Fortifying Federated Learning: Security Against Model Poisoning Attacks

by

Fabiha Anan

20101085

Kazi Shahed Mamun

20301471

Md Sifat Kamal

20101231

Nizbath Ahsan

23341119

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material that has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Fabiha Anan 20101085	Kazi Shahed Mamun 20301471
Md. Sifat Kamal 20101231	Nizbath Ahsan 23341119

Approval

The thesis titled "Fortifying Federated Learning: Security Against Model Poisoning Attacks" submitted by

1. Fabiha Anan (20101085)
2. Kazi Shahed Mamun (20301471)
3. Md. Sifat Kamal (20101231)
4. Nizbath Ahsan (23341119)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January, 2024.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Md. Tanzim Reza
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, Ph.D.
Chairperson
Department of Computer Science and Engineering
Brac University

Ethics Statement

We hereby declare that this thesis is our own work and the discoveries are from our research. The sources have been appropriately acknowledged which we used in our study. Therefore, we are confirming that this thesis has not been submitted published, or presented in any other educational institution for receiving any degree.

Abstract

Distributed machine learning advancements have the potential to transform future networking systems and communications. An effective framework for machine learning has been made possible by the introduction of Federated Learning (FL) and due to its decentralized nature it has some poisoning issues. Model poisoning attacks are one of them that significantly affect FL's performance. Model poisoning mainly defines the replacement of a functional model with a poisoned model by injecting poison into models in the training period. The model's boundary typically alters in some way as a result of a poisoning attack, which leads to unpredictability in the model outputs. Federated learning provides a mechanism to unleash data to fuel new AI applications by training AI models without letting someone see or access anyone's confidential data. Currently, there are many algorithms that are being used for defending model poisoning in federated learning. Some of them are really efficient but most of them have lots of issues that don't make the federated learning system properly secured. So in this study, we have highlighted the main issues of these algorithms and provided a defense mechanism that is capable of defending model poisoning in federated learning.

Keywords: Machine learning, Federated learning, Model poisoning, Defense

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Federated learning (FL) is a relatively new concept, a promising solution in light of this new reality. The process through which a model is trained across multiple decentralized devices without sending the data of the user directly to the central server is known as federated learning. In federated learning, decentralized devices mainly refer to individual devices such as smartphones, tablets, or laptops that are not connected to a central server. In the context of federated learning, decentralized devices can be used to train a machine learning model locally, based on the data available on the device. Here basically, the model is trained locally on each device and the updates are sent to a central server and aggregated to improve the model. The main target of federated learning is to protect the privacy of users. It offers a privacy protection approach that trains the model by preventing the leakage of sensitive data during data transmission. As there is the availability of numerous rich dataset resources due to the abundance of mobile and other types of devices and so federated learning can fully utilize them. Federated learning initially safeguards user privacy to prevent attackers from accessing source data by communicating encrypted processed parameters. But here along with this safeguard, there is still a chance of privacy issues like: Model Poisoning.

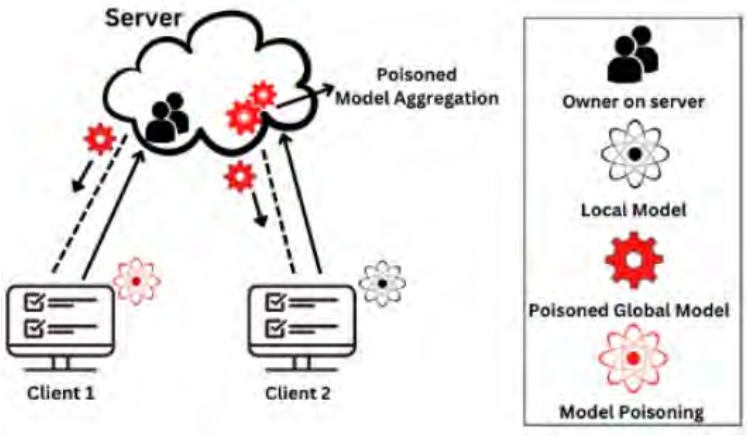

Figure 1.1: Model Poisoning Attack

Model poisoning has now become a great threat while working in these fields. The process through which poison is injected into the local model before sending it to the server or inserting hidden backdoors into the global model during the training process is known as model poisoning. We know poison can be a chemical term in chemistry but in model poisoning, the poison indicates or defines the way or method to pollute a machine learning-trained model by manipulating its training data. It is also known as an adversarial attack in which malicious clients aim to poison the global model by sending constructed local model updates to the central server. The main aim of model poisoning is to reduce the working performance of models by degrading their accuracy. It mainly occurs in the training phase of the model. Due to model poisoning the trained models failed to show the correct accuracy in the output or result. The method through which multiple attackers are involved to manipulate the machine learning model is called distributed model poisoning. In distributed model poisoning the attacker mainly tries to alter or change a small portion of data of the trained model which changes the overall behavior of the model to act in a wrong way. In centralized machine learning, several sources of data are collected and stored on a single server or location in order to train a global model. The model's parameters are then repeatedly updated using the combined dataset, and the model is then utilized to perform tasks or predictions [1]. In decentralized machine learning distributes data among nodes or devices and conducts local training, providing superior privacy and scalability but in centralized machine learning gathers data into a single location for building a global model. In addition, distributed model poisoning mostly occurs in federated learning systems. Analyzing the Model poisoning attack, it is observed that it is more vulnerable to federated learning.

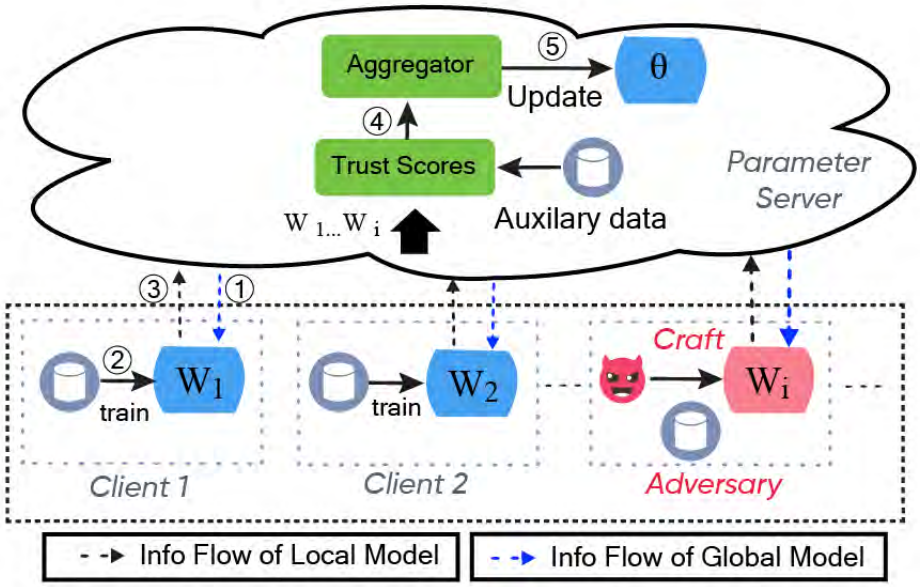

Figure 1.2: Federated Learning system model

The action of this poison attack mainly differs in various fields in various ways as their working procedure is different. Based on the attacker's intention model poisoning can be divided into two categories, targeted attacks and untargeted attacks. If the attacker wants to decrease the overall accuracy of the global model, it is called an untargeted attack. And when the attacker wants to decrease the accuracy of targeted test inputs then it is called a targeted attack. In this paper, we tried to mitigate the untargeted attack model poisoning attack. This untargeted attack decreases the accuracy of the global models and as a result, the model cannot show the actual output due to the degradation of performance. For these types of attacks, defensive algorithms have been discovered. Depending on aggressive poison attacks in federated learning two types of defense have been initialized, mentioning evaluation methods for local model updates and aggregation methods for the global model. By studying the related works of model poisoning in federated learning we came to know about different types of poison attacks and learned about various algorithms to defend against these attacks and solve the security problems in federated learning. So overall we have studied the attacks, existing algorithms and constructed a new defense mechanism for the model poisoning attack. Our mechanism is very much capable of defending a model from the model poisoning attack.

1.1 Motivation

Federated Learning is a very promising machine learning approach since it is capable of training models across multiple decentralized edge devices including Smartphones, IoT devices, or any local servers, etc. without any user privacy issues. However, there is an issue of model poisoning which happens during the training process. This model poisoning is a very serious concern since by manipulating the model, it will not provide the proper outputs. Hence it will not be able to perform properly. There are existing defense mechanisms but most of them have issues that can not resolve all kinds of attacks properly. It represents that there is a research gap on this. So this research paper aims to address these gaps and present a better defense mechanism for the federated learning environment.

1.2 Research Organization

- In Chapter 1 (Introduction), we discuss the importance of federated learning and the challenges posed by model poisoning. We define the problem and outline the objectives of the research.
- In Chapter 2 (Literature Review), we review existing literature on federated learning, model poisoning, and defense mechanisms.
- In Chapter 3 (Implementation), we describe various types of model poisoning attacks and existing defense algorithms. We discuss the dataset used and the preprocessing steps applied.
- In Chapter 4 (Proposed Model), we detail the architecture of the proposed federated learning model and describe its components. We discuss the handling

of non-IID data, the criteria for model selection, the experimental setup, and the implemented attacks.

- In Chapter 5 (Result Analysis), we discuss the performance of the proposed model against various model poisoning attacks. We present the experimental results and discuss the precision, recall, and F1 score of the proposed algorithm.
- In Chapter 6 (Conclusion and Future Work), we summarize the findings of the research, discuss potential improvements, and future research directions.

1.3 Research Problem

Our world is currently progressing with technology very fast. Along with development, it is generating huge amounts of data per day which are being used for various fields like Artificial Intelligence, Big Data, Data Analysis, etc. Meanwhile, advancement in Artificial Intelligence is changing society very rapidly. For the activities related to AI, some industries currently prefer to train data centrally which is very hassle in a traditional way. So this has become very challenging for many industries.

So, In the case of working with decentralized training data, many industries currently use Federated Learning. It's a machine learning algorithm that has the ability to train a set of small models on individual devices and then aggregate it with the purpose of forming the final model with no security issues. However, there is still a weakness in the security of the federated learning algorithm which makes it an easy target for the model poisoning attack. Attacking the raw data of a model during training is called Model Poisoning.

For preventing model poisoning, here [2] the author has investigated the current defense methods for the model poisoning attacks on the Federated Learning algorithm and classified them into two categories. These two categories are Evaluation Methods for local model updates and aggregation methods of the global model. Evaluation methods for local model updates are testing the submissions of the clients for the raw data which is a kind of straightforward way to defend the model from model poisoning. On the other hand, aggregation methods of the global model are aggregating raw data by making sure of proper security, though usually they are used together to protect the model. According to the author's [2] analysis aggregation method also can be classified into two categories such as: Adjusting the weights of local model updates based on specific criteria and designing aggregation algorithms using statistical methods. In the case of the criteria-based method, some criteria are being used for the evaluation of the model poisoning. For example Trust, Similarity, Reliability, etc. On the other hand, Statistic Based Aggregation methods use statistical methods instead of following simple criteria during the aggregation process. Trimmed Mean, Krum, Bulyan, etc. are examples of this method [2].

Some of the defense models related to these are discussed below:

1) Truth Inference-Based Evaluation Method:
Truth Inference-Based Evaluation (TIBE) is a method for evaluating the robustness of machine learning models against adversarial attacks or "model poisoning". This method is based on the idea of inferring the "ground truth" or the true label of a given data sample and comparing it with the model's prediction. The basic idea behind TIBE is that a robust model should be able to make accurate predictions even in the presence of adversarial attacks. Reducing the weighted deviation from the true aggregated parameters is the goal of this method in FL. The reliability score is first calculated here for each update. Secondly, to aggregate the global model they bring in two methods where the first method uses the reliability score according to the weight of each update. Another method eliminates updates with low-reliability scores [2].

Truth Inference-Based Evaluation Method assumes that the true label of a data sample is known, which may not be the case in many real-world scenarios. Moreover, it can be time-consuming, as it requires the inference of the ground truth for a large number of data samples, which can be computationally expensive. Also, the Truth Inference-based evaluation method only works when there is IID data but it is not efficient for non-IID cases.

2) Trimmed Mean Aggregation Method:
The trimmed Mean Method is a technique used in Federated Learning to mitigate the impact of malicious or unreliable participants on the model's performance. It's a coordinate-wise AGR (robust aggregation algorithm). Aggregating each dimension of input gradients separately is the basic idea of this method. In this method, instead of using the average of all the models from different participants, a trimmed mean is used, which is calculated by discarding a certain percentage of the models with the highest or lowest values. The idea is that these extreme values are likely to be outliers, which can be caused by malicious participants [3].

It is challenging to determine the right percentage of models to be trimmed as discarding too many models can lead to a reduction in the accuracy of the model, while discarding too few models may not be effective in mitigating the impact of poisoning attacks. The Trimmed Mean Method is not very robust to attacks that are specifically designed to bypass this method. For example, an attacker can try to manipulate the model's parameters in such a way that the trimmed mean still includes their malicious models.

3) Median Aggregation Method:
The Median Method is used to counter model poisoning attacks by computing the median of the model updates received from all devices and then using that median to update the global model. This is based on the assumption that the median of a set of values is less susceptible to manipulation by outliers than the mean or other aggregate statistics. It's also a coordinate-wise AGR that aggregates input gradients by calculating the median of the values of individual dimensions of the gradients [3].

One of the key limitations is that it only works well when the number of malicious devices is small compared to the total number of devices. When the number of

malicious devices is large, the median can be skewed towards the malicious updates, making the global model behave maliciously. Another limitation of the Median Method is that it is not robust to more sophisticated attacks and can not provide enough fidelity since a server is required to compute pairwise distances of local model updates.

4) Krum Aggregation Method:
The Krum method works by robustly aggregating the model updates from each client, by computing the median of the model updates and taking it as the final model update. This reduces the impact of any outliers or malicious updates, as the median is less sensitive to individual values compared to the mean. Krum chooses the gradient in the squared Euclidian norm space that is close to its m-c-2 neighboring gradients from the set of its input gradients. Here m is an upper bound on the number of malicious clients in Federated Learning [3].

The median may not always be the best estimate of the true model update, and the Krum method may result in a lower accuracy compared to other aggregation methods. While the Krum method can mitigate the impact of model poisoning attacks, it does not guarantee full protection against such attacks, especially if the attacker has a large influence over the global model. Also, it becomes computationally expensive if the client numbers become large.

5) Bulyan:
It's a Byzantine fault-tolerant algorithm. The Bulyan method works by adding a layer of randomization to the aggregation process so that the contribution of each local model to the global model is weighted randomly, rather than equally. This makes it more difficult for an attacker to manipulate the global model by poisoning their local model, as their contribution to the global model will be diluted. It constantly cycles via the updates. Along with this, it does trimmed mean. Especially it uses Krum for the selection. Consequently, this method is a combination of Krum and Trimmed Mean [2].

The Bulyan method has some limitations. For one, it requires a significant amount of communication overhead, as each device must exchange a large amount of random data with the server. Additionally, it can lead to decreased performance, as the random weighting of the local models can lead to a loss of information and accuracy. Moreover, it is non-scalable because it does Krum many times throughout every iteration to compute pairwise distances within the local models. It is possible that the Euclidian distance within two local models is influenced by individual model parameters which can be resulted that Krum will be affected by anomalous model parameters.

6. Dynamic Defense Against Byzantine Attacks (DDaBA):
DDaBA is a defense mechanism that helps protect against model poisoning attacks by detecting and excluding malicious devices from the aggregation process. It dynamically adjusts the set of trusted devices based on their past behavior and the consistency of their model updates. This makes it more robust to attacks compared to static methods that use a fixed set of trusted devices, as the malicious devices may change over time

DDaBA is also not a perfect solution since it has limitations. For example, it requires devices to have enough computational resources to perform the necessary computations, and the accuracy of the defense depends on the quality of the initial set of trusted devices. Additionally, DDaBA can still be vulnerable to attacks if the attackers are able to coordinate their efforts and present a consistent false model. These models are capable of defending the attacks but they have various limitations that actually do not represent enough robustness to these models. So there is a research gap on these limitations. So in our study, we aim to investigate this and construct a good defense mechanism for this attack.

1.4 Research Objective

This research aims to introduce an updated or better version of the defense algorithm for the model poisoning of Federal Learning. Current defense algorithms for model poisoning have several issues that can affect the accuracy of the model. So security for this model is very concerning. The objectives of this research are:

- To deeply understand the Federated Learning algorithm and how it works
- Understand the security of federated Learning
- To analyze the security issues of the defense models for the Federated Learning
- Constructing a new defense model for the Federated Learning

Chapter 2

Literature Review

As computing devices are increasing, people are generating a huge amount of data day by day. Collecting this type of data in centralized storage is very expensive and time-consuming. Unfortunately, traditional Machine Learning can not support this kind of issue due to infrastructure shortcomings like limited communication bandwidth, network connectivity, delay constraints, etc. [4] For this kind of scenario Federated Learning is playing a great role.

From the beginning, the role of federated learning can be observed while working in different industries for example Healthcare, Internet of Things (IoT), and Transportation. Since federated learning can train models in different devices and aggregate them without any security issues. Certainly, the application of this algorithm is implemented for the development of 5G and 6G which can offer more secure effective schemes for communication in the future [2].

Although the aggregator has no access to the info, model poisoning could create data accuracy problems which are possibly considered as faulty for the model. In addition during the training phase model poisoning is performed by adding crafted data to training data resulting in low accuracy in the model. The model won't provide the actual result due to low accuracy. So ensuring the security of federated learning algorithms became an important concern for researchers.

Model poisoning attack for federated learning (MPAF), which sends expertly prepared fake local model refreshes to the host using phony clients. MPAF is able to decrease the efficiency of the model structure in an untargeted approach and it can still considerably reduce accuracy even when conventional defenses are used. In contrast to current attacks with untargeted model poisoning, which are based on compromised legitimate clients and construct distinct round-wise optimization problems, A fixed base model is the goal of MPAF, which is conceptualized as a global optimization problem that departs from the model structure. Future work will focus on extending MPAF, carrying out focused model poisoning attacks and improving it with new information and data. An attacker can choose a base model that has the desired targeted behavior, such as a base model with a gateway. If the learned global model is pushed to be substantially close to a corrupted base model, it may display the same gateway behavior as the base model and anticipate attacker-selected target categories for attacker-selected test data [5].

SparseFed is a defense against model poisoning attacks in federated learning that uses global top-k update sparsification and device-level gradient clipping. The algorithm computes gradients locally, clips them, aggregates them in the cloud, extracts the top-k values, and broadcasts them as sparse updates to participating devices in the next round. SparseFed is effective in training high-quality models under targeted model poisoning attacks, in which the goal is to reduce performance on specific data points or sub-tasks without compromising overall accuracy. The authors propose a framework for analyzing the robustness of defenses using the certified radius metric and show that SparseFed minimizes this radius by sparsifying aggregate model updates. Limitations of the work are the lack of access to actual federated datasets at the scale of tens of thousands of devices, our work is empirically constrained because we are compelled to create faulty simulations of cross-device federated settings. Our simulation technique is to simulate each device only drawing samples from the distribution of one class rather than many classes for CIFAR-10, CIFAR-100, and FMNIST, which lack any naturally occurring non-iid partitioning. However, this may not always be the case in the actual world. To get around these constraints in the future, we urge the federated learning community to submit large-scale, real-world datasets [6].

Defense mechanisms for model poisoning attacks in federated learning are classified into two categories: evaluation methods for local model updates and aggregation methods for the global model. There are many types of attacks on federated learning, including poisoning attacks, backdoor attacks, and inference attacks. Poisoning attacks can be further divided into data poisoning attacks, which manipulate raw data on the clients, and model poisoning attacks, which manipulate local model updates. Model poisoning attacks are more likely to cause damage to federated learning than data poisoning attacks. Defense mechanisms against model poisoning attacks involve identifying malicious submissions through evaluation mechanisms for local model updates and designing novel, Byzantine fault-tolerant aggregation algorithms based on mathematical statistics. These approaches are often used in combination. A good defense mechanism should be effective against attacks, conserve resources, and protect data privacy. In the future, research should consider the impact of different attack strategies, such as the number of malicious clients and the timing of the attack, on the effectiveness of the attack, and design effective defense mechanisms accordingly. The deployment of defense mechanisms should also take into account computational resource limitations. One potential solution is the combination of blockchain and federated learning, in which updates are selected and aggregated through a voting process and written to blocks through a blockchain network [2].

One of the research articles describes research on Dynamic Defense Against Byzantine Attacks, a dynamic aggregation operator used in federated learning (FL) to fight against byzantine (adversarial) attacks (DDaBA). This operator dynamically discards hostile clients, allowing you to stop the global learning model from being tainted. Based on an induced-ordered function and a linguistic quantifier, the induced ordered weighted averaging (IOWA) operator underlies DDaBA. It aggregates clients on a weighted basis. Through the use of deep learning classification models in FL and a variety of image datasets, including Fed-E MNIST Digits, FM-

NIST, and CIFAR-10, the research evaluates the effectiveness of DDaBA as a defense against adversarial attacks. According to the study, DDaBA is more effective than FL's current byzantine defenses. Additionally, the authors proposed a static version of DDaBA, named Static Defense Against Byzantine Attacks (SDaBA), which addresses the weakness of DDaBA when there's a very high presence of adversarial clients in the system [7].

Federated learning is a widely used method for securing data in Industrial Internet of Things (IoT) devices, but it is vulnerable to model poisoning attacks, which are difficult to detect, particularly in Industrial IoT applications. This study proposes TIMPANY, a framework for detecting model poisoning attacks using accuracy as a measure. Theoretical and experimental analysis shows that TIMPANY is effective and efficient against model poisoning attacks in federated learning, with a true positive rate and accuracy of 100% and no false positives. TIMPANY is particularly well-suited for resource-constrained Industrial IoT devices used in various industrial applications. A popular machine learning method known as federated learning (FL) enables decentralized learning and data privacy for end devices. However, it also introduces a new security risk in the form of model poisoning attacks, making it particularly suited for industrial Internet of Things (IIoT) applications. These attacks are challenging to identify because, first, the server and FL participants cannot identify tainted local models using only the specified weights, and second, each FL iteration involves a large number of random participants, making it impossible to verify each one individually. The authors suggested the TIMPANY detection framework, which uses accuracy as a detection metric, to overcome this problem. The TIMPANY system has been theoretically and empirically assessed by the authors, who concluded that it is reliable and effective for spotting FL model poisoning attacks. Future research will examine more complex and automated model poisoning attacks, and TIMPANY will be improved to address additional challenges like system and statistical heterogeneity [8].

The author developed a brand-new kind of poisoning attack known as a "semi-targeted attack" in which the perpetrator tries to trick the classifier into classifying them as a different group. But switching to a different class makes the attack less effective. Therefore, they suggested ADA (Attacking Distance-aware Approach), a novel kind of semi-targeted model poisoning attack that seeks to undermine the classifier by carefully choosing the target class. The major objective was to make it difficult for the global classifier to correctly classify the source class's data. They also looked at harder cases where the foe knows nothing about the client's information. With the aid of backward error analysis, they were able to resolve this difficult scenario. They completed three distinct image categorization tasks while altering the element of attacking frequency for the evaluation. In the most difficult scenario, with an attacking frequency of 0.01, it was successful in raising the attack performance by 1.8 times [9].

The concept of model poisoning in the field of artificial intelligence (AI) is introduced. Model poisoning involves injecting malicious triggers into machine learning models during the training phase to compromise their performance and use them for malicious purposes. The paper discusses two types of model poisoning attacks:

targeted and untargeted. It also examines the effectiveness of model poisoning attacks on Pythia and GPT-2, two advanced neural code completion models. The paper also discusses the goals and steps of model poisoning attacks, as well as the various types of "baits" that attackers may use. Finally, the paper discusses three methods of defense against model poisoning in neural code completion: activation clustering, spectral signature, and fine pruning. By combining these methods, the impact of model poisoning attacks can be reduced [10].

The topic of neural architecture search (NAS) is discussed, specifically how it can be affected by model poisoning attacks. The focus is on cell-based search spaces and differentiable NAS using the example of DARTS, which contains two types of cell structures. The main attacks on NAS, including adversarial evasion, model poisoning, backdoor injection, and functionality stealing, are analyzed and defenses against them are explored. The limitations of this work, including the focus on cell-based search spaces and differentiable NAS, and the exclusion of other search strategies and NAS frameworks, are also mentioned [11].

The practical impact of poison attacks on neural networks introduces a new type of attack called "black card" which manipulates a pre-trained model to create targeted poison instances for deceiving the intended model. The paper shows that this attack is effective, with a high success rate and misclassification confidence, while maintaining certain desired properties. The paper also describes a method for solving blackcard using optimization algorithms and examines targeted and untargeted poisoning attacks. The paper also points out that the proposed method, BlackCard, is effective with a high success rate using only one or a few poisoning samples, not dependent on the target model knowledge and labeling process, it can also use arbitrary test-time instances [12].

The problem of model poisoning in deep neural networks (DNNs), is where an attacker can manipulate a small portion of the training data in a way that alters the behavior of the DNN model when it is presented with a specific trigger. The paper discusses different types of attacks that can be used for this, and how existing defense methods work against them. The authors then propose a new verification method, called VPN, for detecting poison in DNN models. This method is based on solving constraints using pre-existing neural network verification tools. The effectiveness of VPN is demonstrated using two types of datasets and it is able to detect small models that were trained to perform image classification tasks including triggers. Future work is suggested to expand the research to more complex attack models and investigate the idea of verification of large, complex models through the process of transfer or abstraction [13].

A Trojan is a kind of (backdoor) attack on deep neural networks that involves the attacker giving the victims a model that has been trained or retrained on harmful data. Classification of the neural model is observed based on two classes of models trained without trojans(clean) and (2) models trained with trojans (poisoned). The objective of this work is to create a baseline strategy for identifying (a) potential reference model architecture manipulation (changing a task-specific NN architecture termed . A range of architectures contain Trojans, as well as (a) a reference model)

and (b). Our strategy is demonstrated. Methods have been demonstrated to solve the problem mentioning the Pruning-based Approach: and Reduction of Search Space. Different types of datasets were used to detect trojans in clean data and poisoned data as the input dataset. It provided a baseline trojan detection method based on pruning that was tested on 2104 NN models from Challenge of Trojan AI [14].

Poisoning attacks on neural networks are attempts to influence the behavior of a classifier on a single test instance. The paper proposes a novel class of attacks called clean-label attacks, in which the attacker injects training instances that are cleanly labeled by a recognized authority, rather than maliciously labeled by the attacker. The idea is that these attacks are challenging to identify and provide attackers a chance to succeed without having direct access to the data collection/labeling process. The research presents a method for creating these clean-label poisoning instances using an optimization algorithm and discusses the transferability of these poison attacks. The paper also suggests the idea of training with poison instances as a way to defend against these attacks and highlights the importance of data provenance and dependability for neural networks [15].

Federated Learning is a Machine learning model that works with decentralized data. It works with multiple clients to train a shared model to their local data without sharing data. The paper "FLDetector: Defending Federated Learning Against Model Poisoning Attacks via Detecting Malicious Clients" proposed a defense mechanism for the malicious clients and the name of that defense mechanism is "FL Detector". This mechanism can able to check the consistency of model updates from every client. It predicts the model update by its historical model updates. So here by any chance, if the model update that was received is not able to show enough consistency with the predicted model then the mechanism considers it as a malicious client. Here evaluation process done by the author happened with three benchmark datasets. The evaluation clarified that this mechanism is effective for detecting malicious clients in multiple state-of-the-art model poisoning attacks. After discarding the malicious clients the existing Byzantine-robust FL methods can learn accurate global models. Overall the paper has made a good contribution to the Federated Learning security system. Also, the proposed method by the author is simple to implement and very effective for detecting malicious clients as well [16].

A new defense mechanism for the model poisoning attack in Federated learning, named DEFL has been introduced in a research study. This mechanism works by identifying critical learning periods (CLPs) during training where even a little gradient error can have a good impact on the accuracy of the final model. So for measuring fine-grained differences among DNN model updates it sets a federated gradient norm vector (FGNV) metric. Here if a client's update by any chance differs from the other client's the mechanism will consider the client as a suspicious client. During CLPs DEFL discards the suspicious clients from aggregating which results in the lesser impact of the model poisoning. For the evaluation process, the researchers evaluate this with three benchmark datasets showing that it provides very effective state-of-the-art model poisoning attacks. By comparing to the conventional defenses DeFL provides very significant performance and maintains robustness in case of

detecting errors. DeFL shows promise as a cutting-edge approach for detecting model poisoning attacks in FL [17].

Another study showed a new defense mechanism named, FLTrust. This mechanism works for the Byzantine model poisoning attack. FLTrust operates by having the server gather a small, clean dataset and train a local model on it in order to initially bootstrap trust in the clients. Then the server uses this local model to determine the trust scores for each client with higher trust scores provided to clients whose updates are similar to the server's local model. In every round, the server aggregates those updates of the clients that have high trust scores. This helps to mitigate the effect of Byzantine attacks. During the evaluation process, the authors evaluated this mechanism on three benchmark datasets and showed that this mechanism is effective in defending against Byzantine attacks even when a large number of clients are flagged as malicious. This mechanism is effective for existing and strong adaptive attacks. For example: using a root dataset with fewer than 100 samples, FLTrust can still train global models that are as accurate as the global models that are trained by FedAvg under no attacks, despite adaptive attacks with 40%–60% of malicious clients where FedAvg is a very prominent technique in non-adversarial settings. Overall, FLtrust is a potential new approach for FL that can lead to an increase in the security and privacy of the FL system [18].

Another research proposes a defensive mechanism for the decentralized framework of federated learning using blockchain. In this research work, the global model has been built up by exchanging the data of the local model. But it faced robustness issues in reality which is solved by the VBFL (Validation Accuracy Difference) framework. This framework has been built using blockchain architecture. VBFL incorporates a validation process, a role-switching policy, and a consensus algorithm to counter malicious worker updates and remove bias toward specific worker nodes. Additionally, alternative methods like blockchain-driven smart contracts and reputation-based worker selection are explored but are noted for their constraints [19].

One research article has demonstrated the critical thinking power over the untargeted poisoning attacks of FL environment though the threat models. This article has also explored the many kinds of poisoning attacks and how they could affect federated learning models at the production level. It evaluates the effectiveness of current defenses critically and suggests fresh tactics to strengthen these systems against hostile attacks. This study delves into the various types of poisoning attacks and their potential impact on production-level federated learning models. It critically assesses the efficacy of existing defense mechanisms and proposes novel strategies to fortify these systems against adversarial attacks [20].

Another research study is conducted with the introduction of the defensive mechanism of federated learning to stop model poisoning. In this research the new defense mechanism which is proposed against poisoning attacks is CONTRA. Poisoning attacks can compromise the integrity of the global model by injecting malicious data into the training process CONTRA utilizes a cosine-similarity approach to assess the reliability of local model parameters in every iteration. It employs a reputation system that dynamically adjusts the status of individual clients. This adjustment is

based on their contributions to the global model, considering both their current and past engagements in each round. Clients are either encouraged or penalized based on their consistency and impact on the collaborative model. Three well-known machine learning datasets were used to evaluate CONTRA: MNIST, CIFAR-10, and Loan. Each dataset had a separate training length defined, with 100 epochs for MNIST, 300 epochs for CIFAR-10, and 1000 epochs for the Loan dataset. While the batch sizes varied (20 for MNIST and CIFAR-10, and 128 for the Loan dataset), the learning rate was consistently set at 0.1. The training arrangement was guided by the convergence rate found in the body of existing literature. The system took an average of 18.43 seconds to finish 100 training cycles in a baseline federated learning setup with 100 truthful clients. On the other hand, the average runtime climbed to 29.36 seconds when CONTRA was implemented and 33% of clients were malicious. In general, the material offers important perspectives on countering poisoning attacks in federated learning (FL) and introduces a new defensive strategy. Yet, additional investigation is necessary to gauge its efficacy in intricate settings and against advanced attack methods [21].

Moreover, this research work addresses the problem of poisoning attacks in federated learning, in which adversaries attempt to intentionally reduce the accuracy of the model. In this work, FLAIR as a countermeasure against the directed deviation attack (DDA), a novel model poisoning method, had been developed. Based on participant behavior, FLAIR provides trust scores, guaranteeing that the model as a whole is shielded from all known untargeted poisoning efforts. By identifying uncommon gradient sign flip patterns and using weighted averaging for convergence, FLAIR finds fraudulent clients. It exceeds FABA with more than 20% of malicious clients while matching the accuracy of the best defenders. It exceeds FABA with more than 20% of malicious clients while matching the accuracy of the best defenders. FLAIR's protections hold up against a skilled attacker. However it necessitates client updates that aren't encrypted, which lessens the value of gradient encryption. Participating clients are evaluated by FLAIR according to their training conduct, and reputation scores are assigned. A weighted mix of these clients' contributions is then included. FLAIR outperforms existing defenses: even in the face of intense malicious activity, it operates remarkably effectively in a variety of federated learning settings. This research, which is funded by prominent funds, presents a viable answer to a significant federated learning problem. Different kinds of datasets have been applied in this project such as FMNIST, MNIST, CIFAR-10. Moreover, different kinds of attacks have been introduced to work for experiments such as Full Krum, None, AND Full Trim. Flip-scores are used by FLAIR to analyze unencrypted client updates and categorize them as benign or suspicious. Because gradient encryption has a large processing load, it is considered less important in cross-device settings. However, these restrictions open the door for further follow-up studies in the field. FLAIR's limitations indicate possible topics for further research to improve its application in many circumstances, despite its promising nature [22].

Another one discusses a defense mechanism against model poisoning attacks in Federated Learning (FL). The paper proposes a client-based defense that can mitigate the effects of model poisoning attacks on the global model. The defense mechanism is designed to work even if the attackers have access to benign clients' data. The

paper also introduces evaluation metrics to measure the effectiveness of the defense mechanism. The proposed defense mechanism outperforms the baselines in mitigating the attack effectively and efficiently. The article examines a Federated Learning (FL) protection tactic against model poisoning attacks. The client-focused response that has been suggested effectively counteracts the effect that these attacks have on the global model. Three metrics are used in the evaluation: attack, robustness, and utility. These measures measure the accuracy of benign data, rounds required to reduce this confidence, and misclassification confidence on harmful data, respectively. FMNIST and CIFAR-10 datasets are used in the study's experimental setting to compare the defense against popular model poisoning attempts. The defense's effectiveness in fending off attacks in fewer rounds with little loss of model utility is demonstrated by the results. Strengths include robustness and convergence guarantees, flexibility to different poisoning attempts, and quantitative evaluation of the impact of attacks. However the emphasis on targeted attacks begs the question of how well they work against non-targeted ones. The paper concludes by stating that the defense mechanism can be extended to other poisoning attacks [23].

The goal of one of the research works is to present FedMLSecurity, an addition to the open-source FedML library that simulates adversarial attacks and related defense mechanisms in the context of Federated Learning (FL). This benchmark, which is integrated into FedML, increases the functionality of the library and makes it possible to assess security issues and possible fixes in FL. FedMLAttacker, which simulates injection attacks during FL training, and FedMLDefender, which simulates defensive measures to neutralize these attacks, are the two main components of FedMLSecurity. This highly customizable open-source application works with a variety of machine learning models (including ResNet, GAN, and Logistic Regression) and federated optimizers (like FedAVG, FedOPT, and FedNOVA). Furthermore, its flexibility is extended to Large Language Models (LLMs), demonstrating its adaptability and usefulness in a range of contexts [24].

In addition, another research paper worked on SAFEFL which tackles the twin issue of security flaws and privacy in federated learning. While traditional federated learning techniques safeguard privacy, they are nevertheless vulnerable to poisoning and privacy inference attacks. To address these problems, SAFEFL uses Secure Multiparty Computation (MPC), which guarantees cooperative computing without disclosing personal information. It supports the assessment of security measures against different types of attacks and combines with MP-SPDZ for secure protocols. The framework offers a safe environment for creating reliable federated learning algorithms and is compatible with PyTorch. It may also be extended to accommodate various models. Though MPC has a significant computing overhead and linear regression is the main focus at the moment, SAFEFL has the potential to improve security in practical federated learning applications [25].

Chapter 3

Implementation

3.1 Attacks

Krum Attack:

In our model, we applied the Krum attack [5] purposefully to modify local model updates on infected devices. It involves creating modified models by arbitrarily choosing parameters. The objective of this attack is to weaken the resilience of model aggregation in federated learning systems by using maliciously constructed updates from hacked devices to subtly affect the global model.

Krum attack selects a local model as a global model in each iteration. To influence the performance of Krum's attack on our model, the selection process was initialized to select a particular manufactured local model rather than the valid model chosen in an attack-free setting, the goal is to generate compromised local models. To guarantee that Krum, using the aggregate rule, chooses the assigned constructed model, this entails creating c compromised models. By increasing the disparity between the chosen constructed model and the valid model, this technique seeks to impact Krum's choice-making process in the process of model aggregation.

The purpose of this attack is to compromise the reliability of the model aggregation procedure. Through deliberate manipulation of compromised local models, the attacker hopes to influence the Krum aggregation rule to choose a particular constructed model over an authentic one.

Gaussian Attack:

Apart from the Krum attack, another attack was applied to our model to compromise the integrity and accuracy of the collaborative model aggregation process. The attack attempts to skew the learning process by introducing modified values, sampled from a Gaussian distribution, into local models from compromised devices. The intention behind the Gaussian attack [26] is to quietly modify the combined model to suit the attacker's goals, which may have an effect on the model's overall performance. In the end, the attack aims to introduce false information while trying to avoid detection to compromise the federated learning system's dependability and efficacy. This attack specifically creates changes to the model parameters by selecting random numbers from a normal, Gaussian distribution. The attack attempts

to fool the aggregation process by inserting these modified variables into the local models, which could have an effect on the integrity and accuracy of the global model. By adding subtle but deceptive information to the aggregated model updates from compromised devices, the goal is to obstruct the learning process while trying to avoid being discovered by the defenses of the federated learning system.

This method of attack involves creating altered versions of the local models on compromised worker devices in a random manner. To execute this, for every parameter of a given model, a Gaussian distribution is approximated using the initial local models across all worker devices before the attack. Subsequently, for each compromised worker device, a value is randomly selected from this Gaussian distribution and substituted as the parameter for that device's local model.

Median Attack:

The Trimmed Median attack in federated learning involves manipulating the model updates contributed by compromised devices to influence the aggregation process. In this attack, rather than sending a single model update, the attacker sends multiple versions of their update, each slightly altered from the original. These alterations are designed to be within a range that won't be immediately flagged as outliers by aggregation methods like the Trimmed Median. By doing so, the attacker aims to subtly shift the aggregated model towards their manipulated updates, potentially impacting the global model's performance without triggering outlier detection mechanisms. To manipulate the Median aggregation rule, we employ similar attack tactics as those employed in trimmed mean attacks. Median attack [6] is used to manipulate the learning process, potentially favoring the attacker's objectives or causing the global model to deviate from its intended performance. Ultimately, the aim is to compromise the integrity and accuracy of the federated learning model by introducing deceptive information while evading detection, thus impacting the overall learning outcomes. The goal is to introduce biased or misleading information into the aggregation process without being detected, thereby compromising the integrity and accuracy of the federated learning model.

3.2 Defense Algorithms

LFR (Loss Function Reduction):

The algorithm operates by focusing on the concept of loss in machine learning, which is a measure of how well the model's predictions align with the actual outcomes. Initially, the algorithm calculates the Sparse Categorical cross-entropy loss for each client. This is done by excluding one client at a time and aggregating the model updates from the remaining clients. The updated model is then used to make predictions on a validation dataset, and the loss is calculated. This process is repeated for each client, resulting in a list of loss values, one for each client. Once the losses have been calculated, the clients are sorted based on these values. The idea here is that a higher loss value could indicate a malicious client, as their exclusion leads to a model that performs better on the validation set. The algorithm then identifies the non-malicious clients as those with the lowest associated loss values. The rationale is that these clients are likely to be contributing beneficial updates that improve

the model's performance. Finally, the algorithm aggregates only the updates from these identified non-malicious clients to update the global model. By doing so, it ensures that potentially harmful updates from malicious clients are not incorporated, thereby defending against model poisoning. Hence, the LFR [26] algorithm provides a robust defense mechanism in federated learning systems, ensuring the integrity and performance of the global model by smartly identifying and excluding potentially harmful updates. It's a significant step towards secure and reliable federated learning.

Union:

The defense algorithm in federated learning is designed to protect against model poisoning. Union [26] initiates by setting the server model weights, defining the loss function, and creating two lists to record the loss and gradient norm for each client. For each client, a temporary list of all other clients is created. The loss and gradient norm for the current client is then calculated using the server model's predictions on the validation data and the defined loss function. Clients are ranked based on their loss and gradient norm, with those having lower loss and higher gradient norm considered non-malicious. The algorithm intersects the top-ranked clients based on loss and gradient norm, selecting the top-ranked clients from this intersection as the non-malicious clients. Finally, the server model weights are set and updates from the non-malicious clients are aggregated. This process mitigates the impact of malicious clients attempting to poison the model during federated learning, ensuring the integrity and robustness of the learned model by only considering updates from non-malicious clients.

3.3 Dataset Description

CIFAR-10: A well-known benchmark dataset used in computer vision and machine learning research is the CIFAR-10 [27] dataset. It consists of 60,000 color photos divided into ten different classes, with 6,000 images in each class. To ensure an even distribution of classes, the dataset is split into a training set of 50,000 photos and a test set of 10,000 images. The CIFAR-10 dataset includes photos that were compiled from diverse sources and manually annotated by humans. Every picture has a fixed resolution of 32x32 pixels and consists of red, green, and blue color channels. While the CIFAR-10 dataset offers a wide range of images with clear class labels, it is vital to recognize that it may also have inherent limitations, such as potential biases in image selection and potential noise or mistakes in the categorization process.

MNIST: In the field of machine learning, the MNIST [28] dataset is a frequently used benchmark dataset, notably for image classification tasks. It is made up of a sizable number of photos in grayscale that represent the handwritten digits 0 through 9. The MNIST Special Database 19 served as the source for the dataset, which was produced by Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The pictures used in the MNIST dataset were gathered from scanned handwriting samples and internet databases and preprocessed to a fixed size of 28x28 pixels, centering normalization, and other operations. To ensure a balanced distribution of each digit class, the dataset is split into a training set with 60,000 photos and a

test set with 10,000 images. The MNIST dataset is frequently used as a trustworthy source for assessing and contrasting image classification techniques.

FMNIST: A well-known and frequently used benchmark dataset in machine learning and computer vision is the FMNIST [29] dataset. It was developed as an alternative to the original MNIST dataset to give image classification algorithms a more difficult task by concentrating on photos linked to fashion. The dataset comprises 28x28 pixel grayscale photos of a variety of clothing items, including T-shirts, pants, dresses, shoes, and more. There are 60,000 photos in the training set and 10,000 images in the test set. The FMNIST dataset is frequently used by academics and industry professionals to assess and compare the effectiveness of machine learning models and algorithms in fashion classification applications.

3.4 Data Preprocessing

For the preprocessing of datasets, firstly we load the predetermined datasets into our FL setting with the help TensorFlow keras library function. Then we perform the next steps of preprocessing.

Normalization: The pixel values in the original images of MNIST and FMNIST datasets range from 0 to 255. The normalization step involves dividing each pixel value by 255. This process scales the pixel values to a range between 0 and 1. Normalization accelerates the training process and helps mitigate issues that may arise from varying pixel value scales, ensuring more stable and efficient learning for the neural network. Concerning mean and standard deviation adjustments, for MNIST and FMNIST datasets, specific mean and standard deviation values are applied after normalization. These values are determined empirically to center and scale the pixel values appropriately. This standardization ensures that pixel values are centered around zero, preventing biases in the model and contributing to better convergence. Like MNIST and FMNIST, the CIFAR-10 dataset also has its pixel values adjusted to a standard range. Each number for a pixel, which goes from 0 to 255, is made smaller by dividing it by 255. This normalization helps make the pixel values the same and helps the neural network learn better. For CIFAR-10, which consists of color images in RGB format. The mean and standard deviation are calculated for each channel (Red, Green, and Blue), and each pixel in a channel is standardized by subtracting the respective mean and dividing by the respective standard deviation. This process ensures that intensity across different color channels is balanced, contributing to better convergence and training stability.

Reshape: To comply with the input requirements of the CNN model, the images of MNIST and FMNIST datasets are reshaped to a standardized format of 28x28x1. The original format, 28x28, captures the image dimensions, while the additional dimension of 1 signifies a single channel, representing grayscale. This reshaping ensures uniformity in the input dimensions across the datasets, facilitating seamless integration with the CNN architecture. Again, reshaping is a crucial step applied to CIFAR-10. The original 32x32 color images went through reshaping to ensure consistency in input dimensions, maintaining compatibility with the selected ResNet

architecture. This transformation is necessary as it establishes a standardized format for the input data, enabling seamless integration. The reshaped images, conforming to the ResNet model's requirements, facilitate effective feature extraction and pattern recognition during the training process, contributing to the overall success of the deep learning model on the CIFAR-10 dataset.

Chapter 4

Proposed Model

4.1 Model Architecture:

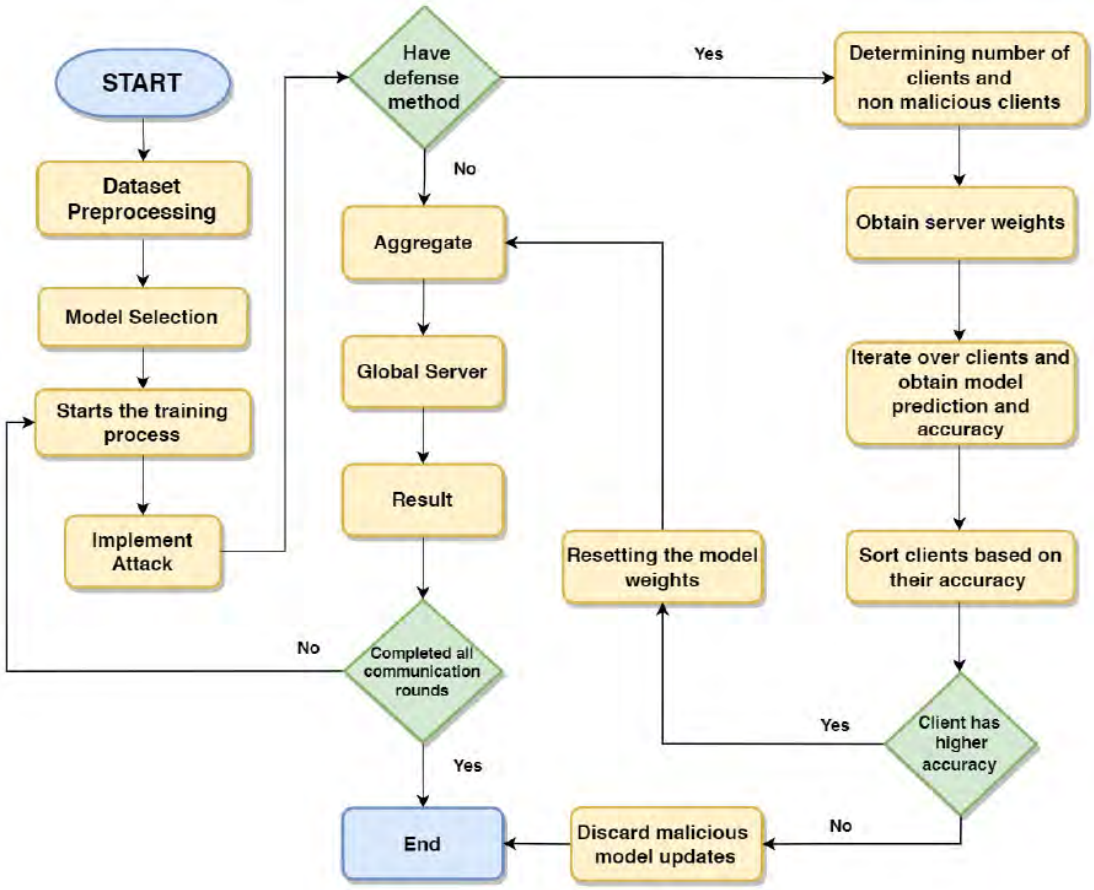

Figure 4.1: Workflow of proposed Decentralized Defense Algorithm Training Model

4.2 Model Description

4.2.1 Non-iid

The type of data that is different in various kinds of devices is known as Non-iid data [30]. In FL, we trained our model on decentralized devices. If the data on the devices vary from each other then it will result in a model aggregation challenge. Since the representation of specific patterns in each device is not good due to a lack of distribution it will affect the convergence of the global model. To make the combined model strong and enhance the working performance application of non-iid data is important. In a model, the training data is dispersed among multiple clients such as devices or servers. Here, the issue of non-identical data is referred to as non-IID data which arises when the data distributions across clients vary, resulting in each client possessing a distinct dataset with potentially different patterns. To mitigate this, our function takes the non-IID nature into account. Non-iid datasets are created where each client has a specific distribution of classes based on the non-iid parameter. This controls the degree of non-iidness, determining how likely a data point is to be assigned to its true class compared to other classes. We first created empty lists to store indices for each class. Calculated probabilities for each class based on the non-iid parameter which sums up to 1. Then we determined the number of clients for each class based on the total number of clients and classes, distributing clients as evenly as possible. It assigns random groups or classes to individual data points based on specified probabilities, subsequently organizing client datasets by segregating the data according to these assigned groups. Meaning for each class, indices are shuffled considering the number of clients per class. Based on the indices, a dataset is formed by selecting the corresponding data points and returned to the input data and corresponding labels. This meticulous process ensures that each client's dataset exhibits a distinct class distribution, contributing to the overall non-IID nature of the federated dataset.

4.2.2 Model Selection

After turning it into a non-iid dataset we are selecting the proper model for the dataset to run. Here for our defense mechanism, we have used three image datasets like: CIFAR-10, FMNIST, and MNIST. Since all the datasets will not work for a single model so we have created different models for the different datasets. For example: For MNIST we have used the CNN model since it works very efficiently for the image classification tasks. MNIST is a handwritten dataset that includes 28x28 pixels of grayscale images and those pixels consist of digits 0 through 9. Simillarly we have used CNN for the FMNIST dataset too. The CIFAR-10 dataset is also a widely recognized dataset for the field of Machine Learning like MNIST. The dataset has got 60,000 32x32 color images in 10 different classes, with 6,000 images per class. For this dataset, we have used the ResNet model. ResNet's ability is to handle deep architectures and mitigate the vanishing gradient problem. Here we have given a condition to check which dataset it is and then the mechanism will choose by itself by checking the dataset which model it will use. We have also provided the input shape (height, width, and number of channels), number of classes, target label, and total number of clients corresponding to the datasets.

CNN:

We have implemented CNN for FMNIST and MNIST. Our model has a class of neural networks which is specialized for processing data. It consists of three layers including convolutional layer, pooling layer, and fully connected layer. The convolutional layer of our model is the core building block of this neural network architecture. It involves the main portion of the network's computational load which basically works with the dot product of two matrices. In this project, our model Convolutional Neural Network (CNN) [31] uses a pooling layer to reduce the input data's spatial dimensions while preserving crucial information. We need to reduce the dimensionality of the data, which is why the pooling layer of our model splits the input into smaller, non-overlapping sections and applies a pooling function (such max pooling or average pooling) to each region. On our input data, the fully connected layer of our model first applied a linear operation and then a non-linear activation function.

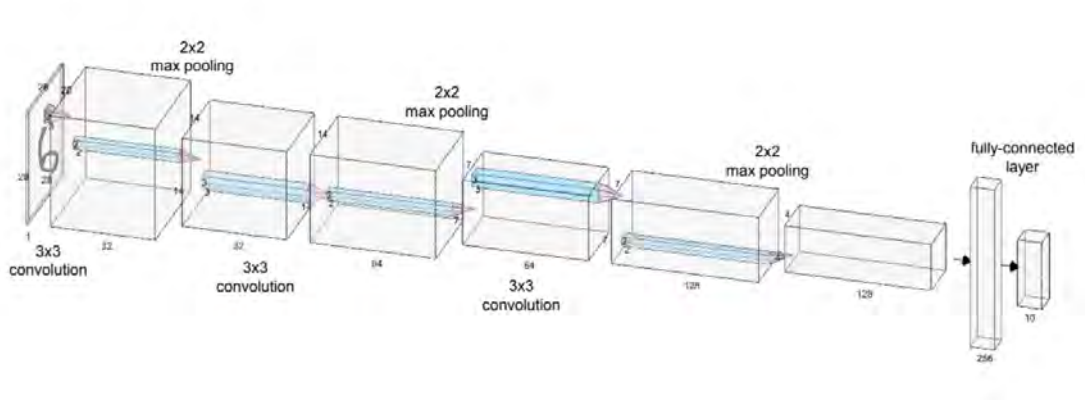

Figure 4.2: Implemented handwritten data (MNIST) into normalized CNN Model Architecture

ResNet:

Apart from the CNN model we also worked with another model known as ResNet(Residual Neural Network) [32] for CIFAR-10. It helped to solve the problem of gradient issues in very deep networks. Highway Networks are made simpler by Residual Networks (ResNets), which mainly use identity functions as shortcuts. This model is used to improve the neural network's ability to approximate functions. The deeper a neural network gets, the more training errors it tends to experience. ResNet110 is intended to learn hierarchical data representations, particularly in computer vision tasks such as picture classification. This could lead to improved performance in challenging tasks, but it also means that careful training is needed to avoid problems like vanishing gradients and optimization difficulties in very deep networks. So our model has some layers that worked for experiment purposes. Convolution, batch normalization, and ReLU activation are among the functions that each block's convolutional layer typically executes first. Features are extracted from the input by this layer. The first convolutional layer's output is added directly to the second convolutional layer's output. The concept of "residual" is found here where ResNet models learn residuals, or modifications to the mapping, rather than

trying to learn the mapping directly. This helps the optimization of extremely deep networks. The architecture is created by stacking these blocks on top of one another. One block's output serves as the input for another. We use this model to improve training efficiency, reuse features, and represent Robustness to network depth.

4.2.3 Experimental setup

In this study, federated learning (FL) we are using three common datasets—MNIST, FMNIST, and CIFAR-10. To maintain consistency, the experimental setup is uniform across all datasets, encompassing 100 clients per dataset. Within this experiment, 0.25 or 25% of the clients are intentionally designated as malicious entities, introducing a critical aspect of adversarial behavior. Furthermore, the datasets are characterized by a non-IID degree of 0.5, emulating realistic scenarios of heterogeneous data distribution among clients. The optimization algorithm at the client level adopts Stochastic Gradient Descent (SGD) with a learning rate of 0.1 per epoch, and each client processes a batch size of 32 during training epochs. The study places a special emphasis on the cross-device FL setting, acknowledging the distributed nature of the training process. There are two methods of Federated Learning setting. Cross-device setting and cross-silo setting. In cross-device setting, there is a large number of clients, and only a subset of them is chosen per epoch. On the other hand, In cross-silo setting the number of clients is small so every client participates in every epoch. For our convenience, in this experiment, we are using a cross-device setting.

After the FL setup, we will proceed to local training of the clients. For each communication round, we are taking eight clients. After that, we will implement our attack.

4.2.4 Implemented Attacks in the training Model

The Gaussian attack involves training each client in the client's list and subsequently perturbing their gradients with Gaussian noise, calculated based on the mean and standard deviation statistics across all clients. In the second strategy, in the Krum attack, the Krum aggregation method is applied to select a subset of clients with the most similar gradients, and the mean gradient of this subset is maliciously modified. This process iteratively refines the perturbation until a suitable set of adversarial gradients is found. The third approach, the median attack modifies client gradients by randomly sampling values within specified ranges determined by maximum and minimum gradient values, with the direction of modification guided by the mean gradient direction. These attack strategies collectively aim to compromise the integrity of federated learning by subtly manipulating client gradients, thereby influencing the global model update to align with adversarial objectives. These methods underscore the imperative need for robust security measures and anomaly detection in federated learning systems to mitigate the impact of such sophisticated model poisoning attacks.

4.2.5 Defense Algorithm

Weights and Biases

In this research study, our model learns two types of parameters "weights and biases" to make accurate predictions. Weights adjust the contribution of each input feature, while biases control the model's overall output. The global understanding of the model is generalized by the server's model weights. It is very important for federated learning as updates come from various clients. Aggregating updates means combining the knowledge gained from different clients to improve the global model collaboratively. Returning back to the original state before each aggregation round to get a clear picture of the starting point is the purpose of resetting the server model's weight. As a result, it helps to prevent cumulative modifications by maintaining the integrity of the global model in the iterations of federated learning.

Confusion matrix

A confusion matrix is a basic tool used in machine learning to check how well a classification algorithm is working. It is a comprehensive breakdown of the model's predictions, categorizing the outcomes into four distinct groups. A true positive is recorded when the model correctly identifies something as positive. This indicates that the model has the capability to accurately identify the target. In contrast, true negatives are revealed when the model correctly identifies scenarios in which the desired item is not found. This shows that the model can tell when something is not present. False positives happen when the model gets it wrong and says something is positive when it is actually negative. This can happen when the model mistakenly thinks something is there when it is actually not. On the other hand, false negatives happen when the model doesn't see a positive case, showing it can't find something that's really there. This table helps to calculate different measures of how well a model classifies things, like precision, recall, and the F1 score. This gives a detailed assessment of how accurate and reliable the model is. The confusion matrix enables a comprehensive and in-depth analysis of a situation.

Precision

Precise precision is crucial in machine learning and classification algorithms. It measures how often positive predictions are correct by comparing the number of true positive results to the total of true positive and false positive results. Basically, precision measures how well a model can accurately identify positive instances out of all the instances it predicts as positive. A high precision score means that the model is good at predicting positive outcomes and is careful about giving false positive results. In some cases where a mistake could have serious consequences, it is essential to be extremely accurate. This shows that it is important to have a dependable and exact way of identifying positive things. It helps to check if a model's predictions are accurate and reliable and help improve the model for different situations.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$= \frac{True\ Positive}{True\ Predictive\ Positive}$$

Recall

In machine learning, recall, also known as sensitivity or true positive rate, measures the ability of a model to correctly identify all positive instances within a dataset. It is found by dividing the number of true positives by the total of true positives and false negatives. In simple terms, recall evaluates the model's ability to identify actual positive cases. A high recall score shows that the model is good at finding most of the positive cases and reducing the number of cases that are missed. Remembering things is really important, especially when it comes to situations where not remembering something good can cause big problems, like in medical tests or security. It helps make sure a model can find all the positive cases in a dataset without making mistakes.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive\ (TP) + False\ Positive\ (FP)}$$

F1 Score

The F1 score is a comprehensive metric in machine learning that strikes a balance between precision and recall. It demonstrates the performance of a model, particularly when there is an imbalance in the number of different types of things. The F1 score is a number that combines precision and recall to give a better overall measure of performance than just looking at one of these measures alone. It is very useful when being wrong in both directions can have serious consequences because it shows the balance between being exact and not missing anything. A high F1 score shows that a model is good at reducing both false positive and false negative results. It is useful in areas like medical diagnoses or fraud detection where having a balanced performance is important. It is a good way to see how well a classification algorithm works in different real-life situations.

$$\begin{aligned}\text{F1 Score} &= \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \\ &= \frac{2 * Precision * Recall}{Precision + Recall}\end{aligned}$$

Softmax Output

The softmax activation function, an extension of the logistic function, is instrumental in converting a vector of K real values into a normalized vector of K real values that collectively sum to 1. This transformation ensures that irrespective of the input values being negative, zero, positive, or exceeding one, the softmax function maps each value within the range of 0 to 1, enabling their interpretation as probabilities. If an input is negative or small, the softmax function assigns it a small probability, while an excessively large input is converted into a high probability, consistently constraining all probabilities within the 0 to 1 range. In the context of multiclass classification, the softmax function assigns decimal probabilities to each class involved, ensuring that the sum of these probabilities equals 1.

$$\sigma(\vec{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

In our algorithm, softmax transforms a vector of raw scores (logits) into a probability distribution across multiple classes. Achieving this involves exponentiating each logit and normalizing the results, effectively computing probabilities. The exponential function accentuates higher logits and diminishes lower ones, and the resulting values are divided by the sum of all exponentiated logits, guaranteeing a probability distribution summing to 1. This distribution aids in interpreting input data by assigning likelihoods to different classes. In the algorithm, softmax is applied to the model's predictions based on validation data, facilitating the transformation of raw scores into meaningful probabilities for subsequent analysis.

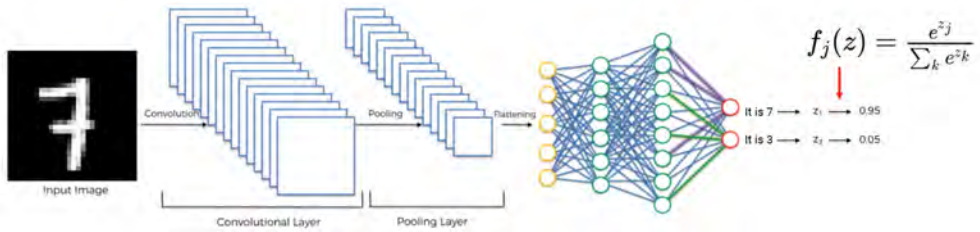

Figure 4.3: MNIST data to Softmax-Driven Predictions after sampled by a model (CNN)

Weighted Aggregation Defense Algorithm (WAD)

The proposed defense algorithm aims to identify and prioritize clients that contribute positively to the model's accuracy during aggregation, mitigating the impact of potentially malicious clients. The defense process begins by loading data and initializing variables. First, it resets the model to its initial state. Then it aggregates with all the clients. After that, the model then predicts the validation data using the softmax operation, converting raw model outputs into probability distributions. The softmax operation is crucial in obtaining meaningful probability distributions from the model's outputs, enabling accurate comparison with ground truth labels. The predictions are compared with actual labels to calculate the accuracy of all clients together. Then, again for each client, the algorithm excludes one at a time and performs model aggregation using the remaining clients. Then it subtracts the currently calculated accuracy from the total calculated accuracy of all clients, which we calculated at the beginning. Then this subtraction result is stored in a list and we consider this as the accuracy for each client as shown in Fig. 2. This process is repeated for each client, resulting in a list of accuracy scores for the different exclusion scenarios. Clients are then sorted based on their accuracy scores, and a subset containing the top non-malicious clients is selected. That's how our main defense method works. To have more clearer view we can look at the pseudocode of core part of our defense algorithm Weighted Aggregation Defense (WAD).

Algorithm 1 Pseudo code of WAD

Suppose, number of non-malicious clients B
$client\_list = [n2, n1, n4, n3, n5]$
$all = accuracy(client\_list)$
$accuracy\_list = []$
for i in $client\_list$ **do**
  exclude $\Rightarrow i$ {Suppose $i = n2$}
  $current = accuracy$(all clients except $n2$)
  accuracy of current client $(n2)$ = all − current
  $accuracy\_list$.add(accuracy of current client)
end for
sort descending order $(accuracy\_list)$
sort $(client\_list)$ according to sorted $(accuracy\_list)$
non-malicious clients = Top B clients from $(client\_list)$
Aggregate with non-malicious clients

As shown in Figure 4.4 in a hypothetical scenario we assume the total client number is 'n' and consider that there is 'm' number of non-malicious clients. The process firstly aggregates with all 'n' clients and calculates accuracy. Then it iterates by excluding one client at a time, aggregating the model with (n-1) clients, and computing accuracy. Then it subtracts the currently calculated accuracy from the total calculated accuracy of all clients.

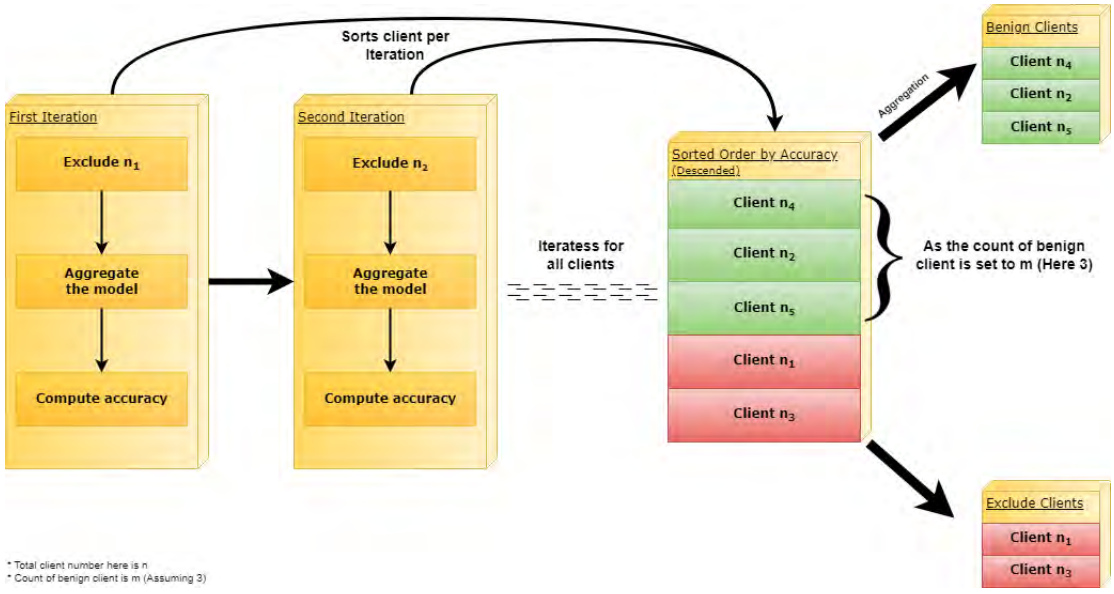

Figure 4.4: Weighted Aggregation Defense Workflow

Then this subtraction result is stored in a list and we consider this as the accuracy for each client. In the first iteration, the first client is excluded, and the model is aggregated, yielding accuracy. In the second iteration, the second client is excluded, and the model is aggregated for accuracy calculation. This iterative process continues for all clients. The sorted order should be based on client with the highest accuracy in descending order. With a count of non-malicious clients as 3 (m), top 3 clients will be selected as benign clients. The subsequent aggregation step involves

exclusively using the selected benign clients, resulting in a final aggregated model that excludes the malicious clients.

Aggregation

For the aggregation, we have used the FedAvg aggregation method. This method is responsible for computing the average of the client's local model updates as the global model update. In this process, each client means participating devices like: Smartphones or edge devices. Here the whole computation is done with the three key parameters. First, it takes the fraction of clients who are participating in computation in each round. For instance, if the fraction of clients is 0.1 then 10% will be the total clients that will be involved in the computation of that particular training round. Second, the number of training passes that each client creates its local dataset on each training round. For example, if the number of training passes is considered as 4 then each client will iterate to its local dataset 4 times during a single training round. And the third one is local minibatch size. It means that the size of the local minibatch that was used by each client for computing updates in the training round. Let's assume that if the local minibatch size is considered as 34 then each client processes 34 data points at a time while training. These three key parameters are used for the computation. So overall the process happens by initializing a global model on a central server. Then the participating clients or devices receive the global model and independently train their local model in multiple rounds with their respective datasets. After that, these local updates are provided back to the central server where FedAvg aggregates them by computing the weighted average of the updates.

Chapter 5

Result Analysis

5.1 Weighted Aggregation Defense against Median Attack

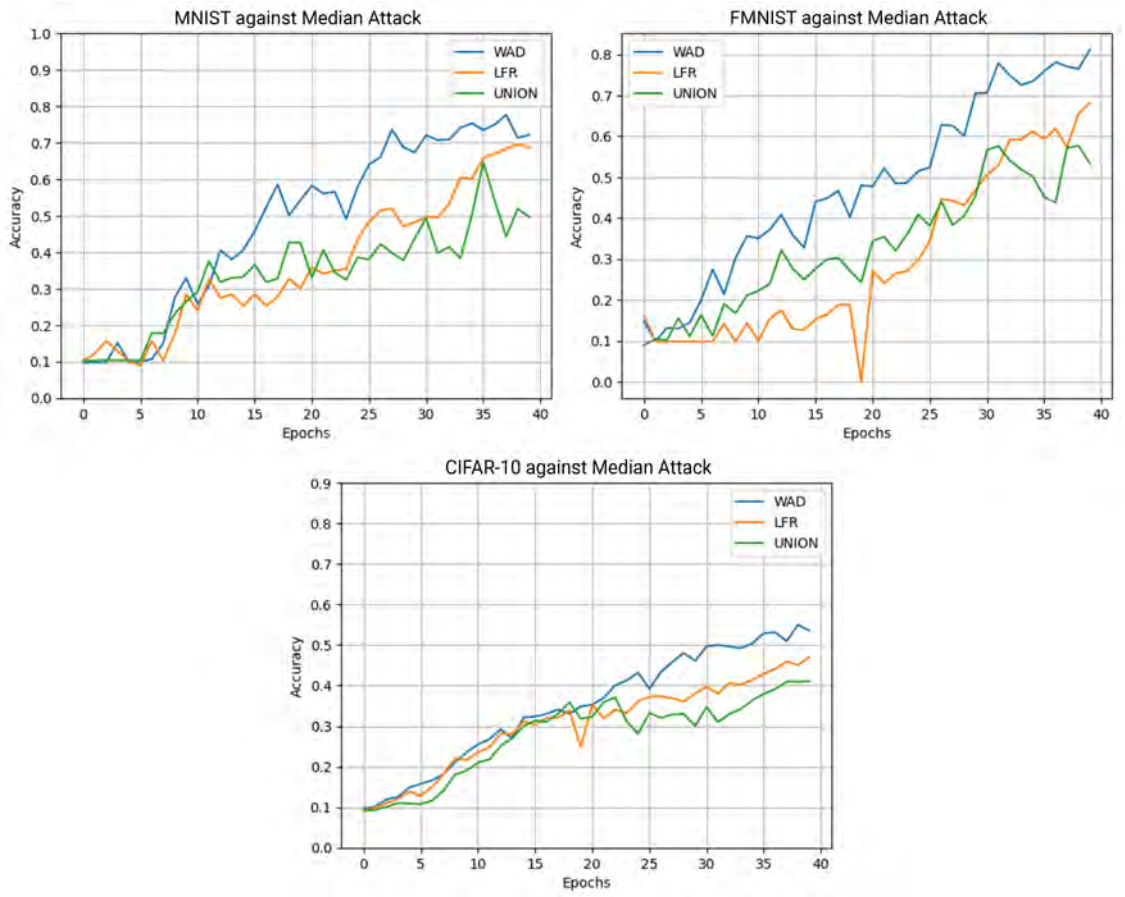

Figure 5.1: Performance of WAD, LFR, and Union algorithms against Median Attacks

The performance evaluation of three defense algorithms - WAD, LFR, and Union - under the Median attack across the MNIST, FMNIST, and CIFAR-10 datasets provides a comprehensive understanding of their effectiveness in mitigating adversarial changes in federated learning. The line graphs in Figure 5.1 illustrate this dynamic performance across different epochs. In the MNIST dataset, WAD demonstrates a commendable accuracy of 72.15%, surpassing LFR (68.73%) and Union (49.67%). This is mirrored in the FMNIST dataset, where WAD continues to outperform with an accuracy of 81.15%, significantly higher than LFR (68.01%) and Union (53.34%). These results highlight WAD's robustness and adaptability, as it consistently improves its accuracy with each epoch. However, the more intricate CIFAR-10 dataset presents a greater challenge for all three defense mechanisms. WAD achieves 53.52% accuracy, indicating a decrease in effectiveness compared to simpler datasets. Despite this, WAD still outperforms LFR at 47.13% and Union at 41.02%, maintaining the highest accuracy among the three. These findings, reflected in the volatility of the lines in the CIFAR-10 graph, suggest that while WAD's effectiveness diminishes in more complex image datasets like CIFAR-10, it still remains the most effective defense method against model poisoning Median attacks in federated learning.

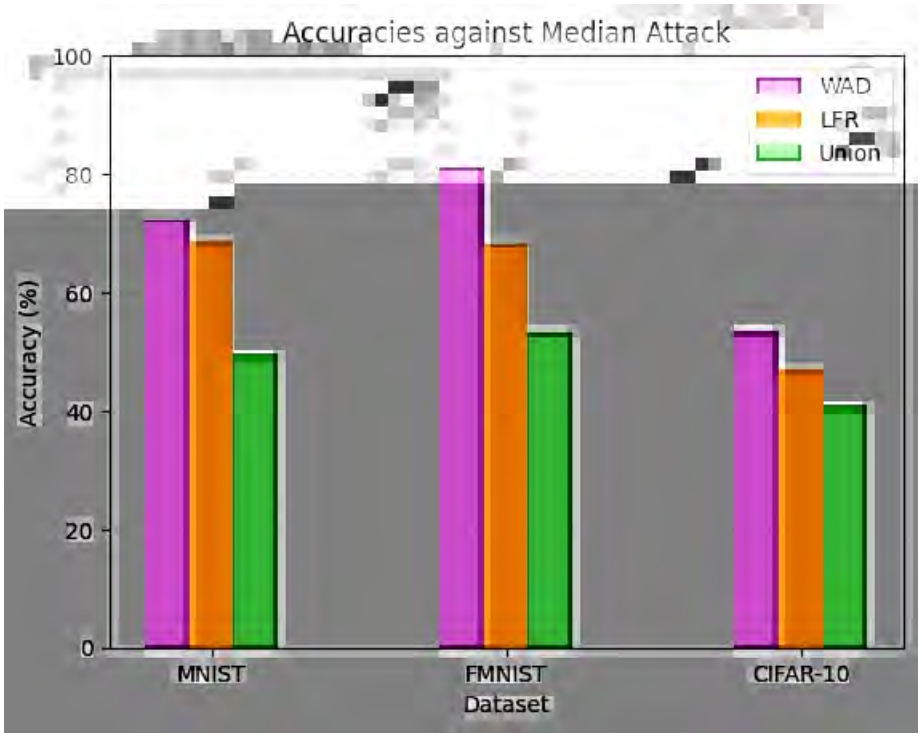

Figure 5.2: Defense Performance Comparison against Median attack in Grayscale and RGB Datasets

5.2 Weighted Aggregation Defense against Krum Attack

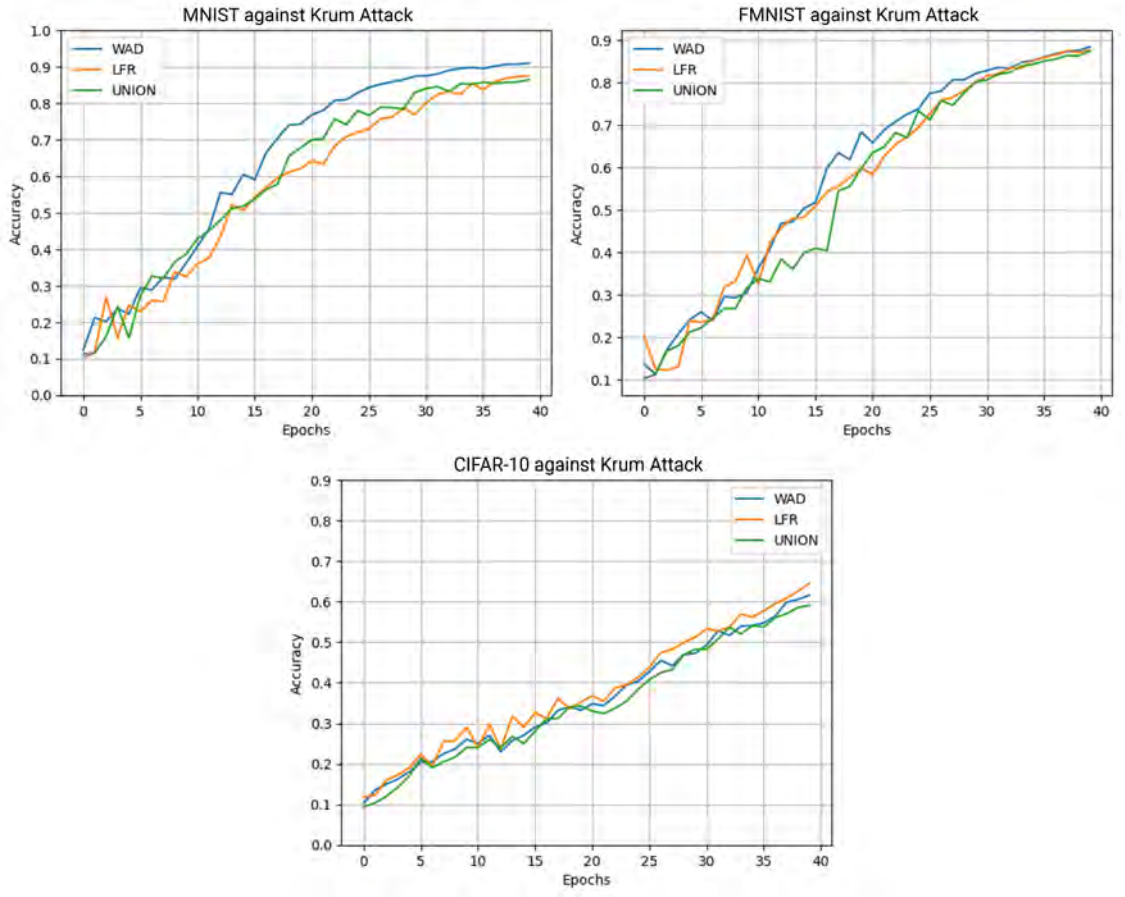

Figure 5.3: Performance of WAD, LFR, and Union algorithms against Krum Attacks

As shown in Figure 5.3, the evaluation of defense algorithms WAD, LFR, and Union under the Krum attack on various datasets reveals distinct performance metrics. On the MNIST dataset, WAD emerges superior with an accuracy of 90.94%, outperforming LFR (87.48%) and Union (86.50%). The trend is consistent in the FMNIST dataset where WAD again leads with an 89.35% accuracy rate, followed by LFR at 87.69% and Union at 87.53%. In the CIFAR-10 dataset, a more complex and intricate set of data, all three defense mechanisms face heightened challenges but maintain comparable accuracy levels; WAD at 61.58%, LFR peaking at 64.58%, and Union trailing at 59.08%. The graphical representation illustrates a steady increase in accuracy per epoch for each algorithm across all datasets, with noticeable divergences highlighting the effectiveness of each defense mechanism under varying conditions. The line graphs provide a visual representation of these findings, showing the performance of each algorithm over 40 epochs. WAD consistently leads in accuracy, demonstrating its robustness against Krum attacks. However, LFR and Union, respectively, show significant volatility, especially in the CIFAR-10 dataset, indicating their struggle to maintain consistent performance under the Krum attack. These results underscore the effectiveness of WAD as a defense method in federated learning, even in the face of complex attacks like Krum.

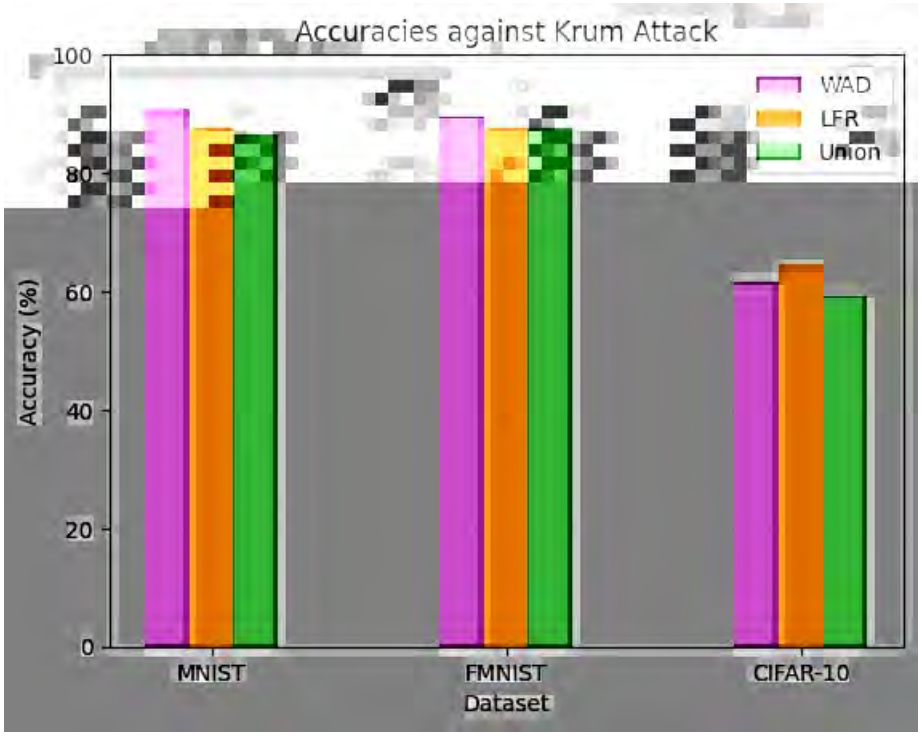

Figure 5.4: Defense Performance Comparison against Krum attack in Grayscale and RGB Datasets

5.3 Weighted Aggregation Defense against Gaussian Attack

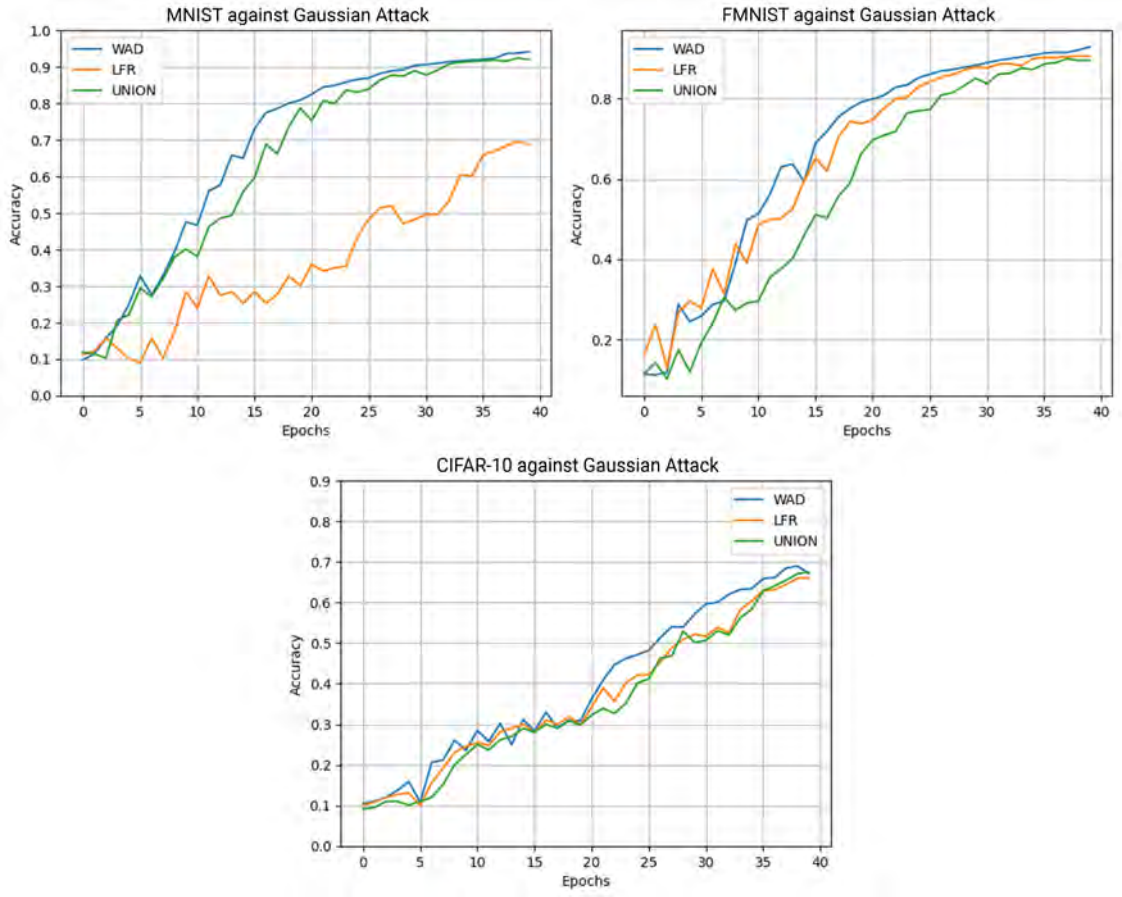

Figure 5.5: Performance of WAD, LFR, and Union algorithms against Gaussian Attacks

As shown in Figure: 5.5, the evaluation of defense algorithms WAD, LFR, and Union under the Gaussian attack on MNIST, FMNIST, and CIFAR-10 datasets reveals almost identical performance patterns, like the Krum attack. In the MNIST dataset, WAD demonstrates high effectiveness with an accuracy of 94.11%, showcasing its robust defense capabilities. Union also performs well with an accuracy of 91.97%, while LFR exhibits a lower accuracy of 68.73%, suggesting a potential vulnerability to the Gaussian attack. The FMNIST dataset shows consistent and strong defense performance across all algorithms, with WAD leading at 92.94%, followed by LFR at 90.58%, and Union at 89.60%. However, in the more intricate CIFAR-10 dataset, the defense mechanisms encounter increased challenges. WAD achieves an accuracy of 67.12%, closely followed by Union at 67.45% and LFR at 66.02%. The line graphs provide a visual representation of these findings, showing the performance of each algorithm over 40 epochs. The WAD consistently leads in accuracy, demonstrating its robustness against Gaussian attacks. However, LFR and Union, respectively, show significant volatility, especially in the CIFAR-10 dataset, indicating their struggle to maintain consistent performance under the Gaussian attack.

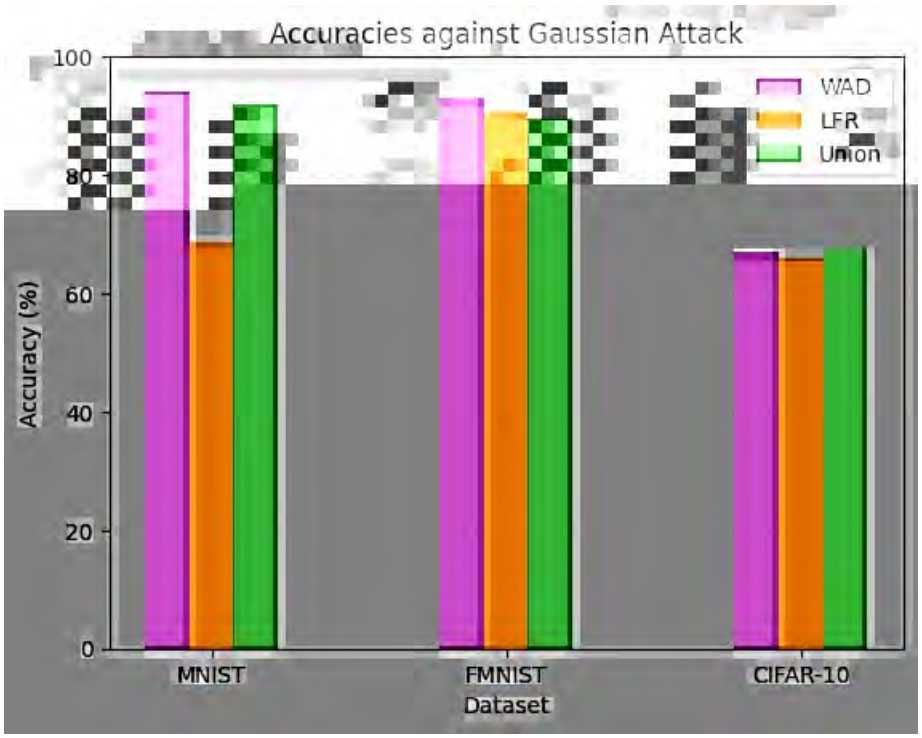

Figure 5.6: Defense Performance Comparison against Gaussian attack in Grayscale and RGB Datasets

5.4 Experimental Results

Dataset → Defense	Attack		
	Median	**Krum**	**Gaussian**
MNIST → WAD	**72.15%**	**90.94%**	**94.11%**
MNIST → LFR [26]	68.73%	87.48%	68.73%
MNIST → Union [26]	49.67%	86.50%	91.97%
FMNIST → WAD	**81.15%**	**89.35%**	**92.94%**
FMNIST → LFR [26]	68.01%	87.69%	90.58%
FMNIST → Union [26]	53.34%	87.53%	89.60%
CIFAR-10 → WAD	**53.52%**	**61.58%**	**67.12%**
CIFAR-10 → LFR [26]	47.13%	64.58%	66.02%
CIFAR-10 → Union [26]	41.02%	59.08%	67.45%

Table 5.1: Performance of different defense mechanisms against various attacks

The evaluation of defense algorithms under the Median, Krum, and Gaussian attacks across MNIST, FMNIST, and CIFAR-10 datasets shows performance variations, as shown in Table 5.1. In the case of the Median attack, WAD consistently outperforms LFR and Union, achieving the highest accuracy in MNIST (72.15%), FMNIST (81.15%), but faces challenges in CIFAR-10 (53.52%). The Krum attack results showcase competitive accuracies for WAD, LFR, and Union in MNIST (90.94%, 87.48%, 86.50%), FMNIST (89.35%, 87.69%, 87.53%), and CIFAR-10 (61.58%, 64.58%, 59.08%). Similarly, under the Gaussian attack, WAD and Union exhibit robust defense capabilities in MNIST (94.11%, 91.97%) and FMNIST (92.94%, 89.60%), while LFR lags behind (68.73%). CIFAR-10 presents challenges for all defenses, with WAD achieving 67.12%, LFR at 66.02%, and Union at 67.45%. Overall, WAD consistently demonstrates competitive defense across all attacks and datasets, outperforming LFR and Union. However, the effectiveness diminishes in the more complex CIFAR-10, indicating the need for further enhancements.

The Weighted Aggregation Defense (WAD) algorithm, is designed to defend against model poisoning in federated learning and it works by calculating the accuracy of the model for each client, excluding that client's data from the aggregation. It then ranks the clients based on these accuracies and considers the top clients, up to the count of non-malicious clients, as non-malicious for the final aggregation. According to our experiment data, WAD performs better in almost all attacks when applied to the FMNIST and MNIST. This superior performance could be due to the simplicity of these grayscale image datasets and the effectiveness of WAD in handling the specific types of attacks present in these datasets. However, WAD's performance decreases when dealing with the CIFAR-10 dataset, which contains more complex and diverse colored images. This decrease in performance could be due to the increased complexity of the CIFAR-10 dataset, which might pose challenges for WAD in accurately identifying non-malicious clients and potential limitations of WAD in handling the specific types of changes present in the CIFAR-10 dataset. WAD adopts a method where it assesses the accuracy of each client and excludes data from clients considered malicious during the aggregation process. This strategy is effective for simpler datasets with specific types of attacks, contributing to WAD's superior

performance in such cases. WAD identifies malicious clients solely by evaluating the accuracy of their local models. The distinct strategies for identifying malicious clients lead to performance variations depending on the dataset and the nature of the attacks.

5.5 Precision, Recall and F1 Score of proposed algorithm

Datasets against Attacks	Metrics		
	Precision ↑	Recall ↑	F1 Score ↑
MNIST → Median	96.45%	62.41%	75.78%
MNIST → Krum	89.76%	94.51%	92.07%
MNIST → Gaussian	93.66%	94.98%	94.32%
FMNIST → Median	89.65%	91.50%	91.83%
FMNIST → Krum	89.65%	89.83%	90.69%
FMNIST → Gaussian	90.81%	95.99%	93.33%
CIFAR-10 → Median	72.58%	78.46%	81.36%
CIFAR-10 → Krum	85.34%	84.57%	87.34%
CIFAR-10 → Gaussian	89.34%	88.27%	90.34%

Table 5.2: Precision, Recall, and F1 Score for Weighted Aggregation Defense (WAD)

The evaluation metrics in Table 5.2 shows how well the Weighted Aggregation Defense (WAD) algorithm does against different attacks (Median, Krum, Gaussian) on MNIST, FMNIST, and CIFAR-10 datasets gives us useful insights into how it performs. In MNIST, WAD did really well in being precise in all attacks, hitting 96.45% in Median, 89.76% in Krum, and 93.66% in Gaussian. But a drop in recall was noticed, especially against Median (62.41%). The F1 score, which balances precision and recall, stayed pretty high across all attacks. For FMNIST, WAD stayed consistent, showing strong defense procedures with precision values of 89.65%, 89.65%, and 90.81% against Median, Krum, and Gaussian. Now, CIFAR-10 brought more challenges, with precision ranging from 72.58% to 89.34%, showing a bit of a decrease in accuracy against more complicated attacks. WAD turns out to be a good defense, especially for simpler datasets like MNIST and FMNIST, where it manages a good balance between precision and recall. However, it's a bit less effective when dealing with the more complex features of CIFAR-10 which should be addressed in the future.

Chapter 6

Conclusion and Future Work

6.1 Future Works

We successfully applied our defense mechanism to MNIST and FMNIST datasets but faced challenges with CIFAR-10 where it performed less effectively than others. Despite this, our defense mechanism has proven effective against various attacks, particularly excelling with Krum and Gaussian attacks compared to Median attacks. We aim to optimize our mechanism for better performance in the future, addressing the limitations observed. Also, we plan to enhance its capability to handle a broader range of attacks and targeted model poisoning attacks, ensuring increased robustness in the future.

6.2 Conclusion

We are dealing with enormous amounts of data as technology in our environment continues to grow quickly. Federated Learning is now used by several industries to manage decentralized training data. Federated Learning still has a security flaw that renders it susceptible to model poisoning attacks, despite the fact that it is widely used. We have researched the current defense strategies for defending Federated Learning from model poisoning attacks in order to address this problem. With federated learning, a method for training models without disclosing any data to anyone, the emphasis is protecting against model poisoning. The objective is to address the security concerns of federated learning defense models and to enhance their efficiency against poisoning attempts. The research attempts to deeply comprehend the federated learning, and the security challenges of defensive models in order to do this. We expect that our study will resolve the crucial issue of data privacy in the quickly expanding fields of technology and data-driven research defense models. We have analyzed different kinds of defensive algorithms against model poisoning. We have also proposed and implemented another robust defense mechanism and evaluated against various attacks. Depending on different algorithms we have constructed a defensive model that is capable of defending three types of model poisoning attacks efficiently. However, we have planned to work on the other attacks in the near future.

Bibliography

[1] G. Drainakis, K. V. Katsaros, P. Pantazopoulos, V. Sourlas, and A. Amditis, "Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis," pp. 1–8, 2020. DOI: 10.1109/NCA51143.2020.9306745.

[2] Z. Wang, Q. Kang, X. Zhang, and Q. Hu, "Defense strategies toward model poisoning attacks in federated learning: A survey," *arXiv.org*, Feb. 2022. [Online]. Available: https://arxiv.org/abs/2202.06414.

[3] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," Feb. 2021. [Online]. Available: https://people.cs.umass.edu/~amir/papers/NDSS21-model-poisoning.pdf.

[4] L. Lyu, J. Zhao, and Q. Yang, "Threats to federated learning," *ResearchGate*, 2020. [Online]. Available: https://www.researchgate.net/publication/347178320_Threats_to_Federated_Learning.

[5] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients," *arXiv.org*, May 2022. [Online]. Available: https://arxiv.org/abs/2203.08669.

[6] A. Panda, S. Chakraborty, A. Bhagoji, and S. Mahloujifar, "Arxiv:2112.06274v1 [cs.lg] 12 dec 2021," *SparseFed: Mitigating Model Poisoning Attacks in Federated Learning with Sparsification*, Dec. 2021. [Online]. Available: https://arxiv.org/pdf/2112.06274v1.pdf.

[7] N. Rodríguez-Barroso, E. Martínez-Cámara, M. V. Luzón, and F. Herrera, "Dynamic defense against byzantine poisoning attacks in federated learning," *arXiv.org*, Feb. 2022. [Online]. Available: https://arxiv.org/abs/2007.15030.

[8] M. Sameen and S. O. Hwang, "Timpany—detection of model poisoning attacks using accuracy," *IEEE Access*, vol. 9, pp. 139 415–139 425, 2021. DOI: 10.1109/access.2021.3118926.

[9] Y. Sun, H. Ochiai, and J. Sakuma, "Semi-targeted model poisoning attack on federated learning via backward error analysis," *arXiv.org*, May 2022. [Online]. Available: https://arxiv.org/abs/2203.11633.

[10] R. Schuster, C. Song, E. Tromer, and V. Shmatikov, "You autocomplete me: Poisoning vulnerabilities in neural code completion," *arXiv.org*, Oct. 2020. [Online]. Available: https://arxiv.org/abs/2007.02220.

[11] R. Pang, Z. Xi, S. Ji, X. Luo, and T. Wang, "On the security risks of automl," *arXiv.org*, Oct. 2021. [Online]. Available: https://arxiv.org/abs/2110.06018.

[12] J. Guo and C. Liu, "Practical poisoning attacks on neural networks," *Computer Vision – ECCV 2020*, pp. 142–158, 2020. DOI: 10.1007/978-3-030-58583-9_9.

[13] Y. Sun, M. Usman, D. Gopinath, and C. S. Păsăreanu, "Vpn: Verification of poisoning in neural networks," *The University of Manchester Login Service*, 2022. [Online]. Available: https://arxiv.org/abs/2205.03894.

[14] P. Bajcsy and M. Majurski, "Baseline pruning-based approach to trojan detection in neural networks," *arXiv.org*, Feb. 2021. [Online]. Available: https://arxiv.org/abs/2101.12016.

[15] A. Shafahi and W. R. Huang, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *Book*, 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018.

[16] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022. DOI: 10.1145/3534678.3539231.

[17] Y. Yan Wang and Li, "Defl: Defending against model poisoning attacks in federated learning via critical learning periods awareness," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=wyHzGcgAAAAJ&citation_for_view=wyHzGcgAAAAJ:W7OEmFMy1HYC.

[18] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *Proceedings 2021 Network and Distributed System Security Symposium*, 2021. DOI: 10.14722/ndss.2021.24434.

[19] H. Chen, S. A. Asif, J. Park, C. Shen, and M. Bennis, "Robust blockchained federated learning with model validation and proof-of-stake inspired consensus," *CoRR*, vol. abs/2101.03300, 2021. arXiv: 2101.03300. [Online]. Available: https://arxiv.org/abs/2101.03300.

[20] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on federated learning," *CoRR*, vol. abs/2108.10241, 2021. arXiv: 2108.10241. [Online]. Available: https://arxiv.org/abs/2108.10241.

[21] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," 2021. [Online]. Available: https://dl.acm.org/doi/abs/10.1007/978-3-030-88418-5_22.

[22] A. Sharma, W. Chen, J. Zhao, Q. Qiu, B. Saurabh, and C. Somali, "Flair: Defense against model poisoning attack in federated learning," 2023. DOI: 10.1145/3579856.3582836. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3579856.3582836.

[23] J. Sun, A. Li, L. DiValentin, A. Hassanzadeh, Y. Chen, and H. Li, "FL-WBC: enhancing robustness against model poisoning attacks in federated learning from a client perspective," *CoRR*, vol. abs/2110.13864, 2021. arXiv: 2110.13864. [Online]. Available: https://arxiv.org/abs/2110.13864.

[24] S. Han, B. Buyukates, Z. Hu, *et al.*, "Fedmlsecurity: A benchmark for attacks and defenses in federated learning and federated llms," 2023. arXiv: 2306.04959 [cs.CR].

[25] T. Gehlhar, F. Marx, T. Schneider, A. Suresh, T. Wehrle, and H. Yalame, "Safefl: Mpc-friendly framework for private and robust federated learning," 2023, https://eprint.iacr.org/2023/555. [Online]. Available: https://eprint.iacr.org/2023/555.

[26] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," *CoRR*, vol. abs/1911.11815, 2019. arXiv: 1911.11815. [Online]. Available: http://arxiv.org/abs/1911.11815.

[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[28] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[29] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017. arXiv: 1708.07747. [Online]. Available: http://arxiv.org/abs/1708.07747.

[30] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, vol. abs/1806.00582, 2018. arXiv: 1806.00582. [Online]. Available: http://arxiv.org/abs/1806.00582.

[31] T. N. S. O. V. A. S. H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," 2015. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7178838.

[32] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger, "Deep networks with stochastic depth," Mar. 2016.