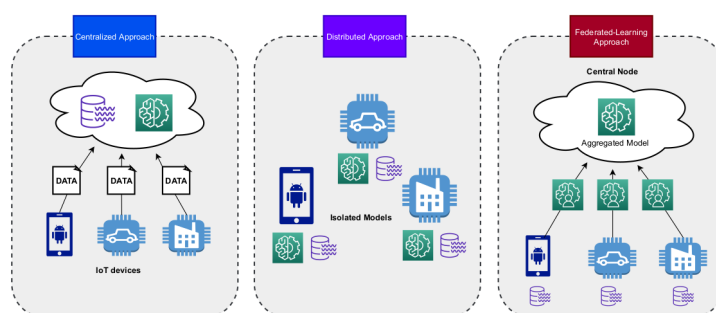


Figure 2.1: Different Types of Learning Process

- **Centralized Learning:** In this learning process, all the local client data is shared to a central server. All the data storage, model creation, and training happen at the central server, and the trained models are shared with the clients. Although the model's performance is high due to a large amount of data available for training, the learning process compromises the privacy and security of the clients as they share raw information with the server. The learning process also has a single point of failure at the server; therefore, the shared data and model performance are vulnerable if the server is compromised.

- **Distributed Learning:** The clients build their model and train them on their local data in this learning process. The clients do not have access to their peer's data or model, and therefore their learning process is independent of each other. As different clients have different data distribution, some clients have high performing models, whereas others do not. The clients suffer from a lack of data for training and data imbalance as well. As the clients don't share their model or data with anyone else, this learning process maintains both the privacy and the security of the clients.
- **Federated Learning:** The clients do not explicitly share their data in this process, but they do share their trained models with the server, which ensures that knowledge can be pooled at the server without compromising the users' security and privacy. The learning process removes the single point failure of centralized learning and improves the quality of service by increasing the speed of detection through distributed training and aggregation. Due to the distributed nature of the learning process, there is no need for large storage or high computational requirements at the central server.

## 2.2 Federated Learning Model

### 2.2.1 Learning Algorithm

The federated learning process consists of seven main steps:

1. The server generates a generic intrusion model where a neural network architecture is built. At this stage, the number of hidden layers neurons, epochs, and so on, are identified.
2. The model is downloaded by the nodes that wish to use the model, whether contributing or not in the FL process.
3. The selected nodes keep their local data private and use it on-device to enhance the studied model.

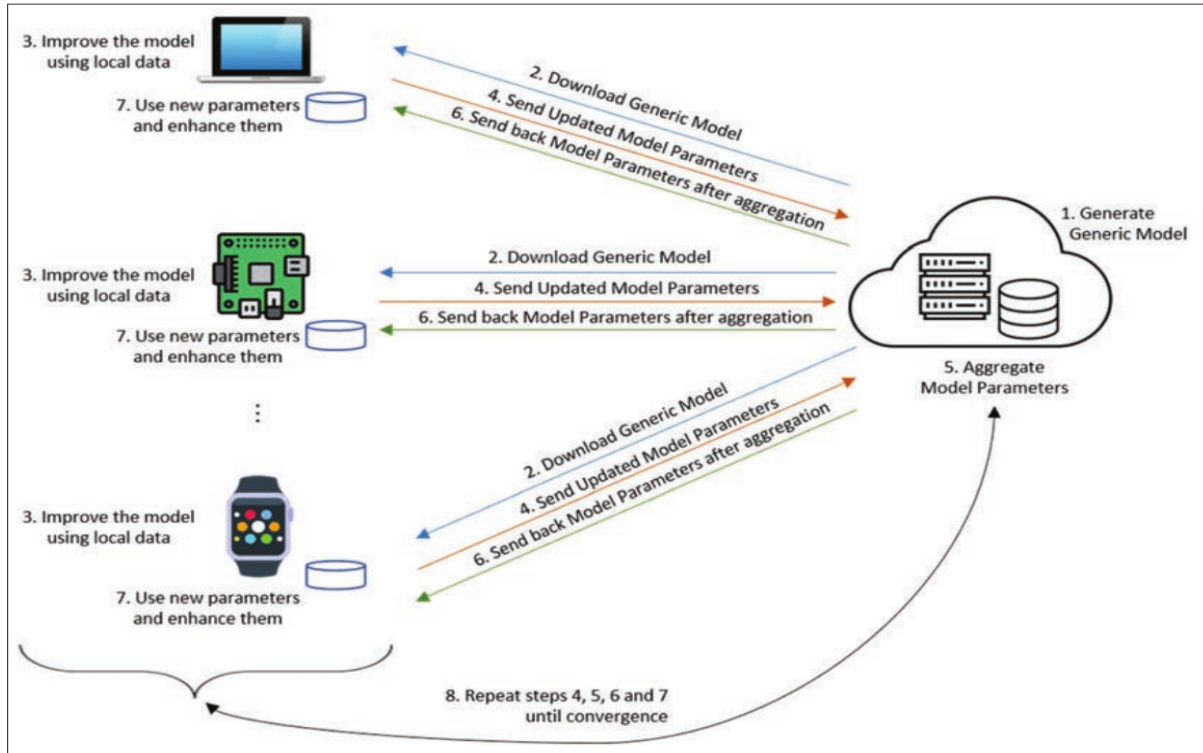


Figure 2.2: Federated Learning Process

- Only the model parameters of the updated intrusion detection model are shared with the central server instead of sending sensitive data and intruding on the privacy of the nodes.
- Once all the updates are received, the server aggregates the weights from the different node models and creates a new updated model. For the aggregation, the FederatedAveraging algorithm is used, in which the parameters are weighted based on the nodes dataset size.
- The server pushes back the updated model parameters to the nodes.
- Each node uses the updated model parameters and improves them based on its generated data.

**Advantages of the FL process:** Federated Learning algorithms are distributed, meaning there is no single point of failure that compromises the whole system, and they can scale efficiently with increasing network size without compromising the integrity of the system. Federated learning algorithms also leverage the computing power of edge devices

by performing their training on them which significantly speeds up the training process and ensures that the central server does not need large storage or high computational requirements. Most importantly, they preserve the privacy and the security of the user information while simultaneously sharing knowledge with their peers.

### 2.2.2 Model Aggregation

After every update round, the server receives a set of neural network weights from all the clients. These client weights are aggregated using certain functions to create the server model for the next iteration, this process is called as Model Aggregation. There are multiple techniques to perform this aggregation, however in this research work we consider the Federated Average(FedAvg) algorithm. The FedAvg algorithm uses the number of data points used by the client to train its model to weight the client model during the aggregation process.

$$\min_{\theta \in R^d} f(\theta) = \sum_{i=1}^n \frac{|n_i|}{N} * F_i(\theta) \quad (1)$$

$$F_i(\theta) = \frac{1}{n_i} \sum_{j \in S_i} f_j(\theta) \quad (2)$$

$$N = \sum_{i=1}^n |n_i| \quad (3)$$

The above equations 1,2 and 3 summarize the functions of the model aggregation process. In equation 2,  $f_i(\theta)$  is a loss function that needs to be optimized by every client by training their models on the local data. In equation 2,  $F_i(\theta)$  is the optimized client model that is shared to the central server. In equation 1, the central server then aggregates the client models such that their combined loss is minimized. To minimize the loss the FedAvg algorithm weights the client models by the number of data points they used to train their local model.

The above algorithm 1 summarizes the federated learning communication and the aggregation process described in sections 2.2.1 and 2.2.2. Note that during the server

---

**Algorithm 1:** Federated Averaging: The  $K$  clients are indexed by  $k$ ;  $B$  is the local minibatch size,  $E$  is the number of local epochs,  $\eta$  is the learning rate, and  $C$  is the fraction of clients selected for each update round.

---

**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

---

execution, we select a random set of  $m$  clients from the available set to optimize the learning process by reducing the communication costs and reducing the time consumed in the process.

## 2.3 Reinforcement Learning Based DQN

Deep Q Learning models leverage the ability of deep neural networks to analyze large datasets with complex relationships between the features to make decisions for a given environment. The DQN take a state vector as their input, which represents the current state of the system, and predict the output or action that have to be taken at this junction. The actions predicted by the DQN models are carried out, and the whole system moves to a different state. Depending on their actions, the DQN models are rewarded. If their actions lead to a favourable state, they get positive feedback; otherwise, they get negative feedback. This feedback trains the neural network to understand the relationship between the input state vector and the possible actions. This process is carried out multiple times until the DQN models learn to make the right decisions given an environment state.

Unlike the supervised learning models the targets for a DQN model aren't fixed or pre-defined as the models keeps learning the rewards and probability distributions of  $p(s', r|s, a)$  the more it explores the state space. Hence, at each time step the optimal actions given a state keeps changing. Therefore, the parameters of the DQN model are optimized using a Temporal Difference loss function, where the model waits till time step  $t + 1$  to validate the actions taken by the model at time step  $t$ .

$$loss = (r + \gamma * \max_{a'} \hat{Q}(s, a') - Q(s, a))^2 \quad (4)$$

The Temporal Difference(TD) Loss function used by the DQN models is described in equation 4. In equation 4,  $r$  is the reward at time  $t$ ,  $Q(s, a)$  is the predicted  $Q$  values at time  $t$ , and  $\hat{Q}(s, a)$  is the  $Q$  values predicted at time  $t + 1$ . The target for the actions taken at time  $t$  are computed at time  $t + 1$  by choosing the action that maximizes the total target. The parameter  $\gamma$  is used to discount the future rewards, as the rewards we receive at time step  $t$  is more valuable than the rewards we receive at time step  $t + k$ .

On top of the DQN model there are other variants such as Double DQN [32] and Generative Memory Replay [33] that helps in stabilizing the training process and improving the performance at the cost of more storage and computational capacity, but in our research work we limit our experiments to the DQN model due to limited resource constraints at the client and server levels.

# Chapter 3

## Methodology

### 3.1 Dataset Description

#### 3.1.1 Dataset Analysis

The KDD-CUP99 dataset is a widely used dataset for testing systems developed to detect computer network traffic anomalies. This dataset contains many records in different attack types. However, in the studies performed on this dataset, it has been determined that there are some cases that adversely affect the performance of the systems tested in the dataset. To solve this problem, it has been suggested to use a new dataset called NSL-KDD to test the proposed systems, eliminating some records in the KDD CUP99 dataset. In the NSL-KDD dataset, the unnecessary samples from the dataset to be used for training are cleared and the size of the dataset is set to a reasonable value for the anomaly detection.

The following table summarizes the attack distribution in the train and test set. The train set consist of 23 different types of attacks but the test set consists of 38, some of which are day 0 attacks meaning the ML model wasn't trained on these attacks. All the data points are divided into 5 major categories namely Normal, DDOS, Probe, U2R and, R2L for multi-class classification task.

Further analysis of the dataset reveals that both the train and the test set share 21 different attack types which constitutes 99.29% and 83.36% of the datapoints in their



<b>Dataset</b>	<b>Normal</b>	<b>DOS</b>	<b>Probe</b>	<b>U2R</b>	<b>R2L</b>
<b>Training</b>	67,343	45,927	11,656	52	995
<b>Test</b>	9,711	7,458	2,241	67	2,887

respective datasets. Two attacks are exclusive to the train dataset which accounts for the remaining 0.71% of the training dataset and seventeen attacks are exclusive to the test dataset accounting for the remaining 16.64% of the test dataset.

To remove the imbalance between the train and the test dataset, we combine and shuffle them before we split them 80-20 train-test for the rest of the research work. We refer to the newly generated data as train and test set in the rest of the paper. This process ensures that there aren't any unknown bias or inconsistencies in the dataset by completely eliminating them if they did exist.

### 3.1.2 Dataset Preprocessing

The raw dataset consists of 41 features values for each datapoint describing the various network parameters. Three('protocol-type', 'service', 'flag') out of the fourty one values are categorical and all else are continuous in nature. We convert all the categorical values to continuous by applying the one-hot encoder technique. This results in a dataset with 122 features. All the features are then scaled between 0-1 using min-max normalization to help the models learn the pattern easily.

Note that we do not perform any feature selection/reduction algorithms due to changing nature of the attack patterns in the long term, which might affect the relative importance of the features across various attacks. Thus we expect our models to learn the importance of features by themselves. The importance of features describing an attack might also vary across time considering the non-stationary nature of the attack domains. Therefore, as the features' importance changes through time and across attacks, we abstain from implementing any selection/reduction algorithms. The continuous nature of the federated learning process ensures that all the involved parties adapt to the changing nature of the attacks.

## 3.2 Experiment Settings

In this section, we present the different parameters and models used for the experiments performed in this research work.

### 3.2.1 Neural Networks

We develop a six-layer deep neural network with an input size of 122, followed by [128, 80, 50, 25, 10, 5] nodes in the following layers. We use a dropout layer with 0.2 between all the subsequent layers. We use a relu activation function for all the layers except the final one, where we use a softmax function as the model predicts the likelihood of an attack for a given data point. There are 31,704 parameters in the model that are tuned using an adam optimization function with a learning rate of 0.01.

### 3.2.2 Federated Learning Parameters

We consider 10 clients for all the experiments proposed in this research work. The batch sizes during the training process is fixed at 512. The clients go through 20 rounds of update and subsequently communicate their learnt parameters to the server. Each round consists of 5 epoch with the length of the local client dataset varying depending on the experiment. We use the FedAvg algorithm for the model aggregation process.

### 3.2.3 Entities Compared

In this research work, we compare the performance of four entities across all the experiments. We cover all the three types of learning process to study the relative performance of the federated learning process with respect to the other algorithms. The following are the entities being studies:

- **Local Client:** This is the performance of the client that trains only on its local dataset, meaning it follows a distributed/self-learning process.

- **Federated Client:** This measures the performance of the local clients in federated learning setting right before they share their models with the central server. Therefore, the performance of the client models is computed on the client-side.
- **Federated Server:** This measures the performance of the server in a federated learning setting. The performance is computed right after the client models are aggregated.
- **Centralized Server:** This is the performance of the server that is trained in a centralized learning process, meaning they have access to all the local client data.

# Chapter 4

## Attacks in Federated Learning

During the distributed training of the federated learning models various entities involved in the process can affect the final outcome of the federated learning model. In this chapter, we study four major problems that can compromise the training process of the FL models.

### 4.1 Local Data Attacks

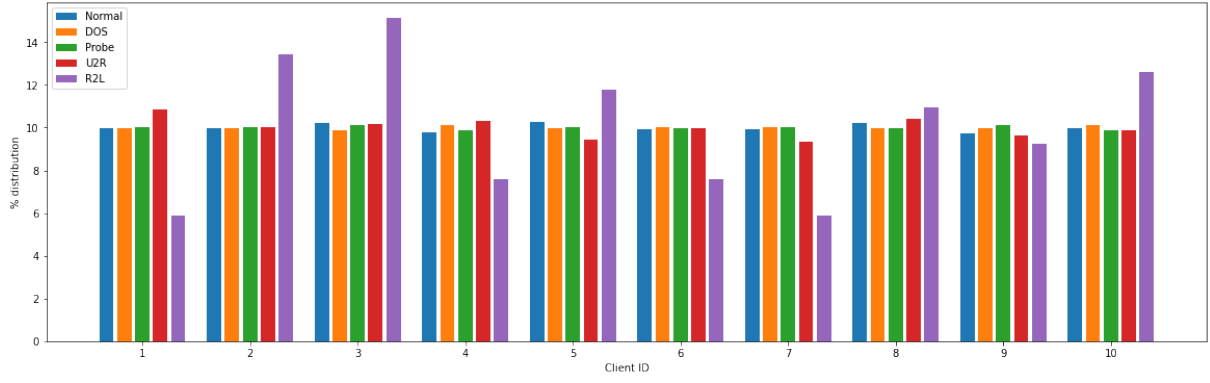
In this section, we study the performance of the federated learning algorithm on different types of dataset distribution. For this research work, we consider the Independent and Identically Distributed (IID) data and non-IID data for the analysis.

#### 4.1.1 Varying Data Distribution

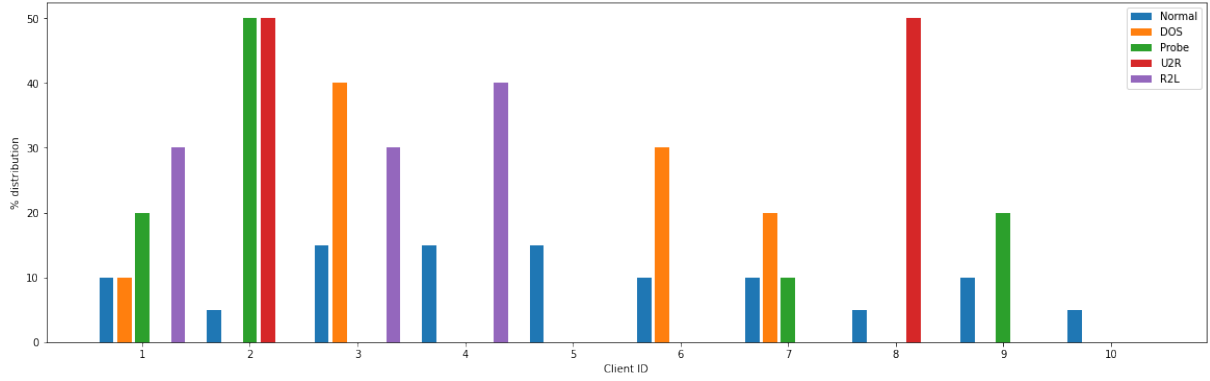
##### **IID**

The train set of the NSL-KDD dataset is uniformly divided amongst 10 different clients. We ensure that all the clients receive all the different attack types in equal proportions.

From Fig.4.1a, we can observe that all the ten clients receive similar data distribution. Many research works have previously shown that federated learning algorithms perform best in IID scenarios as the goals of the central server and the local clients align with one another. Therefore, the knowledge gained by the clients through local training can



(a) IID Data



(b) Non-IID Data

Figure 4.1: Varying Data Distribution

efficiently help the central server to perform better, thereby showcasing the potential of federated learning systems.

## Non IID

Unlike IID distributions, where all the clients receive similar data distributions, the Non-IID scenario reflects real-world situations where some clients have all the data, and the others do not have any. In our experiments, we have created specific distributions to ensure that some clients suffer from a lack of data and others suffer from an imbalance in their local dataset. In real-world scenarios, as the clients are distributed, they encounter different attack patterns and data distributions depending on their environment.

From Fig.4.1b, we can observe that different clients have different distributions and that no two clients have similar datasets. In Non-IID distribution, client 5 and 10 only has normal data points in their local dataset, whereas client 1 and 2 have a mixture of

normal and other attack data points. This is possible because specific clients might not be attacked in a real-world setting because of which they do not have any attack data points. Despite the unequal data distribution, note that all the clients have access to at least some normal data points in their local dataset, as they cannot always be under attack.

Most federated learning models fail to effectively perform on Non-IID datasets as the goals of the local client, and the server are not the same due to the skewed nature of the local dataset. For example, client 10, which only has normal data, can achieve high accuracies by always predicting the data points as normal, but the server has to ensure that all the attacks are detected efficiently as the server also caters for clients 1 and 2 who have a mixture of attack data points. Therefore, the federated learning model is expected to perform poorly on non-IID distributions.

### 4.1.2 Compromised Clients

In the federated learning setting, the server trusts that the clients share their actual models without manipulating them. However, some local clients can be compromised by adversaries, and their performance can be severely affected. In this section, we discuss two major attacks on federated learning models:

#### Dataset Poisoning Attacks

In this type of attack, the dataset with which the client trains its local model is manipulated either by adding malicious samples or by changing the values in the existing data. The label flipping process affects the alignment of the client and the server goals. Figure 4.2 illustrates the process by which the data can be poisoned. By changing the labels in the dataset of the poisoned clients, the data distribution changes; therefore, the weights learnt by the poisoned clients is significantly different when compared to normal clients. Therefore, when the models are aggregated at the server, the combined weights of the client models starts to diverge from the weights of the poisoned clients. If enough clients are compromised, this can lead to significant deterioration on the performance of

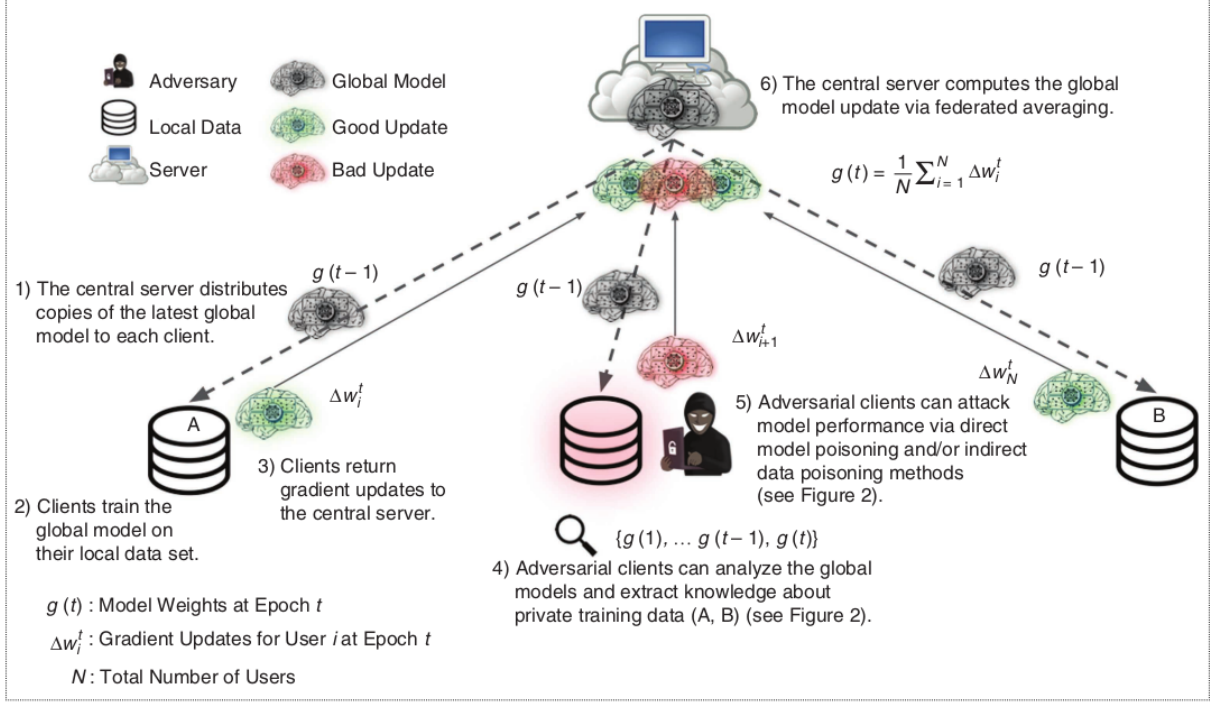


Figure 4.2: Data Poisoning Attack

the server model. In summary, the difference in goals of the poisoned clients and the central server leads to a weight divergence problem, where the weights of the neural network trained by the client and aggregated by the server are significantly different.

In this research work, we perform label flipping attacks where all the attack data points are labelled as  $["1", "2", "3", "4"] \rightarrow "4"$  before the client trains its model. We consider 20% of the clients to be compromised, that is 2 out of the 10 client models.

### Model Manipulation:

In the model manipulation attack, the datasets are not directly targeted, but the models learnt by the clients are manipulated by multiplying the weights with a negative constant; this ensures that the weights of the models cancel each other out and the model is not capable of performing the assigned task. Compared to data poisoning attacks, model manipulation attacks are much easier to perform as there is no need for affecting the local dataset of the clients. Moreover, unlike data poisoning attacks, model manipulation attacks will surely affect the performance of the client models significantly due to the direct

changes in weights.

We can consider the model manipulation attacks as a form of Man-in-the-middle(MITM) attacks where an attacker captures the model shared by the client by trapping the communication channel and altering the model parameters without both the client and the server knowing. Therefore, the accuracy of the federated clients remains high as it is computed before the model is shared, but the overall model after aggregation suffers from a lack of performance. Therefore, due to the clever nature of the attack the client doesn't realize that it's under attack. If the attacker chooses to attack clients with high accuracy rates, the aggregated client models performance will decrease further.

In this research work, we consider 20% of the shared models to be manipulated by an adversary, that is 2 out of the 10 models. The adversaries, manipulate the weights by multiplying them with "-2". The manipulated models are then shared with the server directly. Choosing the magnitude with which the model weights are being multiplied with is really important. A high magnitude means a huge deviation in the aggregation but also makes it easier to detect the poisoned models. On that other hand, small magnitudes might be difficult to detect but does not affect the model's performance significantly.



## 4.2 Privacy Invasion: Membership Inference

While the local data attacks originate at the client level, privacy invasion attacks usually happen during the model weight/gradient communication process between the client and the server. Due to the distributed nature of the federated model’s training process, any attacker can tap into the communication between the client and the server and download all the model parameters and gradient updates. By using this information an attacker will be able to identify whether or not a particular data point was used for training the local client model. These attacks are termed as “Membership Inference Attacks” and can compromise the privacy and security of the individuals whose data is being used for training the models. Although federated learning models secure the privacy of the user data, they are not fully secure and can be exploited by targeted attackers.

**Attack Setting:** We assume that the attacker has access to a set of data  $D_a$  and is interested in targeting client  $i$  to determine the datapoints used by client  $i$  for training its local model. We consider the local client dataset as  $D_c$  and the intersection between the datasets  $D_a \cap D_c \neq \emptyset$  is not null. We also assume that the distribution of  $D_a \setminus D_c \sim D_c$ . In our research work, we download the model parameters  $\theta_c$  of client  $i$  during communication to the server and store the prediction probability of the correct class  $\theta_c(x_j)[k]$  for all  $x_j \sim D_a$  where  $k$  is the actual class of  $x_j$ .

When client models optimize their model parameters they tend to overfit on their local dataset, and therefore do not generalize to other similar data points. Moreover, the gradient updates used for optimizing the parameters store information about the noises in the local data distribution through which the membership of a datapoint can be effectively identified. In our research work, we collect the correct class probability of all the data points in  $D_a$  across multiple update rounds of the federated learning training process. We label all the data points in  $D_a$  that belongs to  $D_c$  as “1” meaning member and other data points as “0” meaning non-members.

We use this labelled dataset of members and non-members and treat them as a binary classification problem. In our research work, we utilize a random forest classifier with  $max\_depth = 10$  and  $n\_classifiers = 20$  to fit this data with a 80-20 train-test split. Once we build our binary membership classifier, we can use the same to classify further data points by observing their probability predictions over multiple update rounds.

### 4.3 Model Deployment Adversarial Attacks

Once the federated learning model has been trained and their performances evaluated, they need to be deployed for real-world utilization. However, attacks can still exploit the loop holes in the federated learning model to significantly affect their performances. One such important attack is the adversarial attacks. In this attack, the attackers exploit the decision boundaries of the deep neural networks by adding small but unrecognizable perturbations to the input datapoints to change the output of the neural networks. Assuming the attackers have access to the model parameters by tapping the communication between the clients and the server, it is safe to assume a white box setting for the attackers.

$$x^{adv} = x + \epsilon * \text{sign}(\nabla_x J(x, y_{true})) \quad (5)$$

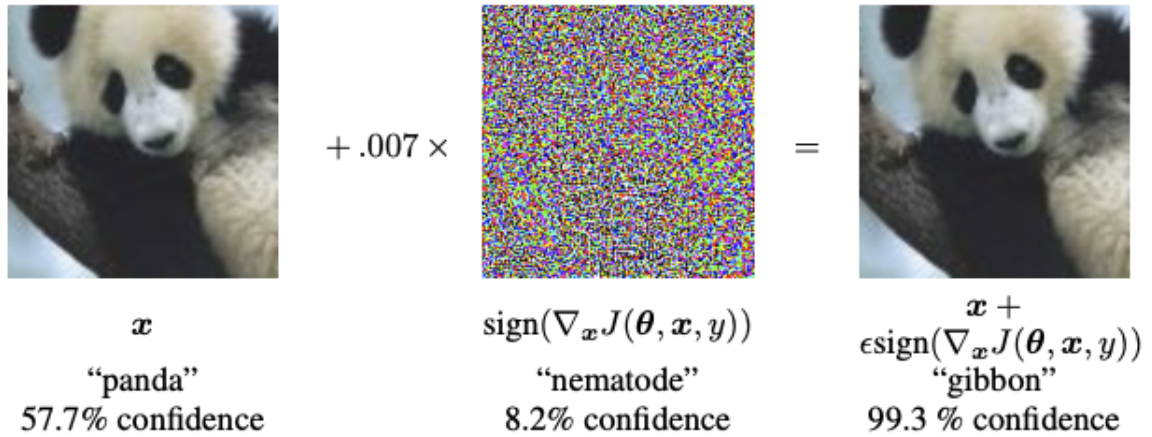


Figure 4.3: Adversarial Attack: Fast Gradient Sign Method

While there are many techniques through which the models can be attacked, in our research work we consider the simple yet efficient Fast Gradient Sign Method(FGSM). During the model training process, the parameters are updated in the opposite direction of their gradients to minimize the loss function. However, in case of FGSM we update

the input parameters in the same direction of their gradients so that the loss is increased and the data points are wrongly classifier.

From equation 5, we can observe that for each data point we update the value by  $\epsilon$ . Larger the value of  $\epsilon$  easier to wrongly classify the data points but creates a lot of distortion and lower the value of  $\epsilon$  more difficult to wrongly classify but lesser the distortion.

Figure 4.3 is a common example of the FGSM attack on a image classifier CNN model. Initially the image  $x$  is correctly predicted as a “panda” with a 57.7% class accuracy. However, by performing the FGSM attack, where we add carefully crafted noises through the use of  $\epsilon$  parameter such that the input  $x$  is wrongly classified. Note that in this case, not only is the input wrongly classified, but is also classified with a high class probability of 99.3% for a wrong class. While the FGSM algorithm is common in computer vision tasks, we extend them to the domain of intrusion detection and cyber security in general.

In our research work, we perform adversarial attacks on the aggregated client models, i.e the server model at the end of each update round. We evaluate the effectiveness of the adversarial attack by measuring the number of samples wrongly classified after adversarial updates from the combined datasets of all the clients. The combining of datasets at the central server is ideally not possible in real-world scenarios, but this can be viewed as each client trying to create adversarial samples on the server model. In this research work, we consider the  $\epsilon$  value as 0.05.

# Chapter 5

## Defenses in Federated Learning

### 5.1 Deep Q Network for Client Selection

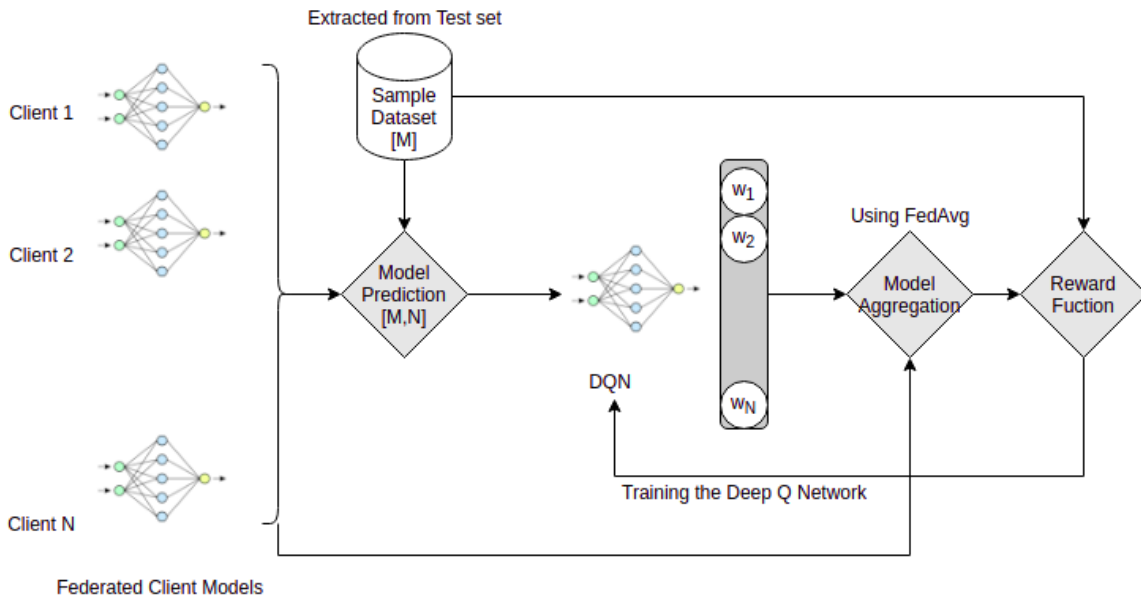


Figure 5.1: Deep Q Learning for Client Selection

The models aggregated by the server aren't always the best because of the non-IID distribution and compromised clients. Therefore, directly aggregating the models without checking their performance will lead to poor federated learning models. Moreover, in real-world scenarios there are millions of available devices to choose from for each round out of which only tens of thousands are selected. Therefore, it is important to choose

clients efficiently such that the aggregated model performs well.

To address this problem, we propose a RL based DQN models that helps in choosing the best clients for each update round. The model takes in the client predictions on a dataset sampled from the test set and outputs the weight distribution that needs to be considered while aggregating the client models. In our research work, we only consider the top  $k$  clients and reassign the weights of the chosen clients such that their sum is equal to 1. Using the aggregated model we compute its accuracy on the sample dataset which serves as the reward and feedback for optimizing the parameters of the DQN model. The whole process of state generation, weight prediction, model aggregation and reward computation is detailed in Figure. 5.1. The three major components of the RL model are:

- **State:** Most research works that implement RL based models for client selection directly considers the model weights as their input state. Although this might be an effective solution, the size of the model parameters can be very large and therefore the whole process can be time and resource consuming. Hence, instead of using the models directly, we use the predictions of the model on a small sample of data as our input state. Hence, in our proposed DQN model we consider the state to be the predictions of our  $n$  clients on a dataset of size  $m$  sample from the test set.
- **Action:** The DQN model takes the model predictions as the input and outputs a vector of size  $n$ (number of available clients) which we use as the weights assigned to the corresponding client model. In this experiment, we only consider the top  $k$  models for the aggregation process, all the other models are ignored by the server.
- **Reward:** Once we compute the aggregated model, the model predicts the outputs for the sample subset that was used during the state generation process. The reward is the product of the one hot encoded target and the output vector of the aggregated model, which measures the strength of the predictions. The goal of the DQN is to maximize this value.

## 5.2 Differential Privacy

---

**Algorithm 2:** Differentially private SGD

---

**Input:** Examples  $\{x_1, \dots, x_N\}$ , loss function  $\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}(\theta, x_i)$ .

Parameters: learning rate  $\eta_t$ , noise scale  $\sigma$ , update rounds  $T$ , gradient norm bound  $C$

Initialize  $\theta_0$  randomly

for  $t \in [T]$  do

**Compute gradient**

    For each  $i \in L_t$ , compute  $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, x_i)$

**Clip gradient**

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

**Add noise**

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L} (\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$

**Descent**

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output  $\theta_T$

---

Differential Privacy techniques help in addressing the privacy invasion attacks such as membership inference. The major reason for privacy invasion attacks is the attacker's ability to download the client model during the communication between the client and the server. Usually techniques such as secure multi-party computation and homomorphic encryption can encode the model parameters before they are transmitted such the even if the attacker interferes they won't be able to gain any useful information. However, such techniques require large storage and computational requires and are therefore expensive and resource intensive. Considering that we are building a federated learning model, we need to optimize our resources while trying to achieve high accuracy levels.

We efficiently solve this problem by using differential privacy algorithms. While we are not able to encode and decode the model parameters before and after transmission, we can try to reduce the attacker's ability to differentiate between member and non-member data points by interfering with the local client model training process. As explained in section 4.2, the reason for the success of privacy invasion attacks is the over fitting problem during local client training. By reducing over fitting we can ensure that the models

generalize better so that members and non members can't be distinguished easily.

The differential privacy algorithm is described in Algorithm 2. Once the gradients are generated for optimizing the model parameters, we clip the gradients and add a random gaussian noise so that the gradients updates don't reflect the exact data points from which they were generated. We utilize the modified gradients for updating the model parameters, thereby ensuring better generalization and reduced over fitting.



## 5.3 Adversarial Defenses

In this section, we discuss two major techniques through which the effect of adversarial attacks on deployed models can be minimized. Note that, unlike centralized training approaches, in case of federated learning the server model does not have access to the client data. Therefore, any defense that requires client data must originate from the client itself and not from the server. In accordance with this, we study two different adversarial defense techniques namely defensive distillation and adversarial training for protecting the models.

### 5.3.1 Defensive Distillation

Adversarial samples exploit the hard decision boundaries of the neural network to generate noises that helps in wrongly classifying an input. The hard decision boundaries of the neural network are majorly created due to the training process where one-hot encoded vectors are used as the targets for the input sample. These one-hot encoded vectors force the neural network to classify each input sample to only one class and not to a probability distribution, which leads to sudden changes in decision boundary.

Using a probability distribution instead of one-hot encoded vectors have shown to create smoother decision boundaries which makes generating adversarial samples more difficult. By not forcing the neural network model to output one-hot encoded vectors, we improve the model's flexibility which makes it less susceptible to attacks and more robust.

Figure 5.2 outlines the process of defensive distillation. Initially we train the model using the one-hot encoded targets. This model will have hard decision boundaries and will be prone to adversarial attacks. After training, we get the output for the training data which will be a probability distribution across all classes. Using the newly obtained probability distribution as the target for the input samples, we train a different neural network once again. The neural network trained using the probability distribution instead

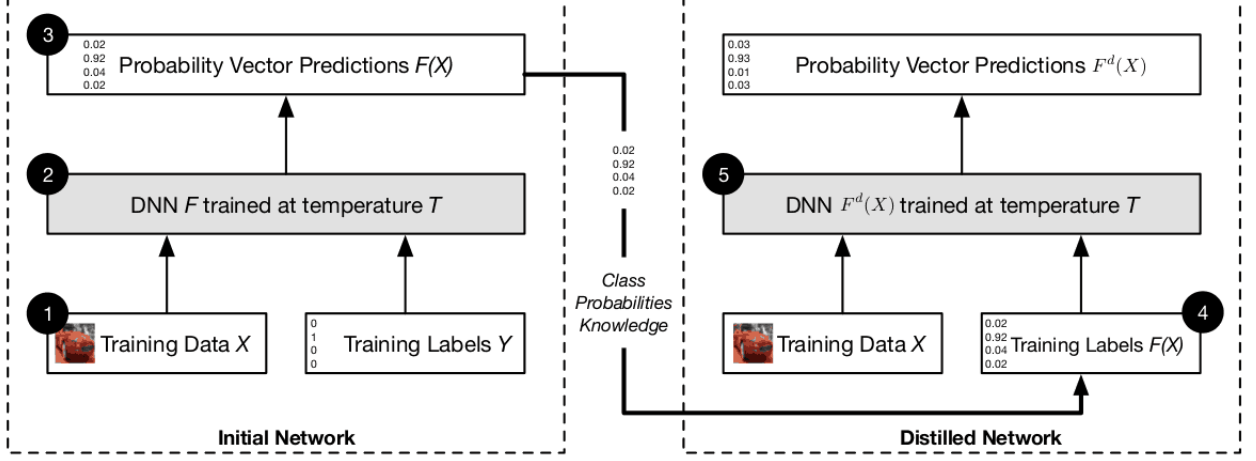


Figure 5.2: Defensive Distillation against Adversarial Attacks

of the one hot encoded vectors will have smoother boundaries and be more robust against adversarial attacks.

In supervised learning techniques, to protect the model against adversarial attacks, we directly perform the defensive distillation process on the model because of availability of the training data. However, in case of federated learning the central server does not have any access to the training data of the local clients, therefore, the defensive distillation process must originate from the local client itself. Hence, in our research work, the clients perform the defensive distillation process before sharing the model with the central server.

### 5.3.2 Adversarial Training

Another popular technique used to defend models against adversarial attacks is adversarial training. Unlike defensive distillation where the models are trained twice, once on hard labels and once on soft labels, in case of adversarial training we train the models only once but increase the size of the dataset to include adversarial samples in them. As adversarial samples are usually located cross to the decision boundaries and are wrongly classified by the models, adding the adversarial samples to the dataset and labelling them with the correct labels before training the model will lead to more robust decision boundaries which makes generating adversarial samples more difficult.

In our research work, we use the Fast Gradient Sign Method (FGSM) described in section 4.3 for generating the adversarial samples that will be included in the dataset before training. Moreover, the labels of the adversarial samples is the sample as the input sample from which the adversarial sample was generated. Similar to the defensive distillation technique where the defense originates from the local client, even in case of adversarial training the defense originated from the local client as the server does not have access to the local training data of the client.

At each update round we train the local client models using adversarial samples. After the model aggregation at the central server, we measure the accuracy of the server model on the actual test dataset and the adversarial accuracy of the server model by generating adversarial samples from the test dataset.

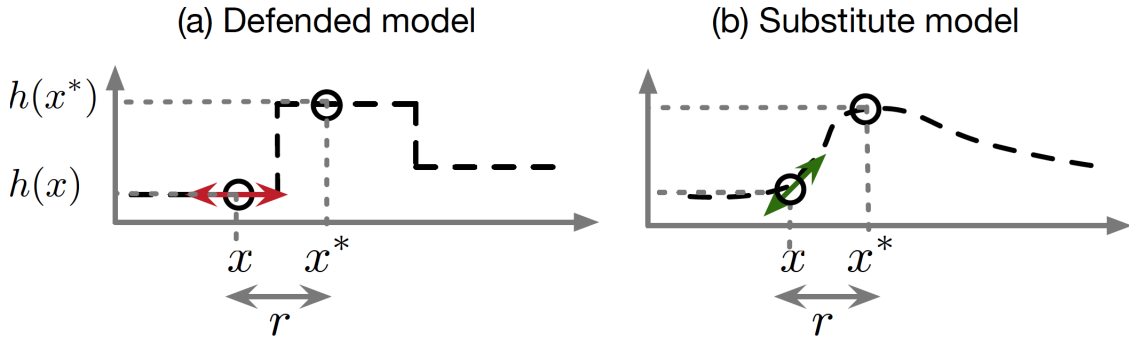


Figure 5.3: Gradient Masking: Adversarial Training and Defensive Distillation

Papernot et al. [34] concluded that adversarial training and defensive distillation techniques result in gradient masking at the decision boundaries. This phenomenon is illustrated in figure 5.3 for a one dimensional model. After applying the adversarial defense techniques, the gradient at the boundaries of the models becomes smooth making it difficult for the attacker to generate adversarial samples with reduced gradients.

# Chapter 6

## Results and Discussion

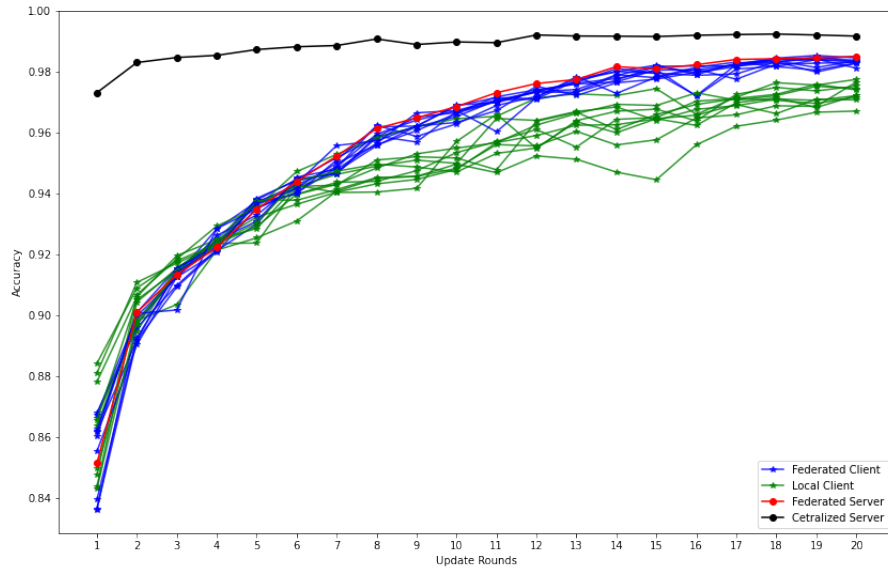
### 6.1 Performance on Local Data Attacks

#### 6.1.1 Varying Data Distribution

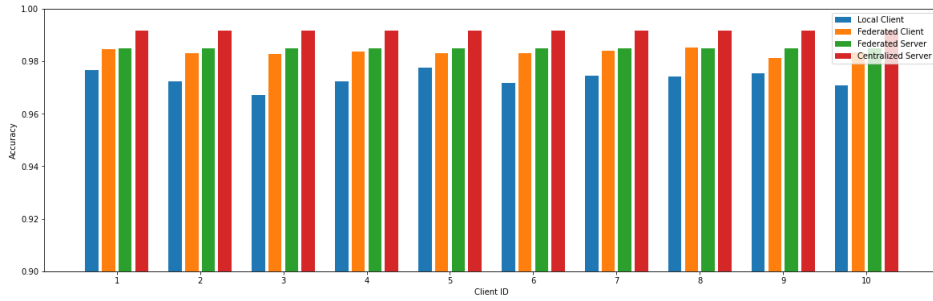
##### IID

In this section, we analyze the results of the federated learning model on the IID data. We compare the accuracies, recall and the f1 score of normal and attack data points to estimate the performance of the FL models. We also compare 4 different entities namely: local client, federated client, federated server and a centralized server; the definitions of these entities is explained in section 3.2.3.

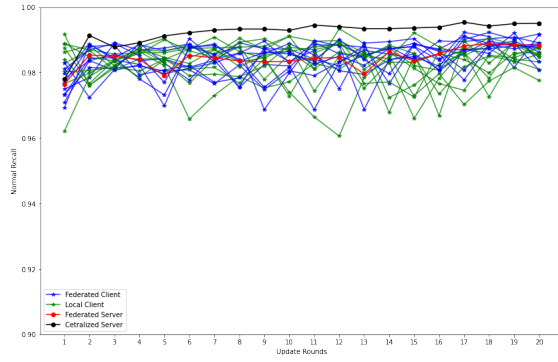
From fig [6.1a](#) we can observe the accuracies of the model across multiple update rounds. Initially the centralized server performs much better than all the other models especially because it has access to all the local client data. However, federated and local clients also improve their accuracies with each update rounds. The federated client model lags behind the local client initially, but over time improves its accuracy, because of sharing their information and knowledge with their peers indirectly through models without compromising their privacy. As the local datasets of the local clients are IID, all the clients are trying to solve the same problem, therefore the performance of the federated



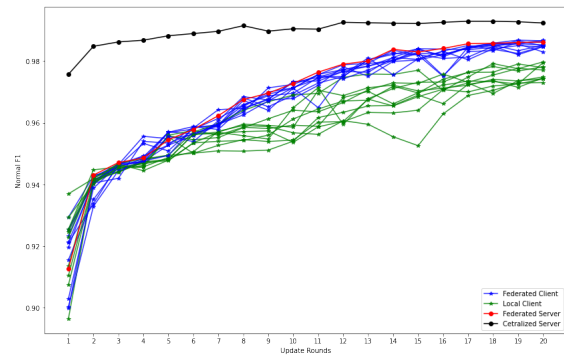
(a) IID Overall Accuracy



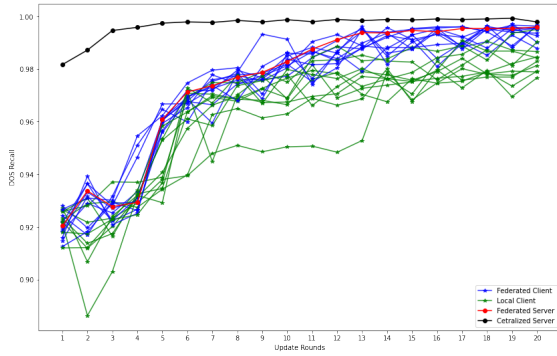
(b) Final Accuracy IID



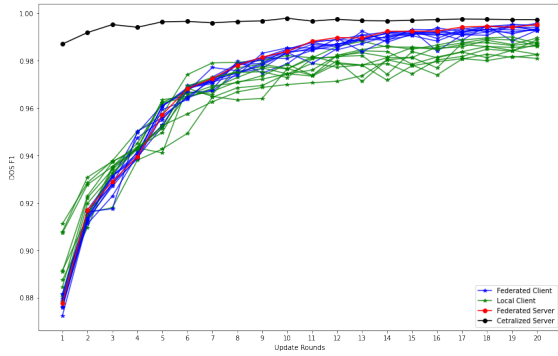
(c) Normal Recall



(d) Normal F1 Score



(e) DOS Recall



(f) DOS F1 Score

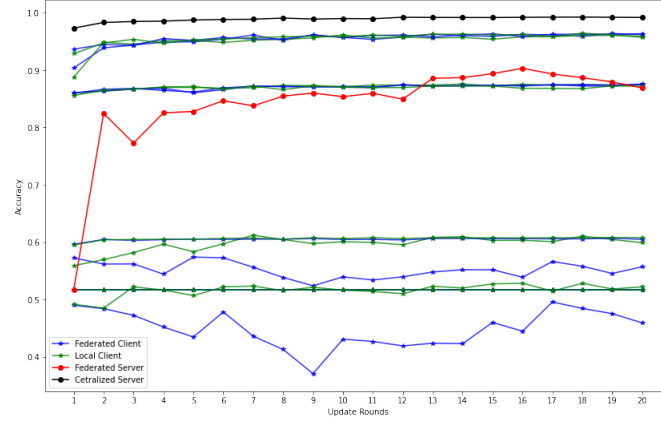
Figure 6.1: Results of IID

client is improved when the local client models are aggregated. From fig 6.1b we can see that, at the end of 20 update rounds, the difference between the performance of federated learning and distributed learning is clear as there is a 3-4% gap between the accuracies. Note that, all the federated learning clients outperform their corresponding self trained model. Also, there is a gap between the federated learning and the centralized learning process, as the centralized learning model has access to all the data directly whereas the federated model needs to get the same information indirectly through parameter sharing.

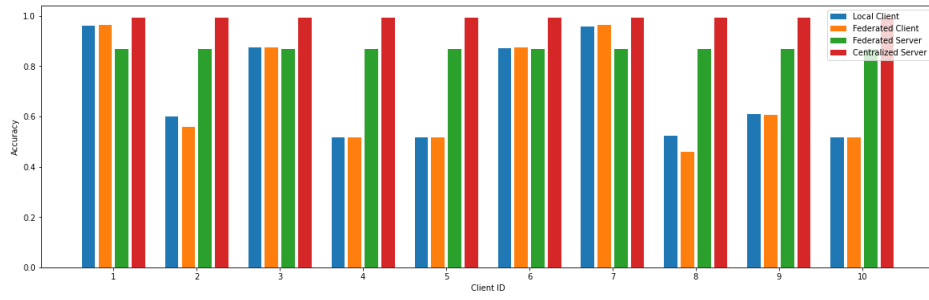
Fig 6.1c and fig 6.1d compare the recall and the f1-score of the clients on their ability to detect normal data points. From these figures we can observe that the recall rate of the federated clients are similar to that of the local clients and are marginally lesser than the centralized server. On the other hand, the normal f1-score of the federated clients is better than the local clients, meaning that their precision is better. This is because of the fact that federated clients share information with their peers that help each other improve their model performance. Similar observations can be made from figures 6.1e and 6.1f which compared the recall and the f1-score of the DOS attacks. In this case, the federated clients almost achieve the recall and the f1 of the centralized server and clearly outperform the local clients in the process. The ability of the federated clients to achieve high f1 scores in detecting DOS attacks while maintaining the privacy of their data is significant.

## Non IID

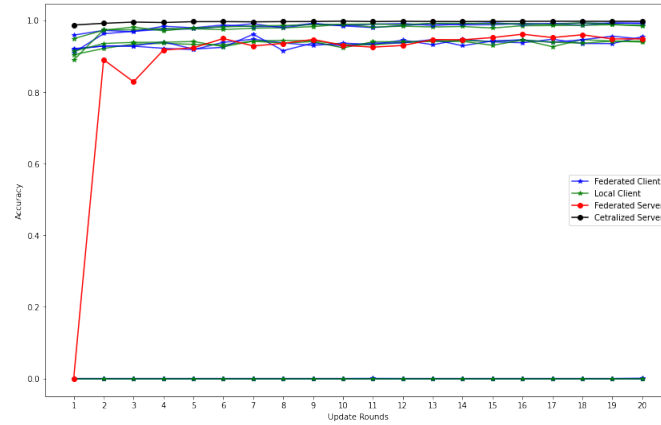
From figure 6.2a, we can observe that, unlike the IID case, the accuracy of the federated client and the server models do not reach high values because of the difference in the data distribution. Due to the variation in their data distribution, the clients and the server are trying to solve different problems that cause their weights to diverge. From fig 6.2b, we can see that some local clients can outperform the federated server, whereas some clients perform much worse than the federated server. Especially clients that do not have a good mix of data suffers from low accuracy rates.



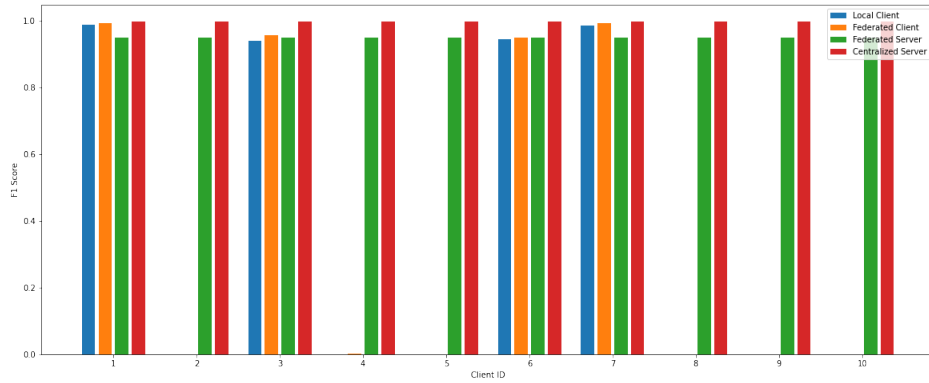
(a) Non IID Overall Accuracy



(b) Final Accuracy Non IID



(c) DOS F1 Score



(d) Final DOS F1 Score

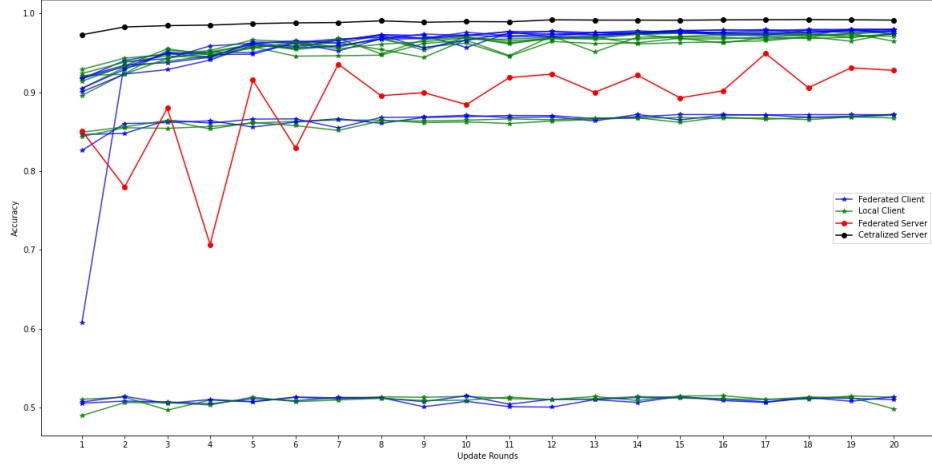
Figure 6.2: Results of Non IID

The advantage of federated learning models is that local clients who have not encountered specific attacks will still detect them as the federated server model shared with them was combinedly developed by multiple other clients. From fig 6.2d, we can observe the ability of the clients to detect DOS attacks by measuring their f1-scores. While some clients perform similar to the high performing centralized server, some federated clients have a 0 f1-score, meaning they were not able to identify DOS attacks because they haven't encountered any DOS attack. Clients 2, 4, 5, 8, 9, and 10, all have never encountered a DOS attack, despite that due to the federated learning algorithm they'll receive the federated server model that can detect DOS attacks with an f1-score around 0.9. Therefore, federated learning models are able to help each other out and identify day-0 attacks efficiently without compromising the security of their local data.

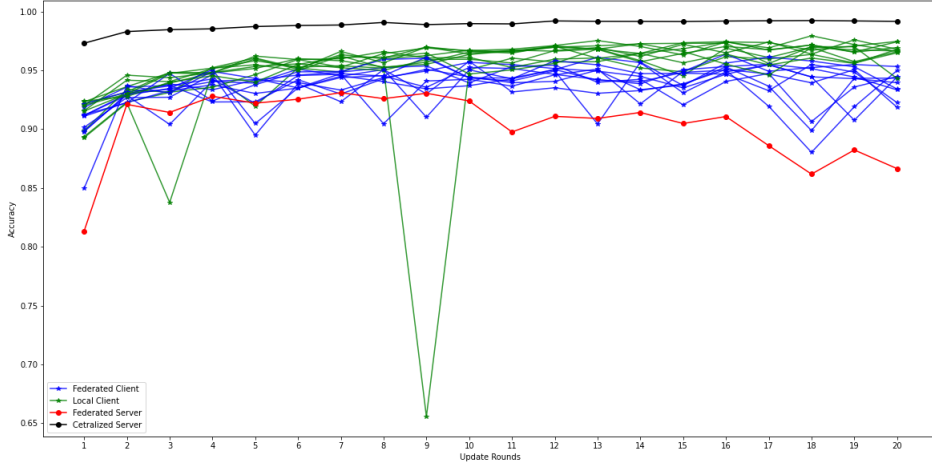


### 6.1.2 Compromised Clients

In this section, we discuss the performance of FL models on data poisoning and model manipulation attacks.



(a) Data Poisoning Attack



(b) Model Manipulation Attack

Figure 6.3: Performance on various attacks

### Data Poisoning Attacks

From figure 6.3a, we can observe that the two clients that are being attacked by adversaries have significant decline in their performance. Interestingly, the poor performance affects two other clients as well, because of which the federated clients aren't able to perform as well as the federated server model. All the other clients perform as expected at high levels, but due to the poor performance of four clients, the overall performance of the

federated learning model is compromised. Attacking only two clients, reduces the model accuracy by 8-9%, therefore by increasing the number of clients that are being attacked, we can cause significant damages to the FL model.

### **Model Manipulation Attacks**

From fig 6.3b, we can see that the performance of the federated server model has decreased by 15-16%. Notably, as we use two clients and “-2” as the model manipulation constant, the attack cancels out four other models. Along with the two models already under attack, more than six models are effectively compromised. Therefore, this attack is much more powerful than the data poisoning attacks. Moreover, as more than half the clients have been compromised, the federated clients perform poorer than their corresponding local clients. In order to increase the extent of the damages, we can either increase the number of clients under attack or increase the model manipulation constant.

## 6.2 Defense against Data Attacks: DQN based Client Selection

In this section, we analyze the results of the RL based DQN model described in section 5.1 which is used for client selection during the model aggregation process. The DQN model helps in choosing the top  $k$  clients from the available  $n$  clients such that the performance of the aggregated model is maximized.

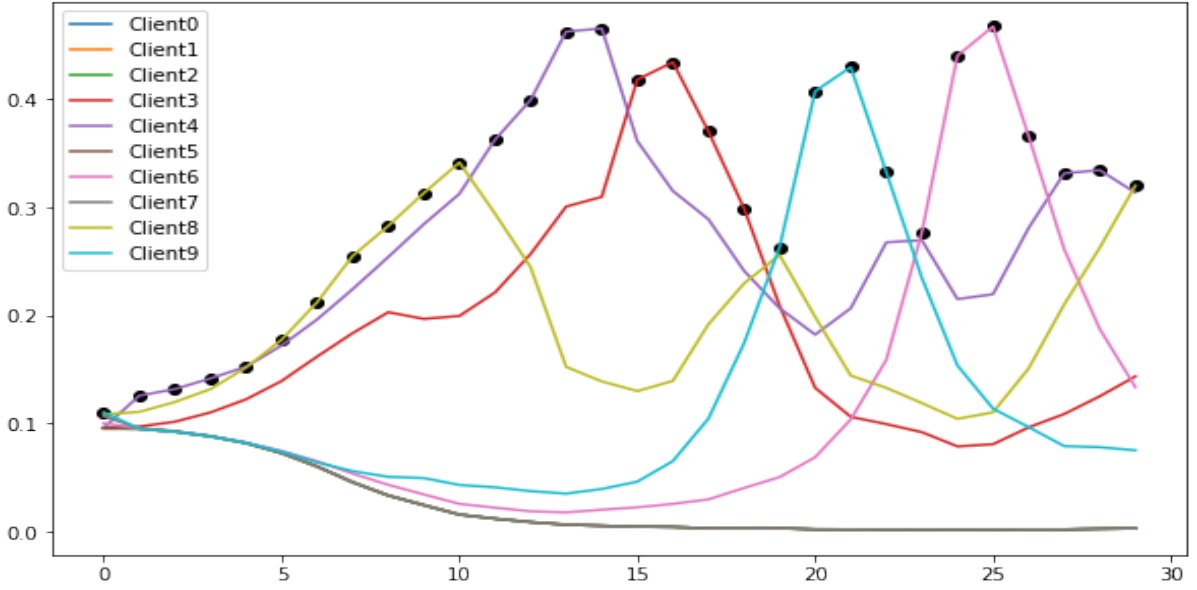


Figure 6.4: Weight Prediction by RL-DQN

In our research work, we consider  $n = 10$  and  $k = 1$  for all the 30 update rounds. We perform data poisoning attacks on clients  $\{0, 1\}$ , model poisoning attacks on clients  $\{2, 5\}$ , and non-IID data for client  $\{7\}$  across all the 30 update rounds. From figure 6.4, we can observe that clients  $\{3, 4, 6, 8, 9\}$  has high weights assigned to them by the DQN but other clients  $\{0, 1, 2, 5, 7\}$  are assigned very low weights which are close to 0. As we only consider the top-1 client (highlighted by black dots in fig 6.4) at each time step, we choose the client with the highest weight allotted to it at each update round. Therefore, across the update rounds all the clients that aren't under attack are considered equally during the model aggregation as they all have similar performance levels. In order to reduce the bias and variance of the aggregated model, we don't rely on a single client,

rather we select all high performing clients randomly. In our implementation, we select clients  $\{3 \rightarrow 4, 4 \rightarrow 10, 6 \rightarrow 4, 8 \rightarrow 7, 9 \rightarrow 5\}$  times in 30 update rounds. Further due to the dynamic client selection, the average accuracy of the server model increases by 5.8% from 90.23% to 96.03%. In conclusion, the RL based DQN for client selection can be really helpful for dynamically selecting the high performing clients by ignoring clients with low performance levels or the ones under attack.

## 6.3 Membership Inference Attacks and Differential Privacy Defense

In this section, we discuss the effectiveness of membership inference attacks and how differential privacy and help solve the problem. As introduced in section 4.2 membership attacks try to differentiate between the members and non-members of a local client given a sample dataset.

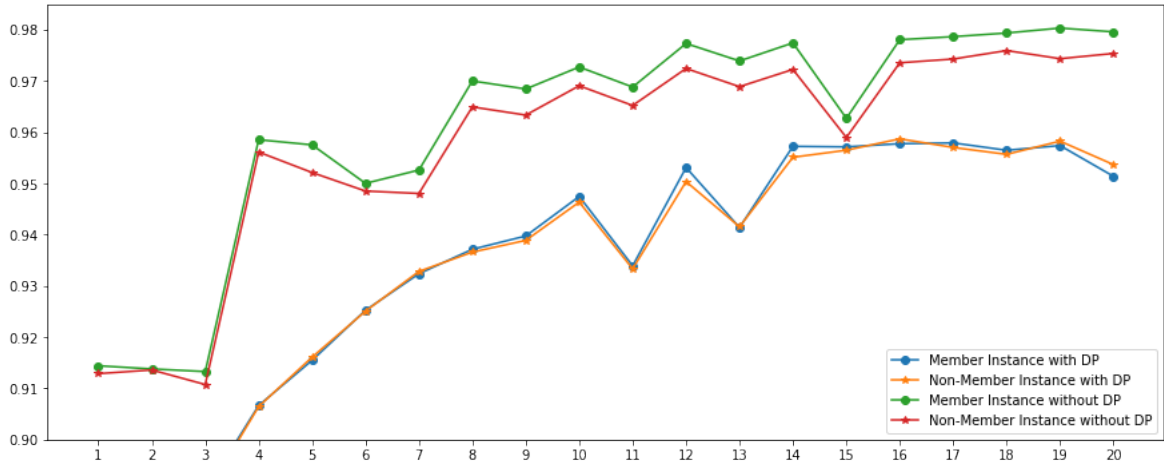


Figure 6.5: Membership Inference and Differential Privacy

Figure 6.5 tracks the average probability assigned to the correct label for members and non-members in a dataset, with and without differential privacy(DP) defense. We can clearly observe that the green and red lines(tracking the probability without DP) diverge with more update rounds. This occurs because with more update rounds and more epochs, the models over fit the member data points whereas the non-members aren't, which leads to a deviation between the probability assigned to them. However, in the orange and blue lines(tracking the probability with DP) we can observe that there is no deviation between the member and non-member data points. The DP technique removes all the information regarding the local client training data from the gradients before they are used for updating the model. This ensures that the model does not memorize anything related to the local training data(member data points), and therefore not creating a difference between the member and non-member probabilities. Although

the DP technique addresses the membership inference problem, it comes at the cost of reduced prediction probability. Without DP, the prediction probability was 0.97-0.98 for the data points, but after DP it falls by 0.3-0.4 to 0.94-0.95. The loss in the prediction probability is because of the addition of noise and clipping of the gradient which restricts the capability of the neural networks to learn more from the data.

Using membership inference techniques, we can accurately identify the membership of 78.34% of the data points in the training dataset. However, with the help of differential privacy, this accuracy drop by 22.81% to 55.53%. Considering that the models are trained on IID datasets, differential privacy only allows an additional 5.53%(above 50%) to be correctly identified, thereby reducing the impact of this privacy invasion technique.

## 6.4 Adversarial Attacks and Defense on Model Deployment

In this section, we evaluate the performance of adversarial attacks and the ability of various defenses such as defensive distillation and adversarial training.

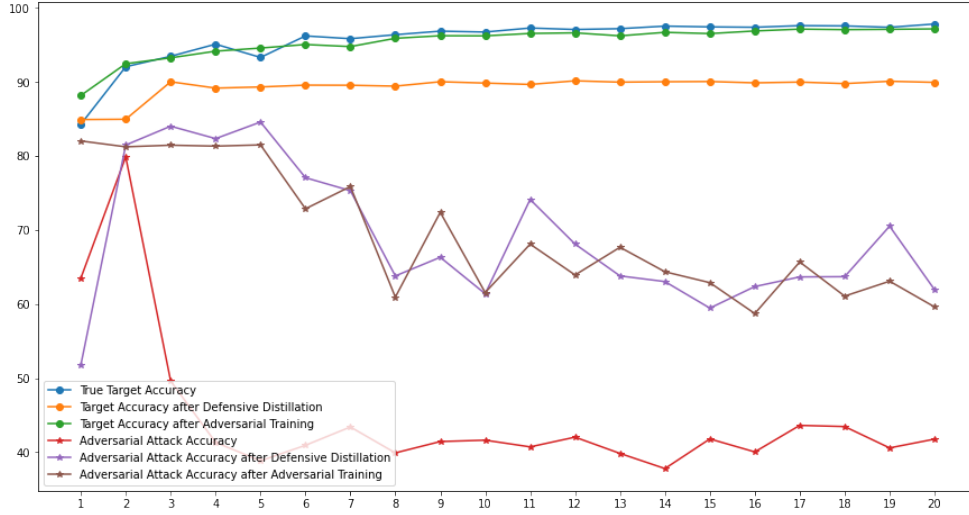


Figure 6.6: Adversarial Attacks and Defenses

To evaluate the performances of the attacks and defenses, we measure the accuracy of the model on the intrusion detection classification task using the test dataset and the adversarial accuracy of the models which measures the performance of the model on the generated adversarial samples from the test dataset. The FGSM attack as described in 4.3 achieves an adversarial accuracy of 44.58% and the target accuracy of the model remains unchanged at 95.73% as the target model is not modified. Therefore, 55.42% of adversarial samples are wrongly classified by the model even with a very low  $\epsilon$  of 0.05.

From figure 6.6, analyzing the defensive distillation and adversarial training defenses, we observe that the model achieves an adversarial accuracy of 68.93% and 69.03% respectively. The defense techniques have improved the adversarial accuracy by around 25% which is a significant increase. However, the target accuracy of the model after implementing defensive distillation techniques falls by 6.41% to 89.32%, where as the target accuracy after adversarial training only falls by 0.29% to 95.44%. Therefore, in this re-

gard the adversarial accuracy technique is superior to defensive distillation as they both achieve the same adversarial accuracy, but the adversarial accuracy technique is able to maintain its target accuracy unlike the defensive distillation technique. Further, from a resource point of view, the adversarial accuracy technique is better because generating adversarial samples using a one step process is much cheaper than training two neural networks. In conclusion, adversarial training technique is a better defense mechanism against adversarial attacks than defensive distillation.



# Chapter 7

## Conclusion

In this research work, we studied the applications of federated learning on intrusion detection systems in the context of the NSL-KDD dataset. By comparing the performance of centralized training, distributed training, and federated training, we illustrate that FL models can perform at high levels while simultaneously maintaining the privacy and security of the users. Along with the accuracy of the models, we compare their recall rates and f1-scores across normal and attack points to analyze their performance in depth. To understand the shortcomings and vulnerabilities of the FL model, we implement various attacks and empirically evaluate their performances. We show that FL models perform poorly on non-IID data distribution due to the difference in goals of the local and the global model but can share valuable day-0 attack detection knowledge with their peers. Moreover, we test the robustness of the federated training process by comparing its relative performance against data poisoning and model manipulation attacks and determine that FL models are vulnerable to such attack strategies. Further, we exploit the distributed learning process of the federated learning models by performing membership inference attacks while violates the privacy of the clients involved in the process. We also show that FL models are not robust against common adversarial attacks such as FGSM during the model deployment phase. For efficiently solving the problems faced by FL models, we propose three major defense techniques. Firstly, to overcome the problems of poor client performance, we implement an RL based DQN capable to selecting clients dynamically based on their performances. Secondly, we implement differential privacy techniques to

protect the client from privacy invasion attacks and finally analyze adversarial defense strategies such as defensive distillation and adversarial training so protect the FL models from adversarial attacks during deployment. We empirically show how these defense techniques can help alleviate the problems caused by various attacks and also highlight the drawbacks and resource constraints of these techniques.

# Chapter 8

## Future Work

Possible direction for future work:

1. The FL models can be deployed in real-world scenarios/test beds to measure their actual potential and also learn the drawbacks of the model(if any) in a real world setting in terms of communication delay and resources consumed.
2. The proposed FL based IDS infrastructure can be extended to other domain such as cloud based systems where distributed training and user privacy and more crucial.

# Bibliography

- [1] Samaneh Rastegari, Philip Hingston, and Chiou-Peng Lam. Evolving statistical rulesets for network intrusion detection. *Appl. Soft Comput.*, 33(C):348–359, August 2015.
- [2] Iram Abrar, Zahrah Ayub, Faheem Masoodi, and Alwi M Bamhdi. A machine learning approach for intrusion detection system on nsl-kdd dataset. In *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, pages 919–924, 2020.
- [3] Abdulsalam Alzahrani and Mohammed Alenazi. Designing a network intrusion detection system based on machine learning for software defined networks. *Future Internet*, 13:111, 04 2021.
- [4] Yuyang Zhou, Guang Cheng, Shanqing Jiang, and Mian Dai. Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Computer Networks*, 174:107247, 2020.
- [5] Yesi Novaria Kunang, Siti Nurmaini, Deris Stiawan, Ahmad Zarkasi, Firdaus, and Jasmir. Automatic features extraction using autoencoder in intrusion detection system. In *2018 International Conference on Electrical Engineering and Computer Science (ICECOS)*, pages 219–224, 2018.
- [6] Chaofei Tang, Nurbol Luktarhan, and Yuxin Zhao. Saae-dnn: Deep learning method on intrusion detection. *Symmetry*, 12(10), 2020.
- [7] Mostofa Ahsan and Kendall Nygard. Convolutional neural networks with lstm for intrusion detection, 08 2020.
- [8] Peilun Wu and Hui Guo. Lunet: A deep neural network for network intrusion detection. pages 617–624, 12 2019.
- [9] Yuelei Xiao and Xing Xiao. An intrusion detection system based on a simplified residual network. *Information*, 10:356, 11 2019.
- [10] Dapeng Man, Fanyi Zeng, Wu Yang, Miao Yu, Jiguang Lv, and Yijing Wang. Intelligent intrusion detection based on federated learning for edge-assisted internet of things. *Security and Communication Networks*, 2021, 2021.

- [11] Sawsan Abdul Rahman, Hanine Tout, Chamseddine Talhi, and Azzam Mourad. Internet of things intrusion detection: Centralized, on-device, or federated learning? *IEEE Network*, 34(6):310–317, 2020.
- [12] Shaashwat Agrawal, Sagnik Sarkar, Ons Aouedi, Gokul Yenduri, Kandaraj Piamrat, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, and Thippa Reddy Gadekallu. Federated learning for intrusion detection system: Concepts, challenges and future directions. *arXiv preprint arXiv:2106.09527*, 2021.
- [13] Ruijie Zhao, Yue Yin, Yong Shi, and Zhi Xue. Intelligent intrusion detection based on federated learning aided long short-term memory. *Physical Communication*, 42:101157, 2020.
- [14] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.
- [15] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *arXiv preprint arXiv:2102.02079*, 2021.
- [16] Valerian Rey, Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Gérôme Bovet, and Martin Jaggi. Federated learning for malware detection in iot devices. *arXiv preprint arXiv:2104.09994*, 2021.
- [17] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.
- [18] Jiale Zhang, Bing Chen, Xiang Cheng, Huynh Thi Thanh Binh, and Shui Yu. Poissongan: Generative poisoning attacks against federated learning in edge computing systems. *IEEE Internet of Things Journal*, 8(5):3310–3322, 2020.
- [19] Oualid Zari, Chuan Xu, and Giovanni Neglia. Efficient passive membership inference attack in federated learning. *arXiv preprint arXiv:2111.00430*, 2021.
- [20] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.
- [21] Ali Hatamizadeh, Hongxu Yin, Pavlo Molchanov, Andriy Myronenko, Wenqi Li, Preerna Dogra, Andrew Feng, Mona G Flores, Jan Kautz, Daguang Xu, et al. Do gradient inversion attacks make federated learning unsafe? *arXiv preprint arXiv:2202.06924*, 2022.
- [22] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

- [23] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.
- [24] Malhar S Jere, Tyler Farnan, and Farinaz Koushanfar. A taxonomy of attacks on federated learning. *IEEE Security & Privacy*, 19(2):20–28, 2020.
- [25] Nader Bouacida and Prasant Mohapatra. Vulnerabilities in federated learning. *IEEE Access*, 9:63229–63249, 2021.
- [26] Sannara Ek, François Portet, Philippe Lalanda, and German Vega. Evaluation of federated learning aggregation algorithms: application to human activity recognition. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers*, pages 638–643, 2020.
- [27] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE, 2020.
- [28] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *International Conference on Information Security and Cryptology*, pages 1–21. Springer, 2012.
- [29] Ahmad Al Badawi, Louie Hoang, Chan Fook Mun, Kim Laine, and Khin Mi Mi Aung. Privft: Private and fast text classification with homomorphic encryption. *IEEE Access*, 8:226544–226556, 2020.
- [30] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.
- [31] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.
- [32] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
- [33] Dongbin Zhao, Haitao Wang, Kun Shao, and Yuanheng Zhu. Deep reinforcement learning with experience replay based on sarsa. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. IEEE, 2016.

- [34] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.