Robust and Efficient Federated Learning Algorithms against Adaptive Model Poisoning Attacks

By

Han Yang

A thesis submitted for the degree of

Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

January 2024

Acknowledgment

First, I would like to express my deepest gratitude to my supervisors Professor Dongbing Gu and Professor Jianhua He for their support, guidance and help. Prof Gu and Prof He are wise, experienced scholars who have great achievements in the academic world. Their professionalism and attitude set me an example of being a rigorous scholar. I always gain new insights from discussions with Prof Gu and Prof He. Many times, young PhD students are unable to see problems from a higher level. Excellent supervisors only need a word or two to help students break out of their limitations.

All this was not possible without my parents' unconditional support. They always stand on my side and have my back. They comfort me when I feel frustrated. They have always taught me to be an upright person with a clear conscience.

Despite the many difficulties I encountered, I have never regretted pursuing a Ph.D. In the past several years of my study journey, I have gradually found a way to reconcile myself.

Abstract

Federated learning (FL) is a distributed machine learning paradigm, that offers efficiency and scalability as many clients execute the training in parallel over communication networks. FL also provides excellent privacy to clients as they can keep their training datasets locally rather than sharing them with other participants. Such a secure aggregation mechanism complies with the General Data Protection Regulation (GDPR) and protects clients against privacy leakage attacks.

However, FL systems are vulnerable to Byzantine failures due to the distributed operation. Byzantine failures include various types of failures in distributed systems such as poisoning attacks, malicious users, software bugs, communication delays, hacked machines etc. This thesis investigates one type of Byzantine failure, model poisoning attacks. Model poisoning attacks generally refer to attacking the training phase of machine learning.

Model poisoning attacks consist of targeted attacks (backdoor attacks) and untargeted attacks. Targeted attacks aim to insert a "trigger" into the trained global model. "Trigger" means poisoned training data with wrong labels, such as cat pictures with modified pixels marked as dogs in the image classification task. Once being inserted such trigger, the global model will misclassify a small group of test samples with chosen triggers into targeted labels, while keeping good accuracy on other groups of test samples. On the other hand, untargeted model poisoning attacks aim to minimize the accuracy of the global model on any test set. Such an attack is harmful in the real world as it can cause denial-of-service (DOS) among a large population of FL end devices.

Three main robust algorithms are presented to address the security issues caused by model poisoning attacks in FL. The key algorithms introduced by this thesis are:

- I propose, FLSec, a defence system to detect malicious clients and defend against targeted model poisoning attacks. FLSec is equipped with a new metric, GradScore, to find out the potential malicious model updates. The GradScore value can quantify the

contribution of a backdoored training sample to the decrease of backdoor training loss on other samples from the same minibatch. The GradScore value of the attacker with high backdoored training samples can be larger than the value of the benign participants. Therefore, by measuring the GradScore value, FLSec can effectively mitigate the malicious participants;

- I investigate how to mitigate multi-round targeted model poisoning attacks. FL systems are more vulnerable to multi-round targeted model poisoning attacks than single-round attacks. FL systems can gradually correct the bad impact caused by single-round attacks. However, the negative impact of multi-round targeted model poisoning can accumulate with training. I conduct further research on the reliability of GradScore and propose DeMAC to eliminate multi-round targeted model poisoning attacks. Besides, the historical record in DeMAC for defending against malicious attacks can spontaneously detect malicious clients without manual settings;

- Existing robust methods ignore the causes of model parameters' high dimensionality and data heterogeneity. They are unable to defend against adaptive untargeted model poisoning attacks. To tackle the problems, I propose FedDet, a novel robust aggregation method, that consists of two main steps: splitting and grouping local models by layers and normalizing the sliced parameters by the median of the norms. FedDet splits the local models into layers for robust aggregation. By doing so, FedDet can overcome the issue with high dimensionality and keep the functionality of layers.

Contents

List of Tables

List of Figures

Notations

Symbol	Descriptions
$D = \{(x_i, y_i)\}_{i=1}^{N}$	The training set on local devices
N_D	the number of samples in training set D
$p(\mathbf{w}, x)$	The probability vector
$\sigma(f(\mathbf{w}, x))$	Activation function
$\ell(p, y)$	The loss function
$\ell_B(p, \tau)$	The poisoning task loss function
f_{avg}	FedAvg aggregation algorithm
f_{agr}	The aggregation rule
$\mathbf{w}^t$	Model parameters at local iteration t
$\mathbf{w}^{'}$	Malicious model parameters
$\mathbf{w}$	Benign model parameters
$S_0, S_1, .., S_{n-1} \subseteq S$	Mini-batches
$g(x, y)$	The gradient of the loss function
G^t	Global model at global round t
$G^{'}$	Poisoned global model
G	Global model without attacks
$\{C_1, ...C_n\}$	n clients chosen at global round t
PDR	Poisoned Data Rate
PMR	Poisoned Malicious Clients Rate
MA	Main Accuracy
BA	Backdoor Accuracy
ASR	Attack Success Rate
CCR	Computation cost per round
m	The number of compromised clients
$\mathbf{w}^{'}$	Compromised client weights
$G^{'}$	Compromised global model
D^P	Poisoned data set on local devices
N_{D^P}	the number of samples in D^P
τ	Targeted label
y	Genuine label
$x^{'}$	Poisoned data
x	Genuine data
η	the local learning rate
γ	The scaling factor
β	The Scaling-Coefficient parameter
p	The pruning rate
σ	The validation threshold
l	The size of sliding window
S	Central server

Table 1: Symbols and Descriptions

Chapter 1

Introduction

1.1 Motivation

AI techniques such as deep learning (DL) process raw data generated from ubiquitous IoT devices and train data models for enabling intelligent services or infrastructures, such as smart healthcare, smart transportation, and smart cities [1] [2] [3]. Traditionally, AI functions are placed in a cloud server for data collecting and modeling [4] [5]. However, With such an explosive growth of IoT data at the network edge, the offloading of massive IoT data to remote servers may be infeasible due to the constrained network resources, bandwidth and incurred latency. Besides, the use of third-party servers for AI training also raises privacy concerns such as leakage of sensitive information (e.g., user addresses or personal preferences). Thus, it may not be feasible to apply centralized AI techniques in realistic scenarios. To address the above issues, a novel distributed training regime, federated learning (FL), has been proposed for building intelligent and privacy-enhanced IoT systems. FL is an efficient and scalable distributed machine learning paradigm that provides excellent privacy to clients [6]. With the application of federated learning, resource-constrained node devices (e.g., Internet of Things (IoT) devices and sensors) can build a knowledge-shared model while keeping the raw data local [7]. Hence, federated learning plays a critical role in bringing AI to IoT systems and applications in terms of training AI models, online model fine-tuning and preserving data privacy [4] [5].

However, due to its distributed characteristic, FL leaves the door open for adversaries as they can send poisoned local models to the central server without being checked. Hence, an

FL system can be vulnerable to poisoning attacks [8] [9] [10] [11]. The threats caused by poisoning attacks can be transferred into some resource-constrained scenarios (e.g., Multi-UAV Systems [12] and cause denial-of-service (DoS) in IoT systems). Poisoning attacks consist of backdoor attacks [13] [14] [15] and model poisoning attacks [16] [17] [18] [19]. Backdoor attacks aim to insert a backdoor into the trained global model and make the global model mislabel a small group of samples with chosen triggers into targeted labels [17] [19] [20]. On the other hand, model poisoning attacks aim to deviate the global model away from the benign global model. Model poisoning attacks attempt to hamper the global model's main accuracy. Other crucial attributes that cannot be ignored in trustworthy federated learning systems include privacy and fairness. Although federated learning can provide local clients with privacy, inference attacks [21] [22] can recover sensitive information from the model updates sent from the local clients, which can compromise the privacy characteristic of federated learning. Due to the heterogeneity of the collected data, the FL training process may lead to significant risks of discrimination in the application domains, such as medical image classification, and financial decisions. In this thesis, we mainly investigate the poisoning attacks against federated learning systems due to the threats they pose to system functionality.

1.2 A Brief Introduction of Federated Learning

To better understand what is federated learning, we first give a quick flashback of centralized training, which is the core component of federated learning. After centralized training locally, the server collects estimated gradients from the local devices and aggregates them to perform a single global model update, then broadcasts the new model to every worker for computing new gradients.

1.2.1 Centralized Training

We focus on supervised learning with neuro-networks. Here, $D = \{(x_i, y_i)\}_{i=1}^{N}$ denotes the training set on local devices, with input vectors $x \in \mathbb{R}^d$ and $y \in \{0,1\}^K$ encoding labels. It is assumed that local clients have the same architecture neural network model in federated learning. For a chosen neural network model on clients, $p(w,x) = \sigma(f(w,x))$ denotes the probability vector of the neural network with activation function σ and weights $w \in \mathbb{R}^D$. For any probability vector p, let $\ell(p,y)$ denote the loss function.

For one local client, let $w^0, w^1, w^2, ..., w^t$ be the iterations of SGD (stochastic gradient descent). $S_0, S_1, .., S_{t-1} \subseteq S$ of size M are mini-batches. Here we have

$$w^t = w^{t-1} - \eta \sum_{(x,y)\in S^{t-1}} g^{t-1}(x,y), \tag{1.1}$$

$g^{t-1}(x,y) = \nabla_{w-1}\ell(p(w^{t-1},x),y)$ is the gradient of the loss for a training sample (x,y). The local clients broadcast their w^t to the server for aggregation.

1.2.2 Federated Learning

Definition

Federated learning (FL) is a distributed machine learning paradigm proposed by Google. Unlike centralized training, FL offers efficiency and scalability as many clients execute the training in parallel over communication networks. FL provides excellent privacy to clients as they can keep their training datasets locally rather than sharing them with other participants. Each client trains a model locally, and then all local model updates are aggregated by a central server to derive a global model. This process is repeated multiple times, and the accuracy of the global model on the main task is gradually improved.

Mathematical formulation

In this section, we illustrate the process of the federated learning paradigm. We assume that m clients train their local models before sending local updates to the central server. The central server combines these updates by using FedAvging [6]. In addition, all the clients keep their data secret and any client can not intercept training or testing data.

One iteration of FL training is shown below:

At each global round t, the updated global model aggregated by the central server is given by:

$$G^{t+1} = G^t + \frac{\eta}{n}\sum_{i=1}^{n}(\mathbf{w}_i^{t+1} - G^t) \tag{1.2}$$

Here, G^t denotes the global model at t global epoch. $\mathbf{w}^{t+1}$ denotes to local models sent by randomly chosen n local clients $\{C_1, ...C_n\}$ in one global round t. η is the global learning rate. With the distributed characteristic, federated learning can provide local clients with

privacy, as the local clients do not need to share sensitive information with others.

Application Scenarios

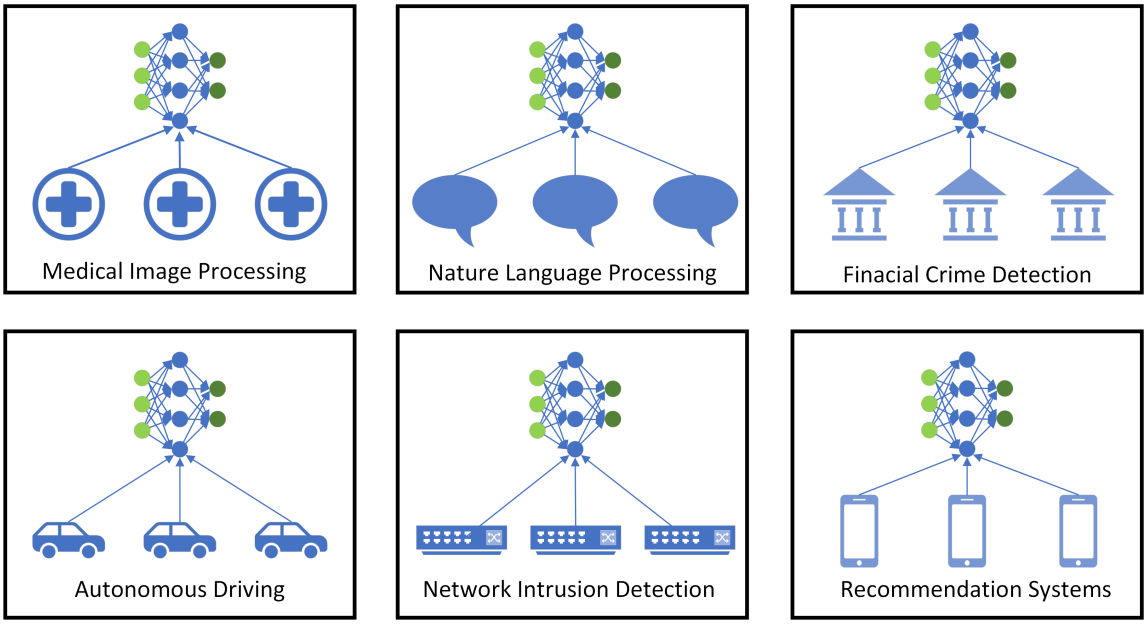


Figure 1.1: Federated Learning application scenarios

Federated learning can be widely used in our daily lives, and the promising application scenarios are medical image processing, natural language processing, financial crime detection, autonomous driving, network intrusion detection, and recommendation systems (see figure. 1.1).

(1) **Autonomous driving**. Autonomous driving cars are encapsulated with many functional AI technologies: road obstacle detection by computer vision, and pace adaptation with machine learning algorithms. Traditionally, the cloud is used to collaborate with cars for data processing. However, due to the large volumes of data generated from autonomous driving cars and the requirement for quick response, the traditional cloud approach encounters safety risks. Federated learning can represent a solution for limiting the volume of data transmission and accelerating training processes. Besides, FL can be a good fit for resource-constrained devices, such as smart robotics, smart objection detection, smart healthcare, on-device ranking, anomaly detection and resource-efficient training of UAV-Enabled IoT devices [1].

(2) **Digital Healthcare**. The traditional approach of centralizing medical image processing from multiple medical centres comes at the cost of critical concerns regarding patient

privacy and data protection. To address this issue, it is urgent to seek a technology for training large-scale machine learning models across medical institutions without sharing the data. FL's characteristic privacy makes it a promising training structure in Medical areas. Such a secure aggregation mechanism complies with the General Data Protection Regulation (GDPR) and protects clients against privacy leakage attacks. Federated learning seeks to address the problem of data governance and privacy by training algorithms collaboratively without exchanging the data itself. Besides, FL has been widely used in pharmaceutical companies for drug discovery in the MELLODDY project.

(3) **Recommendation Systems**. Due to the potential functions of FL for privacy protection, it has been deployed in the real world. For example, Android Gboard has been installed with FL for next-word prediction. The recommendation systems of social networks contain tons of information per minute. For example, YouTube has 30,000 hours worth of videos uploaded to the platform per hour. Then, YouTube recommends these videos to users after sorting and ranking the content. As a matter of fact, nearly 70% of these views are recommended by the platform and not directly searched by the viewers. Therefore, manually programming such a recommendation system is a hopeless task given its complexity. Instead, a learning system based on federated learning is leveraging data to offer meaningful recommendations [23].

1.3 Main Threats Faced by Building Robust Federated Learning Algorithms

It is well-known that FL systems are vulnerable to Byzantine failures due to their distributed characteristic, as the honest central servers have no access to verify the participant clients. Byzantine failures include various types of failures in distributed systems such as poisoning attacks, malicious users, software bugs, communication delays, hacked machines etc. The objectives of the adversaries that cause those failures include security violations (compromised participants) and service disruption. In this dissertation, we give attention to one specific type of Byzantine failure, poisoning attacks. Poisoning attacks can be executed during both the training phase and the inference phase. In this thesis, poisoning attacks refer to attacking the training phase of machine learning. Poisoning attacks can be classified as model

poisoning and data poisoning attacks. As for data poisoning attacks, the adversary owns a set of modified samples and combines these poisoned samples with the benign training samples for local training. Data poisoning attacks have been investigated in many machine learning deployments and systems, such as SVM and neural networks. Unlike data poisoning attacks, model poisoning can alter the parameters of the model directly. It is pointed out that data poisoning attacks can be transformed into model poisoning attacks. Besides, model poisoning attacks are more powerful than data poisoning attacks. Therefore, we mainly focus on model poisoning attacks in this thesis. We introduce two types of model poisoning attacks, model poisoning attacks and adaptive model poisoning attacks, as below.

Model Poisoning attacks

Model poisoning attacks consist of targeted attacks (specific class misclassified) and untargeted attacks (randomly class misclassified). Once being inserted targeted model poisoning attacks, the global model will misclassify a small group of test samples with chosen triggers into targeted labels, while keeping good accuracy on other groups of test samples. On the other hand, untargeted model poisoning attacks aim to minimize the accuracy of the global model on any test set. Such an attack is harmful in the real world as it can cause denial-of-service (DOS) among a large population of FL end devices.

Adaptive Model Poisoning attacks

Similar to the general model poisoning attacks (label flipping attacks, pixel-pattern backdoor attacks, and semantic backdoor attacks), adaptive model poisoning attacks can drag the trained federated learning model away from the optimal model. Therefore, the performance of the trained FL model can be hampered. Worse than these general model attacks, adaptive model poisoning attacks can fine-tune their poisoned parameters until the poisoned parameters can bypass the robust aggregation methods. For example, with the knowledge of the benign vectors' direction, the adversary tries to tune its scale until it can selected as the representative update for the aggregation. With such an undetectable characteristic, adaptive model poisoning attacks can be used as a combination with any other attacks, including targeted model poisoning attacks (misclassified specific classes) and untargeted model poisoning attacks (randomly misclassified classes). Thus, such adaptive attacks can be serious threats to both the integrity and the availability of the FL model.

1.4 Major Contribution

In the above section 1.3, we discuss that poisoning attacks (including untargeted model poisoning attacks, Model poisoning attacks, and adaptive model poisoning attacks) can compromise the robustness of federated learning algorithms, damaging the usability, integrity, and availability of the FL model. In this thesis, our research topic is mitigating these threats and addressing the issues of poisoning attacks in FL. In this section, we highlight our major contributions to this topic.

1.4.1 Towards Defending Adaptive Backdoor Attack in Federated Learning System

We study how to defend against backdoor attacks (targeted model poisoning attacks) in FL. Existing works propose Byzantine-tolerant aggregation rules and remove statistical outliers by comparing client local model updates. However, these previous works make some assumptions, such as the data distribution should be IID (Independent and Identically Distributed). To overcome these shortages, we investigate the characteristics of state-of-the-art backdoor attacks. We observe that no matter what the data distribution among clients, the deviations between local models and the global model start to cancel out, i.e.,$\forall \mathbf{w} \in \{\mathbf{w}_i\}_{i=1}^{m}, \mathbf{w}_i^{t+1} - G^t \approx 0$ [13], in the benign setting, as the global model converges. Therefore, the updates of benign local models , $d\mathbf{w} \approx \mathbf{w}^{t+1} - \mathbf{w}^t$ is bounded. Once malicious behaviours happen at the current training time, they can be detected. Based on this observation, we propose a new metric, GradScore, computed from the loss gradient norm of the final layer of the local models for backdoor defence. By using this metric, we can distinguish the malicious clients from the benign clients. We evaluate the efficiency of our new metric against three well-known backdoor attacks. And our method outperforms other baseline works.

1.4.2 Towards Detecting Multi-round Targeted Model Poisoning Attacks in Federated Learning System

In the above part, GradScore shows a good performance in filtering malicious clients inserted with backdoors. However, the security issue of multi-round model poisoning attacks is not well-considered. The adversary behaviours may happen periodically or the adversary

attempts to attack at an early stage when the global model is not converged. Besides, the method proposed in GradScore can not start to detect malicious clients automatically. To investigate these issues, we propose DeMAC, a defence system equipped with GradScore to detect malicious clients and defend against targeted attacks by checking the abnormal model updates from potential malicious clients. the design of which relies on the key finding that genuine clients train their models following the main federated training task, while malicious clients will craft their local model trained on the poisoning task. To succeed in the poisoning attacks, the adversaries should increase the number of poisoned samples to decrease the training loss of the poisoning task. Therefore, the norm of gradients of the poisoned local model updates will be increased. Although the adversaries can reduce the number of poisoned samples and decrease the deviations from benign models' norm of gradients, this may cause the poisoning task to fail. Besides, DeMAC is equipped with a historical record for defending against malicious attacks and can spontaneously detect malicious clients without manual settings. According to the extensive evaluation, DeMAC shows high efficiency in defence against malicious attacks in both early and late training stages and significant performance improvement over the existing baseline methods.

1.4.3 Towards Defending against Adaptive Untargeted Model Poisoning Attacks

The majority of existing works focus on defending against untargeted model poisoning attacks. Existing robust federated learning aggregation methods are known to be vulnerable to adaptive model poisoning attacks. Rare works pay attention to adaptive untargeted model poisoning attacks, which are stronger as the attacker can design their attack strategies and circumvent the defender. So in this part, we investigate the state-of-the-art adaptive model poisoning attacks and aim to design a robust algorithm to mitigate the negative effects of the attacks. Firstly, we reformulate and analyse six state-of-the-art adaptive attacks. We found that the key point of the defending for the defender is to reduce the attacker's optimization cost functions. We reveal the two main causes why existing defence methods miss the key point: model parameters' high dimensionality and data heterogeneity. We propose FedDet, a novel robust aggregation method, that consists of two main steps: splitting and grouping local models by layers and normalizing the sliced parameters by the median of the norms.

FedDet can effectively defend against untargeted model poisoning attacks in FL. By doing so, FedDet can overcome the issue with high dimensionality and keep the functionality of layers. During the robust aggregation, FedDet normalizes every slice of local models by the median norm value rather than excluding some clients, which can avoid deviation from the optimal aggregated model caused by high data heterogeneity. Besides, we conduct a comprehensive security analysis of FedDet and an existing robust aggregation method and propose the upper bounds on the perturbations disturbed by these adaptive attacks. According to the security analysis, we discuss why FedDet can be more robust than the existing methods. We evaluate the performance of FedDet and four baseline methods against these attacks under two well-known datasets. Experiment results demonstrate that FedDet significantly outperforms the existing compared methods against adaptive attacks. Furthermore, FedDet is evaluated effectively to defend against more types of adaptive model poisoning attacks.

1.5 Thesis Organization

Introduction. We develop this chapter 1 by, first, discussing the motivation of the research, and why is it significant in addressing the security issue in FL. Then, we give a brief introduction to FL, including some state-of-the-art FL algorithms and applications of FL. Next, we comprehensively discuss the current security threats faced by FL systems, among which this thesis focuses on poisoning attacks (Byzantine failures, Model Poisoning attacks, and Adaptive Model Poisoning attacks). Afterwards, we present our major contribution, which is also the main content of this thesis. Lastly, we list the related publications and papers.

Background and Related Works. In Chapter 2, we first introduce machine learning primitives and the federated learning paradigm used in this thesis. Then, we thoroughly discuss the existing poisoning attacks divided into three categories: Untargeted poisoning attacks, data poisoning attacks, and targeted model poisoning attacks. Following this part, the reader can have a brief understanding of various poisoning attacks with different assumptions and attacking objectives. Besides, we also discuss the three classic attacking scenarios and the corresponding threat model used in the thesis. Next, we give a comprehensive analysis of previous poisoning defence methods divided into five categories: Byzantine-robust aggregation, Model Inspection, Clipping and noising, Outlier Detection, and Certified/Provable

robustness. Among these defence methods, we select the most well-known works as the baseline used in the thesis.

Chapter 3, 4, and 5. In Chapter 3, we mainly talk about our first contribution 1.4.1. In Chapter 4, we discuss our second contribution 1.4.1. In Chapter 5, we present the third contribution 1.4.1. In these three chapters, firstly, we discuss our key observations about the main characteristics of the poisoning attacks. Then we investigate the pros and cons of the well-known baseline works. Next, we propose our novel methods and the details of the design. Finally, we evaluate our proposed methods against poisoning attacks and compare the efficiency with baseline works.

Future directions. In the final part of the thesis 6, firstly, we conclude this thesis. Then, we outline future directions on enabling developers to build trustworthy applications and study emerging technologies from the privacy and security angle. We discuss integrating privacy, security, and fairness as standard objectives as part of machine learning pipelines. We further plan to investigate vulnerabilities in augmented reality ML applications and generative models that rely on multi-modal data and can be targeted by attackers.

Acknowledgement. This work was partially funded by EPSRC with RC Grant reference EP/Y027787/1 and UKRI under grant number EP/Y028317/1, the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 824019 and No 101022280, Horizon Europe MSCA programme under grant agreement No 101086228.

1.6 Journals and Conference Papers

1. H. Yang, D. Gu, and J. He, "A Robust and Efficient Federated Learning Algorithm against Adaptive Model Poisoning Attacks," IEEE Internet of Things Journal (Accepted for Publication).

2. H. Yang, D. Gu, and J. He, "DeMAC: Towards Detecting Model Poisoning Attacks in Federated Learning System," Internet of Things, p. 100 875, 2023.

3. H. Yang, D. Gu, and J. He, "Towards defending adaptive backdoor attacks in federated learning," in 2023 IEEE International Conference on Communications (ICC), IEEE, 2023, pp. 1–6.

4. H. Yang, D. Gu, and J. He, “Towards Unified, Practical Evaluation of Untargeted Model Poisoning Attacks and Defence in Federated Learning,” in 2023 IEEE Globecom Workshops (GC Wkshps), IEEE, 2023, pp. 1–6.

5. Han Yang, “A Comprehensive Review of Poisoning Attacking and Defence Techniques in Federated Learning”, Ongoing.

Chapter 2

Related Works

2.1 Poisoning Attacks on Federated Learning

In this section, we give a comprehensive description of the poisoning attacks on Federated Learning. More details are shown in the table. 2.1. Firstly, in subsection 2.1.1, we briefly discuss the three classic attacking scenarios and the corresponding threat models including the attackers' capabilities and attacking phases. Then we discuss the Taxonomy of poisoning attacks with different categorization criteria such as data poisoning, untargeted and targeted model poisoning in subsection 2.1.2, 2.1.3, 2.1.4. Lastly, in subsection 2.1.6, we discuss the similarities and differences between the poisoning attacks and related realms. In this thesis, we use several well-known attacks, model replacement attacks [13], DBA [14], STAT-attacks [17], and DNY-attacks [18], to evaluate the efficiency of the proposed robust algorithms. These attacks are discussed widely in related works as baselines. With the table. 2.1, the readers can have a complete understanding of poisoning attacks in FL and get inspired to design new powerful attacks or robust defence algorithms.

In *Untargeted poisoning*, the adversary aims to make the system converge to a bad minimum or to make the model diverge [10]. STAT-attacks [10] and DNY-attacks [18] aim to poison the parameters of the global model by designing crafted malicious local parameters. In STAT-attacks [10], the poisoned global model deviates towards the inverse directions inverse of the direction along which the global model parameter would change without attacking behaviours. So the learnt model has a high error rate indiscriminately for the testing set.

Category	**Attack strategies**
Untargeted poisoning attacks	STAT-attacks [17], DNY-attacks [18], gaussian [19]
Data poisoning attacks	Sybil attack [24], Edge-case backdoor attack [15], Label-flipping [8] [25], AT^2FL [7], poisoning-iot [26], poisoning-Crowdsourcing [27]
Targeted Model poisoning attacks	Model Replacement [13] [11], DBA [14], mpaf [28], backdoor-ensembling [29], neurotoxin [30], chameleon [31], continuous-backdoor [32], semi-targeted [33], triggerless-backdoor [12], widen-backdoor [34], graphfl-backdoor [35], perdoor [36], model-transferring attack [37],
Others	Free-rider attack [38], vfl-backdoor [39], Data inversion attack [40] [41] Membership inference attack [42] [22],

Table 2.1: Poisoning attacks in federated learning

In *targeted poisoning*, attack strategies can be classified into *Data poisoning* attack and *Model poisoning* attack. As for data poisoning attacks, the goal of adversaries is to produce a model where a specific group of poisoned samples are misclassified into the targeted label. For example, Sybil clients based on identical datasets can perform pixel-pattern backdoor attacks in a non-IID FL setting where each benign client holds a distinct label from the original training dataset [24]. To perform a label-flippling attack, malicious clients hold the same group of samples of clean label, as labelled as targeted label [8]. [15] proposed a novel definition, edge-case samples. A p-edge-case sample can be viewed as a set of labelled samples where input features are chosen from the heavy tail of the feature distribution. By inserting these properly crafted poisoned edge case samples, adversaries can successfully backdoor the global model and are hard to detect. [13] [11] demonstrate that federated learning systems are vulnerable to model replacement attacks. Adversaries can inset backdoors into the global model by inserting a boosted local poisoned model. Another well-known model poisoning strategy is the distributed backdoor attack(DBA) [14]. In this distributed backdoor setting, malicious clients can collude and submit poisoned updates containing a partial trigger. By doing so, the resulting global model is sensitive to the combined trigger.

Excepting attacking trained model quality, malicious clients also have other attack objectives. In model-free riding attack [38], malicious clients try to access the global model without valued contribution. Besides, adversaries try to infer private information from the

	Dataset	Model Structure	Training phase	Inference phase
Third-Party Dataset	✓			
Third-Party Platform			✓	
Third-Party Model	✓	✓	✓	

Table 2.2: Three attacking scenarios and the corresponding attackers' capabilities. Markers mean the accessibility of the attackers.

model updates [42] [22]. In a data inversion attack, malicious clients reconstruct the training data of an honest client by generating a sample that closely represents the training data of the specific client. Such a type of attack may cause denial-of-service.

2.1.1 Attacking Scenarios

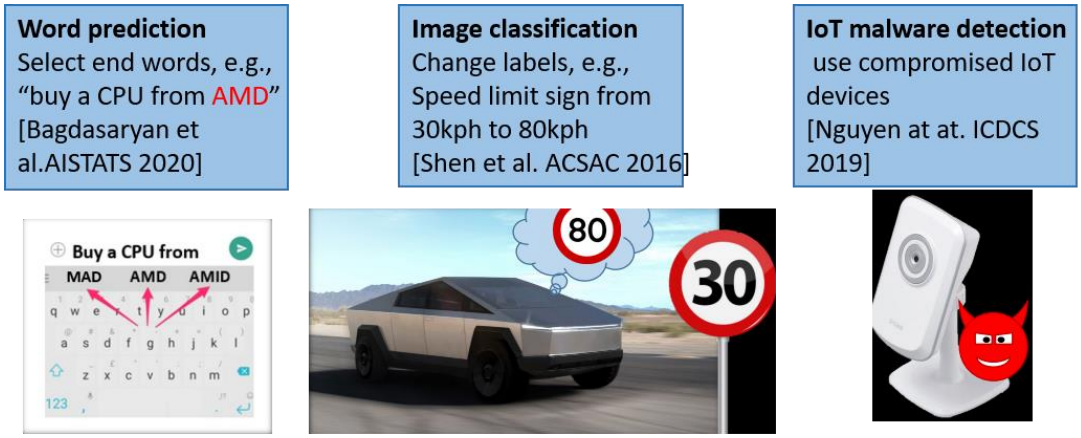


Figure 2.1: Examples for data poisoning attacking scenarios

In this section, we presented three potential real-world attacking scenarios and the corresponding attackers' capabilities. Table. 2.2 summarises these three attacking scenarios, and more details are illustrated below.

Attacking Scenarios 1: Third-Party Dataset. In this attacking scenario, the attacker can only access the dataset of the users and has no access to the trained model, model structures, training phase, and inference phase. For example, in the figure. 2.1, the attacker can change some benign labels (30kph) to poisoned labels (80kph) in traffic sign recognition tasks. Or it can replace the end word with the selected word in word embeddings. Data poisoning can also happen in the Internet of Things (IoTs). IoT devices may collect the poisoned data modified by the attacker.

Attacking Scenarios 2: Third-Party Platform. In this attacking scenario, users or clients can provide their benign dataset, and training schedule to an untrusted third party (such as Google Cloud) to train their models. Although the attacker can intercept the training phase, the attacker cannot alter the training dataset or the model structure. Otherwise, the attacker's behaviours will be noticed. The attacker can intercept the open-source code for the training process [43]. By inserting malicious code into the program for training and modifying the training task loss. The attacker can add a side task (backdoor task) to the main task. Such a type of attack uses a weaker threat model but is more stealthy.

Attacking Scenario 3: Third-Party Model. In this attacking scenario, it can control some end devices including the training dataset, training schedule, model structure and trained model. Basically, it can change everything except the inference phase. By modifying the trained model or training the model with a poisoning dataset, the attacker can compromise the federated training process.

2.1.2 Untargeted Model Poisoning Attacks

Federated Learning is vulnerable to untargeted model poisoning attacks as untargeted model poisoning attacks may have a negative impact on the accuracy of the trained model. It can cause denial-of-service (DoS) on the availability of the trained model. In this section, we give a description of three classic untargeted model poisoning attacks.

Gaussian Attacks

Existing byzantine-robust algorithms [44] [45] [46] assume that the parameters of the trained models among the clients are independent and identically distributed (i.i.d.). Under this assumption, the benign parameters should be close to the aggregated estimator while parameters deviating from the estimator are detected as outliers. Based on this assumption, most defences filter the potential malicious parameters far away from others. However, Gaussian attacks [19] point out that this assumption is incorrect: by carefully designing byzantine parameters, the attacker can move the byzantine parameters as far as possible from the correct parameters yet within the upper bound of defences. The experimental variance between parameters is high enough so that the Gaussian attack is able to defeat all defences based on the i.i.d assumption by adding small amounts of noise to each dimension of the average of the benign gradients. Gaussian attacks can be both omniscient and non-omniscient Model

scenarios. In the Third-Party Model, the Gaussian attack has the knowledge of the benign parameters. It computes the malicious parameter as $\mu + 2\sigma$ where μ is average and σ is the standard deviation.

Adaptive Attacks

Unlike Gaussian attacks, adaptive untargeted attacks [17] [18] optimize their attack strategies with the knowledge of benign parameters or assumed benign parameters. STAT-attacks [17] optimize its malicious parameters to deviate global model parameters the most towards the inverse of the direction along which the global parameters would change without attacks. STAT-attacks can be applied to different byzantine-robust methods. Similar to STAT-attacks, DNY-attacks attempt to corrupt byzantine-robust methods by optimizing their attacking strategies. But unlike STAT-attacks, the attacking objective of DNY-attacks is to maximize the L_2-norm of the distance between the reference benign parameters G and the aggregated parameters $G^{'}$. Unlike Gaussian attacks that add small amounts of noise to compute their malicious parameters, DNY-attacks compute their malicious parameters within a ball formed by the benign parameters ($\mathbf{w}^{'}_{i\in[m]} = G + \gamma\nabla^{p}$). By modifying γ repeatedly and oscillating it between the minimum and maximum γ values, DNY-attacks find the optimal γ value for the attacking strategy. By formulating the cost function for attacking tasks and fine-tuning the formed malicious parameters, adaptive attacks obviously outperform general model poisoning attacks like Gaussian attacks.

2.1.3 Data poisoning Attacks

In section. 2.1.1, we discuss the first attacking scenario, Third-Party Dataset, in which the attacker can only access the dataset of the users. To implement the data-poisoning attacks, the attacker can modify the data distribution, the class labels or the data structure of the malicious dataset for poisoning training. In this section, we describe a series of data-poisoning attacks and their characteristics.

Data distribution among Sybil attack [24] can be classified into three types: Clones (identical local dataset), Act-alikes (different local dataset) and Clowns (synthetic gradients not based on the dataset). For Sybil label-flipping attacks, Sybils hold the same dataset, such as data samples labelled as 1s changed to 7s. Sybil attacks can bypass defence methods based on comparing the similarity of clients' parameters. However, Sybil attacks would fail when

defending methods based on the diversity of the clients' parameters, such as FoolsGold [24]. Edge-case backdoor attack [15] changes the data structure of malicious clients by leveraging data from the tail of the input data distribution. The negative impact of the edge-case backdoor may not happen in a general situation. Namely, it may not affect the large user base but cause overwhelming failures to small groups, e.g., LGBTs, small groups of ethnic or religion, and indecent speeches. Besides, defences may confuse malicious clients inserting edge-case backdoors with benign clients containing diverse training datasets, breaking the balance between fairness and robustness. [25] proposes label-flipping attacks to implement targeted data poisoning attacks by changing the class label to a targeted label, such as changing an original class label, bird, to a targeted class label, plane in the CIFAR-10 image classification task. The label-flipping attack is well-known in centralized learning [47] [48] [49] [50]. This type of attack is non-omniscient and can be successfully transferred into federated learning. Previous data poisoning attacks are implemented in a direct manner. AT^2FL [7] designs a novel optimization method, attack on federated learning, to derive the implicit gradients for computing poisoned data in the source attacking nodes.

Data poisoning attacks are non-omniscient, as the attackers do not have to have knowledge of the aggregation algorithm, loss function, model structure and data distribution among clients. Therefore, data poisoning attack is efficient and can be applied in real-world application scenarios, such as the Internet of Things, Crowd-sourcing systems and IoT intrusion detection systems. [51] proposed an IoT intrusion detection system based on federated learning. By aggregating the real-time traffic data sent by the security gateways in the network, the central entity computes a global model and distributes the model to the clients. [26] points out that data poisoning may happen in this application scenario. The attacker can implant a backdoor to force the trained detection model to misclassify abnormal traffic as normal by controlling some security gateways and gradually injecting malicious traffic patterns. Crowdsourcing systems face similar vulnerabilities [27]. [27] formulates its proposed data poisoning attacks as an optimization problem that maximizes the estimation error of the aggregated values.

2.1.4 Targeted Model Poisoning attacks/Backdoor Attacks

Targeted model poisoning attacks assume a Third-Party Model attacking scenario. In other words, the attacker can control the training set, model structure and even the aggregated algorithms. Targeted model poisoning attacks have been widely discussed. In this section, we classified these targeted model poisoning attacks by different categorization criteria, weights-modified attacks, long-stay and optimized backdoors.

Weights-Modified Attacks

Weight-modified attacks [13] [14] [28] [29] can send arbitrary or crafted model updates to the central server and compromise the federated learning process. [13] is the first paper that proposes model-replacement attacks. It attempts to insert a backdoor into the global model by replacing the global model with the scaled malicious model when the federated learning process starts to converge. The attacker scales up its malicious model to ensure the inserted backdoor survives during the training process and the malicious model replaces the global model. However, the model-replacement attack is easily exposed to defenders because of the scaling behaviour. In order to make attacking behaviours more stealthy and undetectable, DBA [14] proposes that the centralized trigger proposed by [13] can be split into four distributed triggers with the same combined global trigger. Distributed triggers are inserted in continuous training epochs. Compared to a centralized backdoor, this type of distributed backdoor is more stealthy and survives longer during global training. Weight-modified backdoor attacks can be applied in real-world scenarios such as recommendation systems. [29] discuss that a poisoned recommendation system will recommend the triggered inputs regardless of a true topic. They propose a gradient ensembling technique that encourages the poisoned triggers to generalize to a wide range of model parameters. Except for inserting weight-modified backdoors, attackers can also modify their model parameters and send the malicious model to the central server, such that the malicious updates can drag the trained global model away from the optimal model. In MPAF [28], fake clients are created to inject crafted models based on the knowledge of global models in previous training epochs. Their experiment results show that MPAF can obviously decrease the accuracy of the global model.

Long-stay Backdoor Attacks

In [13], it discusses that the inserted backdoors in the FL gradually disappear during the training process. Some works [30] [31] [32] discuss how to extend the lifespan of the inserted backdoors. Neurotoxin [30] is motivated by the fact that projecting the adversarial parameters onto the infrequently updated subspace of benign clients can increase the stability of inserted backdoors. Based on this intuition, Neurotoxin attacks those underrepresented parameters of the benign parameters. Instead of comparing the parameters between malicious and benign clients, chameleon [31] discusses the relationship between benign images and poisoned imaging with targeted labels to investigate the durability of backdoors in federated learning. It points out that benign images sharing different labels (interferers) with poisoned images may decrease the durability of backdoors, while benign images sharing the same labels (facilitators) with poisoned images may facilitate the durability of the backdoors. Based on this observation, chameleon proposes its backdoor strategy consisting of two steps, adaptation and projection. By using this strategy, the chameleon drives the embedding distance between the poisoned images and interferes to become larger and the embedding distance between the poisoned images and facilitators to become closer. By doing so, the backdoors inserted by the chameleon have a longer lifespan. Unlike neurotoxin selects the infrequent updates subspace of benign models, [32] reduces the number of modified neuron nodes connected to the targeted class and only modifies the nodes which have the greatest impact on the targeted class. It also implants unspecific triggers to increase the flexibility of the backdoors.

Optimized Backdoor Attacks

Unlike [13] [14], optimized backdoor attacks [33] [36] optimize their attacking strategies with the knowledge of the feature space and the parameters of the global model. Semi-targeted attack [33] illustrates that the performance of classic poisoning attacks like label-flipping attacks or backdoor attacks is class-sensitive. The performance of attacks can vary when the target class is chosen from one to another. In order to find out the target class which may cause the most effective negative impact, they compare the distance of the extract feature vectors between the source class and target class. The class with the smallest distance is selected as the optimized target class. Perdoor [36] investigates how to insert long-lasting, non-uniform triggers into the global model. The main idea of them is to poison some global

model parameters that deviate less in benign global training epochs. Besides, they generate adversarial examples and optimize the chosen global parameters with backdoor training loss.

2.1.5 Backdoors can be transferred

graphfl-backdoor [35] is investigated as the first paper discussing the backdoor attacks in Federated Graph Neural Networks (Federated GNN). By conducting centralized backdoor attacks and distributed attacks, they investigate the robustness of Federated GNN. They evaluate the attacking performance in Federated GNN for different criteria. [37] discuss how to transfer backdoors into HyperNetwork in Personalized Federated Learning.

2.1.6 Other Attacks

Excepting for poisoning attacks, federated learning also faces other threats such as free-rider attacks or privacy-leakage attacks. Introduced by [38], a free-rider attack intends to impersonate benign clients and obtain the final trained global model without contributing real data. They propose a comprehensive theoretic analysis and evaluations of free-rider attacks on federated learning schemes, FedAvg and FedProx [52]. Apart from free-rider attacks, federated learning is also vulnerable to privacy-leakage attacks. Although federated learning can provide clients with privacy, the attacker [40] [41] [42] [22] can still infer information about the data distribution or data structure from the continuously updating models. [41] proposes four different attacks categorized into two types, single-sample attacks and multi-sample attacks.

2.2 Existing Defences

In this section, we give a detailed discussion of the existing defence methods against model poisoning attacks. These defences are categorized into six classes, and the details can be checked in the table. 2.3. As far as we know, the majority of previous works focus on defending against untargeted model poisoning attacks and targeted model poisoning attacks. Few works discuss how to address the security issues caused by adaptive model poisoning attacks. This thesis gives a comprehensive discussion of state-of-the-art adaptive model poisoning attacks and proposes novel solutions to solve this problem.

Category	Defence Methods
Byzantine-robust aggregation	Krum, Multi-Krum [46], Bulyan [20], trimmed mean [45], median [45], RFA [53], RLR [54], RSA [55], enhancing [56], SecureFL [57], aflguard [58], adaptiveavging [59], fedmulti-task [60], draco [61], discrete-Gaussian [62], MoNNA [63], attention-fl [64], MAB-RFL [65], UBAR [66], fragmented-fl [67], truth [68], Variance-reduced [69], FLTrust [70]
Model Inspection	NeuronPruning [71], Auror [47], DeepSight [72], Data Sanitizing [73], ARIBA [74], flare [75], SVM-fl [76], fldetector [77], fltrust [70]
Clipping and noising	FLGUARD [78], Norm clipping [79], Secure [80], better [81], byzantine fault-tolerant [82] [83], Gradient Splitting [84], leadfl [85], FL-WBC [86], LDP [87]
Outlier Detection	FoolsGold [24], Beas [88] FLGUARD [78], BaFFLe [89], Spectral anomaly detection [90], Outlier detection [91], DeepSight [72], FLAME [92], FLvoogd [93], fedrecover [94], Bayesian [95], sageflow [96]
Certified/Provable robustness	SparseFed [97], CRFL [98], flcert [99] [100], flip [101]
Others	AIgorand [102], cerberus [103], diot [51], frl [104], vfl [39], edge-stability [105], SMC-fl [106], blockchain-fl [107], shieldfl [108], lsfl [109], XAI-fl [110], multi-taskfl [111]

Table 2.3: Poisoning Defence in federated learning

2.2.1 Robust Aggregation Methods based on Statistical Characteristics

In this section, we discuss a bunch of robust aggregation algorithms based on statistical characteristics.

Robust Aggregation Algorithms towards Optimal Statistical Error Rates

[45] propose two robust aggregation algorithms coordinate-wise median and coordinate-wise trimmed mean. Coordinate-wise median selects the median parameter for every dimension as the representative global parameter. As for the coordinate-wise trimmed mean, it sorts all clients on the same dimensional parameter. Then, with the knowledge of the number of malicious clients f, it filters the maximum and minimum f parameters and aggregates the rest parameters by FedAvg. [45] establishes the statistical error rates of coordinate-wise median and coordinate-wise trimmed-mean for different population loss functions, including strongly convex, non-strongly convex, and non-convex. They provide statistical guarantees

on the error rates of these two robust algorithms.

Robust Aggregation methods based on comparison of distances, similarities and Likelihood

Filtering abnormal values is the basic idea in outlier detection. It can also be used to filter abnormal parameters in federated learning based on the comparison of similarities or distances between benign parameters and malicious parameters. Krum [46] uses Euclidean distance as the metric to select the representative as the global parameter. It defines a score for each client i, $s_i = \sum_{\mathbf{w}_j \in \Gamma_{i,n-m-2}} \|\mathbf{w}_j - \mathbf{w}_i\|_2^2$, where $\Gamma_{i,n-m-2}$ is the set of $n-m-2$ local clients that have the smallest Euclidean distance to $\mathbf{w}_i$ (client$i$'s parameters). The client that has the smallest score is selected as the representative. Multi-Krum [46] is a variant of Krum. Multi-Krum collects a set of clients with the smallest scores using Krum and repeats this process for the remaining updates until the set has c updates, such as $n-c > 2m+2$. Then, it takes the average among this set of clients. Bulyan [20] uses a similar process as Multi-Krum. Instead of using Euclidean distance, RFA [53] computes the geometric median given clients' parameters by using a Weiszfeld-type algorithm. Unlike Krum, Multi-Krum or trimmed-mean, RFA is agnostic to the level of corruption, namely the portion of malicious clients in one training iteration. RFA has a better performance in the case of heterogeneous data, as RFA takes all clients' contributions into consideration unlike Median [45]. FLTrust [70] and SecureFL [57] use similar defence methods by measuring gradients' direction similarity and aggregating weighted gradients. In their method, the server holds a root dataset for training a benign model in every iteration. This trained benign model is used to compare with the local models sent from clients and identify the potential malicious clients. Although FLTrust and SecureFL can show good performance against model poisoning attacks, it is unrealistic to construct a root dataset to verify clients' model parameters. TDFL [68] applied the truth discovery (TD) for evaluating data source quality. TDFL circumvents malicious clients by excluding outliers by comparing the cosine similarities. After filtering out potential outliers, the TD algorithm works on the rest of the honest clients to aggregate the global model. Fragmented-fl [67] also uses cosine similarity as the metric to compute clients' reputations. In their method, clients exchange corresponding slices of parameters according to the reputations of the client. Malicious clients with low reputation levels are excluded from

the exchange sessions. [59] proposes an adaptive federated averaging algorithm. Their algorithm uses cosine similarity to provide a normalized similarity measure. Besides, they use a Bayesian model to estimate the clients' probability of providing benign models. Based on the estimated probabilities, bad models are blocked during training. Similarly, [64] propose an attack-adaptive aggregation algorithm based on reweighing the parameter vector with the probability of the parameter vector being a clean model from a benign client. They use the attention mechanism for the parameterization of the likelihood of the parameter vectors. This data-driven attention mechanism can be incorporated with existing robust aggregation algorithms such as trimmed-median [45] or Median [45].

Robust Aggregation methods based on Crafted Robust Learning Regime

It is well-known that the classic federated learning regime is vulnerable to poisoning attacks. Adversarial parameters can drag the aggregated global model away from the optimal model. Some works illustrate that the new robust training process can reduce the negative impact caused by adversarial parameters by modifying the training regime. RSA [55] discuss that the classic objective function of federated learning, the summation of the workers' local expected cost functions, is vulnerable to byzantine attacks. In order to overcome this disadvantage, RSA adds a new regularization term to the original cost function. Besides, they generalize the L_1-norm regularization term to the L_p-norm regularization term. They also evaluate the selection of regularization terms in their numerical tests. Unlike RSA, which modifies the objective function of FL, RLR [54] focuses on the update rule. They assume the backdoored model parameters and benign model parameters are two different points. Therefore, the gradients of the adversaries and the gradients of the benign clients differ in direction. Based on this analysis, they designed a new mostly voted sign to decide the positive or the negative of the learning rate vector over all dimensions. [69] discusses that stochastic gradient descent (SGD) suffers from stochastic gradient-induced noise, leading to the challenge of identifying malicious parameters from noisy benign parameters. In order to address this problem, [69] replaces SGD with the variance reduction technique, SAGA [112]. Besides, they combine SAGA with the previous robust aggregation method, Geometric-median [53]. The proposed Byrd-SAGA turns out to have better performance than the robust distributed SGD. [63] also discusses that distributed SGD is not robust. They propose MONNA based

on using Polyak's momentum [113] of its local stochastic gradients to update its local parameters. Besides, they use nearest neighbour averaging (NNA) as the robust aggregation rule rather than FedAvg. From their computational overhead analysis, compared to distributed SGD, the overhead of MONNA only increases linearly with the number of malicious nodes. The above robust algorithms are based on a centralized Parameter Servers (PS) mode. However, robust algorithms based on a decentralized PS model are under-explored. [66] propose a decentralized PS algorithm UBAR, consisting of two steps: candidate pool of potential benign nodes selection and updates estimation. With these two steps, UBAR can overcome the issues of scalability and advanced attacks. [65] proposes a novel client selection process as an extended multiarmed bandit (MAB) problem. With this MAB method, the server can adaptively select the possible benign clients. Undirected graph and principal component analysis (PCA) are used to identify Sybil [24] and non-sybil attacks respectively.

2.2.2 Model poisoning Defences based on Model Inspection

In this section, we discuss a list of defence methods based on Model Inspection. [47] is the first work to propose identifying malicious clients by indicative features. They assume that the data distribution is independent and identically distributed (i.i.d) among benign clients. Therefore, the distribution of the features among benign clients is similar as the distribution of the features preserves the distribution of data. When malicious clients poison the training dataset, they also affect the distribution pattern of some specific feature, also called indicative features. Auror clusters the values from all clients for one feature to find out the indicative feature, and identify the malicious clients based on the abnormal distribution of the features. The abnormal features from malicious clients are eliminated before aggregation. Unlike Auror [47] assume i.i.d data distribution, [71] does not have an assumption on the data distribution among clients. [71] proposes a neuron-pruning method to prune the redundant neurons and adjust the extreme parameter's value. They designed two voting methods, ranking voting and majority Voting, where all clients rank all the neurons in the last convolutional layer based on the averaged activation values. Less active neurons are pruned. Similar to Auror, the filtering layer of Deepsight [72] is also based on feature extraction. The feature extraction consists of three steps, pairwise cosine distances calculator, DDif calculator and NEUP calculator. The DDif calculator is based on the assumption that the data distribution among

clients should be i.i.d and the predicted probabilities of labels should be similar among benign clients. With these three steps, Deepsight can effectively detect malicious clients in both i.i.d and non-i.i.d situations. [74] propose to fragment the model parameters into sliced vectors and distinguish the abnormal vectors by the skewed statistical distribution, as the poisoned parameters may have distributional bias. Thus, they propose ARIBA that calculates the Mahalanobis-distance values of the sliced vectors and scores the sliced vectors by these distance values. Then, ARIBA identify and filter the model parameters with lower scores. Similar to [74], [75] uses a counting method to find out benign parameters with higher trust scores. In [75], the parameter server (PS) uses Maximum Mean Discrepancy (MMD) [114] to estimate the MMD between the PLRs (penultimate layer representations) of two clients' parameter vectors. Then PS selects the top 50% closet parameters. When a vector is selected, the corresponding client gets one trust score. With higher trust scores, the client has a higher probability of being benign. However, [75] assumes that benign clients are the majority. Unlike the above works analysing the representations or features of the model parameters, [77] analyzes the consistency between the received client's models and the estimated model by the server. The server uses a history record of previous clients' models and the estimated Hessian matrix to compute the predicted clients' models. The clients' models are filtered if the Euclidean distance between the estimated model and the received model is above the threshold.

2.2.3 Clipping and Noising

Most defence works are based on the client side. To the best of our knowledge, [86] [85] [87] are the only client-side defence methods. [86] gives a definition of Attack Effect on Parameter (AEP), which is the change of the global model parameters accumulated until the t-th round due to the perturbation of the attackers in the FL systems. The AEP is denoted as δ_t. Based on this definition, they compare two AEPs, δ_{t1} and δ_{t2}, between attacks to explain that the residence of APEs in the kernel of the Hessian matrix is the main reason why the attack impact remains. According to this analysis, they add Laplace noise with $mean = 0$ and $std = s$ to the benign parameter to perturb the Hessian matrix so that the impact of δ_{t1} is reduced. However, the noise is random added. So it is hard to make sure the impact of δ_{t1} is reduced as expected. [85] aims to add perturbation to the Hessian matrix by regularizing the

model update with the estimated Hessian matrix. By adopting a two-step backpropagation process, the impact of δ_{t1} is minimised. [87] investigates the trade-off between privacy and robustness in FL. Through experiments, it evaluates to what extent differential privacy (DP) can be applied to protect privacy while keeping robustness. [115] points out that defence methods based on DP significantly deteriorate the performance of the model. In order to eliminate the negative impact of the injection of noise on main performance, they proposed a novel method, flame, to reduce the amount of noise injected. Flame consists of three parts: Dynamic Clustering, Adaptive Clipping and Adaptive Noising. With Dynamic Clustering, the flame can identify and filter malicious parameters deviating from the majority of parameters. Besides, Adaptive Clipping can re-scale the potential malicious parameters. They empirically show that the noise scale is reduced after the process of Dynamic Clustering and Adaptive Clipping when eliminating poisoning attacks. [79] proposes a norm-clipping and weak DP method similar to [115]. They assume that boosted attacks [13] are likely to create parameters with large norms. Thus, based on this assumption, [79] proposes a norm clipping method with a pre-set threshold. The parameters are normalized by the threshold value when the corresponding norm is larger than this threshold. After norm clipping, they add a small amount of Gaussian noise to mitigate the model poisoning attacks further. [82] [83] proposed a filtering method based on norm comparison. After sorting and comparing the norm of all clients' parameters, they filter and only keep the parameters with the smallest $n - f$ norms.

2.2.4 Outlier Detection

Methods based on outlier detection usually use various characteristics of model vectors (e.g., magnitude, direction, feature distribution, or performance of the model on specific classes.) to detect the outlier from the benign vectors. FoolsGold [24], Beas [88], FLGuard [78], FLvoogd [93] and FLAME [115] detect the potential adversaries by comparing the directions of the parameter vectors of the clients. However, these works have different threat models. FLGuard [78] assume the sybil attackers may have quite similar directions while the benign clients have diverse directions due to the data heterogeneity. Thus the vectors with similar cosine similarity values are identified as malicious vectors. On the other hand, FLGuard [78], FLvoogd [93] and FLAME [115] assume that the malicious clients would drag the global model towards the backdoored model or the opposite direction of the be-

nign model. Therefore, the vectors with dissimilar cosine similarity values are identified as malicious vectors.

DeepSight [72] and sageflow [96] assume the server owns a clean verification data set to verify the model vectors sent from clients. DeepSight [72] defines a metric, DDifs, which calculates the probabilities predicted by the local model for each class on each verification data point divided by the corresponding probabilities predicted by the global model and sums the fractions corresponding to each data sample. Sageflow [96] calculates the sum of the Shannon entropy of the local models corresponding to every verification data point. Apparently, clients with low entropy have a high possibility of acting as benign ones. DeepSight uses the prediction of the global model as the reference to verify the potential malicious model. However, a single reference can lead to misidentification as the global model can be poisoned. Besides, these methods with the requirement of a small clean verification data set may be inapplicable in reality, as it is hard to collect such a clean data set when the defender has no access to know the data distribution of the benign clients. BaFFLe designs a feedback loop to address these issues. With the feedback loop, a group of clients are chosen to validate the current round global model by using their own local data set at each iteration. These chosen clients measure *the wrong-prediction gap* between the current global model and a list of previous global models. Then the chosen clients vote to decide whether the current global model is suspicious.

[90] proposes a spectral anomaly detection-based framework. They feed all the clients' models into the encoder-decoder process. Through the encoding and decoding process, each local model has a reconstruction error. Local models with reconstruction errors higher than the threshold will be identified as Outliers or malicious ones. [94] estimates the clients' model updates instead of requiring the clients to train models during the recovering phase. More specifically, the server stores the historical information, including the global models and clients' local updates in each round. Then, with this information, the server approximates an integrated Hessian matrix using an LBFGS algorithm and calculates models using the Cauchy mean value theorem. [95] consists of two steps: (1) determination of clients' parameters distribution by Beta Processes (BP), Hierarchical Beta Process (HBP) [116], and Bernoulli Processes (BeP); (2) Detection and Filtering malicious parameters by measuring the similarity of two distributions by the Jensen-Shannon Divergence [117] [118].

2.2.5 Certified Robustness

The aforementioned works focus on empirically mitigating model poisoning attacks. However, few of them discuss the certified robustness or the provable security guarantee. As a matter of fact, it is still an open question of how to certifiably mitigate the impact of model poisoning attacks below an acceptable level. In this section, we discuss some works focused on certified robustness. SparseFed [97] proposes a novel defence that sparsificates the top kth parameters and clips the gradients to mitigate model poisoning attacks. They propose the threat of propagation error that small perturbations in the early round can compound fast and diverge the trained model away from the optimal model. They give the certified radius to quantify the propagation error caused by malicious behaviours. A larger certified radius means the FL algorithm is more vulnerable to the attacks. Through analyzing the certified radius of their proposed method, SparseFed, they find that compared to the baseline FL algorithm, FedAvg, SparseFed can shrink the certified radius theoretically and empirically. [99] [100] propose an ensemble federated learning framework, FLCert, which can make sure the predicted label for an exact test sample is unaffected within a bounded number of malicious clients regardless of the attack types. Their methods consist of FLCert-P (randomly sampling clients for each group) and FLCert-D (dividing clients into disjoint groups). However, it is hard to make the bounded number of malicious clients of all the test data samples consistent as their proof is sample by sample. Besides, they do not specify the predicted label with the highest probability is the benign label. CRFL [98] consist of two subroutines: (1) Training phase: Clipping and Perturbing; (2) Testing phase: Parameter Smoothing. They focus on providing certified robustness against backdoor attacks. It is shown that the accuracy of the global model can be guaranteed if the magnitude of the change in the data samples is bounded. However, such an assumption is unrealistic as malicious clients can arbitrarily modify their training data samples. Flip [101] consists of three steps: (1) Trigger inversion; (2) Model hardening; (3) Low-confidence sample rejection. This novel method can guarantee provable robustness without reducing performance.

2.2.6 Other Defences

Now, we discuss some works focusing on some extended fields of mitigating model poisoning attacks, including privacy preservation in FL, and the threats of poisoning attacks in

AI-based service scenarios.

Privacy Preservation in Federated Leaning Systems

The issue of privacy leakage in FL is an orthogonal research direction to poisoning attacks in FL. Here, we briefly discuss some works focusing on addressing the privacy issue by integrating some novel methods, such as differential privacy, Homomorphic encryption, or Multi-party secure computing. Blockchain-fl [107] provides the FL system with privacy while mitigating poisoning attacks. Their poisoning attacks defence is based on the work of [70] with a trust root. Then they use fully homomorphic encryption to encrypt the communication process. To accelerate transparent processes, they use blockchain. [106] also takes homomorphic encryption as the basic privacy mechanism. Different from [107], their method allows the server to detect malicious behaviours under ciphertext by extracting gradient data of the logarithmic function. Similar to [107], ShieldFL [108] uses the cosine similarity to detect poisoning behaviours. However, they use two-trapdoor homomorphic encryption as the underlying privacy mechanism instead of fully homomorphic encryption.

The Threats of Poisoning attacks in AI-based Service scenarios

Nowadays, intelligent edge computing applications based on federated learning are widely used in our daily lives. However, it is well-known that deep learning methods are vulnerable to backdoor attacks. [105] discusses the threat of propagation of the backdoors among federated works, which causes negative effects on intelligent edge computing. They attempt to develop a stability-based mechanism to address the issue of backdoors on edge devices. In addition to model poisoning attacks, LSFL [109] attempts to address the privacy leakage on intelligence edge devices by utilizing two servers to enable secure Byzantine robustness and model aggregation. RFL [104] is proposed as a novel federated pruning paradigm to train a robust pruned global model. Within this paradigm, the clients are requested to rank the edges of their local models and find a supermask within the initial models by using the Edge-popup Algorithm. Then, the clients send the ranks of the edges to the server. The server takes a majority vote mechanism to calculate the reputations of all the edges and aggregate all the edges by their reputations. XAI [110] discusses the backdoor issue in the large-scale Industrial Internet of Things (IIoT) applications. The workflow of [110] involves both the server and the IIoT applications. The server has three parts: (1) the attack library,

collecting the characteristics of existing backdoor trigger patterns; (2) the filter library; (3) the XAI model library. When an IIoT application is selected for training, the server provides the participants with the backdoor filter built on the filter library and the XAI model library. With this backdoor filter, the participants can de-trigger the malicious inputs by blurring the trigger area. By using the two-step blur-label-flipping strategy, the participants can mitigate the backdoor completely. [103] discusses the problem of how to accurately predict cyber-attack behaviours based on past events among organizations without disclosing sensitive information. They propose Cerberus, an FL system for predicting security events by training a Recurrent Neural Network (RNN) model with historical security events and predicting future security incidents. Their work bridge FL into intrusion detection.

Chapter 3

Towards Defending Adaptive Backdoor Attack in Federated Learning System

In this chapter, we mainly focus on defending against backdoor attacks (targeted model poisoning attacks) in FL. As we discuss in the subsection. 1.2, local clients can participate in FL training without revealing their sensitive information or being checked. Therefore, this privacy characteristic of FL leaves the door open for the adversary. Malicious clients can send poisoned model updates to the central server without being identified, which makes FL vulnerable to backdoor attacks. To address this security issue, we present FLSec, a novel defence approach, to mitigate backdoor attacks caused by adversarial local model updates. FLSec utilizes an original measurement, GradScore, computed from the loss gradient norm of the final layer of the local models for backdoor defence. We give a detailed explanation of why this GradScore can be an effective metric for distinguishing malicious behaviours. We show that GradScore is efficient and robust in identifying malicious model updates through analysis and experiments. Our extensive evaluation also demonstrates that FLSec is highly efficient in mitigating three state-of-the-art backdoor attacks (Single-shot, Constrain-and-Scale, and Distributed Backdoor) on well-known datasets, MNIST, LOAN, and CIFAR-10. According to the experiments, the accuracy of FLSec on a benign dataset with the proposed defence approach is nearly unchanged, with the accuracy on the backdoor dataset being reduced to 0%. In addition, our experiments show that FLSec significantly outperforms existing backdoor defences.

3.1 Introduction

FL is a collaborative machine learning paradigm proposed by McMahan et al. [6]. Unlike centralized training, FL offers efficiency and scalability as many clients execute the training in parallel over communication networks [6]. FL also provides excellent privacy to clients as they can keep their training datasets locally [6] rather than sharing them with other participants.

However, due to its distributed operation, FL leaves the door open for adversaries. An FL system is vulnerable to poisoning attacks, especially *backdoor attack* that aims to insert a trigger into the trained global model [13]. The existence of a backdoor makes the global model mislabel a small group of samples with chosen triggers into targeted labels. However, these backdoored global models can have good accuracy in benign and backdoored datasets.

Existing defences against poisoning attacks can be divided into two major classes, *certified robustness* and *empirical robustness* (e.g. [45], [89]). In this work, we mainly discuss empirical robustness, which is currently investigated by inspecting distinguishable factors, such as indicative features [47], source-focused error [89], or pair-wise cosine similarities [24] [71] [90] [89] [47]. However, these existing approaches are only efficient under specific assumptions about the data distribution of the clients [89] [47] [71] [78], or specific attack strategies [24] [89]. Therefore, these works have poor efficacy in generic adversary models. Defence approaches [119] based on robust statistics suffer from targeted poisoning attacks as they seek robustness against untargeted attacks.

To address the challenges mentioned earlier and limitations, in this work, we propose an effective defence approach applicable to a generic adversary model without assumptions about data distribution and attack strategies. This proposed technique can mitigate adaptive attacks while maintaining the main tasks' performance. Specifically, the contributions of our work are as follows:

- We proposed FLSec, a novel generic defence to mitigate backdoor attacks on federated learning systems. FLSec uses a pruning scheme. By carefully setting the pruning rate, malicious clients can be pruned largely.

- We propose and utilize scores (GradScore) of local client models, which are computed by the loss gradient norm of the final layer of the local models. It measures the updates

of each client to its local model. Clients with larger GradScore would be regarded as suspicious.

- We demonstrate the effectiveness of FLSec against backdoor attacks by evaluating multiple datasets and various attack scenarios. Experiments show that FLSec can effectively mitigate several state-of-the-art backdoor attacks without affecting the performance of the global model on main tasks.

3.2 System and Threat Model

The related preliminaries and system settings can be checked in the section. 1.2.

3.2.1 Attack Strategies

Data poisoning: In this attack strategy, adversary A is only able to manipulate the local training dataset of end devices. By varying the Poisoned-Data-Rate (PDR), the attacker can trade between attack impact and stealthiness. Let D_i denote the number of the combined and poisoned dataset of a compromised client i and D_i^A the number of modified or poisoned data. Then the PDR is given by:

$$PDR = \frac{D_i^A}{D_i} \tag{3.1}$$

Model Poisoning: In this attack strategy, adversary A can control a subset of the clients fully. To increase the attack's impact on the aggregated model, Adversary A^c can deliberately modify the model updates before submitting them to the aggregator.

Single-Shot: As proposed in [13], the adversary can scale up the model weights by γ up to the bound β set by simple weight-based anomaly detectors. The scaled malicious local updates $\mathbf{w}_i^t$ is given by:

$$\mathbf{w}_i'^t = (\mathbf{w}_i^t - G^t)\gamma_i^t + G^t \tag{3.2}$$

Here, $\mathbf{w}_i^t$ denotes a backdoored local model trained by a malicious client. $\mathbf{w}_i'^t$ refers to the scaled malicious local model.

Model replacement attack (Single-Shot) can ensure a good attack performance even when only one malicious client submits one malicious update $\mathbf{w}_i'^t$ in a single training round t

(Single-shot attack [13]). We use this attack strategy as a benchmark for evaluating our proposed defence technique.

Anomaly-Evasion: An adaptive loss function is used in [13] [11]. They added a term $\ell_{anomaly}$ that measures the cosine distance similarity between the known global model and the original poisoned model. Let $\ell_{original}$ denote the normal loss function and $\ell_{anomaly}$ denote the evasion loss function. Then the adaptive loss function $\ell^{'}$ is given by:

$$\ell^{'} = \alpha\ell_{original} + (1-\alpha)\ell_{anomaly} \tag{3.3}$$

The parameter α is used to control the weights of each part. If α is close to zero, the impact of backdoor attacks would be decreased. On the other hand, large α (α close to one) can make the malicious behaviours conspicuous by the anomaly detector. The combination of the anomaly-evasion and Scaling attack strategies is called **Constrain-and-Scale** attack [13]. This **Constrain-and-Scale** is another benchmark for evaluation in that

DBA: This novel backdoor strategy is proposed by [14]. By splitting the trigger and clients into different parts, this attack strategy performs better in clients and stealthiness than centralized backdoor strategies. We use this attack strategy as the third benchmark.

3.2.2 Adversary Model

The goal of an adversarial client A is to insert a backdoor into the aggregated model, inducing the learned classifier to achieve high accuracy on both its main task and a targeted backdoor task. We assume that D_B denotes benign dataset, and D_M denotes backdoored samples $\{x_i\}_{i=1}^{m}$ with true labels y_i that should be misclassified as targeted label τ_i. The adversary's objective is to maximize the sum of misclassified backdoored samples:

$$A(D_B \cup D_M, G^t) = max_{G^t}\{\sum_{i=1}^{m} 1[f(x_i;G^t) = \tau_i] + \sum_{D_B} 1[f(x;G^t) = y]\} \tag{3.4}$$

From the above equation, two main objectives for adversary A are:

O1: Performance on the backdoor task. The aggregated model should have a good performance on the backdoor task. Namely, the aggregated model should misclassify triggered samples into targeted labels [120].

O2: Stealthiness. The adversary should ensure that the aggregator server is unaware of

malicious behaviours. An obvious drop in the main task accuracy (MA) should be avoided.

Similar to previous works on backdoor attacks and defence [13] [11] [26] [72] [78], we consider a strong adversary model: (1) The attacker fully controls the compromised end device; (2) The attacker has full knowledge of the aggregating algorithm and configuration hyper-parameters, i.e., learning rate and the number of epochs; (3) The attacker can modify the updated weights adaptively before sending back to the aggregator.

3.2.3 Defense Objectives

To defeat Adversary objectives, the proposed defensive technique needs to meet the below security requirements:

R1: Poisoning elimination: The defence should eliminate the backdoor attack. In other words, the performance on the backdoored dataset should remain at the same level without the attack.

R2: No Interruption of the original Training Process: The defence should not interrupt or negatively impact the main training process. The main task accuracy should achieve the same level as without defence.

3.3 Proposed Approach

In this section, we introduce our proposed approach, FLSec, by deeply inspecting and analyzing model updates to discover models whose training data were poisoned for a specific backdoor task. First of all, to find out the potential malicious model updates, we define a new metric, GradScore. We analyze that the poisoned training dataset rate(PDR) directly impacts the value of GradScore of a poisoned model. Then we describe how to detect malicious clients by evaluating the corresponding GradScore values in federated learning. Finally, we give the details of our FLSec algorithm.

3.3.1 GradScore and analysis

Definition 1 *The GradScore of training set* $S = (x_i, y_i)_{i=1}^{N}$ *on a local client at global iteration* t *is* $GradScore(C_i^t) = \|g(\{(x_i, y_i)\})\|_2$.

It is approximated that training dynamics are in continuous time. For a labelled example (x, y) from local data set $S = \{(x_i, y_i)\}_{i=1}^{N}$, the time derivative of the loss on this labelled

sample is $\Delta_t((x,y),S^t) = -\frac{d\ell(f_{\mathbf{w}^t}(x),y)}{dt}$ at time t. By the chain rule,

$$\Delta_t((x,y),S^t) = g_t(x,y)\frac{d\mathbf{w}^t}{dt} \tag{3.5}$$

The instantaneous rate of change in $\mathbf{w}^t$ at time t, $d\mathbf{w}^t \approx \mathbf{w}^{t+1} - \mathbf{w}^t = -\eta \sum_{(x,y)\in s^t} g_t(x,y)$. The goal is to understand how poisoned samples from minibatch S^t affect the time derivative of the loss for any samples (x^*,y^*) from the same minibatch.

Lemma 1 *Let $S_{\neg j} = S\backslash(x_j,y_j)$ denotes training set excluding sample (x_j,y_j). Then for all rest samples $(x_\prime,y_\prime)$, there exists c such that*

$$\|\Delta_t((x_\prime,y_\prime),S) - \Delta_t((x_\prime,y_\prime),S_{\neg j})\| = c\|g_t(x_j,y_j)\|. \tag{3.6}$$

proof 2 *For a given example $(x_\prime,y_\prime)$, the chain rule yields $\delta_t((x_\prime,y_\prime),S) = g_t(x_\prime,y_\prime)\frac{d\mathbf{w}_t}{dt}$. Therefore, for the left part of Eq.(3.8),*

$$\begin{aligned}
&\|\Delta_t((x_\prime,y_\prime),S) - \Delta_t((x_\prime,y_\prime),S_{\neg j})\| \\
&= \|\frac{d\ell(f_t(x_\prime,y_\prime))}{d\boldsymbol{w}_t}(-\eta\sum_{S_t} g_t(x,y)) \\
&- \frac{d\ell(f_t(x_\prime,y_\prime))}{d\boldsymbol{w}_t}(-\eta\sum_{S_{\neg jt}} g_t(x_\prime,y_\prime))\| \\
&= \|\frac{d\ell(f_t(x_\prime,y_\prime))}{d\boldsymbol{w}_t}(-\eta g_t(x_j,y_j))\| \\
&= \eta\|\frac{d\ell(f_t(x_\prime,y_\prime))}{d\boldsymbol{w}_t} g_t(x_j,y_j)\|
\end{aligned} \tag{3.7}$$

Let $c = \eta\|\frac{d\ell(f_t(x_\prime,y_\prime))}{d\boldsymbol{w}_t}\|$, we can get the right part of Eq.(3.8).

It is not difficult to see from above that the contribution of a training sample (x_j,y_j) to the decrease of loss on other samples from the same minibatch can be quantified by Eq.(8). The value of $\|g_t(x_j,y_j)\|$ is the GradScore of sample (x_j,y_j). Samples with large GradScore have a high influence on learning. For backdoor training on local devices, malicious clients should try to reduce backdoor training loss $\ell_B((x,y),G_t)$. Hence, malicious clients should increase the poisoned data rate(PDR). In Fig.4.3(a)(c), we evaluated this inference, running backdoor training on MNIST dataset with a minibatch of 64 samples. With the same pre-

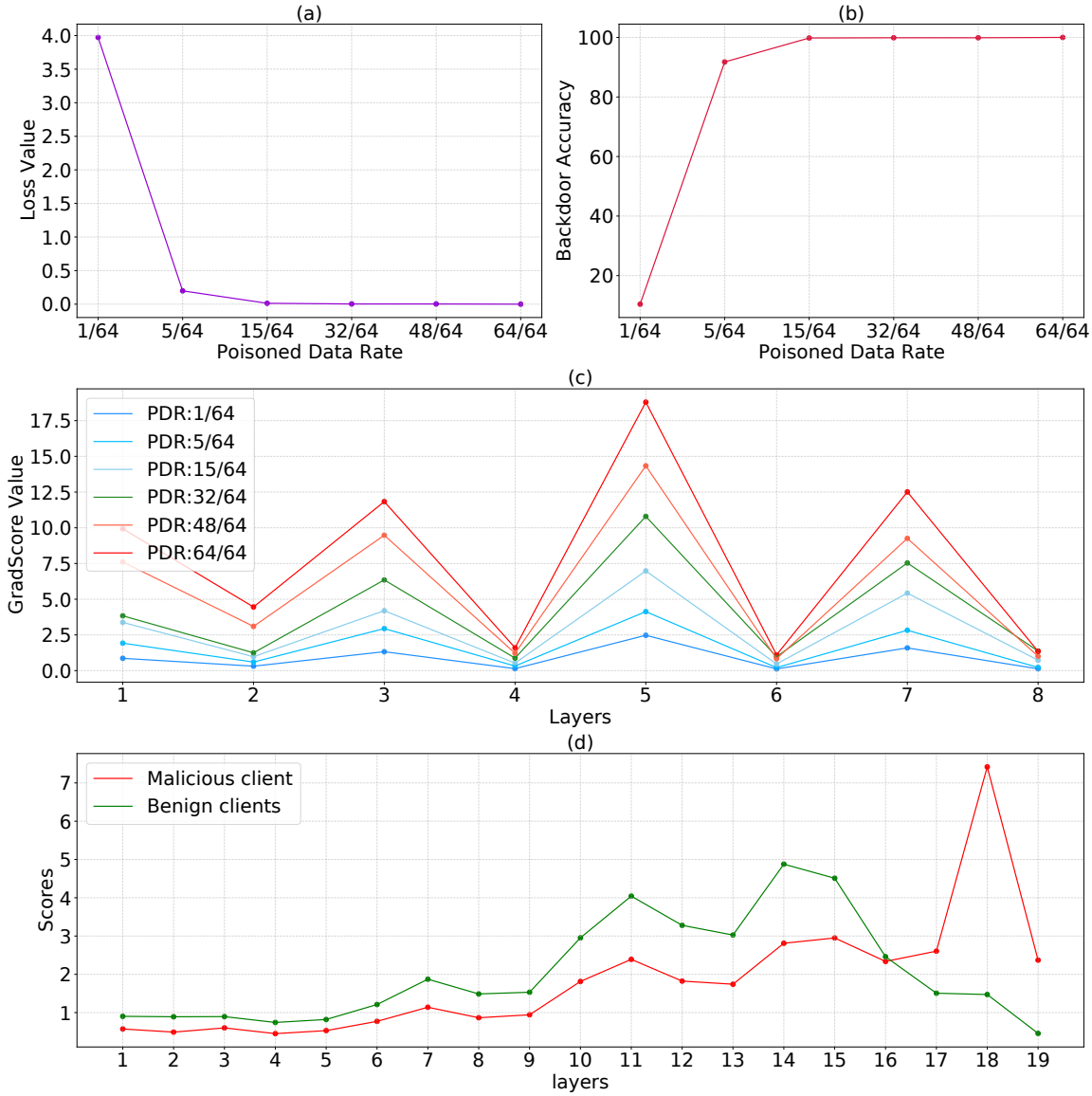


Figure 3.1: Impact of the poisoned data rate (PDR) on loss value, Backdoor Accuracy, and GradScore. value

trained model, a model trained with a higher poisoned data rate causes an obvious decrease in backdoor training loss and has a higher GradScore value.

3.3.2 FLSec Design

Key Observations. Our first observation is that no matter what is the data distribution among clients, the deviations between local models and global model start to cancel out, i.e.,$\forall \mathbf{w} \in \{\mathbf{w}_i\}_{i=1}^{m}, \mathbf{w}_i^{t+1} - G^t \approx 0$ [13], in the benign setting, as the global model converges. Therefore, the updates of benign local models , $d\mathbf{w} \approx \mathbf{w}^{t+1} - \mathbf{w}^t$ is bounded. According to Eq.(8), it is not difficult to see $\|\Delta_t((x_\prime, y_\prime), S) - \Delta_t((x_\prime, y_\prime), S_{\neg j})\|$ is bounded. Therefore, $\|g_t(x_j, y_j)\|$

of one example from the benign dataset is small. The second observation is that when the global model starts to converge, poisoning behaviours on the malicious client would deviate from the malicious updates from the current iteration global model [115]. The GradScore of benign clients is small, while the GradScore of malicious clients is larger. In fig.4.3, it is shown that the GradScore of the last layer gradients of malicious clients' model is larger than benign clients'. Therefore, malicious clients can be detected by comparing the GradScore of the last layer of local models.

Algorithm 1: Design of FLSec

Input: n, G^0, T, $C_0^{t+1},...,C_{n-1}^{t+1}$

// n is the number of clients in one iteration, G^0 is the initial global model, T is the number of global iterations, $C_0^{t+1},...,C_{n-1}^{t+1}$ is a group of clients chosen for training in one round

Output: G^T

// G^T is the updated global model after T iterations

for $t \in [1,..,T]$ **do**
  for $i \in [C_0^{t+1},...,C_{n-1}^{t+1}]$ **do**
    $GradScore(C_i^{t+1}) = \|g\{(x,y)\}\|_2$ // $\|g\{(x,y)\}\|_2$ is the L_2-norm of gradients of parameters in final layer of models
  end
  $SCORE \leftarrow [GradScore(C_0^{t+1}),...,GradScore(C_{n-1}^{t+1})]$;
  $Sort(SCORE)$;
  $Pruned(\mathbf{w}_0^{*t+1},...,\mathbf{w}_{m-1}^{*t+1}) \leftarrow Pruning_{p\%}([\mathbf{w}_0^{t+1},...,\mathbf{w}_{n-1}^{t+1}])$ // $p\%$ is the pruning rate
  $SendPruned(\mathbf{w}_0^{*t+1},...,\mathbf{w}_{m-1}^{*t+1}) \rightarrow Aggregator$;
  $G^{t+1} \leftarrow G^t + \frac{\eta}{m}\sum_0^{m-1}(\mathbf{w}_i^{t+1} - G^t)$ // Global Aggregating, η is the global learning rate
end

Now, we discuss the steps of FLSec. Algorithm 1 outlines the procedure of FLSec.

Identifying malicious behaviours. In designing FLSec, the first step is identifying and measuring malicious behaviours in the federated learning system.

Pruning and excluding malicious clients. After malicious behaviours are identified, the next step in FLSec is to identify and exclude anomalous clients based on corresponding *GradScore* values. First, *GradScore* corresponding to clients is sorted in descending order. The top p per cent clients with the highest scores are pruned and excluded from the benign

client list. The parameter p depends on the number of anomalous clients in one global iteration. For example, when the adversary takes a single-shot attack strategy, only one malicious client should be excluded. It is generally assumed that the fraction of malicious clients are within the range $(0 < f < n/2)$, and at least half of clients with smaller *GradScore* values are identified as benign.

Generally, p is set to 0.5. In Section.3.4, we evaluate the validity of FLSec with different pruning rate p. The sorting and pruning step is shown in lines 5-7 of Alg.1.

Aggregation The aggregator excludes the updates sent by malicious users in the current iteration and trains the global model on the remaining model updates(line 9 of Alg.1). The global training algorithm varies based on the underlying training algorithm used in the application. We use FedAvg [6] to train the global model in this proposed work.

3.4 Evaluation Results

In this section, we test the efficiency of FLSec against three adaptive backdoor attacks. We conduct several experiments to analyze the detection accuracy of FLSec under multiple configurations and varying system parameters (pruning rate p). All evaluations are implemented based on the PyTorch framework provided by [13] and [14].

3.4.1 Experimental Setup

Datasets.

MNIST. The MNIST dataset consists of 70000 handwritten digits [121]. The learning task is to classify images to identify digits. The adversary clients mislabel labels of images before starting poison training task [47].

CIFAR-10. The CIFAR-10 dataset consists of 60000 coloured images with 32×32 pixels and 24-bit colour per pixel (3 colour channels). 50000 samples of this dataset are used for training, and 10000 samples are used for testing.

LOAN. A non-i.i.d financial dataset consists of 1.808,534 data samples. 80% of these data are divided as training samples, and 20% are for testing.

Attack strategy and setting. We consider four backdoor attacks: Single-Shot attack [13], Constrain-and-Scale attack [13], DBA [14] and Multi-round backdoor attack.

Single-Shot, Constrain-and-Scale, and DBA. The details of these three attacks are discussed in 4.3.4. It is assumed that only one out of ten clients is malicious in one global epoch. Adversary only attacks once when the global model starts to converge. We enable the proposed approach after the first ten global epochs for the MNIST dataset, ten epochs for the LOAN dataset, and 200 epochs for the CIFAR-10 dataset, respectively.

Multi-rounds backdoor attack. Unlike the three attacks above, it is assumed that malicious clients take attack actions every global epoch after the global model starts to converge. The multi-round backdoor attack is a strong attack strategy. Small perturbations caused by malicious clients in every round can compound and negatively affect the global model.

Evaluation Metrics.

BA(Backdoor Accuracy). the model's accuracy in the backdoored dataset.

MA(Main Task Accuracy). the accuracy of the model in the benign dataset.

3.4.2 Effectiveness of FLSec

Choice of pruning rate p. Fig.3.2 shows the impact of the pruning rate on MA and BA rates. As for MNIST, FLSec completely mitigates four types of backdoor attacks ($BA \approx 0\%$) for three datasets (meet R1) and does not affect the main task performance as the main task accuracy is the same as baseline (meet R2). In the case of MNIST, we set $\alpha = 0.5$. In section 4.3.4, we discuss that the Scaling Coefficient parameter α can balance the effect and stealthiness of backdoor attacks. When α is set to 0.5, the performance of backdoor attack is greatly weakened ($BA = 32\%$) in Fig.3.2(b). Therefore, we do not set α less than 0.5, as the attack impact is too weak. In CIFAR, FLSec can easily mitigate single-shot and Constraint-and-scale attacks ($BA \approx 0\%$). However, backdoors cannot be completely mitigated under another two attack strategies. As discussed in section 4.3.4, unlike centralized learning, DBA splits triggers and malicious clients into different parts. Split trigger images alone are unable to change to prediction into targeted labels until they are assembled as a global trigger. This characteristic makes split triggers much tougher to distinguish from benign images. But as shown in Fig.3.2(g)(h), the negative effects of DBA and DBA with Constraint-and-scale can still be effectively decreased to a lower level. Similar results on the LOAN dataset are shown in Fig.3.3, FLSec can mitigate all backdoor attacks with different pruning rates.

	(p=90%)MNIST		(p=80%)LOAN	
	BA	MA	BA	MA
Benign Setting	0.0	97.68	0.0	76.16
No Defense	99.83	39.47	98.12	71.25
Single-shot	0.44	97.32	0.0	73.36
Scaling Coefficient($\alpha = 0.9$)	0.46	97.39	0.0	73.70
Scaling Coefficient($\alpha = 0.8$)	0.44	97.32	0.0	71.25
Scaling Coefficient($\alpha = 0.7$)	0.46	97.35	0.0	71.25
Scaling Coefficient($\alpha = 0.6$)	0.39	97.28	0.16	73.05

Table 3.1: Resilience of FLSec to Constrain-and-Scale attacks with varying α values

	(p=90%)MNIST		(p=80%)LOAN	
	BA	MA	BA	MA
Benign Setting	0.0	97.68	0.0	76.16
No Defense	93.25	77.44	98.58	75.37
DBA	0.43	97.64	0.0	73.77
Scaling Coefficient($\alpha = 0.9$)	0.39	97.59	0.0	73.63
Scaling Coefficient($\alpha = 0.8$)	0.39	97.55	-	-
Scaling Coefficient($\alpha = 0.7$)	0.43	97.64	-	-
Scaling Coefficient($\alpha = 0.6$)	0.37	97.53	-	-

Table 3.2: Resilience of FLSec to DBA with varying α values

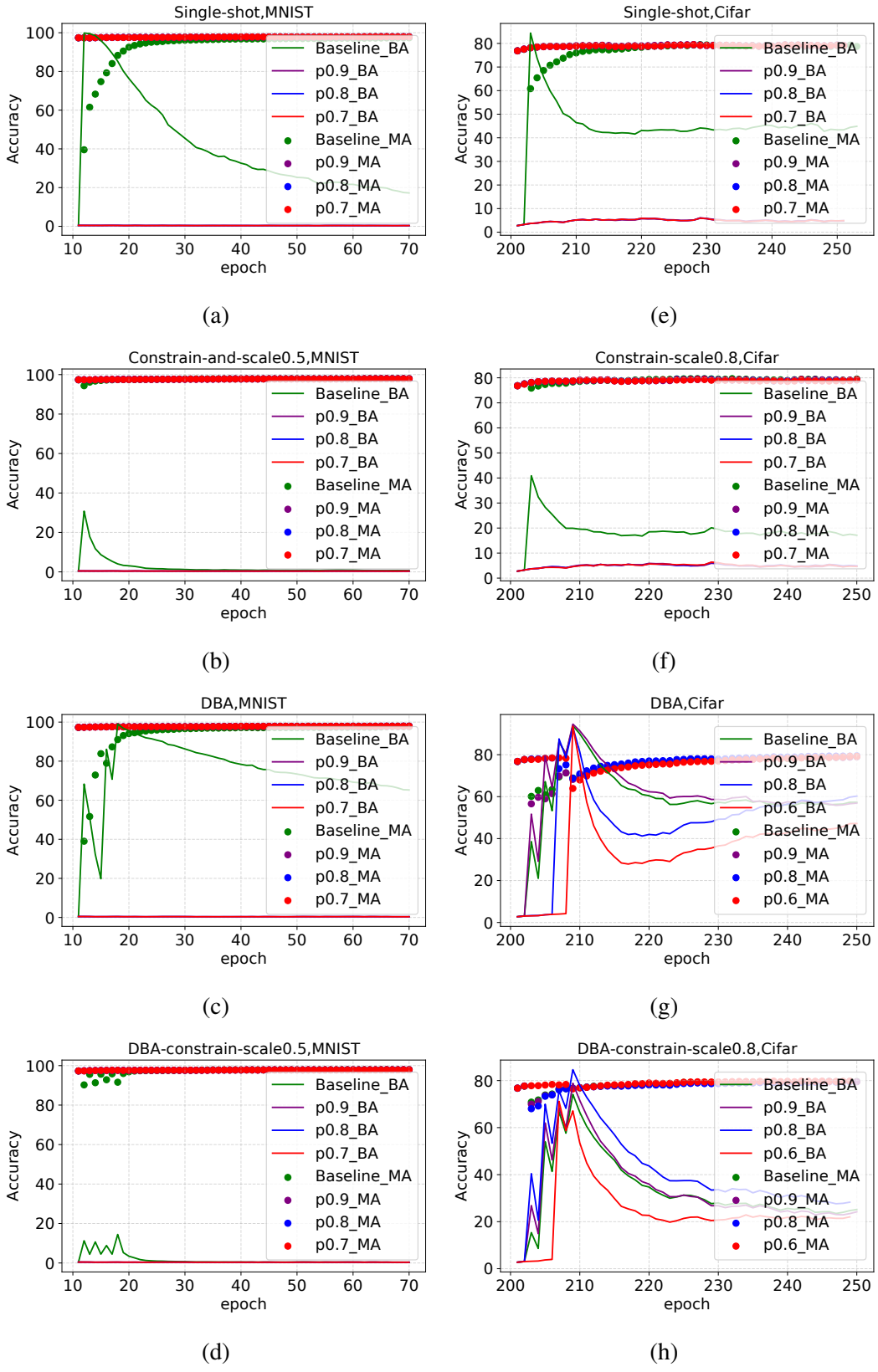


Figure 3.2: Effectiveness of FLSec with different pruning rate p under Single-shot, Constrain-and-Scale, and DBA attack strategies on two data sets, MNIST and CIFAR

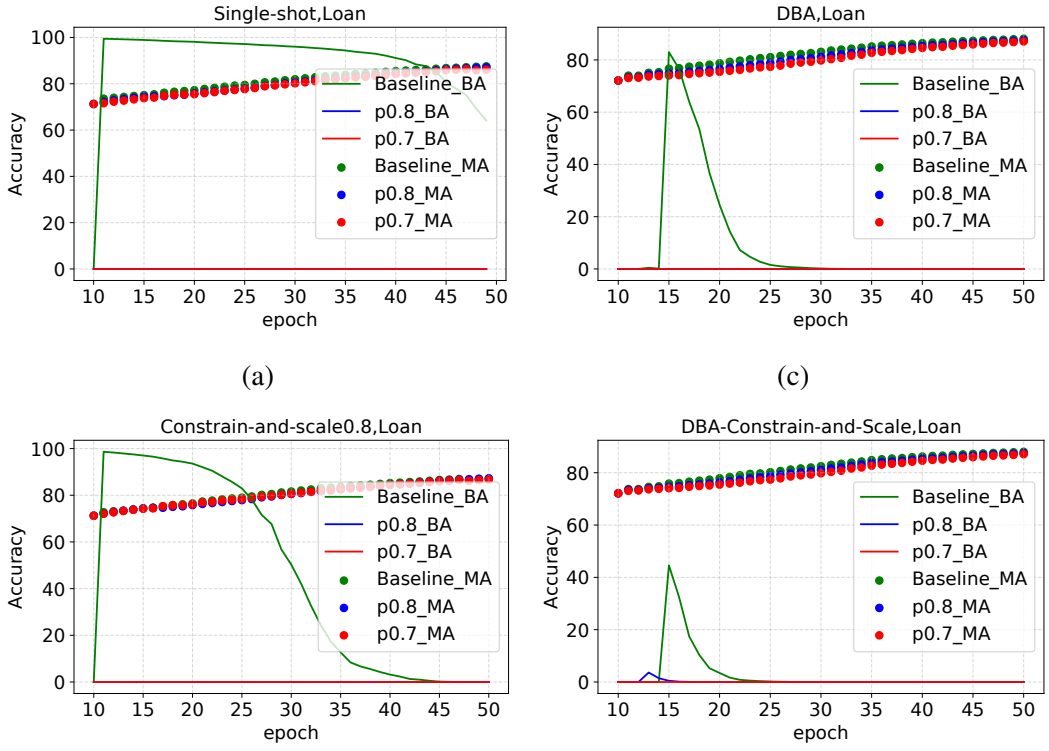


Figure 3.3: Effectiveness of FLSec with different pruning rate p under Single-shot, Constrain-and-Scale, and DBA attack strategies on Loan

3.4.3 Resilience to Adaptive Attacks

Varying Scale Coefficient Parameter (α) For the Constraint-and-scale attack strategy, the malicious clients can adjust the Scale Coefficient parameterα to bypass defence techniques. We evaluate different Scale Coefficient values α from 1 to 0.1 and keep other parameters, including global and local learning rates, the scaling factorγ, PDR, and PMR, consistently on two datasets, MNIST and LOAN. FLSec effectively reduced the impact of single-shot replacement attacks without misclassifications on the MNIST dataset and did not impact the main accuracy. The results are shown in Table.3.1. From Table.3.1, FLSec also had a good performance on LOAN with α value from 1 to 0.6. For the DBA constrain-and-scale attack strategy, the proposed approach can still recognize all the malicious models on the MNIST dataset(Table.3.2). More results on CIFAR can be checked in the Table.3.3 and Table.3.4.

	(p=80%)CIFAR	
	BA	MA
Benign Setting	2.788	76.93
No Defense	76.47	66.66
Single-shot	3.477	77.74
Scaling Coefficient($\alpha = 0.9$)	3.500	77.70
Scaling Coefficient($\alpha = 0.7$)	2.788	76.78
Scaling Coefficient($\alpha = 0.3$)	2.755	75.40
Scaling Coefficient($\alpha = 0.1$)	3.622	77.11

Table 3.3: Resilience of FLSec to Constrain-and-Scale attacks with varying α values on CIFAR dataset

	(p=80%)CIFAR	
	BA	MA
Benign Setting	2.788	76.93
No Defense	81.377	71.86
DBA	3.222	78.10
Scaling Coefficient($\alpha = 0.9$)	3.200	78.14
Scaling Coefficient($\alpha = 0.7$)	3.244	78.13
Scaling Coefficient($\alpha = 0.3$)	26.66	73.13
Scaling Coefficient($\alpha = 0.1$)	6.988	78.71

Table 3.4: Resilience of FLSec to DBA with varying α values on CIFAR dataset

3.4.4 FLSec's Resilience to Adaptive Attacks on CIFAR

In Table.3.3 and Table.3.4, FLSec shows similar performance on CIFAR on defending Constrain-and-Scale attacks and DBA attacks. In Table.3.3, FLSec can reduce the backdoor accuracy to a low level($2.755\% - 3.622\%$) compared to 76.47% without defence. In Table.3.4, when defending against DBA attacks, the backdoor accuracy can be reduced effectively from 81.377%(No Defence) to 3.200%($\alpha = 0.9$). However, when α is set to 0.3, the backdoor accuracy is 26.66% with FLSec defence. In section.3.2, we discuss that when α is small, $l_{anomaly}$ is assigned a large weight. So the trained poisoned model is more similar to the known global model, which makes the attack more inconspicuous by the defender. Therefore, the stealth of distributed backdoors is strengthened with $\alpha = 0.3$, and one or two distributed attack actions avoid detection.

Varying Poisoned Data Rate (*PDR*) The adversary may attempt to vary the poisoned data rate to circumvent FLSec. We evaluate FLSec against Constrain-and-Scale attacks for different *PDR* values on CIFAR-10 with $p = 0.9$, $\gamma = 100$. In Fig.3.4, we can see that compared with 99.83% in no defence setting, the attacks show a significant decrease in the backdoor accuracy in all cases. When *PDR* is below 0.1, the malicious clients should set a large scaling factor(γ) to inject backdoors. However, large malicious updates can be easily detected by outlier detectors [13]. Another interesting result is that it is hard for attackers to make a trade-off between backdoor accuracy and attack stealthiness. A smaller α means that the malicious updates are more similar to benign models, and the model is more stealthy. However, too small α can highly impact the backdoor accuracy. When α is set to 0.1, the backdoor attacks fail in all the cases.

3.4.5 Comparison to previous defences

Preventing multi-rounds backdoor strategy. We perform the multi-rounds backdoor strategy in the Dirichlet distribution with hyperparameter 0.5 on the MNIST dataset to demonstrate the effectiveness of the proposed technique compared to RFA [119]. Four of ten malicious clients collude in every global iteration and perform distributed backdoor attacks without boosting.

The single-shot attack is not considered here, as boosted malicious model updates can be easily detected by RFA [119]. Fig.3.5 shows that FLSec outperforms RFA in mitigat-

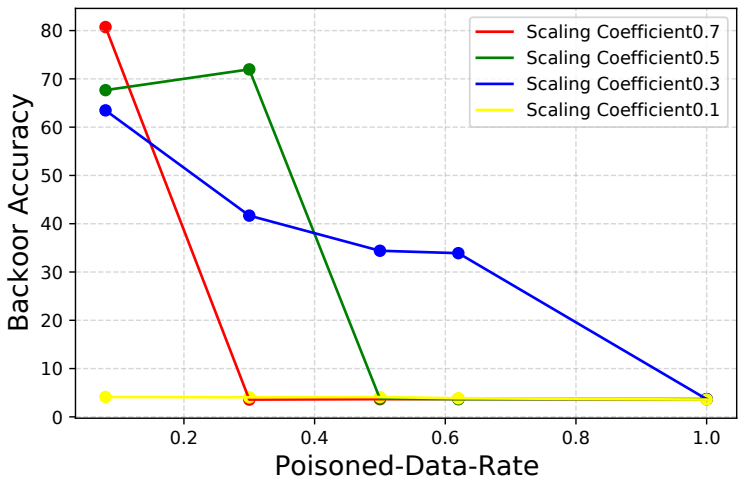


Figure 3.4: Poisoned data rate vs Backdoor accuracy

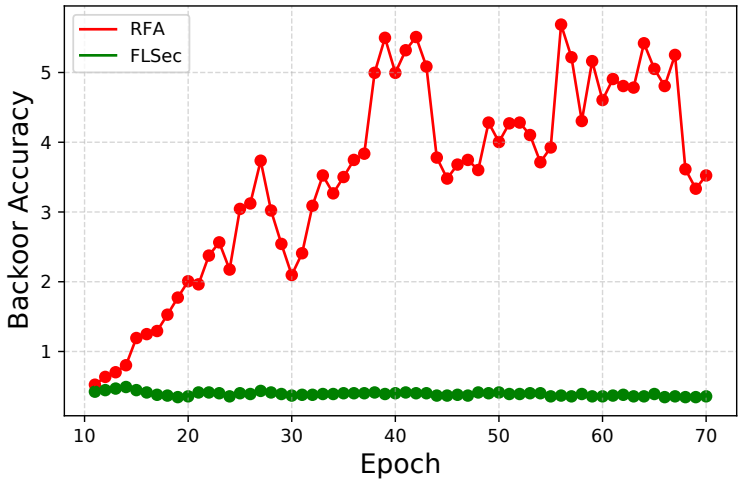


Figure 3.5: Effectiveness of FLSec in comparison to RFA on MNIST dataset. The Y-axis label refers to the accuracy of muti-rounds backdoor attacks in federated learning

ing multi-round backdoor attacks. The backdoor accuracy of the global model with FLSec remains at 0%, while there is a non-negligible increase in the global model with RFA. Therefore, the performance of RFA is poor in the non-IID data distribution among clients, while FLSec can mitigate malicious behaviours completely.

3.5 Conclusion

We proposed a novel approach FLSec for federal learning that can resist backdoor attacks. It analyzes the difference between benign and malicious clients' contributions to the global model and uses new measurements on local model updates to identify malicious updates.

FLSec was evaluated with various attack strategies and datasets. Experiment results show that FLSec can effectively mitigate backdoor attacks without sacrificing the performance of the main task. We compared FLSec with state-of-the-art defence techniques, and FLSec was able to address complicated backdoor attacks in FL systems.

In this chapter, we investigate the efficiency of FLSec with GradScore against the three well-known backdoor attacks. We do not consider the attack timing of the adversary. The attacks may happen in the early training stage or the attacks can appear periodically. Besides, the FLSec should be set manually before the FL training. Thus, the adversary can foreknowledge of the defender method and try to avoid it. In the next chapter, we will dive more deeply and discuss how to overcome these problems.

Chapter 4

DeMAC: Towards Defending Model Poisoning Attacks in Federated Learning System

In this chapter, we investigate how to mitigate multi-round targeted model poisoning attacks, in which malicious clients can send poisoned model updates to the central server in multiple training rounds. The bad impact of single-round attacks can be gradually eliminated with training, however, the negative impact of multi-round model poisoning can overlay with training. We propose a new system named DeMAC, equipped with GradScore, to improve the detection and defence against model poisoning attacks by malicious clients. The main observation is that, as malicious clients need to reduce the poisoning task learning loss, there will be obvious increases in the norm of gradients. It is shown through a toy experiment that the GradScores of malicious and benign clients are distinguishable in all training stages. Therefore, DeMAC can detect malicious clients by measuring the GradScore of clients. Furthermore, a historical record of the contributed global model updates is utilized to enhance the DeMAC system, which can spontaneously detect malicious behaviours without requiring manual settings. Experiment results over two benchmark datasets show that DeMAC can reduce the attack success rate under various attack strategies. In addition, DeMAC can eliminate model poisoning attacks under heterogeneous environments.

4.1 Introduction

Federated learning is a distributed machine learning paradigm [6] [122] [123]. Each client trains a model locally, and then all local model updates are aggregated by a central server to derive a global model. This process is repeated multiple times, and the accuracy of the global model on the main task is gradually improved. FL offers efficiency and scalability compared to centralised training since many clients execute the training in parallel [124]. In particular, FL provides clients' privacy as they can keep their training dataset locally [6] rather than sharing it with other participants. Such a secure aggregation mechanism complies with General Data Protection Regulation (GDPR) [125] and protects clients against privacy leakage attacks. Due to FL's potential functions of privacy protection, it has been deployed in the real world. For example, Android Gboard [126] has been installed with FL for next-word prediction. In finance, WeBank [127] has applied FL for credit risk prediction. FL has been widely used in pharmaceutical companies for drug discovery in MELLODDY project [128].

A major challenge faced by FL is that it leaves the door open for malicious clients. A FL system is vulnerable to model poisoning, especially *backdoor attack* that may insert backdoors into the trained global models [13]. The backdoors can make the global machine-learning model misclassify a small set of samples with chosen triggers into targeted labels, while the backdoored global model can show good performance on both main and backdoor tasks.

Existing defences such as [46] [45] [119] propose Byzantine-tolerant aggregation rules and remove statistical outliers by comparing client local model updates. However, these previous works make some assumptions, such as the data distribution should be IID (Independent and Identically Distributed), which is not valid in non-IID setting [46]. [119] relies on the aggregation of updates using the geometric median, not the standard arithmetic mean aggregation. However, our work shows that this method can be bypassed by malicious clients who carefully design their poisoned updates. Besides, they do not consider the situation when the attackers participate in the global training round more than one time.

In view of the above drawbacks of the existing works, we propose a system (called DeMAC) to detect and defend against model poisoning attacks from malicious clients. Our design relies on the key finding that genuine clients train their model updates following the

main federated training task and their local benign datasets, while malicious clients will craft their local model updates trained on the poisoning task and poisoned datasets. To succeed in the poisoning attacks, the adversaries should increase the number of poisoned samples to decrease the training loss of the poisoning task. Therefore, the $L2$-norm of gradients of the poisoned local model updates will be increased. Although the adversaries can reduce the number of poisoned samples and decrease the deviations from benign models' norm of gradients, this may cause the poisoning task to fail. Hence, to measure the $L2$-norm of gradients and capture the abnormal changes, we define a new metric which is called GradScore as the $L2$-norm of gradients in the last layer of the client model updates after the first local epoch training. Using GradScore DeMAC can effectively detect potential malicious clients with abnormally large GradScore values. When the global model training converges, the loss and the norm of gradients of the main federated training task will be small. If there are poisoning attacks in this training stage, the difference between model GradScores of genuine clients and malicious clients will be more obvious. Therefore, DeMAC can easily detect and mitigate malicious clients. As for poisoning attacks starting from an early stage, our experiments also show that DeMAC can work effectively.

Furthermore, to improve the defence performance, a historical record of the contributed global model updates is utilised in DeMAC to help spontaneously estimate the convergence trend of the global model and determine the time to start detecting malicious behaviours. This historical record will store a list of variables within a flexible look-back window size. The variables are the absolute values of two adjacent accuracy values of the global model on the validation set. Once the maximum value among this historical record is below the default threshold, DeMAC would be triggered and start to detect.

We evaluate DeMAC on two benchmark datasets and model-replacement attack [13], distributed attack [14], scaling-scale attack and muti-poisoning attack [77]. Experiment results show that DeMAC can effectively mitigate model-replacement, distributed, and scaling-scale attacks. When malicious clients participate in every training iteration and insert perturbations, the Attack Success Rate (ASR) of the baseline algorithms increases gradually while the ASR of DeMAC remains at a low level. Therefore, DeMAC can effectively suppress the propagation errors. In brief, our contributions include: 1) We propose DeMAC, a defence system to detect malicious clients and defend against model poisoning attacks via

checking the abnormal model updates from potential malicious clients. 2) We utilize the historical record in DeMAC for defence against malicious attacks, which can spontaneously detect malicious clients without manual settings; 3) We extensively evaluate DeMAC by experiments against multiple model poisoning attacks and backdoor attacks on benchmark datasets, which shows high efficiency in defence against malicious attacks in both early and late training stages and significant performance improvement over the existing baseline methods.

The remainder of our paper is organized as follows. Section 4.2 discusses the research related to poisoning attack and defence. In section 4.3, we introduce the system and threat models with specific descriptions of adversaries, objectives and requirements for attacks and defences. In Section 4.4 and 4.5, we present our novel DeMAC system to defend against model poisoning attacks. We present the evaluation setup in Section 4.6.1. In Section 4.7, the evaluation results of DeMAC are presented. Section 4.8 concludes the paper and presents our future work.

4.2 Previous Defence Mechanisms

Defence mechanisms (against backdoor attacks) in the literature can be roughly classified into two categories: *Byzantine-robust federated learning methods* [46] [45] [44] [20] [129] [130] [131] [119], and *anomaly detection-based methods* [24] [115] [89] [47] [72]. Byzantine-robust federated learning methods aim to tolerate Byzantine clients failures. In contrast, anomaly detection-based methods attempt to filter potential malicious clients.

4.2.1 Byzantine-robust federated learning methods

The principle of existing Byzantine-robust defences [46] [45] is to train a global model with high performance, even if there are some malicious clients.

Krum [46] tries to find a representative model update as the aggregated model update. Suppose there are n local clients in every iteration. And f among these local clients is malicious. The score for the ith client is calculated as $s_i = \sum_{\mathbf{w}_j \in \Gamma_{i,n-m-2}} \|\mathbf{w}_j - \mathbf{w}_i\|_2^2$, where $\Gamma_{i,n-m-2}$ is the set of $n-m-2$ local clients that have the smallest Euclidean distance to $\mathbf{w}_i$. So the representative model update is the one that has the smallest score. This representative model update will be the global model for the next iteration. Krum attempts to limit the iter-

ation between poisoned and clean models in a single iteration. However, it does not consider compound propagation errors [97]. Therefore, the iterative nature of learning ensures that small deviations at the start of training compound exponentially.

Median [45] is a coordinate-wise aggregation rule. The coordinate-wise median of sorted local models is selected as the aggregated global model update. Instead of using the mean value among local clients, this aggregation rule considers the coordinate median value of the parameters as the corresponding parameter in the global model for the next iteration.

Trimmed-mean [45] is another coordinate-wise aggregation rule. Given the trim parameter $k < \frac{n}{2}$, the server removes the k maximum and minimum values of the coordinates in client model updates and then computes the mean of the remaining $n - 2k$ values as the corresponding parameter in the global model for the next iteration. Trimmed-mean relies on the assumption that the coordinate of the attacker would either be the minimum or the maximum value of the corresponding parameters. However, this assumption does not hold for model poisoning attacks [97]. Therefore, even a single attacker can compromise the trimmed mean.

RFA [119] is a robust aggregation rule based on similarity metrics. RFA aggregates the model updates and appears robust to outliers by replacing the weighted arithmetic mean in the aggregation process with an approximate geometric median. Model-replacement attack [13] is more easily detected by RFA due to the scaling operation [119]. However, by strictly controlling the total weights of the outliers with only a few attackers poisoning a small set in every batch, the attacker model updates can have lower distances and can be assigned higher aggregation weights [14]. By doing so, the attackers can bypass RFA and perform a successful backdoor attack.

4.2.2 Anomaly detection-based methods

Many existing defences [24] [115] [89] [47] [72] follow an anomaly-detection-based strategy and exclude anomalous model updates. FoolsGold [24] defines indicative features. By measuring the cosine similarity on the indicative features and checking the Sybil clones, Sybil attacks can be detected in no-IID data scenarios as Sybils have highly similar updates. However, FoolsGold shows poor performance on one Sybil attack scenario. FLAME [115] uses similar detecting strategies by calculating the angular differences between all model updates. Rather than comparing the probabilities of global models, DeepSight [72] compares

the local model updates with the previous global model. However, it does not work in no-IID scenarios. Auror [47] defines indicative features and finds that all the indicative features come from the final layer. Auror assumes that the indicative features from benign clients would have a similar distribution while the indicative features from malicious clients would have an anomalous distribution. However, it does not work in no-IID scenarios.

4.3 Background

In this section, we give some background knowledge of federated learning and attack strategies against federated learning systems.

Symbol	Descriptions
$D = \{(x_i, y_i)\}_{i=1}^{N}$	The training set on local devices
N_D	the number of samples in training set D
$p(\mathbf{w}, x)$	The probability vector
$\sigma(f(\mathbf{w}, x))$	Activation function
$\ell(p, y)$	The loss function
$\ell_B(p, \tau)$	The poisoning task loss function
$\mathbf{w}^t$	Weights at local iteration t
$S_0, S_1, .., S_{n-1} \subseteq S$	Mini-batches
$g(x, y)$	The gradient of the loss function
G^t	Global model at global round t
$\{C_1, ...C_n\}$	n clients chosen at global round t
PDR	Poisoned Data Rate
PMR	Poisoned Malicious Clients Rate
m	The number of compromised clients
$\mathbf{w}^{'}$	Compromised client weights
$G^{'}$	Compromised global model
D^P	Poisoned data set on local devices
N_{D^P}	the number of samples in D^P
τ	Targeted label
y	Genuine label
$x^{'}$	Poisoned data
x	Genuine data
η	the local learning rate
γ	The scaling factor
β	The Scaling-Coefficient parameter
p	The pruning rate
σ	The validation threshold
l	The size of sliding window
S	Central server

Table 4.1: Symbols and Descriptions

4.3.1 Definition of symbols and corresponding descriptions

The overall definition of symbols and corresponding descriptions is listed in table.4.1.

4.3.2 Preliminaries

The related preliminaries can be checked in the section. 1.2.

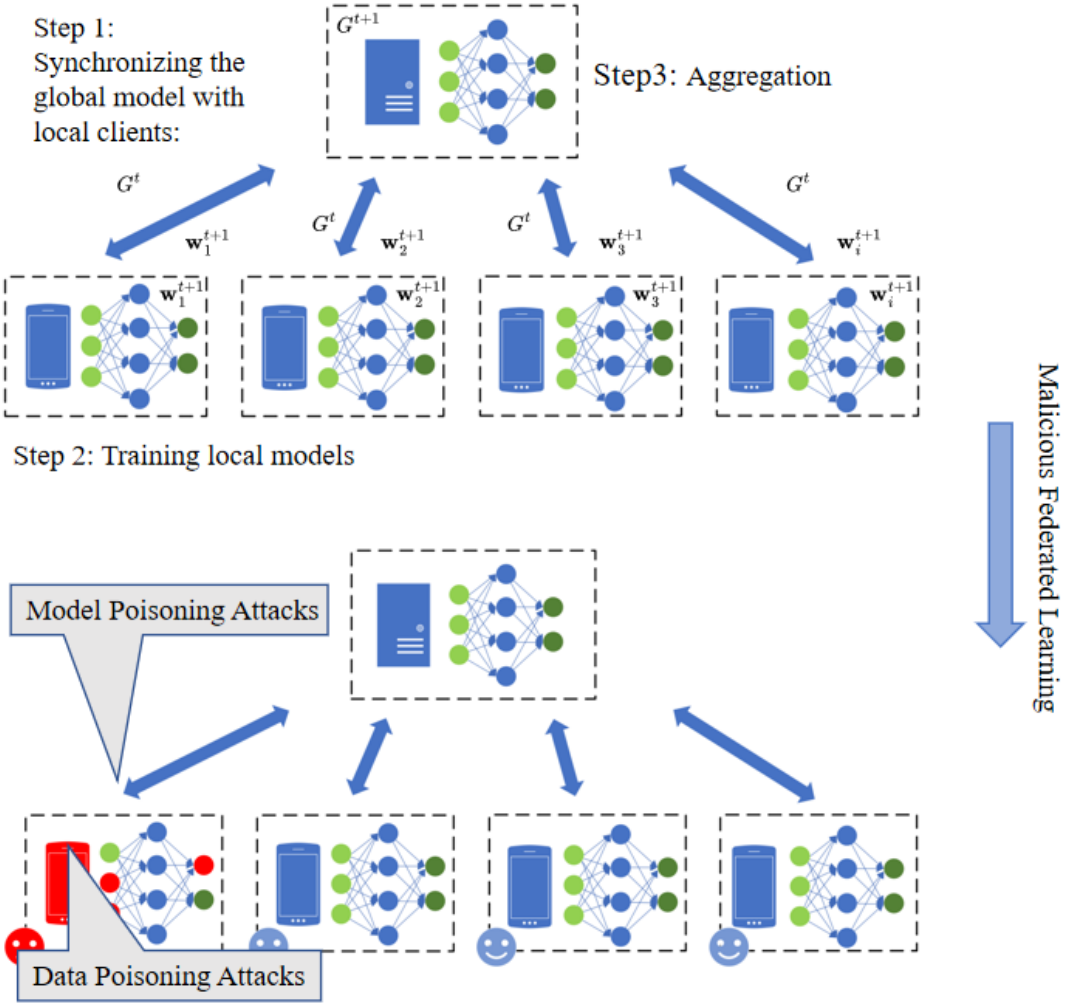


Figure 4.1: On the left are the three steps of Federated Learning. On the right is the malicious Federated Learning

4.3.3 System Setting

We assume that n clients train their local models before sending local updates to the central server. The central server combines these updates by using FedAvg [6]. In addition, all the clients keep their data secret, and no client can intercept training or testing data. The opti-

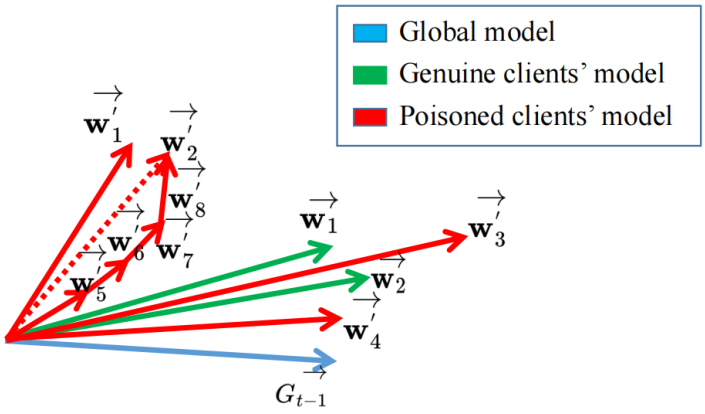


Figure 4.2: Weight vectors of genuine and poisoned models

mization problem of FL is $\min_G F(G)$, where $F(G) = \mathbb{E}_{D\sim\mathbb{Z}}[\ell(p(G,x),y)]$ is the expectation of the empirical loss $\ell(p(G,x),y)$ on the local training set D [70].

One iteration of FL training is shown below (see the left part in Fig.4.1):

- **Step 1:** Synchronizing the global model with local clients: The server sends the current global model G^t to the chosen clients.
- **Step 2:** Training local models: Each client initializes its local model as the global model G^t and trains a local model using its training set $D = \{(x_i, y_i)\}_{i=1}^N$. The optimization problem of clients is minimizing $\ell(p(\mathbf{w},x),y)$, where $\mathbf{w}$ is the local model. By using SGD, the client updates the local model as described in Eq. (**??**). Then the client sends its updates $\mathbf{w}_i^{t+1}$ to the server.
- **Step 3:** Aggregation: The server the updated global model via aggregating the local updates by some aggregation rules. The FedAvg [6] is given as: $G^{t+1} = G^t + \frac{\eta}{n}\sum_{i=1}^n(\mathbf{w}_i^{t+1} - G^t)$

To simulate a non-IID distribution, we assign data to clients according to the Dirichlet distribution [132].

4.3.4 Attack Strategies

In this paper, we will focus on targeted model poisoning attacks. The adversary manipulates the local models $\mathbf{w}$ to obtain the compromised clients model $\mathbf{w}'$ before being aggregated into

the global model G^{t+1}. The adversary wants the compromised global model $G^{'}$ to behave normally on all samples except for poisoned samples $x^{'} \in D^{P}$. Once the adversary attacks successfully, the compromised global model $G^{'}$ would misclassify poisoned samples into the attacker-chosen label τ rather than the genuine label y.

Data poisoning

In this attack strategy (see the right part in Fig.4.1), the adversary can only manipulate the training set on local clients by adding triggers into data samples or by changing the labels of a group of attacker-chosen data samples. By varying the Poisoned Data Rate (PDR), i.e., $PDR = \frac{N_{D^P}}{N_D}$, the attacker can make a trade-off between attack impact and attack stealthiness.

Model Poisoning

In this attack strategy (see the right part in Fig.4.1), the adversary can fully control a subset of the clients. Here, we denote the fraction of compromised clients as Poisoned Malicious Clients Rate (PMR), i.e., $PMR = \frac{m}{n}$. To increase the attack's impact on the aggregated model, the adversary can deliberately modify the model updates before submitting them to the aggregator. This is done by (1) turning up the scaling factor γ to increase attack impact (e.g., model-replacement attack [13]) and (2) constraining the training process by setting the scaling-coefficient parameters β to evade anomaly detection (e.g., constrain-and-scale [13]). In this attack strategy, the adversary can create multi-objective optimization $(\beta\ell(p,y) + (1-\beta)\ell_B(p,\tau))$. By tuning the scaling-coefficient parameter β, the adversary can attack more stealthily.

Propagation Error

[97] firstly introduces the *propagation error*. Suppose clients conduct a protocol $f = (\rho, A, \eta)$ at global iteration $t \in [T]$. Here, $\rho(G^t, D, t)$ is a gradient oracle that inputs the tth global model G^t, local dataset D and outputs the updated weights $\mathbf{w}^t$. Malicious clients conduct a poisoned protocol $f^* = (\rho^*, A, \eta)$ with $\rho^*(G^t, D^P, t)$. For any round t and any global model G^t and any dataset D, we have $\rho^*(G^t, D^P, t) = \rho(G^t, D, t) + \varepsilon$ with $\|\varepsilon\|_1 \leq \rho$. At each iteration t, the upper bound ρ on ε gives the *additive error* introduced by poisoning. Small additive errors introduced at early iterations can build upon each other and create large divergences. This is referred to as *propagation error*.

In this work, we design a multi-poisoning attack to instantiate such propagation errors.

In this attack strategy, the adversarial clients perform model poisoning or data poisoning attacks at every iteration. The adversarial clients can vary the PDR or the PMR. The upper bound ρ would vary with different PDR or PMR.

4.3.5 Characterization of Model Poisoning Attacks

To illustrate various model poisoning attacks more visually, we use a two-dimensional representation of the weight vectors of models. Then each model can be characterized by two factors: direction and magnitude. The cosine distance of the weight vectors can measure the direction between the two given models. The L_2 norm of the distance between the weight vectors can measure their magnitude difference. Fig.4.2 shows several types of poisoned models. The first poisoned client model $\overrightarrow{\mathbf{w}_1}$ is trained by adding a large fraction of poisoned data D^P into genuine dataset D. $\overrightarrow{\mathbf{w}_1}$ has an obvious direction deviation from the benign client models. The second type $\overrightarrow{\mathbf{w}_2}$ consist of four small vectors ($\overrightarrow{\mathbf{w}_5}$, $\overrightarrow{\mathbf{w}_6}$, $\overrightarrow{\mathbf{w}_7}$ and $\overrightarrow{\mathbf{w}_8}$). This type of attack is achieved by four distributed attacks (Distributed Backdoor Attack (DBA) [14]). Poisoned models trained by distributed attacks are more undetectable in direction and magnitude than centralised attacks. The next type is $\overrightarrow{\mathbf{w}_3}$, which has a less direction deviation but a larger magnitude difference. Such poisoned client models can be obtained by boosting the poisoned models with a large scaling factor γ (model-replacement attack [13]). The last type $\overrightarrow{\mathbf{w}_4}$ has similar representations with genuine models. It is more stealthy compared to the first three types. This poisoned model can be crafted by constrain-and-scale attacks [13].

4.4 DeMAC Design Principle and key observation

In this section, we introduce our proposed approach, DeMAC. First of all, we reintroduce the novel scoring method, GradScore as it is an essential component. We analyze that the PDR directly impacts the value of GradScore of a poisoned model. Then we describe how to detect malicious clients by evaluating the corresponding GradScore value in federated learning. Meanwhile, we analyze that based on GradScore, DeMAC can detect model poisoning attacks no matter the data distribution among clients. Finally, we give a security analysis that the proposed scoring method is not affected no matter how the adversary scales its model updates.

To reduce the poisoning task training loss $\ell_B(p,\tau)$, malicious clients should increase the

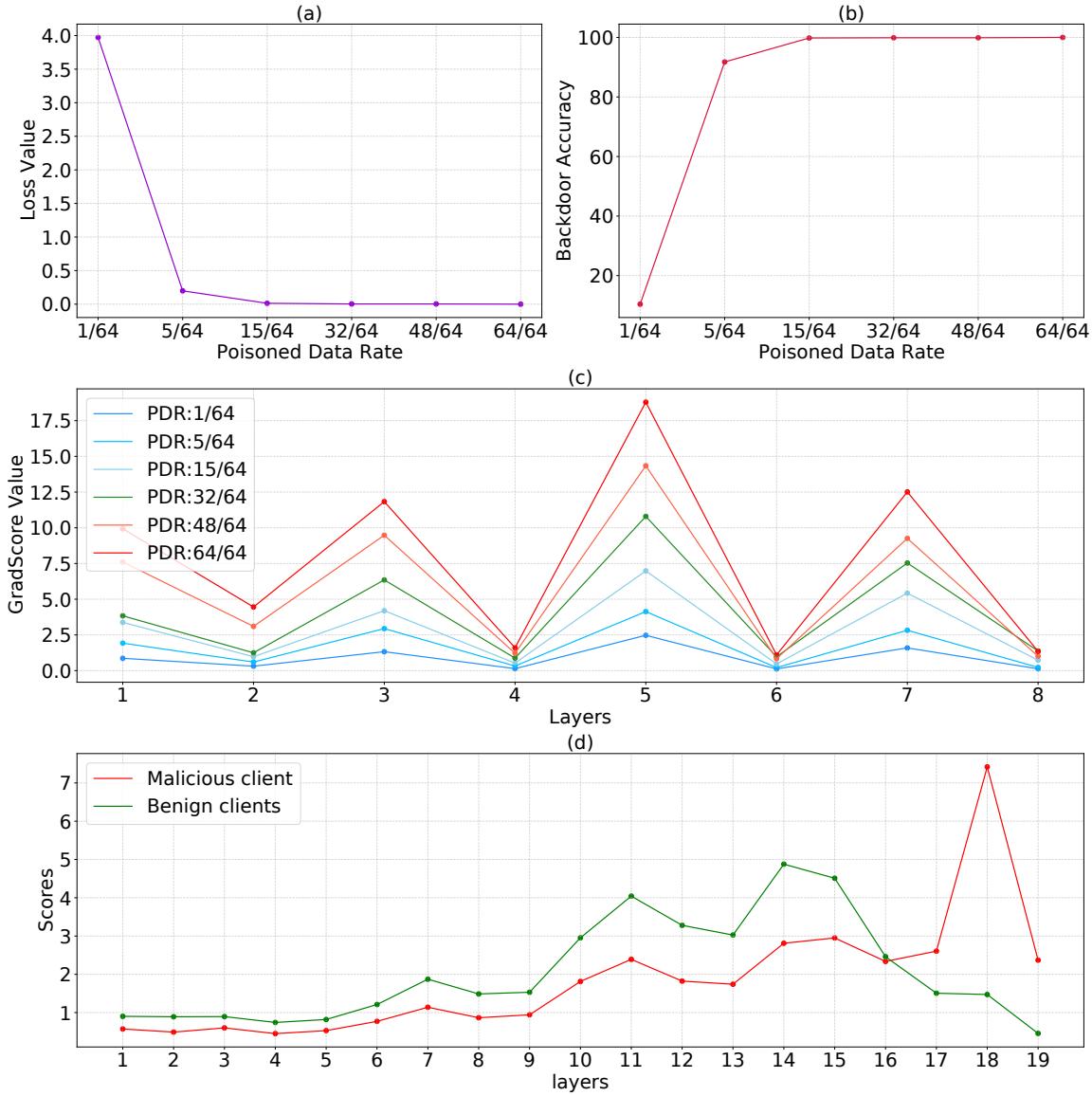


Figure 4.3: Impact of the PDR on loss value (a), Backdoor Accuracy (b) and GradScore value (c), the GradScore of the last layer gradients of the malicious updates and benign updates (d)

PDR. Therefore, the sum of the GradScore of samples is larger with higher PDR on malicious client datasets. In Fig.4.3 (a)(b)(c), we evaluated this inference, running backdoor training on MNIST dataset with a minibatch of 64 samples. With the same pre-trained model, the model trained with a higher poisoned data rate causes an obvious decrease in the backdoor training loss and a higher GradScore value.

Now we analyze how this scoring method can detect the model poisoning attacks in federated learning. No matter what the data distribution is among clients, the deviations

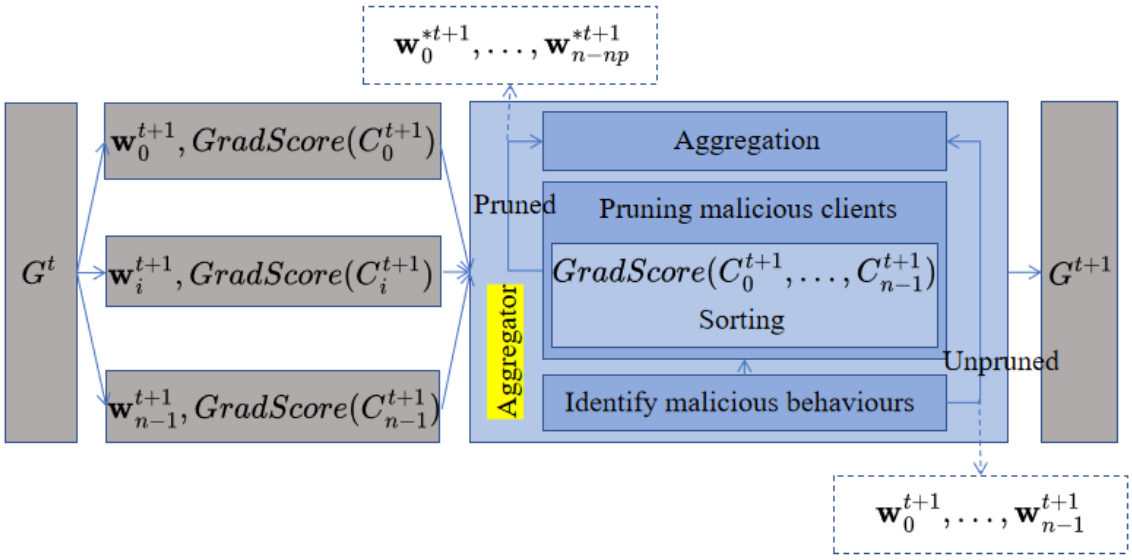


Figure 4.4: Illustration of DeMAC's workflow in global iteration t

between local models and the global model start to cancel out, i.e.,$\forall \mathbf{w} \in \{\mathbf{w}_i\}_{i=1}^m, \mathbf{w}_i^{t+1} - G^t \approx 0$ [13], in the benign setting. Therefore, the updates of benign local models, $d\mathbf{w} \approx \mathbf{w}^{t+1} - \mathbf{w}^t$ is bounded. The second observation is that when the global model starts to converge, poisoning behaviours on the malicious client will deviate the malicious updates from the current iteration global model [115] to reduce the training loss on the poisoning task. The GradScore of benign clients is small, while the GradScore of malicious clients is larger. Fig.4.3 (d) shows that the GradScore of the last layer gradients of the malicious client model is larger than benign clients. Therefore, malicious clients can be detected by comparing the GradScore of the last layer of local models.

To avoid detection, the adversary can try weak model poisoning attacks by limiting scaling up the poisoned model.

4.5 Overview and Design of DeMAC

In this section, we instantiate DeMAC for deep inspection and analysis of model updates to discover model poisoning attacks. We describe the design details of DeMAC below.

4.5.1 DeMAC Design

Fig.4.4 shows the main components and the workflow of DeMAC during global iteration t. It follows a deterministic algorithm and does not know the attack strategies or data distri-

butions. DeMAC is deployed during the training session before the testing phase. Firstly, it should identify malicious behaviours in federated learning systems. Here comes a design challenge. At the beginning of federated learning training, benign local models should update their parameters continually to make the global model converge to the global minimum. So how can we perceive the convergent trend? To solve this problem, we design a historical global model update record with a flexible look-back window size l. This history record records the continuous variation of model accuracy on the validation set. When the maximum value among this history record is lower than a threshold value α, the defence approach can start to process.

After malicious behaviours are identified, the GradScore values of corresponding clients would be sorted in ascending order to detect anomalous clients. The top p with the highest scores can be pruned and excluded from benign clients. The value of p depends on the number of malicious clients in one global iteration. Finally, DeMAC trains the global model, excluding the updates sent by malicious clients. DeMAC automatically generates a clean global model given the training algorithm, which means the proposed technique, DeMAC, is an efficient defence for federated learning systems.

In the rest of this section, we demonstrate a detailed description of every main component of DeMAC. Algorithm 2 outlines the procedure of DeMAC.

4.5.2 Identifying malicious behaviours

In designing DeMAC, the first step is identifying and measuring malicious behaviours in the federated learning system.

$D = \{(x_i, y_i)\}_{i=1}^{N}$ denotes the training dataset on client C_i. If the *GradScore* value for a client is significantly higher than other client *GradScore* values in the same global iteration round, it indicates that malicious behaviours might happen on this client. The step of calculating *GradScore* is shown in line 7 of Algorithm 2.

To avoid benign clients being misidentified when the global model is not yet converging, we demonstrate a validation phase to monitor the convergent trend. This validation phase consists of recording a group of previous global models and measuring the distance between the validation accuracies of two neighbouring global models. In the first step, we design a historical global model record, $history(G^0, ..., G^l)$, where $(G^0, ..., G^l)$ refers to a list of

Algorithm 2: Design of DeMAC

Input: n, G^0, T

// n is the number of clients in one iteration, G^0 is the initial global model, T is the number of global iterations

Output: G^T

// G^T is the updated global model after T iterations

1 **for** $t \in [1,..,T]$ **do**

2 $\quad historyRecord \leftarrow (G^{t-l},...,G^t)$;

$\quad$ // l is the look-back size

3 $\quad [V(G^{t-l},G^{t-l+1}),...,V(G^t,G^{t+1})] \leftarrow VALIDATE(historyRecord)$;

4 $\quad$ **if** $max[V(G^{t-l},G^{t-l+1}),...,V(G^t,G^{t+1})] < \sigma$ // σ is the convergence threshold; Malicious behaviours exist

5 $\quad$ **then**

6 $\quad\quad$ **for** $i \in [C_0^{t+1},...,C_{n-1}^{t+1}]$ **do**

7 $\quad\quad\quad GradScore(C_i^{t+1}) = \|g\{(x,y)\}\|_2$ // $\|g\{(x,y)\}\|_2$ is the L_2-norm of gradients of parameters in final layer of models

8 $\quad\quad$ **end**

9 $\quad\quad SCORE \leftarrow [GradScore(C_0^{t+1}),...,GradScore(C_{n-1}^{t+1})]$;

10 $\quad\quad Sort(SCORE)$;

11 $\quad\quad Pruned(\mathbf{w}_0^{*t+1},...,\mathbf{w}_{n-np}^{*t+1}) \leftarrow Pruning_{p\%}([\mathbf{w}_0^{t+1},...,\mathbf{w}_{n-1}^{t+1}])$ // $p\%$ is the pruning rate

12 $\quad\quad SendPruned(\mathbf{w}_0^{*t+1},...,\mathbf{w}_{n-np}^{*t+1}) \rightarrow Aggregator$

13 $\quad$ **else**

14 $\quad\quad SendUnpruned(\mathbf{w}_0^{t+1},...,\mathbf{w}_{n-1}^{t+1}) \rightarrow Aggregator$ // Malicious behaviours do not exist

15 $\quad$ **end**

16 $\quad G^{t+1} \leftarrow G^t + \frac{\eta}{L-1}\sum_0^L(\mathbf{w}_i^{t+1} - G^t)$ // Global Aggregating, η is the global learning rate

17 **end**

previous global models, and l is the size of the sliding window. In the second step, we define the distance between neighbouring global model validation accuracies as:

$$v(G^{i-1},G^i) = |Acc(G^i) - Acc(G^{i-1})| \tag{4.1}$$

Where $Acc(G^i)$ is the accuracy of global model G^i on the validation dataset. We use a list $V(G^0,G^1),...,V(G^{l-1},G^l)$, to contain neighboring validation variations of a historical record, $history(G^0,...,G^l)$. The related steps are shown in lines 2-3 of Algorithm 2. If

the maximal$V(G^i, G^{i+1}) \in [V(G^0, G^1), ..., V(G^{l-1}, G^l)]$ is below the threshold σ, the global model can be regarded as convergence.

When the aggregator detects unusually large *GradScore* values sent by some clients and the global model converges, malicious behaviours can be identified in the federated system.

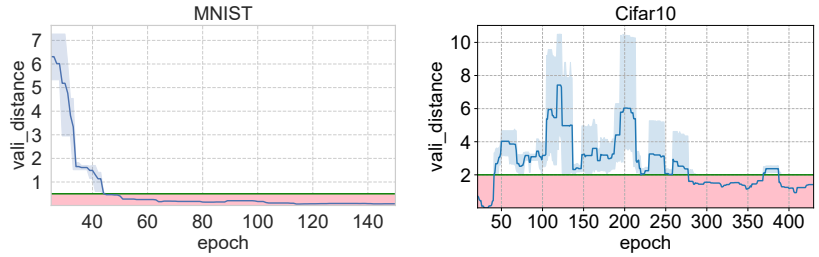
4.5.3 Pruning and excluding malicious clients

After malicious behaviours are identified, the next step in DeMAC is to identify and exclude anomalous clients based on corresponding *GradScore* values. First, the *GradScores* to corresponding clients are sorted in ascending order. The top p clients with the highest scores are pruned and excluded from the benign client list. The parameter p depends on the number of anomalous clients in one global iteration. Only one malicious client should be excluded when the adversary takes model-replacement attack strategies. However, malicious clients may collude and strengthen the impact of poisoning in one iteration. Considering the real-world federated learning deployments, it is unrealistic to assume that the fraction of malicious clients is above the range $(0 < m < n/2)$. In an application scenario like Gboard [126], over 50% malicious clients mean the adversary should control at least 500 million Android devices. That is incredible [133]. So we only consider below 50% cases. Generally, p is set to 0.5. In this work, we assume the server has a knowledge of the number of malicious clients at one iteration. Hence, the server can decide the value of p. The sorting and pruning step is shown in lines 9-11 of Algorithm 2.

The aggregator excludes the updates sent by malicious users in the current iteration and trains the global model on the remaining model updates (line 16 of Algorithm 2). The global training algorithm varies based on the underlying training algorithm used in the application. We use FedAvg [6] to train the global model in this proposed work.

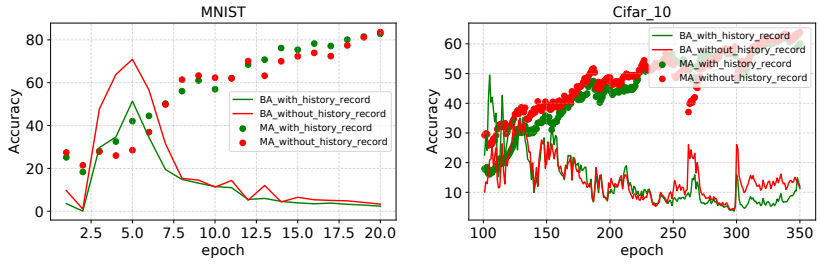
Layer	**Size**
Input	$28 \times 28 \times 1$
Convolutional + ReLU	$3 \times 3 \times 30$
Max Pooling	2×2
Convolutional + ReLU	$3 \times 3 \times 5$
Max Pooling	2×2
Fully Connected + ReLU	100
Softmax	10

Table 4.2: the CNN Network Architecture



(a) DeMAC is triggered by the threshold *vali_distance* value 0.5, MNIST

(c) DeMAC is triggered by the threshold *vali_distance* value 2, CIFAR10



(b) (d)

Figure 4.5: Measuring the maximum *vali_distance* $V(G^i)$ enables DeMAC to detect convergence of the global model (a)(c). DeMAC with history global model record vs DeMAC without history global model record (b)(d)

4.6 Evaluation Setup

In this section, we give the details of the experimental setup and evaluation metrics used in this work for evaluating the effectiveness of DeMAC.

4.6.1 Experimental Setup

Datasets and global-model settings: In this work, two well-known benchmark datasets MNIST [121] and CIFAR10 [134] are considered to evaluate DeMAC. MNIST dataset is a ten-class-balanced image classification task with 70000 grey-scale images. And CIFAR10 dataset is a ten-class image classification task with 60000 RGB images. It is assumed there are 100 clients for global training. To simulate non-IID distribution, data is assigned to clients according to Dirichlet distribution with concentration parameter α, so client datasets are unbalanced to classes. The distribution is more concentrated when the value of α is smaller. It was different from some previous works, which forced the clients to have the same number of data. In our data distribution generation, the data were divided into disjoint partitions with varying sizes. On the contrary, the distribution tends to be more uniform.

Without Dirichlet sampling, data are uniformly distributed (IID) to clients. Unless otherwise mentioned, the concentration parameter α is set to 0.5. For MNIST dataset, a four-layer Convolutional Neural Network (see Tab.4.2) is used. For CIFAR10, ResNet18 [135] architecture is considered the global model.

Federated Learning settings: FedAvg [6] is considered as the FL method. In each global round, 10 of 100 clients are randomly selected. Considering the different characteristics of the datasets, we adopt the following parameter settings for federated training: for MNIST, clients train for 1 local epoch with a local learning rate of 0.1. For CIFAR10, clients train for 2 local epochs with a local learning rate of 0.1.

Attack strategy. We consider four targeted model poisoning attacks, single-shot model-replacement attacks [13], constrain-and-scale [13], and DBA [14] and multi-poisoning attacks.

(1) Model-replacement attacks, constrain-and-scale and DBA. In the case of MNIST, we modify the pixels of the digital image at training time, causing the images with pixel-pattern to be classified towards a target class. On CIFAR10 dataset, we apply the same attack strategy as on MNIST dataset. The attackers can set the scaling parameter γ for single-shot and DBA to control the impact of model poisoning. Unless otherwise mentioned, we set γ to 30, and PDR is set to $30/64$ with a local batch size of 64. For single-shot model replacement attacks and constrain-and-scale attacks, we assume that attackers perform attacks after 60 rounds for MNIST and 400 rounds for CIFAR10. For DBA, as attackers split triggers into four equal parts, it is assumed that malicious clients attack at round 62, 64, 66, and 68 for MNIST and at round 402, 404, 406, and 408 for CIFAR10.

(2) Multi-poisoning attacks. In the case of MNIST, adversarial clients perform the multi-poisoning attack at round 10 and 20, respectively. In the case of CIFAR10, adversarial clients perform the multi-poisoning attack at round 80 and 300, respectively. Unlike the three types of attacks described above, the multi-poisoning attack executes every round after being performed. We set γ to 1, and the PDR is set to $30/64$.

Detecting time.

(1) Single-shot model-replacement attack, constrain-and-scale and DBA. Unlike existing works [89] [77] manually setting detecting time, DeMAC can spontaneously detect malicious clients according to the information provided by the historical record. We set the sliding

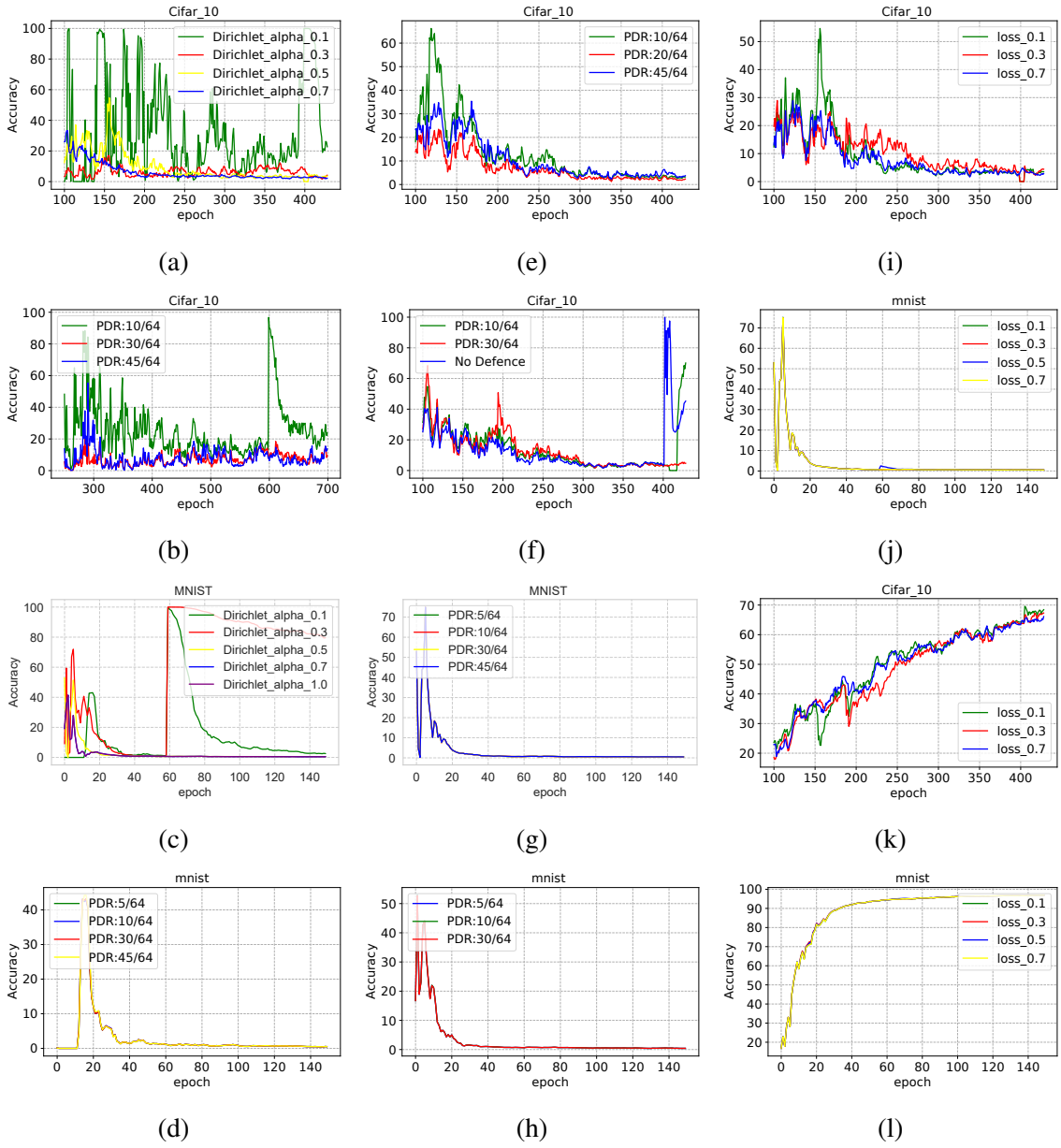


Figure 4.6: The ASR for DeMAC against model-replacement attack under different non-IID settings (a)(c). The ASR for DeMAC against model-replacement attack, where 0.1 of concentration parameter α, threshold σ value 2 for CIFAR10, and σ value 0.6 for MNIST are used (b)(d). Impact of the poisoned data rate on DeMAC against model-replacement attack (e)(g) and distributed backdoor attack (f)(h). MA and BA of the global model under the protection of DeMAC against constrain-and-scale attack with different β values (i)(j)(k)(l).

window size l to 15 for MNIST and 20 for CIFAR10. DeMAC will be triggered when the maximum distance between neighbouring global model accuracies on the validation set $V(G^i)$ is below the predefined threshold σ. We choose $\sigma = 0.5$ for MNIST and $\sigma = 2$ for CIFAR10.

(2) Multi-poisoning attacks. Our experiments show that the increase in global model

accuracy has an obvious oscillation rather than increasing monotonically during the early training stage. Hence, the predefined threshold σ is looser than the above setting in this case.

4.6.2 Evaluation Metrics

We consider two evaluation metrics for evaluating the accuracy and efficiency of DeMAC. *Main task accuracy (MA)* is used to evaluate the accuracy of the global model on the main task. MA is the ratio of testing examples that are correctly classified. *Backdoor Accuracy (BA) or Attack Success Rate (ASR)* is the ratio of poisoned examples that are classified as target labels by the global model. We define an evaluation metric for measuring the performance of DeMAC. *Computation cost per round (CCR)* measures the computation cost at one round in the Byzantine-robust FL system.

4.7 Evaluation Results

Efficiency of history record. Detecting/Attacking timing is rarely discussed in previous defence works. [89] [13] discussed that the impact of model-replacement attacks in early rounds is not durable as the ASR decreases sharply within several rounds. The poisoning impact tends to stay long in the later training rounds. The simple way is to detect when the global model starts training. However, it is not cost-friendly to start detecting from train-from-scratch to defend against poisoning attacks, such as model-replacement attacks. To solve this problem, we combine DeMAC with a historical record. By applying this historical global model record, DeMAC can track the convergence of global training. Fig.4.5(a)(b) shows that DeMAC will be triggered when the maximum $vali\_distanceV(G^i)$ is below the predefined threshold (pink area in Fig.4.5(a)(b)). The DeMAC defence is performed only when the global model starts to stabilize. Fig.4.5(b)(d) shows the comparison between DeMAC with a historical global model record and DeMAC without a historical global model. It is not difficult to see that enabling DeMAC in early rounds may cause a delay in the convergence of the global model. It might be a drawback in federated learning deployments. In Fig.4.5(b) for MNIST, in the initial 20 rounds, DeMAC without a historical global model record shows a higher error rate than DeMAC with a historical global model record. The same result is shown in Fig.4.5(d).

Dirichlet (α)	**PDR**	σ	**Round**	**ASR** (%)
0.1	$30/64$	2	400	99.98
0.1	$30/64$	2	600	2.411
0.3	$30/64$	2	400	7.888
0.5	$30/64$	2	400	5.133
0.7	$30/64$	2	400	3.033

Table 4.3: Impact of the degree of non-IID on CIFAR10

Impact of the degree of non-IID: Fig.4.6(a)(b)(c)(d) shows the impact of the non-IID degree on DeMAC. First, from Fig.4.6(a)(c), we observe that with PDR ($30/64$) and threshold σ fixed, the ASR can be reduced to nearly 0% when the non-IID degree is larger than some threshold. When α is set to 0.1 for CIFAR10 or α is set below 0.3 for MNIST, DeMAC cannot detect malicious clients, which causes a high ASR on the global model. In Fig.4.6(b), we postpone the attacking time at round 600, when the global model stabilises. We observe that at the same PDR ($30/64$), threshold σ value (2) and non-IID setting ($\alpha = 0.1$), DeMAC can mitigate poisoned updates and reduce backdoor accuracy to a low level compared with the failure of detection in Fig.4.6(a). In Fig.4.6(d), we set threshold σ value to 0.6 rather than the default value 0.5 with non-IID setting($\alpha = 0.1$). So, DeMAC would be triggered and start to detect malicious clients earlier. BA of the global model can be reduced to nearly 0% under all PDR settings. From the above analysis, it is not difficult to see that the success of model poisoning attacks is highly rated to the convergent trend of the global model.

We also add some tables corresponding to the results in Tables 4.3, 4.4 show the impact of the non-IID degree on DeMAC. From Table 4.3, with other hyperparameters fixed, DeMAC can decrease the ASR to a very low level when α is larger than 0.1. When α is set to 0.1 and the attacking time is postponed to 600 round, DeMAC can mitigate the poisoned updates. Table 4.4 shows similar results. When threshold σ is set from 0.5 up to 0.6, DeMAC is able to reduce the ASR to a low level. From the above analysis, we can see that the success of model poisoning attacks is closely related to the convergence of FL training.

Impact of Poisoned Data Rate (PDR): Fig.4.6(e)(f)(g)(h) shows the impact of the poisoned data rate on DeMAC. In Fig.4.6(e)(g), DeMAC can mitigate malicious client updates and reduce the ASR to a low level for both two datasets. In Fig.4.6(f)(h), we evaluate the efficiency of DeMAC against distributed backdoor attacks. Fig.4.6(f) shows that DeMAC cannot mitigate the last split backdoor attack when PDR is low. One possible reason is that

Dirichlet (α)	**PDR**	σ	**Round**	**ASR** (%)
0.1	30/64	0.6	60	1.260
0.1	30/64	0.5	60	99.77
0.3	30/64	0.5	60	99.97
0.5	30/64	0.5	60	0.657
0.7	30/64	0.6	60	0.680

Table 4.4: Impact of the degree of non-IID on MNIST

PDR	σ	**Round**	**ASR** (%)
10/64	2	400	3.022
20/64	2	400	2.211
30/64	2	400	4.722

Table 4.5: Impact of Poisoned data rate (PDR) on CIFAR

compared with the central backdoor attack, the distributed property of DBA makes attack behaviour more stealthy. In Fig.4.6(h), DeMAC can decrease the attack impact under all the attack strategies.

We also add some tables corresponding to the results in Tables 4.5, 4.6, 4.7, 4.8 show the impact of PDR on DeMAC. From Tables 4.5, 4.6, DeMAC can effectively mitigate malicious behaviours. From Table 4.7, DeMAC cannot work well when PDR is set to 10/64. The possible reason is that with few data samples being poisoned, DBA is too stealthy to be detected.

Defending Anomaly-Evasion Attack: As discussed in section 4.3, attackers can balance the impact and stealth of attack by varying the scaling-coefficient parameter β. $\ell_B(p,\tau)$ is calculated as the L_2 norm between the current poisoned and round global models. Figure 4.6(i)(j)(k)(l) shows that DeMAC can successfully mitigate attack impact for both datasets and different β values.

We also add some tables corresponding to the results in Tables 4.9, 4.10 show DeMAC can successfully mitigate attack impact for two datasets.

PDR	σ	**Round**	**ASR** (%)
5/64	0.5	60	0.635
10/64	0.5	60	0.646
30/64	0.5	60	0.635
45/64	0.5	60	0.644

Table 4.6: Impact of Poisoned data rate (PDR) on MNIST

PDR	σ	**Round**	**ASR** (%)
10/64	2	400	34.42
30/64	2	400	3.488

Table 4.7: Impact of Poisoned data rate (PDR) on CIFAR for defending DBA

PDR	σ	**Round**	**ASR** (%)
5/64	0.5	60	0.657
10/64	0.5	60	0.747
30/64	0.5	60	0.724

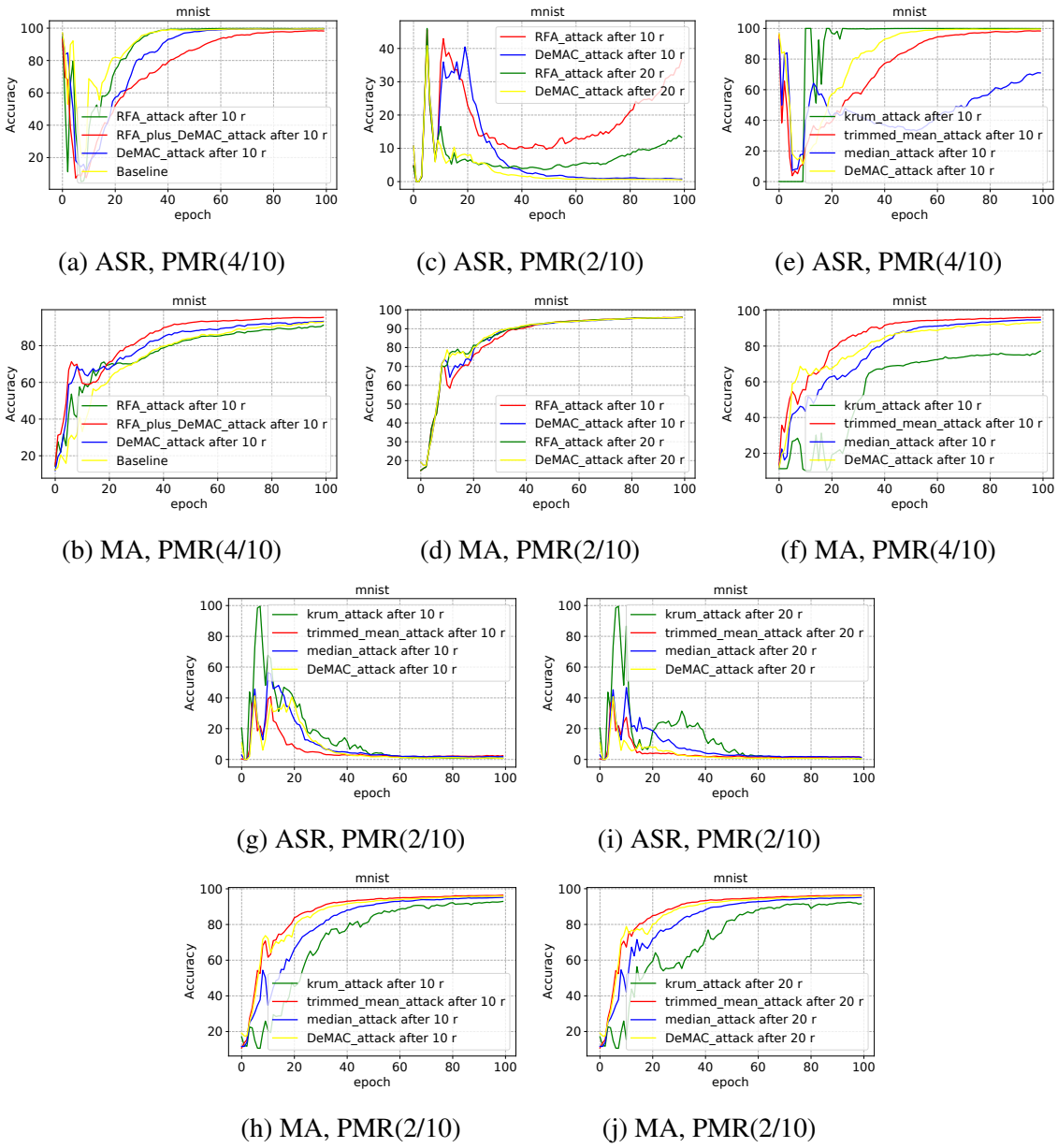
Table 4.8: Impact of Poisoned data rate (PDR) on MNIST for defending DBA

β	σ	**Round**	**ASR** (%)	**MA** (%)
0.1	2	400	5.133	64.25
0.3	2	400	5.6	66.65
0.7	2	400	3.133	63.43

Table 4.9: Defending Anomaly-Evasion Attack on CIFAR

β	σ	**Round**	**ASR** (%)	**MA** (%)
0.1	0.5	60	0.635	94.4
0.3	0.5	60	0.646	94.45
0.5	0.5	60	2.531	94.07
0.7	0.5	60	0.635	94.42

Table 4.10: Defending Anomaly-Evasion Attack on MNIST



(a) ASR, PMR(4/10) (c) ASR, PMR(2/10) (e) ASR, PMR(4/10)

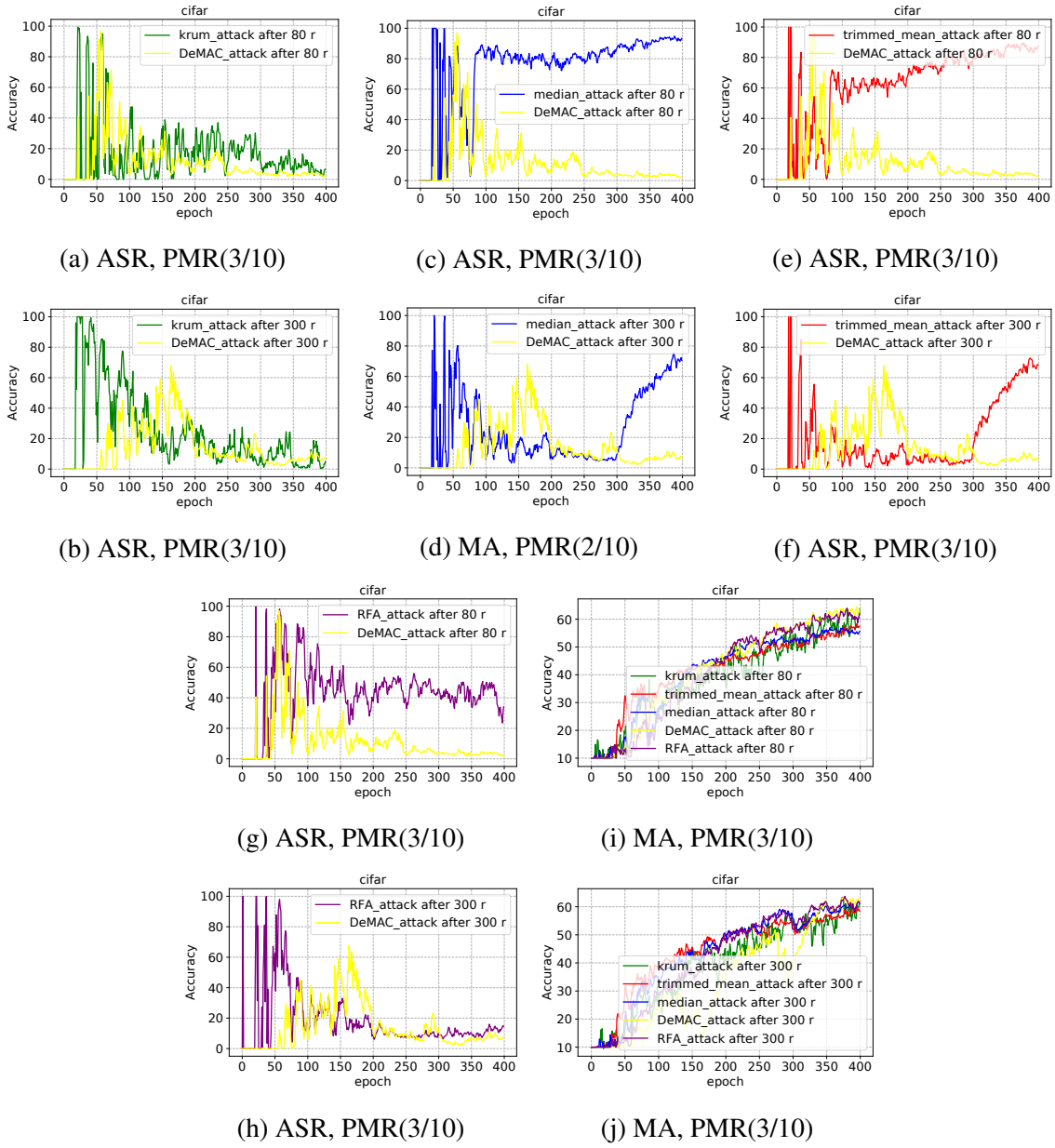
(b) MA, PMR(4/10) (d) MA, PMR(2/10) (f) MA, PMR(4/10)

(g) ASR, PMR(2/10) (i) ASR, PMR(2/10)

(h) MA, PMR(2/10) (j) MA, PMR(2/10)

Figure 4.7: ASR and MA of malicious detection for different detection methods. Concentration parameter $\alpha(0.5)$, MNIST dataset and scaling parameter $\gamma(1)$ are used.

Detecting multi-poisoning attacks: Fig.4.7 and Fig.4.8 show the comparison results on two datasets for detection methods, different attack timing, and different numbers of malicious clients. From our results, it is not difficult to see that malicious perturbations in every iteration can gradually compromise baseline Byzantine-robust FL algorithms and cause high ASR. DeMAC can effectively suppress such propagation errors. Here are several observations. Firstly, DeMAC can mitigate attack impact and reduce the ASR to a low level in most cases, except in the case (MNIST dataset, PMR(4/10), attack after ten rounds, Fig.4.7(a)(b)).



(a) ASR, PMR(3/10) (c) ASR, PMR(3/10) (e) ASR, PMR(3/10)

(b) ASR, PMR(3/10) (d) MA, PMR(2/10) (f) ASR, PMR(3/10)

(g) ASR, PMR(3/10) (i) MA, PMR(3/10)

(h) ASR, PMR(3/10) (j) MA, PMR(3/10)

Figure 4.8: ASR and MA of malicious detection for different detection methods. Concentration parameter $\alpha(0.5)$, MNIST dataset and scaling parameter $\gamma(1)$ are used.

All the Byzantine robust methods fail in this case (Fig.4.7(a)(b)). The main reason could be that the detection methods are too hard to distinguish benign clients from many malicious clients, as the global model is unstable in the first ten rounds. Second, in the case (MNIST dataset, PMR(2/10), attack after ten rounds) and case (MNIST dataset, PMR(2/10), attack after 20 rounds), all the defending methods except RFA [119] can mitigate attack impact. In subsection 4.2.1, we discuss that by carefully setting the scaling factor γ and then controlling the total weights of the outliers, the attacker can bypass RFA. We set the scaling

factor γ as 1 and PDR as $30/64$. RFA fails to detect malicious behaviours. This conclusion is in line with conclusions from previous work [14]. DeMAC, trimmed_mean [45], and median achieve comparable main accuracy and outperform Krum [46]. The main reason could be that Krum selects one client update to represent the global model. Therefore, due to the heterogeneous data distribution, these chosen model updates cannot achieve the same performance as on the global test dataset. This conclusion is in line with conclusions from previous work [97]. Third, in cases (CIFAR10 dataset, attack after 80 rounds), all the defending methods except DeMAC fail to eliminate the attack impact. In cases, (CIFAR10 dataset, attack after 300 rounds), DeMAC, Krum, and RFA can reduce the ASR to a low level, but median and trimmed-mean still cannot defend against attack behaviours. As we discuss in subsection 4.2.1, the assumption of trimmed-mean does not hold for model poisoning attacks. Therefore, this observation is in line with the discussion from prior sections. Fourth, in the case (CIFAR10 dataset, attack after 300 rounds), other defending methods can achieve comparable main accuracy as DeMAC. However, in the case (CIFAR10 dataset, attack after 80 rounds), the main accuracy of DeMAC outperforms other defending methods after 400 rounds.

Performance Comparison: In this work, we define the CCR as the time for one iteration of training in the FL system equipped with the chosen defence method. We use two tables to show the comparison of the effectiveness of DeMAC with other Byzantine-robust methods on two different datasets. In these experiments, we take the multi-poisoning attack strategy. The details of this attack strategy are described in section 6.1. Experimental Setup. In Table 4.11, RFA [119] is the most time-consuming method. Median [45] and trimmed-mean [45] are the most time-saving methods. DeMAC and Krum [46] show similar performance.

Defense methods	**Before Attack** (sec)	**After Attack** (sec)
Krum	62.648	127.47
Median	44.867	105.304
RFA	61.892	141.256
Trimmed-mean	43.933	108.236
DeMAC	58.688	126.37

Table 4.11: The comparison of the effectiveness of DeMAC with other Byzantine-robust methods on CIFAR set

Defense methods	**Before Attack** (sec)	**After Attack** (sec)
Krum [46]	19.837	59.609
Median [45]	19.652	61.403
RFA [119]	18.086	59.699
Trimmed-mean [45]	18.700	62.448
DeMAC	21.473	67.260

Table 4.12: The comparison of the effectiveness of DeMAC with other byzantine-robust methods on MNIST set

4.8 Conclusion

Backdoor attacks and, more specifically, model poisoning attacks are a big challenge faced by federated learning. To address the shortcomings of the existing defence approaches, we proposed a novel defence system, which is called DeMAC to defend against malicious attacks by measuring the difference in the contribution of benign clients and malicious clients to the global model. We defined a new metric GradScore, to compute the L2-norm of the gradients of the last layer of contributed model updates, which is shown to be effective in detecting updates from malicious clients. Furthermore, we utilized the history records of the contributed model updates to enhance the malicious client detection performance. We evaluated and compared DeMAC with state-of-the-art defence techniques over various attack strategies and datasets. Experiment results show that DeMAC can effectively mitigate the model poisoning attacks without sacrificing the performance of the main task and significantly outperforms the existing defence approaches. Future research directions include extending the proposed method to defend against adaptive attacks based on well-known non-targeted model poisoning frameworks. In the next chapter, we focus on addressing the security issues caused by adaptive untargeted model poisoning attacks.

Chapter 5

A Robust and Efficient Federated Learning Algorithm against Adaptive Model Poisoning Attacks

In this chapter, we investigate how to defend against adaptive model poisoning attacks. With the undetectable characteristic, adaptive model poisoning attacks can combine with any other attacks, bypassing the detection and violating the availability of federated learning systems. Existing defences are vulnerable to adaptive model poisoning attacks, as model poisoning-related features are tailored to these methods and compromise the accuracy of the FL model. We first present a unified reformulation of existing adaptive model poisoning attacks. Analyzing the reformulated attacks, we find that the detectors should reduce the attacker's optimization cost functions to defeat adaptive attacks. However, existing defences do not consider the causes of model parameters' high dimensionality and data heterogeneity. We propose a novel robust FL algorithm, FedDet, to tackle the problems. By splitting the local models into layers for robust aggregation, FedDet can overcome the issue with high dimensionality while keeping the functionality of layers. During the robust aggregation, FedDet normalizes every slice of local models by the median norm value instead of excluding some clients, which can avoid deviation from the optimal model. Furthermore, we conduct a comprehensive security analysis of FedDet and an existing robust aggregation method. We propose the upper bounds on the perturbations disturbed by these adaptive attacks. It is found that FedDet can be more robust than Krum [46] with a smaller perturbation upper bound un-

der attacks. We evaluate the performance of FedDet and four baseline methods against these attacks under two classic datasets. It demonstrates that FedDet significantly outperforms the existing compared methods against adaptive attacks. FedDet can achieve 60.72% accuracy against min-max attacks.

5.1 Introduction

The Internet of Things (IoT) plays an important role in our daily lives as it provides intelligent services and applications empowered by artificial intelligence (AI) [1] [2] [3]. AI techniques such as deep learning (DL) process raw data generated from ubiquitous IoT devices and train data models for enabling intelligent services or infrastructures, such as smart healthcare, smart transportation, and smart cities. Traditionally, AI functions are placed in a cloud server for data collecting and modeling [4] [5]. However, With such an explosive growth of IoT data at the network edge, the offloading of massive IoT data to remote servers may be infeasible due to the constrained network resources, bandwidth and incurred latency. Besides, the use of third-party servers for AI training also raises privacy concerns such as leakage of sensitive information (e.g., user addresses or personal preferences). Thus, it may not be feasible to apply centralized AI techniques in realistic scenarios. To address the above issues, a novel distributed training regime, federated learning (FL), has been proposed for building intelligent and privacy-enhanced IoT systems. FL is an efficient and scalable distributed machine learning paradigm that provides excellent privacy to clients [6]. With the application of federated learning, resource-constrained node devices (e.g., Internet of Things (IoT) devices and sensors) can build a knowledge-shared model while keeping the raw data local [7]. Hence, federated learning plays a critical role in bringing AI to IoT systems and applications in terms of training AI models, online model fine-tuning and preserving data privacy [4] [5]. However, due to its distributed characteristic, FL leaves the door open for adversaries as they can send poisoned local models to the central server without being checked. Hence, an FL system can be vulnerable to model poisoning attacks [8] [9] [10] [11]. Poisoning attacks consist of backdoor attacks [13] [14] [15] and model poisoning attacks [16] [17] [18] [19]. Backdoor attacks aim to insert a backdoor into the trained global model and make the global model mislabel a small group of samples with chosen triggers into targeted labels [17] [19] [20]. Model poisoning attacks attempt

to hamper the global model's main accuracy. In this work, we focus on model poisoning attacks in FL as model poisoning attacks can cause denial-of-service among a large population of end services in FL deployments [126] [127] [128]. In this work, we investigate the model poisoning attacks against federated learning systems, which can be transferred into some resource-constrained scenarios (e.g., Multi-UAV Systems [12] and cause denial-of-service (DoS) in IoT systems). And we propose a robust algorithm to defend against such attacks.

The aforementioned Byzantine-robust algorithms [98] [59] [46] [45] [11] [59] [46] [45] [55] have been discussed widely in the literature and perform well against general model poisoning attacks such as label-flipping attacks [17]. However, the adversary can optimize poisoning strategies when the adversary has knowledge of the aggregation methods [17] [18]. Such types of attacks are called adaptive model poisoning attacks. During the FL training, malicious clients can adaptively manipulate local model parameters tailored to the aggregation rule. By being well-designed, these local model parameters could bypass the defence methods like Krum [46] and compromise the FL training model.

In view of the above research issues, we are motivated to design an efficient, robust aggregation method and defeat adaptive attacks. We first reformulate adaptive attacks and discuss their main characteristics. We found that the defender should try to reduce the attacker's optimization cost functions to defeat adaptive attacks. However, some existing methods like Krum or Muti-Krum [46] ignore the curse of model parameters' high dimensionality. The attacker may update partial parameters or infrequent parameters. The attacker can cause negative effects when the value of the cost function is not large. Therefore, the adaptive attacks can bypass existing methods. On the other hand, some methods [45] ignore the data heterogeneity. They exclude potential malicious parameters or misclassified honest parameters, which cause deviation from the optimal global model. In this paper, we propose a Byzantine-robust FL method, FedDet, which consists of two main steps. In the first step, FedDet splits and groups local models by layers. Then, the sliced parameters in one group are normalized by the median of the norms. The first step, splitting, can decrease the high dimensionality of parameters. Besides, splitting by layers rather than random splitting [84] can keep the functionality of different layers. The second step, normalization, considers all parameters in the same group. We also discuss the certified robustness of FedDet based on

an existing certified radius proposed by [97]. As an extension, we provide a detailed security analysis of FedDet and give the upper bounds of the perturbations given by the adaptive attacks. We evaluate the performance of FedDet against six types of attacks. Experiment results demonstrate that FedDet outperforms other baseline works against adaptive attacks.

The main contributions of our work are summarized as follows:

- We present a unified reformulation of existing adaptive model poisoning attacks. The summary of the reformulation can be used for related works to verify the efficiency of their methods against adaptive attacks. To the best of our knowledge, no existing works give such a comprehensive discussion of state-of-the-art adaptive model poisoning attacks.

- Based on our discussion of the main characteristics of adaptive model poisoning attacks, we reveal the two main causes of why existing defence methods are not efficient: model parameters' high dimensionality and data heterogeneity. Then we propose FedDet, consisting of two steps, with the first step splitting overcoming the issue of high dimensionality and the second step normalizing overcoming the issue of heterogeneity.

- We evaluate FedDet against six designed adaptive attacks tailored to it. From the results, FedDet significantly outperforms baseline works against adaptive attacks. Besides, we discuss the certified radius of FedDet. As an extension, we provide a detailed security analysis of FedDet and an existing robust aggregation method, Krum. By comparing the upper bounds of the perturbations caused by DNY-OPT attacks corresponding to these two robust methods, we can see that FedDet outperforms Krum according to the upper bounds.

5.2 Existing Byzantine-robust Algorithms

The principle of existing Byzantine-robust defences [46] [45] is to train a global model with high performance, even if there are some malicious clients.

Krum [46] attempts to select a representative as the aggregated model update for every training round. Suppose there are n chosen local clients in every training round. And m clients among local clients are malicious. The score for the ith client is calculated as $s_i = \sum_{\mathbf{w}_j \in \Gamma_{i,n-m-2}} \|\mathbf{w}_j - \mathbf{w}_i\|_2^2$, where $\Gamma_{i,n-m-2}$ is the set of $n-m-2$ local clients that have the

Adaptive attacks	Optimization cost functions	Constraints
STAT-OPT attacks	$\max_{\mathbf{w}_1^{'},..,\mathbf{w}_m^{'}} \mathbf{s}^T(G-G^{'})$	$s.t. \quad G = f_{agr}(\mathbf{w}_1,..,\mathbf{w}_m,\mathbf{w}_{m+1},..,\mathbf{w}_n)$ $G^{'} = f_{agr}(\mathbf{w}_1^{'},..,\mathbf{w}_m^{'},\mathbf{w}_{m+1},..,\mathbf{w}_n)$
DNY-OPT attacks	$\max_{\gamma,\nabla^p} \|G-G^{'}\|_2$	$s.t. \quad G = f_{avg}(\mathbf{w}_1,..,\mathbf{w}_m,\mathbf{w}_{m+1},..,\mathbf{w}_n)$ $G^{'} = f_{agr}(\mathbf{w}_1^{'},..,\mathbf{w}_m^{'},\mathbf{w}_{m+1},..,\mathbf{w}_n)$ $\mathbf{w}_{i\in[m]}^{'} = G+\gamma\nabla^p$
Min-max attacks	$\max_{\gamma,i\in[m+1,n]} \|\mathbf{w}^{'}-\mathbf{w}_i\|_2$	$s.t. \quad \mathbf{w}_{i\in[m]}^{'} = G+\gamma\nabla^p$ $\|\mathbf{w}^{'}-\mathbf{w}_i\|_2 \leq \underset{i,j\in[m+1,n]}{maximum} \|\mathbf{w}_i-\mathbf{w}_j\|_2$
Min-sum attacks	$\max_{\gamma} \sum_{i\in[m+1,n]} \|\mathbf{w}^{'}-\mathbf{w}_i\|_2^2$	$s.t. \quad \mathbf{w}_{i\in[m]}^{'} = G+\gamma\nabla^p$ $\sum_{i\in[m+1,n]} \|\mathbf{w}^{'}-\mathbf{w}_i\|_2^2 \leq maximum \sum_{i,j\in[m+1,n]} \|\mathbf{w}_i-\mathbf{w}_j\|_2^2$

Table 5.1: Summary of reformulation of existing adaptive attacks

smallest Euclidean distance to $\mathbf{w}_i$ (client$i$'s parameters). So, the client with the smallest score will be selected as the representative. This representative model update will be the global model for the next training round.

Multi-Krum [46] is a variant of Krum. Multi-Krum collects a set of clients with the smallest scores using Krum and repeats this process for the remaining updates until the set has c updates, such as $n-c > 2m+2$. Then, it takes the average among this set of clients.

Median [45] is a coordinate-wise aggregation rule. The coordinate-wise median of sorted local models is selected as the aggregated global model update. Instead of using the mean value among local clients, this aggregation rule considers the coordinate median value of the parameters as the corresponding parameter in the global model for the next iteration. The coordinate-wise median is agnostic to the actual malicious rates.

Trimmed-mean [45] is another coordinate-wise aggregation rule. Suppose the trimmed parameter is $k < \frac{n}{2}$. The server removes the k maximum and minimum coordinates in the model updates and then uses FedAvg to aggregate the remaining parameters for the next training round. Trimmed-mean relies on the assumption that the coordinate of the attacker would either be the minimum or the maximum value of the corresponding parameters. However, this assumption does not hold for model poisoning attacks [97]. Therefore, even a single attacker can compromise the trimmed mean. Unlike the coordinate-wise median, the Trimmed-median uses exact knowledge of the malicious rates.

5.3 Reformulation of previous adaptive attacks

This section introduces four state-of-the-art adaptive attacks that can optimize local model poisoning attacks for any given aggregation rules. Although these adaptive attacks are proposed in [17] [18], they do not have an identical formulation. Out of convenience, We reformulate these adaptive attacks and list a table 5.1 to describe the optimization formulations of these attacks. In the following paragraphs, we give a detailed description of the formulations. They can be used for any work focusing on robust aggregation methods to verify the robustness of their methods against adaptive attacks.

Static Optimization (STAT-OPT) Attack [17]: STAT-OPT attacks consider the attacker's objective to deviate global model parameters the most towards the inverse of the direction along which the global parameters would change without attacks. Suppose that in one global training process, G denotes a set of the aggregated global parameters without attacks, and $G^{'}$ denotes the compromised global parameters. $\mathbf{s}^T$ is the column vector of changing directions of all global parameters without attacks. Then, the cost function $\mathbf{s}^T(G-G^{'})$ (see table 5.1) measures the direction deviation. The attacker's goal is to maximize the value of $\mathbf{s}^T(G-G^{'})$. $\mathbf{w}_1,..,\mathbf{w}_n$ denotes a set of the model parameters shared by the clients in one training process and f_{agr} denotes the robust aggregation method, which the attacker aims to compromise. The first m clients $\mathbf{w}_1^{'},..,\mathbf{w}_m^{'}$ are assumed to be compromised. The attacker aims to find an optimal set of values for $\mathbf{w}_1^{'},..,\mathbf{w}_m^{'}$ and substitute for the benign parameters $\mathbf{w}_1,..,\mathbf{w}_m$. After replacing these benign parameters with malicious parameters $\mathbf{w}_1^{'},..,\mathbf{w}_m^{'}$, the deviation between the compromised global parameters and the benign global parameters in the directions can be maximized.

Dynamic Optimization (DYN-OPT) Attack [18]: DYN-OPT attacks aim to decrease the similarity between compromised and benign global models. The cost function is $\|G-G^{'}\|_2$, where $\|\|_2$ is the L_2-norm value. The attacker aims to maximize the $\|G-G^{'}\|_2$ value (see table 5.1). Unlike STAT-OPT Attack, it restricts malicious models as $\mathbf{w}_{i\in[m]}^{'}=G+\gamma\nabla^p$, where γ is the scaling factor and ∇^p is the perturbation vector. [18] introduces three types of perturbation vectors: Inverse unit vector, Inverse standard deviation and Inverse sign. In this work, we consider the Inverse sign. Other perturbation vectors will be discussed in further works.

AGR-agnostic Attacks [18], Min-Max: Previous robust FL algorithms attempt to distinguish malicious parameters from benign ones based on two main criteria: 1) distances between malicious and benign parameters such as cosine similarities [115] [70], 2) difference in L_p-norms of malicious and benign parameters. To bypass these robust FL algorithms, the attacker must ensure that the malicious parameters lie close to the cluster of benign parameters while maximizing the distance or difference in L_p-norm from benign parameters. The cost function is $\|\mathbf{w}^{'} - \mathbf{w}_i\|_2$ (see table 5.1). The attacker aims to maximize the distance of the malicious parameters from benign parameters. The constraint is the distance from benign parameters should be smaller than the maximum of benign parameter distances.

AGR-agnostic Attacks [18], Min-Sum: Min-Max attack maximises the distance of malicious updates from benign model updates while ensuring the maximum distance from other benign updates is upper bounded by the maximum distance between any two benign updates. Like Min-Max, Min-sum ensures that the sum of squared distances of malicious gradients from all the benign updates is upper bound by the sum of the squared distances of any benign updates from the other benign updates. The cost function is $\max_{\gamma} \sum_{i\in[m+1,n]} \|\mathbf{w}^{'} - \mathbf{w}_i\|_2^2$ (see table 5.1). The constraint is $\sum_{i\in[m+1,n]} \|\mathbf{w}^{'} - \mathbf{w}_i\|_2^2 \leq maximum \sum_{i,j\in[m+1,n]} \|\mathbf{w}_i - \mathbf{w}_j\|_2^2$.

5.4 Proposed Defence approaches

According to table 5.1, the defender should reduce the optimization objectives to defeat the adaptive attacks. For example, the L_2-norm distance of the robust aggregated $G^{'}$ should be close to the benign G when the defender attempts to reduce the negative impact of DNY-OPT attacks. As for Min-max or Min-sum attacks, the defender should try to reduce the distance between the malicious and benign parameters ($\max_{\gamma, i\in[m+1,n]} \|\mathbf{w}^{'} - \mathbf{w}_i\|_2$ or $\max_{\gamma} \sum_{i\in[m+1,n]} \|\mathbf{w}^{'} - \mathbf{w}_i\|_2^2$). However, there are two root causes why existing robust methods fail to defend: high dimensional parameters and data heterogeneity.

(1) the High dimensional parameters. The attacker can only choose partial parameters to alter or poison. In [30], the attacker only poisons the unused or infrequently updated parameters by benign clients. Such attack behaviours are more stealthy when the training models contain many parameters. The attacker can cause negative impacts when the value of optimization objectives is not large. Therefore, the attacker can bypass existing methods based on distance or similarity comparisons of full model parameters, such as Krum [46],

Multi-Krum [46], Flame [115], FLtrust [70].

(2) the data heterogeneity. Some previous methods are parameter-wise, like Median [45] and Trimmed-mean [45]. However, these methods may not consider the cause of data heterogeneity. The Median selects a median value to represent the global parameter. On the other hand, Trimmed-mean prunes a list of potential malicious parameters. Hence, the aggregated model may deviate from the optimal training model. Therefore, robust methods should try to take all clients' parameters into consideration.

Based on the above discussion, we propose FedDet, a novel Byzantine-robust FL algorithm, FedDet. This robust method groups the clients' parameters by layers. Then, it normalizes the layer-wise parameters by the median norms. Using this layer-wise robust aggregation method, FedDet can avoid the curse of high dimensions. Unlike [84] splitting the parameters with random fragments, we choose to split the parameters by layers. Our splitting skill can keep the functionality of layers compared to random splitting. Besides, the layer-wise normalization considers all clients' corresponding parameters rather than filtering potential malicious or misclassified honest parameters.

Now, we describe the details of FedDet. Suppose that the local model is a neural network with l layers.

(1) Splitting and Grouping the model parameters $\mathbf{w}_{\{1,...,n\}}$ by layers.

The server collects a list of parameters of ith layer from all clients $\mathbf{w}_{\{1,...,n\}}$, i.e.,

$$G_{i\in[l]} = \{\mathbf{w}_{1,i}, \mathbf{w}_{2,i}, ..., \mathbf{w}_{n,i}\}, \tag{5.1}$$

The server collects all the groups of the split parameters $G_{\{1,...,l\}}$.

(2) Normalizing the ith layer clients' parameters $\mathbf{w}_{\{1,...,n\},i}$ by the median $L2$-norm value.

Firstly, the server collects a list of the $L2$-norm values of every layer L_i from all clients $\mathbf{w}_{\{1,...,n\}}$, i.e.,

$$L_{i\in[l]} = \{\|\mathbf{w}_{1,i}\|_2, \|\mathbf{w}_{2,i}\|_2, ..., \|\mathbf{w}_{n,i}\|_2\}, \tag{5.2}$$

where, $\mathbf{w}_{i,i}$ denotes the i-th layer of $\mathbf{w}_i \in \mathbf{w}_{\{1,...,n\}}$, and $\|\mathbf{w}_{i,i}\|_2$ denotes the $L2$-norm values of the corresponding $\mathbf{w}_{i,i}$. And $L_{i\in[l]}$ is the set containing all the $L2$-norm values. Then, it

sorts out all the clients by their $L2$-norm in ascending order, i.e.,

$$L_{i\in[l]} = \{\|\mathbf{w}_{s1,i}\|_2, \|\mathbf{w}_{s2,i}\|_2, ..., \|\mathbf{w}_{sn,i}\|_2\}. \tag{5.3}$$

Here, $\|\mathbf{w}_{s1}\|_2 \leq \|\mathbf{w}_{s2}\|_2 \leq ... \leq \|\mathbf{w}_{sn}\|_2$. Then, the server selects the median value $med(L_{i\in[l]})$ of $L_{i\in[l]}$. The server collects all the median values corresponding to each layer. The collection of $med(L_{i\in[l]})$ is represented as follows,

$$H^{'} = \{med(L_1), med(L_2), ..., med(L_l)\}. \tag{5.4}$$

Then, The server scales the L_2-norm values of all layer's parameters $L_{\{1,...,l\}}$ of clients by $H^{'}$. The scaled L_2-norm values of the ith layer's updates can be represented as follows,

$$\tilde{L}_{i\in[l]} = \left\{ \frac{med(L_i)}{\|\mathbf{w}_{1,i}\|_2}, \frac{med(L_i)}{\|\mathbf{w}_{2,i}\|_2}, ..., \frac{med(L_i)}{\|\mathbf{w}_{n,i}\|_2} \right\}. \tag{5.5}$$

Then, the weighted updates of the ith layer can be represented as follows,

$$\left\{ \frac{med(L_i)}{\|\mathbf{w}_{1,i}\|_2}\mathbf{w}_{1,i}, \frac{med(L_i)}{\|\mathbf{w}_{2,i}\|_2}\mathbf{w}_{2,i}, ..., \frac{med(L_i)}{\|\mathbf{w}_{n,i}\|_2}\mathbf{w}_{n,i} \right\}. \tag{5.6}$$

We repeat the same process for all l layers. Then, the server executes the FedAvg algorithm on each layer to obtain the new global model.

5.5 Adaptive attacks

In this section, we leverage the adaptive attacks discussed in 5.3 to design adaptive untargeted attacks for the proposed defence method.

5.5.1 STAT-OPT tailored to FedDet

The idea is to instantiate the aggregation rule f_{agr} with our proposed aggregation rule, *FedDet* in the poisoning framework. So we formulate a specific optimization problem using table. 5.1 (STAT-OPT attacks) as follows:

$$\sum_{i=1}^{l} \max_{\mathbf{w}^{'}_{1,i},...,\mathbf{w}^{'}_{m,i}} \mathbf{s}_i^T (G_i - G_i^{'}). \tag{5.7}$$

As the proposed aggregation rule is layer-wise, unlike Krum [46] or Median [45], we

solve the optimization layer by layer and optimize $\mathbf{w}^{'}_{1,i},...,\mathbf{w}^{'}_{m,i}$, where i denotes the ith layer. Then we concatenate all $\mathbf{w}^{'}_{1,i},...,\mathbf{w}^{'}_{m,i}$ by layers and get the final solutions $\mathbf{w}^{'}_{1},...,\mathbf{w}^{'}_{m}$.

In 5.4, the proposed aggregation rule for ith layer can be written as follows:

$$G_i = \frac{1}{n}\left(\frac{med(L_i)}{\|\mathbf{w}_{1,i}\|}\mathbf{w}_{1,i} + \frac{med(L_i)}{\|\mathbf{w}_{2,i}\|}\mathbf{w}_{2,i} + ,...,+\frac{med(L_i)}{\|\mathbf{w}_{n,i}\|}\mathbf{w}_{n,i}\right). \tag{5.8}$$

We denote by $e_{j,i} = \frac{\mathbf{w}_{j,i}}{\|\mathbf{w}_{j,i}\|}$. So Eq. (5.8) can be rewritten as follows:

$$G_i = \frac{1}{n}\sum_{j\in[1,n]} med(L_i)e_{j,i}. \tag{5.9}$$

Let $e^{'}_{j,i}(j \in [1,m])$ denote the poisoned unit vector sent by malicious clients. So, the poisoned aggregated parameters can be rewritten as follows:

$$G^{'}_i = \frac{1}{n}\left(\sum_{j\in[1,m]} med(L^{'}_i)e^{'}_{j,i} + \sum_{j\in[m+1,n]} med(L^{'}_i)e_{j,i}\right), \tag{5.10}$$

where $L^{'}_i$ is $\{\|\mathbf{w}^{'}_{1,i}\|_2,...,\|\mathbf{w}^{'}_{m,i}\|_2,\|\mathbf{w}_{m+1,i}\|_2...,\|\mathbf{w}_{n,i}\|_2\}$ and $med(L^{'}_i)$ is the new median after poisoning.

We substitute Eq. (5.9) and (5.10) into (5.7) and get the following optimization problem:

$$\begin{aligned}&\sum_{i=1}^{l}\ell(\mathbf{e}^{'}_{1,i},...,\mathbf{e}^{'}_{m,i})\\ =&\frac{1}{n}\sum_{i=1}^{l}\max_{\mathbf{e}^{'}_{1,i},...,\mathbf{e}^{'}_{m,i}}\mathbf{s}_i^T(\sum_{j\in[1,n]} med(L_i)e_{j,i} - \sum_{j\in[1,m]} med(L^{'}_i)e^{'}_{j,i} - \sum_{j\in[m+1,n]} med(L^{'}_i)e_{j,i}).\end{aligned} \tag{5.11}$$

We consider a strong attacker who knows $e_{j\in[1,n],i\in[1,l]}$, $e^{'}_{j\in[1,m],i\in[1,l]}$, and f_{agr}. We use a standard gradient ascent approach to solve the optimization problem 5.11. We optimize $\mathbf{e}^{'}_{1,i},...,\mathbf{e}^{'}_{m,i}$ one by one. When optimizing $e^{'}_{j,i}$, all other $e^{'}_{k\neq j,i}$ are fixed. The steps are as follows:

Computing the gradient $\nabla_{e^{'}_i}\ell$ with respect to $e^{'}_i$: As it is hard to compute this gradient directly, we use a standard method, a zeroth-order method [136], to estimate this gradient as follows:

$$\nabla_{e'_{j,i}} \ell \approx \frac{\ell(e'_{j,i} + \gamma \mathbf{u}) - \ell(e'_{j,i})}{\gamma} \cdot \mathbf{u}, \tag{5.12}$$

where ℓ denotes the eq. (5.11). Where $\mathbf{u}$ is a random vector sampled from the multivariate Gaussian distribution $\mathcal{N}(0, \sigma^2 I)$ and $\gamma > 0$ is a smoothing parameter.

Updating $e'_{j,i}$: We multiply the estimated gradient by a learning rate η and add it to $e'_{j,i}$. Then we normalize $e'_{j,i}$ by its L_2 norm value to ensure it is a unit vector.

$$e'_{j,i} = e'_{j,i} + \eta \nabla_{e'_{j,i}} \ell. \tag{5.13}$$

When estimating the gradient $\nabla_{e'_{j,i}} \ell$ and updating $e'_{j,i}$, we fix the value of $med(L'_i)$ for simplicity. The $med(L'_i)$ value will be updated after $e'_{j,i}$ is updated.

Repeating the above two steps for n iterations. The $\mathbf{w}'_{j,i} = med(L'_i) \cdot e'_{j,i}$ after $e'_{j,i}$ is solved.

Repeating the gradient ascent process over all $\mathbf{e}'_{1,i}, ..., \mathbf{e}'_{m,i}$.

The procedure is summarized in Algorithm. (3). We initialize $\mathbf{w}'_{1,i}, ..., \mathbf{w}'_{m,i}$ using Trim attack in [17].

5.5.2 DYN-OPT tailored to FedDet

As discussed in 5.3, DYN-OPT attacks restrict the malicious clients as $\mathbf{w}'_{i\in[m]} = G + \gamma \nabla^p$, where ∇^p is the perturbation vector. We instantiate the aggregation rule f_{agr} in table. 5.1 (DNY-OPT attacks) with our proposed aggregation method. Unlike STAT-OPT attacks, DYN-OPT is a plug-in adaptive attack framework for robust aggregation algorithms. We use Algorithm.1 in [18] to solve the optimal γ value. Algorithm. 1 in [18] sets an initial γ and modifies the value of γ until the change of γ is below a set threshold value.

We assume a strong attacker who knows $\mathbf{w}_{i\in[1,n]}$ and f_{agr}. Hence, the attackers can estimate the perturbation vectors based on $\mathbf{w}_{i\in[1,n]}$. We also consider a weaker attacker who has no knowledge of $\mathbf{w}_{i\in[1,n]}$ and f_{agr}. We compare the efficiency of FedDet in these two different adversary models in the experiment 5.8.3.

Algorithm 3: STAT-OPT tailored to FedDet

Input: $\mathbf{w}_{1,i}^{'},...,\mathbf{w}_{m,i}^{'}$, l, $\mathbf{s}_i$

// $\mathbf{w}_{1,i}^{'},...,\mathbf{w}_{m,i}^{'}$ are a list of initialized malicious updates; l is the number of layers; $\mathbf{s}_i$ is the direction along which the global parameter would change without attacks.

Output: $\mathbf{w}_{1,i}^{'},...,\mathbf{w}_{m,i}^{'}$

for $i \in [0,l]$ // optimization per layer
do
  for $j \in [1,m]$ // optimization $e_{j,i}^{'}$ one by one
  do
    for *iterations* $\in n$ **do**
      Random sample $\mathbf{u}$ $\mathcal{N}(0,\sigma^2 I)$; $\nabla_{e_{j,i}^{'}}\ell \approx \frac{\ell(e_{j,i}^{'}+\gamma\mathbf{u})-\ell(e_{j,i}^{'})}{\gamma}\cdot\mathbf{u}$;
      $e_{j,i}^{'} = e_{j,i}^{'} + \eta\nabla_{e_{j,i}^{'}}\ell$;
    end
    $\mathbf{w}_{j,i}^{'} = med(L_i^{'})\cdot e_{j,i}^{'}$;
  end
end

5.5.3 AGR-agnostic attacks tailored to FedDet

AGR-agnostic attacks do not know the aggregation rules. So, these agnostic attacks can be applied in various robust aggregation algorithms. We use the attack methods in [18] to test the proposed FedDet.

We assume a strong attacker who knows $\mathbf{w}_{i\in[1,n]}$ and f_{agr}. We also consider a weaker attacker who has no knowledge of $\mathbf{w}_{i\in[1,n]}$ and f_{agr}. We consider both adversary models for evaluating FedDet's efficiency.

5.6 Security analysis of FedDet

To conduct the security analysis of FedDet, we fit FedDet into the theoretical framework of [97]. Firstly, we briefly describe the definition of poisoning attacks and the certified radius proposed by [97]. Here are the notation, definitions, and assumptions.

Notation 1 *Let Z be the data domain and D^t be the data sampled (not necessarily i.i.d) from Z at iteration t. Let $\mathscr{L}:\Theta\times Z^*\to\mathscr{R}$ be a loss function and Θ be the class of models with d dimensions. Let $f=(\mathscr{G},\mathscr{A},\lambda(t))$ be the federated learning protocol with update algorithm*

$\mathscr{A} : \mathbf{w}^t \in \mathscr{R}^d \to \mathscr{R}^d$ and $\mathscr{G}(G,D,t) \to \mathbf{w}^t$ that takes a model G and outputs the update $\mathbf{w}^t$. $G^{t+1} = G^t - \lambda(t)\mathscr{A}(\mathbf{w}^t)$ is the updates rule of the FL protocol. For the proposed FedDet, $A(\mathbf{w}^t) = \mathbf{w}^t$.

Definition 2 *(poisoning attacks) Let $f^* = (\mathscr{G}^{'}, \mathscr{A}, \lambda(t))$ be the poisoned federated learning protocol with poisoned $\mathscr{G}^{'}(G,D,t) \to \mathbf{w}^{'t}$. We have $\mathscr{G}^{'}(G,D,t) = \mathscr{G}(G,D,t) + \varepsilon$ with $\|\varepsilon\|_1 \leq \rho$ (or $\|\varepsilon\|_2 \leq \rho$).*

Notation 2 *We use $(G^0,...,G^t)$ and $(G^{'0},...,G^{'t})$ to denote the global model trained through a benign G and a poisoned $G^{'}$ respectively. We use $(\mathbf{w}^1,...,\mathbf{w}^{t+1})$ and $(\mathbf{w}^{'1},...,\mathbf{w}^{'t+1})$ to denote the updates produced by a benign G belongs to global models $(G^0,...,G^t)$ and by a poisoned $G^{'}$ belongs to global models $(G^{'0},...,G^{'t})$. We use $(\mathbf{w}^{*1},...,\mathbf{w}^{*t+1})$ to denote the poisoned updates produced by a poisoned $G^{'}$ belongs to models $(G^{'0},...,G^{'t})$.*

Assumption 1 *A protocol $f(\mathscr{G},\mathscr{A},\lambda(t))$ is a c-layerwise-Lipschitz. Specifically, for any layer index $i \in [L]$*

$$\|\mathscr{G}(G^{'t},D,t)[i] - \mathscr{G}(G^t,D,t)[i]\| \leq c \cdot \|G^{'t} - G^t\|. \tag{5.14}$$

Theorem 3 *[97] Let FedDet be a c-layerwise-Lipschitz protocol on a dataset D. Then $R(\rho) = \Lambda(T)(1+dc)^{\Lambda(T)}\rho$ is a certified radius for f. Namely,*

$$\|G^{'T} - G^T\| \leq \Lambda(T)(1+dc)^{\Lambda(T)}\rho. \tag{5.15}$$

From Equation. (5.15), it is not difficult to see the certified radius $R(\rho)$ relies on ρ when the $\Lambda(T)$, l and c are fixed. Now, we analyze how these adaptive attacks can disturb FedDet with maximum ρ. We attempt to give an upper bound on ρ in the following subsections.

5.6.1 Security analysis for FedDet against STAT-OPT attacks

In subsection 5.5.1, we discuss how STAT-OPT can be adapted to FedDet. Based on Equation. (5.8), we have the benign aggregated parameters per layer:

$$G_i = \frac{1}{n}(\frac{med(L_i)}{\|\mathbf{w}_{1,i}\|}\mathbf{w}_{1,i} + \frac{med(L_i)}{\|\mathbf{w}_{2,i}\|}\mathbf{w}_{2,i} +, ..., + \frac{med(L_i)}{\|\mathbf{w}_{n,i}\|}\mathbf{w}_{n,i}), \tag{5.16}$$

with $L_i = \{\|\mathbf{w}_{1,i}\|, ..., \|\mathbf{w}_{m,i}\|, ..., \|\mathbf{w}_{n,i}\|\}$ and $med(L_i) = l$. The poisoned aggregated parameters per layer are as follows:

$$G_i^{'} = \frac{1}{n}(\frac{med(L_i^{'})}{\|\mathbf{w}_{1,i}^{'}\|}\mathbf{w}_{1,i}^{'} + \frac{med(L_i^{'})}{\|\mathbf{w}_{2,i}^{'}\|}\mathbf{w}_{2,i}^{'} +, ..., + \frac{med(L_i^{'})}{\|\mathbf{w}_{n,i}\|}\mathbf{w}_{n,i}), \tag{5.17}$$

with $L_i = \{\|\mathbf{w}_{1,i}^{'}\|, ..., \|\mathbf{w}_{m,i}^{'}\|, ..., \|\mathbf{w}_{n,i}\|\}$ and $med(L_i^{'}) = l^{'}$.

Assumption 2 *To avoid the impact of attack being restricted,* $\|\mathbf{w}_{1,i}^{'}\|, ..., \|\mathbf{w}_{m,i}^{'}\|$ *should be close to the median value of* $\{\|\mathbf{w}_{1,i}^{'}\|, ..., \|\mathbf{w}_{m,i}^{'}\|, ..., \|\mathbf{w}_{n,i}\|\}$. *Hence, we assume that* $med(L_i^{'}) = \|\mathbf{w}_{1,i}^{'}\| = \|\mathbf{w}_{2,i}^{'}\| =, ..., = \|\mathbf{w}_{m,i}^{'}\| = l^{'}$.

Theorem 4 *Suppose FedDet is a c-layerwise-Lipschitz protocol on dataset D and Assumption 2 holds. Suppose m out of n clients are potentially malicious at one round. The upper bound on perturbation* ρ *caused by STAT-OPT attack on FedDet is given as*

$$\rho \leq \frac{n}{n-cm}\left|\mathbf{w}_{1,i} - G_{i-1}\right| + \frac{cm}{n-cm}\left|G_{i-1}\right| + \frac{c}{n-cm}\left|\sum_{i=m+1}^{n}\frac{\|\mathbf{w}_i\|_{max}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}\right|. \tag{5.18}$$

For the proof of theorem. 4, please see appendix. A.1.

5.6.2 Security analysis for FedDet against DNY-OPT attacks

According to the definition of fedAvg [6], we have the benign aggregated parameters per layer as follows:

$$G_i = \frac{1}{n}\sum_{i=1}^{n}\mathbf{w}_{i,i}. \tag{5.19}$$

Based on Equation. (5.8), we have the poisoned aggregated parameters per layer as follows:

$$G_i^{'} = \frac{1}{n}(\frac{med(L_i^{'})}{\|\mathbf{w}_{1,i}^{'}\|}\mathbf{w}_{1,i}^{'} + \frac{med(L_i^{'})}{\|\mathbf{w}_{2,i}^{'}\|}\mathbf{w}_{2,i}^{'} +, ..., + \frac{med(L_i^{'})}{\|\mathbf{w}_{n,i}\|}\mathbf{w}_{n,i}). \tag{5.20}$$

Theorem 5 *Suppose FedDet is a c-layerwise-Lipschitz protocol on dataset D and Assumption 2 holds. Suppose m out of n clients are potentially malicious at one round. The upper bound on perturbation ρ caused by DNY-OPT attack (DPAs and PGA attacks) on FedDet is given as*

$$\rho \leq \|\mathbf{w}_{1,i} - G_{i-1}\| + \frac{c}{n-cm} \cdot \sum_{i=m+1}^{n} \|(\frac{l'}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\|. \tag{5.21}$$

For the proof of theorem 5, please see appendix. A.2.

5.6.3 Security analysis for FedDet against Agnostic attacks

In section 5.3, we introduce AGR-agnostic attacks. Now, we discuss the possible upper bound on ρ for the Min-max attacks.

Theorem 6 *Suppose m out of n clients are potentially malicious at one round. The upper bound on perturbation ρ caused by Min-max attacks on FedAvg is given as*

$$\rho \leq \|\mathbf{w}_{1,i} - G_{i-1}\| + \max_{i,j \in [m+1,n]} \|\mathbf{w}_i - \mathbf{w}_j\|. \tag{5.22}$$

The security analysis of the Min-sum attacks is similar to that of the Min-max attacks.

Theorem 7 *Suppose m out of n clients are potentially malicious at one round. The upper bound on perturbation ρ caused by Min-sum attacks on FedAvg is given as*

$$\rho \leq \sqrt{\frac{1}{n-m} \cdot (\sum_{i=m+1}^{n} \|\mathbf{w}_{1,i} - G_{i-1}\|^2 + max \sum_{i,j \in [m+1,n]} \|\mathbf{w}_i - \mathbf{w}_j\|^2)}. \tag{5.23}$$

For the proof of theorem. 6 and theorem. 7, please see appendix. A.3.

5.6.4 Security analysis of Krum against DNY-OPT attacks

As a comparison, we also establish the security analysis of Krum [46]. In [17] [18], they design similar adaptive attack strategies to compromise Krum. So, the discussion of the security analysis of Krum does not need to be separated into different situations.

According to the definition of Krum [46], malicious clients' parameters $\mathbf{w}^{'}$ should satisfy that the sum of the squared distances to its closest $n-m$ parameters is the smallest if the malicious parameters $\mathbf{w}^{'}$ could be selected as the next representative global parameters. Namely,

$$\sum_{i \in \Gamma_{\mathbf{w}'}^{n-m-2}} \|\mathbf{w}^{'} - \mathbf{w}_i\| \leq \min_{j \in [m+1,n]} \sum_{i \in \Gamma_{\mathbf{w}}^{n-m-2}} \|\mathbf{w}_j - \mathbf{w}_i\|, \tag{5.24}$$

where $i \in \Gamma_{\mathbf{w}'}^{b}$ denotes a set of parameters that are closest to $\mathbf{w}^{'}$.

Theorem 8 *Suppose m out of n clients are potentially malicious at one round. The upper bound on perturbation ρ caused by adaptive attacks on Krum is given as*

$$\rho \leq \frac{1}{n-2m-1} \min_{j \in [m+1,n]} \sum_{i \in \Gamma_{\mathbf{w}}^{n-m-2}} \|\mathbf{w}_{j,i} - \mathbf{w}_{i,i}\| + \max_{i \in [m+1,n]} \|\mathbf{w}_{i,i} - G_{i-1}\|. \tag{5.25}$$

For the proof of theorem. 8, please see appendix. A.4.

5.6.5 Further analysis

We further analyse the robust aggregation methods' upper bounds on perturbation ρ under various adaptive attacks. We give a table 5.2 that collects all the upper bounds on ρ.

FedDet versus Krum

We notice that the right side of 5 in table 5.2 can be further replaced as below:

$$\leq \|\mathbf{w}_{1,i} - G_{i-1}\| + \frac{cn}{n-cm} \max_{i \in [1,n]} \|(\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\|. \tag{5.26}$$

The coefficient of the right part of (5.26) $\frac{cn}{n-cm}$ is always less than one when $c < 1$. In table. 5.2, the coefficient of the right part of the right side of (8) $\frac{n-m-2}{n-2m-1}$ is greater than one when $m > 1$. Besides, The left part of the right side of (8) $max_{i \in [m+1,n]} \|\mathbf{w}_{i,i} - G_{i-1}\|$ is larger than the left part of (5.26) $\|\mathbf{w}_{1,i} - G_{i-1}\|$, so the upper bound on ρ of Krum is larger than the upper bound of FedDet against DNY-OPT attacks. Therefore, theoretically, Krum is more likely to be compromised than FedDet.

Defense methods	Adaptive attacks	Upper bound on perturbation ρ
FedDet	STAT-OPT attacks	$\rho \leq \frac{n}{n-cm}\left\|\mathbf{w}_{1,i}-G_{i-1}\right\|+\frac{cm}{n-cm}\left\|G_{i-1}\right\|+\frac{c}{n-cm}\left\|\sum_{i=m+1}^{n}\frac{\|\mathbf{w}_i\|_{max}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}-\sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}\right\|$ (4)
	DNY-OPT attacks	$\rho \leq \|\mathbf{w}_{1,i}-G_{i-1}\|+\frac{c}{n-cm}\cdot\sum_{i=1}^{n}\|(\frac{l'}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}-\mathbf{w}_{i,i})\|$ (5)
FedDet	Min-max attacks	$\rho \leq \|\mathbf{w}_{1,i}-G_{i-1}\|+\max_{i,j\in[m+1,n]}\|\mathbf{w}_i-\mathbf{w}_j\|$ (6)
	Min-sum attacks	$\rho \leq \sqrt{\frac{1}{n-m}(\sum_{i=m+1}^{n}\|\mathbf{w}_{1,i}-G_{i-1}\|^2+max\sum_{i,j\in[m+1,n]}\|\mathbf{w}_i-\mathbf{w}_j\|^2)}$ (7)
Krum	DNY-OPT attacks	$\rho \leq max_{i\in[m+1,n]}\|\mathbf{w}_{i,i}-G_{i-1}\|+\frac{1}{n-2m-1}min_{j\in[m+1,n]}\sum_{i\in\Gamma_{\mathbf{w}}^{n-m-2}}\|\mathbf{w}_{j,i}-\mathbf{w}_{i,i}\|$ 9 8)

Table 5.2: Comparison of perturbation ρ for adaptive attacks

Min-max versus Min-sum

Agnostic attacks can be performed in any robust aggregated FL system since these attack strategies do not require knowledge of the aggregation method. In Table.5.2, the right side of (7) can be further replaced as

$$\rho \leq \sqrt{(\max_{i\in[m+1,n]}\|\mathbf{w}_{1,i}-G_{i-1}\|^2+\max_{i,j\in[m+1,n]}\|\mathbf{w}_i-\mathbf{w}_j\|^2)}, \tag{5.27}$$

then we have

$$\rho^2 \leq (\max_{i\in[m+1,n]}\|\mathbf{w}_{1,i}-G_{i-1}\|^2+\max_{i,j\in[m+1,n]}\|\mathbf{w}_i-\mathbf{w}_j\|^2). \tag{5.28}$$

In Table. 5.2, equation. (6) can be further converted to

$$\begin{aligned}\rho^2 &\leq (\|\mathbf{w}_{1,i}-G_{i-1}\|+\max_{i,j\in[m+1,n]}\|\mathbf{w}_i-\mathbf{w}_j\|)^2\\ &\leq \|\mathbf{w}_{1,i}-G_{i-1}\|^2+\max_{i,j\in[m+1,n]}\|\mathbf{w}_i-\mathbf{w}_j\|^2+2\cdot(\max_{i,j\in[m+1,n]}\|\mathbf{w}_i-\mathbf{w}_j\|\cdot\|\mathbf{w}_{1,i}-G_{i-1}\|).\end{aligned} \tag{5.29}$$

Compared to the right side of (5.28) and (5.29), Min-max attacks may incur worse perturbation errors to aggregation methods as the upper bound on perturbation ρ caused by Min-max attacks has an extra item $2\cdot(\max_{i,j\in[m+1,n]}\|\mathbf{w}_i-\mathbf{w}_j\|\cdot\|\mathbf{w}_{1,i}-G_{i-1}\|)$.

DNY-OPT attacks versus Agnostic attacks

In theorem 6, we propose the perturbation error caused by Min-max attacks on FedAvg, which is not robust to malicious attacks. This perturbation can be reduced when robust aggregation methods are applied in FL training. It is not difficult to see that the coefficient of the right part of the right side of (8) is larger than the right side of (6) in the table. 5.2. Therefore, Krum is more vulnerable to DNY-OPT attacks than Min-max attacks.

Analysis for the upper bound of DNY-OPT attacks against FedDet

Now we analyse the real distance $\|G^{'t} - G^t\|$ and estimate the certified radius. The theorem 3 proposed by [97] analyzes the certified radius. We combine the theorem 3 and the theorem 5. Then we get the certified radius of FedDet against DNY-OPT attacks as follows:

$$\|G^{'T} - G^T\| \leq \Lambda(T)(1+dc)^{\Lambda(T)}(\|\mathbf{w}_{1,i} - G_{i-1}\| + \frac{c}{n-cm} \cdot \sum_{i=1}^{n} \|(\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\|). \quad (5.30)$$

We compare the real distance between $G^{'T}$ and G^T and the certified radius at T iteration. To estimate this certified radius, We set $\Lambda(T) = 0.001$, which is the learning rate of the FL training. $(1+dc)^{\Lambda(T)}$ is nearly 1 when $\Lambda(T)$ is a very small number. We get the corresponding values for calculating $\sum_{i=1}^{n} \|(\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\|$ at iteration T. We record the benign G^T and poisoned $G^{'T}$ to calculate the real distance $\|G^{'T} - G^T\|$. We assume a 30% malicious rate at one iteration. This comparison is implemented in the FEMNIST dataset. In figure 5.1, we can see that for $T \in [0, 500]$ epoch, the real distance is always under the estimated certified radius, which means the certified radius 5.30 is a valid upper bound. Besides, the real distance and the certified radius show similar trends as the training epochs.

5.7 Evaluation Setup

In this work, similar to other poisoning or defences-related works, we focus on image classification tasks. We use two natural image recognition datasets, FEMNIST and CIFAR10. Natural image recognition may require security guarantees. For example, in a federated learning-based recommendation system, natural images on social websites can be poisoned with sensitive labels, which can cause discrimination or unfairness.

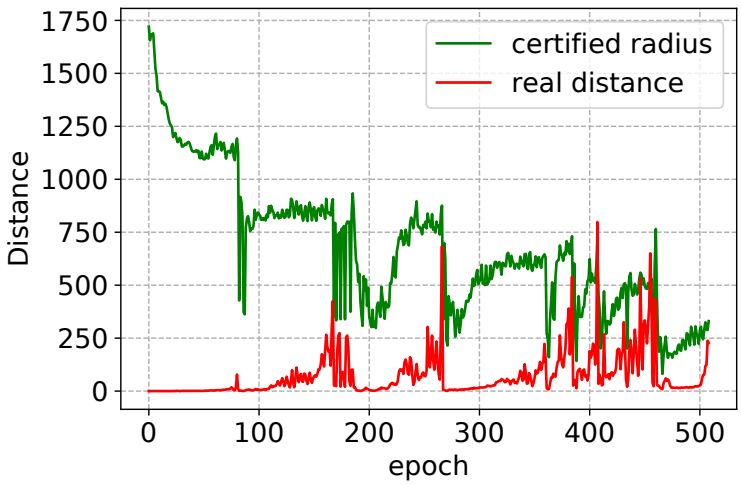


Figure 5.1: Comparison between the real distance $\|G'^{T} - G^{T}\|$ and certified radius

FEMNIST [137] is a 62-class non-IID, class-imbalanced classification task with 3400 clients and 671585 grey-scale images. Each client has their own handwritten digits or letters (52 for upper and lower case letters and ten classes for digits). We select 24 out of 3400 clients for federated training; each client has 1000 samples for local training. We use a four-layer CNN as the local training model.

CIFAR10 [134] is a 10-class class-balanced classification task with 60,000 RGB images, each of size 32 × 32. This class-balanced dataset has the same number of samples per class. Each class of CIFAR10 has 6,000 images. We use 25 clients, each with 1,000 samples, use validation and test data of sizes 5,000. We use Alexnet [138] as the global model architecture.

We use a batch size of 250 and an SGD optimizer with learning rates of 0.001 for FEMNIST. We use a batch size of 250 and an SGD optimizer with learning rates 0.05 for CIFAR10. We repeat the evaluation five times for each attack scenario and use the average as the final result. We conducted five repeated experiments for each attack scenario and took the average value.

5.8 Evaluation Results

In this section, we test the efficiency of FedDet against all six designed adaptive untargeted attacks and compare it with other well-known baseline robust aggregation methods. We use PyTorch to implement all evaluations.

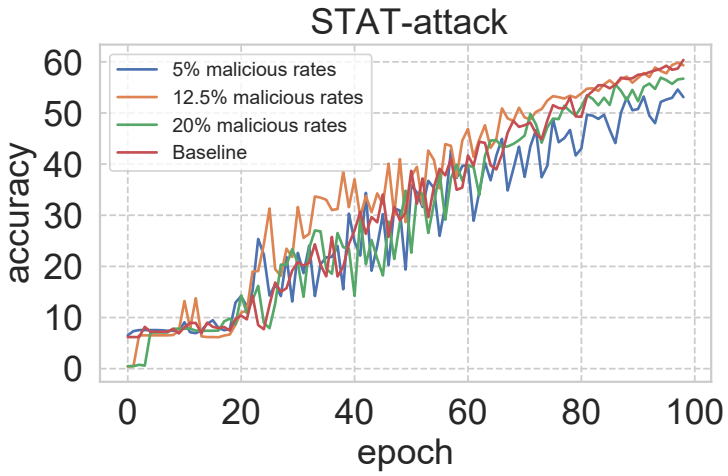


Figure 5.2: Performance of FedDet against STAT-OPT attacks, PDAs and PGA attacks

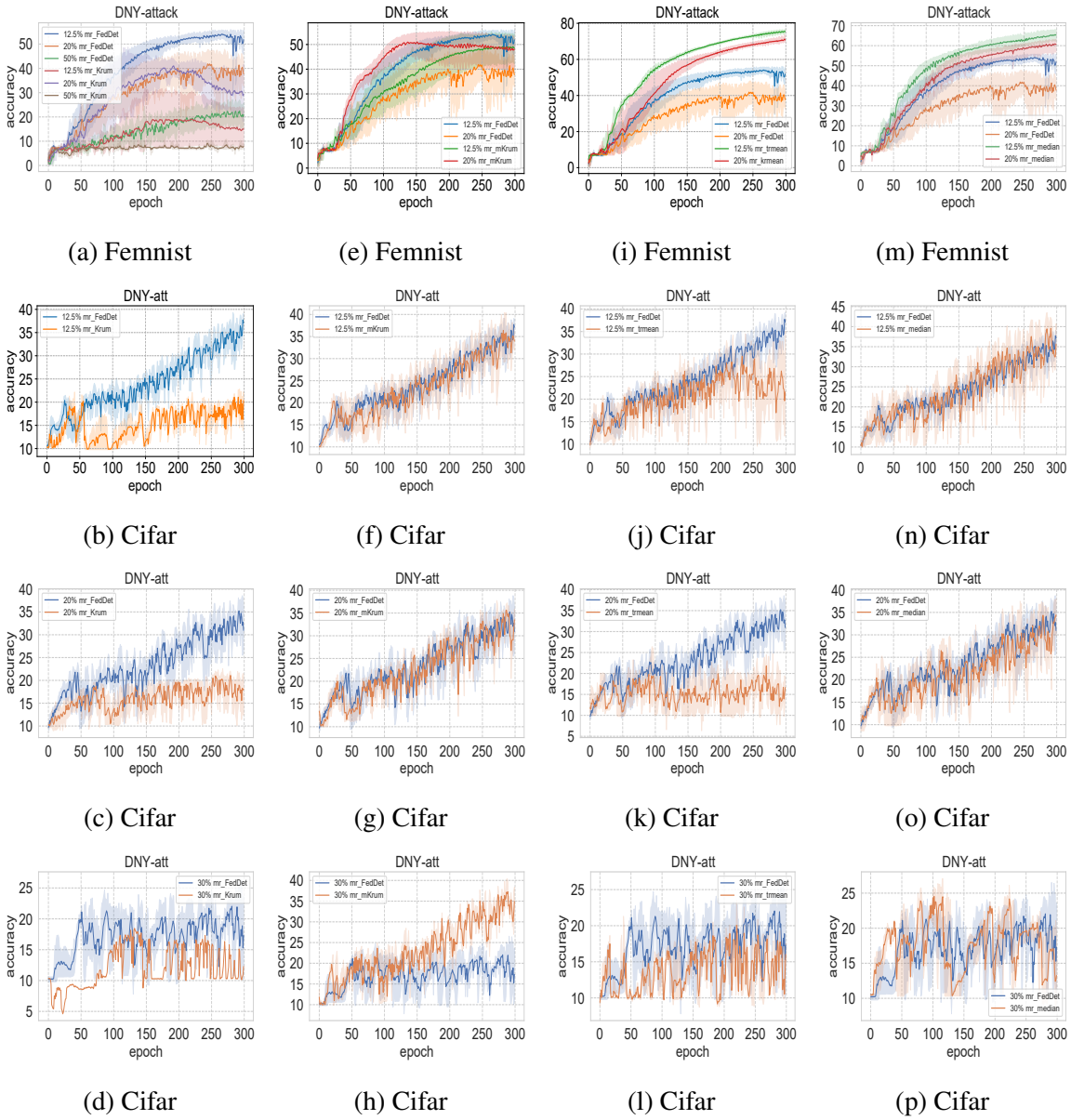
5.8.1 Robustness of FedDet

In figure 5.2, we evaluate the effectiveness of FedDet under different malicious rates. STAT-OPT attacks have minor impacts on the performance of FedDet. Under STAT-OPT attacks, the accuracy of FedDet decreases from 60.43% to 54.00%, 59.60% and 57.11% after 100 global epochs when 5%, 12.5% and 20% malicious clients respectively. As discussed in 5.5.1, this optimization-based model poisoning attack starts from a reference initialized $\mathbf{w}'_{1,i}, ..., \mathbf{w}'_{m,i}$ and keeps updating. It is cumbersome to find the optimal initialization for this attack. Poor initialization might negatively affect the attack performance.

5.8.2 Comparison with previous methods

We compare FedDet with other robust aggregation schemes, Krum [46] , Multi-Krum [46], Trimmed-Mean [45] and Median [45].

In 5.6.2 and 5.6.4, we proposed the upper bounds on perturbation ρ with which the DNY-OPT attacks can disturb the local updates. In 5.6.5, we compare these two upper bounds of FedDet and Krum and draw a conclusion that Krum is more vulnerable to DNY-OPT attacks than FedDet as ρ of Krum is larger. Figure 5.3(a)(b)(c)(d) validate our discussion. In figure 5.3(a), FedDet outperforms Krum under all situations. For example, when the malicious rate is 20%, the main accuracy of FedDet is 41%, but for Krum, the accuracy decreases to 30%. Krum fails the training when half of the clients are malicious. Figure 5.3(b)(c)(d) shows similar results. FedDet is still robust when the malicious clients' rates are 12.5%



(a) Femnist (e) Femnist (i) Femnist (m) Femnist

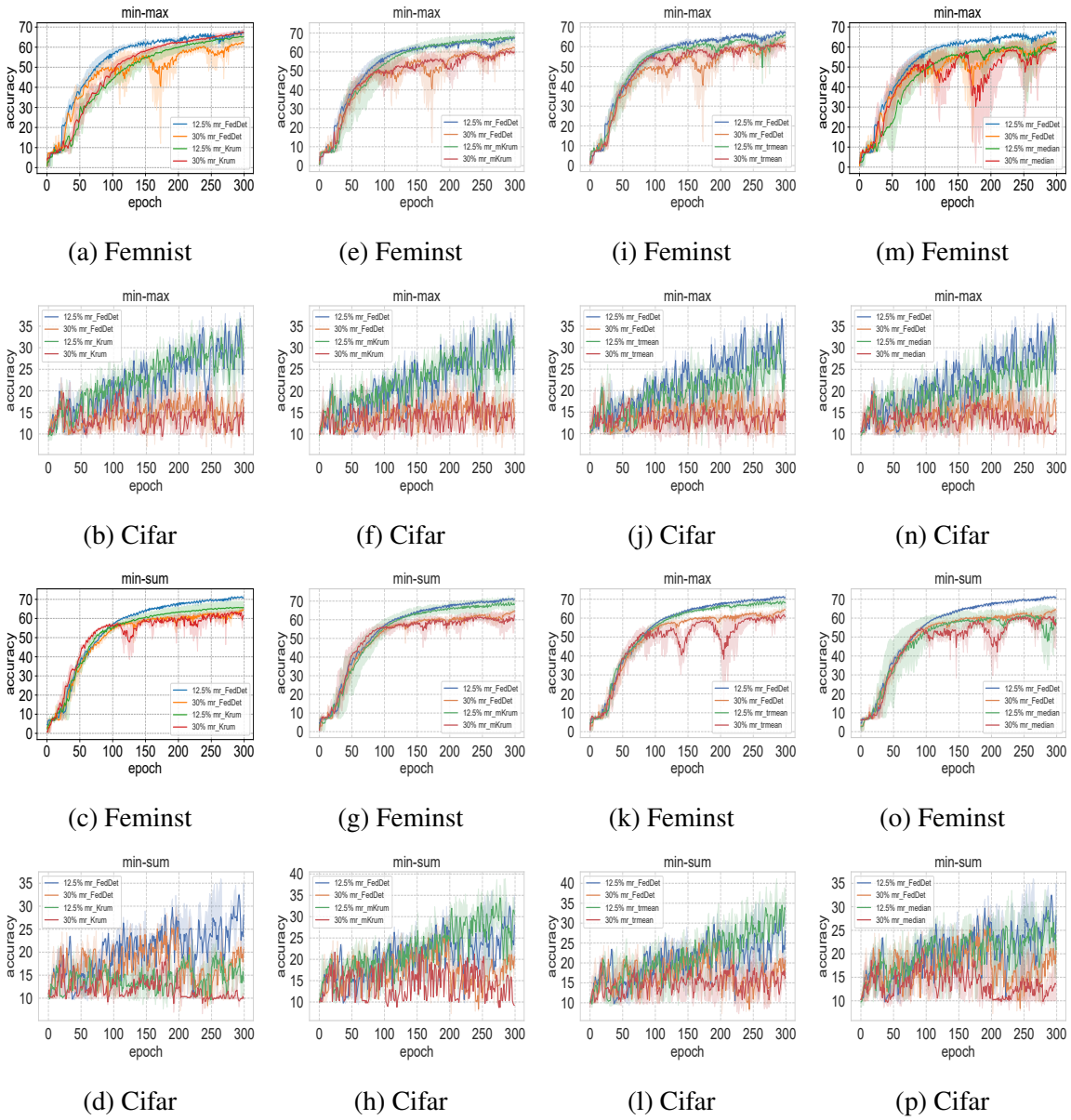
(b) Cifar (f) Cifar (j) Cifar (n) Cifar

(c) Cifar (g) Cifar (k) Cifar (o) Cifar

(d) Cifar (h) Cifar (l) Cifar (p) Cifar

Figure 5.3: Comparison of the robustness of FedDet and other well-known robust aggregation methods against DNY-OPT attacks in two datasets.

and 20%, but Krum has poor accuracy. FedDet also performs better than Krum with 30% malicious rates. The main accuracy of Krum keeps below 20% with all malicious rates. Besides, according to 5.26 in 5.6.5, it is not difficult to see that the upper bound of ρ is larger when m is larger. Namely, larger amounts of malicious clients may cause a worse impact on FedDet. The results of figure 5.3(a)(b)(c)(d) are also in line with this analysis. For example, in figure 5.3(a), when the malicious rate increases from 12.5% to 20%, the main accuracy of FedDet decreases from 52% to 41%. And in figure 5.3(b)(d), the main accuracy decreases from 37% to 15% when the malicious rate increases from 12.5% to 30%. In figure



(a) Femnist (e) Feminst (i) Feminst (m) Feminst

(b) Cifar (f) Cifar (j) Cifar (n) Cifar

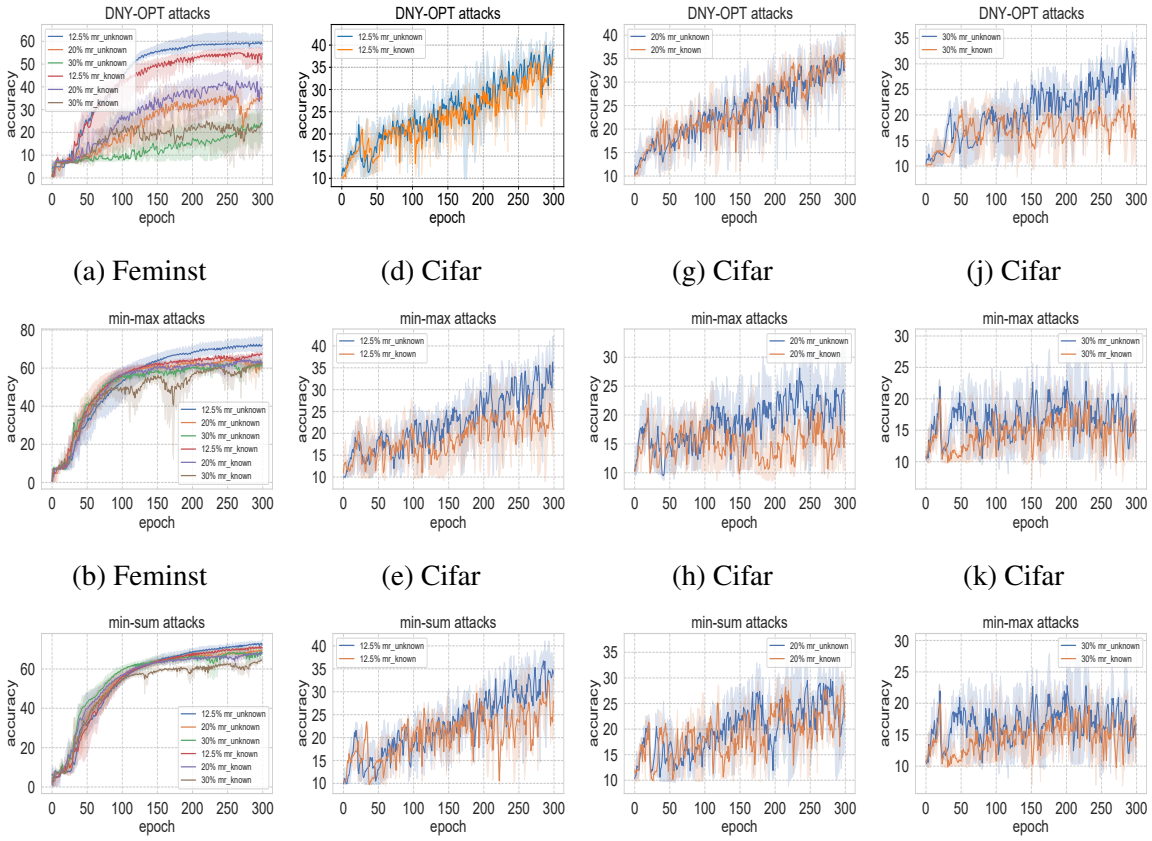
(c) Feminst (g) Feminst (k) Feminst (o) Feminst

(d) Cifar (h) Cifar (l) Cifar (p) Cifar

Figure 5.4: Comparison of the robustness of FedDet and other well-known robust aggregation methods against min-max and min-sum attacks in two datasets.

5.3(e)-(h), we can see that FedDet is competitive with Multi-Krum. In figure 5.3(e), FedDet achieves better performance when the malicious rate is 12.5%, but Multi-krum has higher accuracy with 20% malicious rate. According to figure 5.3(j)(k)(l), FedDet is also more robust than Trimmed-Mean, which uses perfect knowledge of malicious client rates. From figure 5.3(n)(o)(p), FedDet is a little more robust than another agnostic method, Median, which is agnostic to the actual malicious clients' rates.

Remark: According to 5.26 in 5.6.5, it is not difficult to see that the upper bound of ρ is larger when m is larger. Namely, larger amounts of malicious clients may cause a worse im-



(a) Feminst (d) Cifar (g) Cifar (j) Cifar

(b) Feminst (e) Cifar (h) Cifar (k) Cifar

(c) Feminst (f) Cifar (i) Cifar (l) Cifar

Figure 5.5: Performance of FedDet against DNY-OPT attacks, min-max attacks and min-sum attacks when the adversary has partial knowledge of $\mathbf{w}_{i\in[1,n]}$

pact on FedDet. Similar to other compared baseline works (Krum, Multi-krum, Median and Trimmed-Mean), the performance of FedDet is gradually degraded as the portion of adversaries increases. However, Krum only satisfies the resilience property under the assumption $2f+2<n$, where f denotes the number of adversaries and n denotes the number of all clients. When the portion of adversaries is above 50%, Krum fails to work. On the other hand, FedDet can achieve a reasonable performance (see Figure. 5.3(a) in the manuscript). Besides, FedDet has a better performance compared to Median and Trimmed-Mean with 12.5%, 20% and 30% portion of adversaries.

Now, we compare the robustness of FedDet with other aggregation methods against Min-max and Min-sum attack situations. From figure 5.4, FedDet outperforms other robust methods in most situations. According to the test results in the femnist data set in figure 5.4, FedDet achieves the best main accuracy compared to the other four robust aggregation methods when the malicious rate of the agnostic attacks is 12.5%. FedDet performs similarly to

other methods with 30% malicious rate in the femnist dataset. Further, in the aforementioned section 5.6.5, we discuss Krum is more vulnerable to DNY-OPT attacks than Min-max or Min-sum attacks. From figure 5.3(a)(b) and figure 5.4(a)(b)(c)(d), the results validate our points. For example, according to the results of figure 5.3(a) and figure 5.4(a)(c), with the same malicious rate 12.5%, Krum can still achieve a good main accuracy under the agnostic attacks, but Krum fails to defend DNY-OPT attacks. Figure 5.3(b)(c)(d) and figure 5.4(b)(d) show similar results. Krum fails to defend against DNY-OPT attacks in all the situations, but it keeps a 27% accuracy under the Min-max attacks when the malicious rate is 12.5%.

5.8.3 Situations when the adversary has no knowledge of benign updates

In the sections above, we assume the malicious clients have full knowledge of the benign clients, which is a strong adversary model. However, the malicious clients rarely access benign updates from clean clients or the central server. Therefore, the malicious clients should store the historical benign updates locally as alternatives when they attempt to attack. Now, we discuss this weaker adversary model that the benign updates are agnostic to the adversary. In 5.5, we mentioned that in the six attack strategies, the strong attacker knows $\mathbf{w}_{i\in[1,n]}$. But in the limited attack strategies, the attacker only knows $\mathbf{w}_{i\in[m+1,n]}$. Hence, manipulating the Byzantine is based on $\mathbf{w}_{i\in[m+1,n]}$. In figure 5.5, we evaluate the performance of FedDet under the weaker adversary model. Roughly speaking, FedDet is more vulnerable to strong adversary attacks. For example, in figure 5.5(a), the main accuracy of FedDet can achieve 60% with 12.5% malicious rates in the weaker adversary model compared to 52% in the strong adversary model. In figure 5.5(j), FedDet can have a 30% accuracy with 30% malicious rates in the weaker model compared to 15% accuracy in the strong model.

5.8.4 Remarks on Existing Robust Aggregation Algorithms

In this section, we discuss the pros and cons of FedDet compared to other baseline works in terms of different characteristics. Unlike Krum, Multi-krum, and Trimmed-Mean, FedDet and Median do not require the exact knowledge of the corruption level of FL systems, which are more realistic in the real world as the defender has no access to know the attacker's actions. The expected time complexity of Krum is $O(n^2 \cdot d)$, where n is the number of

	Agnostic to the actual corruption level	Expected Time Complexity
Krum [46]	No	$O(n^2 \cdot d)$
Multi-krum [46]	No	$O(mn^2 \cdot d)$
Trimmed-Mean [45]	No	$O(n \cdot d)$
Median [45]	Yes	$O(n \cdot d)$
FedDet	Yes	$O(n \cdot \frac{n}{l})$

Table 5.3: Comparison of Existing Robust Aggregation Algorithms

selected clients in one training iteration and d is the dimension of the parameter vectors. The parameter server computes the squared distance between a client's vector with the resting parameters' vectors ($O(n \cdot d)$. Then, the parameter server repeated this process for all selected clients ($O(n)$). Thus, the square distance computing time is $O(n^2 \cdot d)$. After computing, the server selects the first $n - f - 1$ of the distances for the clients ($O(n)$ with Quickselect) and repeats the process for all clients ($O(n^2)$). Therefore, the expected time complexity for Krum is $O(n^2 \cdot d)$. For Multi-krum, a variant of Krum, selects the $m \in \{1, ..., n\}$ vectors with the smallest sum of distances. m varies between 1 and n. Thus, the expected time complexity of Multi-krum is $O(mn^2 \cdot d)$. Trimmed-Mean sorts (Quicksort) the values of all clients' vectors in dimension ($O(n \cdot d)$). Similarly, Median selects the median value of all client vectors in dimension with Quickselect ($O(n \cdot d)$). For FedDet, it computes the $L2$-norm values of split client vectors ($O(n \cdot \frac{n}{l})$). Then, FedDet sorts the norm values (Quicksort) and selects the median value ($O(n)$). Thus the expected time complexity of FedDet is $O(n \cdot \frac{n}{l})$. From the above analysis, we can see that compared to Krum and Multi-krum, Trimmed-Mean, Median and FedDet have less expected time complexity. The summary of the remarks is shown in the table. 5.3.

5.9 More Adaptive Attacks

This section introduces more untargeted model poisoning frameworks.

5.9.1 Reformulation of DPAs and PGA

Data Poisoning Attacks (DPAs) [133]: DPAs are the first to formulate an optimization problem to construct systematic data poisoning attacks on federated learning. The objective function of DPAs is the same as DNY-OPT's. The attacker aims to find an appropriate amount of label-flipped data as poisoned data set D^P to maximize the value of $\|G - G^{'}\|_2$.

DPAs can be formulated as follows:

$$\begin{aligned} & \max_{D^P \in D} \|G - G^{'}\|_2, \\ s.t. \quad & G = f_{avg}(\mathbf{w}_1, .., \mathbf{w}_m, \mathbf{w}_{m+1}, .., \mathbf{w}_n), \\ & G^{'} = f_{agr}(\mathbf{w}^{'}_1, .., \mathbf{w}^{'}_m, \mathbf{w}_{m+1}, .., \mathbf{w}_n), \\ & \mathbf{w}^{'}_{i\in[m]} = A(D^P, G^{rec}), \end{aligned} \tag{5.31}$$

Where $A(D^P, G^{rec})$ is the poisoning training algorithm and G^{rec} is the received initial global parameters. By using this poisoning training algorithm, the attacker can obtain the poisoned parameters $\mathbf{w}^{'}_{i\in[m]}$. By doing so, data poisoning attacks are transformed into model poisoning attacks in FL. The intuition behind DPAs is that the cross-entropy loss of poisoned models $\mathbf{w}^{'}_{i\in[m]}$ on the main task would be increased after being trained on poisoned dataset D^P. Therefore, the poisoned global model $G^{'}$ would greatly differ from benign aggregated updates G.

Projected Gradient Ascent (PGA) Attack [133]: Similar to DPAs, the intuition behind PGA Attacks is to increase the loss of poisoned model $\mathbf{w}^{'}_{i\in[m]}$ on the main task. Unlike DPAs injecting poisoned dataset D^P for poison training, PGA Attacks use the stochastic gradient ascent(SGA) to increase the loss of $\mathbf{w}^{'}_{i\in[m]}$ on the main task. We have $\alpha = \frac{1}{n-m}\sum_{i\in[m+1,n]}\|\mathbf{w}_i\|$, which is the average of norms of benign parameters. Then we scale $\mathbf{w}^{'}_{i\in[m]}$ by α. By fine-tuning $\mathbf{w}^{'}_{i\in[m]}$ and a scaling factor γ, we attempt to maximize the value of objective function $\|G - G^{'}\|_2$. The formulation is described in Eq.5.32. A simplified version of this PGA attack is using SGA to fine-tune $\mathbf{w}^{'}_{i\in[m]}$ without projecting and re-scaling the norm of $\mathbf{w}^{'}_{i\in[m]}$. These simplified PGA attacks do not need to know benign parameters.

$$\begin{aligned} & \max_{\mathbf{w}^{'}_{i\in[m]}, \gamma} \|G - G^{'}\|_2, \\ s.t. \quad & G = f_{avg}(\mathbf{w}_1, .., \mathbf{w}_m, \mathbf{w}_{m+1}, .., \mathbf{w}_n), \\ & G^{'} = f_{agr}(\gamma(\mathbf{w}^{'}_1, .., \mathbf{w}^{'}_m), \mathbf{w}_{m+1}, .., \mathbf{w}_n), \\ & \mathbf{w}^{'}_{i\in[m]} = \frac{\alpha \mathbf{w}^{'}_{i\in[m]}}{\|\mathbf{w}^{'}_{i\in[m]}\|_2}, \\ & \alpha = \frac{1}{n-m}\sum_{i\in[m+1,n]}\|\mathbf{w}_i\|. \end{aligned} \tag{5.32}$$

5.9.2 Adaptive attacks

In this section, we leverage the state-of-the-art frameworks discussed in 5.3 to design adaptive untargeted attacks for the proposed defence method.

DPAs tailored to FedDet

For *FedDet* AGR as f_{agr}, we use the objective in 5.31, but it is hard to solve it directly as the search space of D^P is too large, and the constraint of the optimization problem is nonlinear. To address this challenge, we make approximations to get the sub-optimal solution to the optimization problem. Like DPAs tailored to Trimmed-mean AGR in [133], we use a classic label flipping poisoning strategy [17] to get the crafted poisoned dataset D^P. We use *PDR* (Poisoned data rate) to estimate the poisoned data rate on malicious local clients. The higher the poisoning rate, the more poisoned data.

Full knowledge We assume that the attacker knows $\mathbf{w}_{i\in[1,n]}$ and f_{agr}. Therefore, attackers can observe and estimate the optimal PDR to maximise the objective value in every round t. In this work, we design new adaptive DPAs against FedDet. In the adaptive DPAs, malicious clients can obtain the optimal PDR in every round t without pre-observations.

The procedure is shown in Algorithm.4.

PGA tailored to FedDet

Unlike the constraint of DPAs is nonlinear, PGA attacks restrict $\mathbf{w}^{'}_{i\in[m]}$ as follows: $\mathbf{w}^{'}_{t} = \mathbf{w}^{'}_{t-1} + \eta(-\nabla_{\mathbf{w}^{'}_{t-1}}\ell(\mathbf{w}^{'}_{t-1};b))$. In this work, we only discuss the performance of FedDet against simplified PGA attacks as described in 5.3.

5.9.3 Evaluation of FedDet Algorithm

Experiment Settings

FEMNIST [137] is a 62-class non-iid, class-imbalanced classification. Each client has their handwritten digits or letters (52 for upper and lower case letters and ten classes for digits). We select 24 out of 3400 clients for federated training; each client has 1000 samples for local training. We use a four-layer CNN as the local training model.

We use a batch size of 250 and an SGD optimizer with learning rates of 0.001 for FEMNIST. We conducted five repeated experiments for each attack scenario and took the average

Algorithm 4: DPAs tailored to FedDet

Input: $\mathbf{w}_1,...,\mathbf{w}_m$, $\mathbf{w}_{m+1},...,\mathbf{w}_n$, pdr, ℓ, $\ell_{pre}=0$, $pdr_{final}=0.1$

// pdr is Poisoned Data Rate; ℓ is the objective value; ℓ_{pre} is the initial objective value; pdr_{final} is the final optimal poisoned data rate.

Output: $\mathbf{w}_1^{'},...,\mathbf{w}_m^{'},\mathbf{w}_{m+1},...,\mathbf{w}_n$

while $pdr \leq 1$ **do**
  for $i \in [1,m]$ **do**
    benign_inputs = inputs*(1-pdr); malicious_inputs = inputs*pdr ;
    outputs = fed_model(inputs) ;
    loss = criterion(outputs, targets) ;
    loss.backward(); $\mathbf{w}_i^{'} = fed\_model.parameters()$;
  end
  $\ell = \|f_{avg}(\mathbf{w}_{m+1},..,\mathbf{w}_n) - f_{agr}(\mathbf{w}_1^{'},..,\mathbf{w}_m^{'},\mathbf{w}_{m+1},..,\mathbf{w}_n)\|$;
  if $\ell_{pre} \leq \ell$ **then**
    $\ell_{pre} = \ell$;
    $pdr_{final} = pdr$;
  end
end

value.

In our extensive version, we also test the performance of FedDet on the class-balanced Cifar dataset. The experiment does not set additional limits on the data. The relationship between the efficiency of defence methods and datasets will be one of our future research directions.

Evaluation Results

In this section, we test the main accuracy of FedDet against all six designed adaptive untargeted attacks.

Figure. 5.6(a)(b) shows that DPAs attacks are more powerful than PGA attacks. DPAs attacks decrease the accuracy from 79.37% to 72.362% after 300 global epochs when 30% malicious clients. However, under the same circumstance, PGA attacks can only reduce the accuracy to 75.95%.

5.10 Conclusion

Federated learning holds great potential in providing privacy for large amounts of distributed end devices. However, it is vulnerable to adaptive poisoning attacks. Existing defence meth-

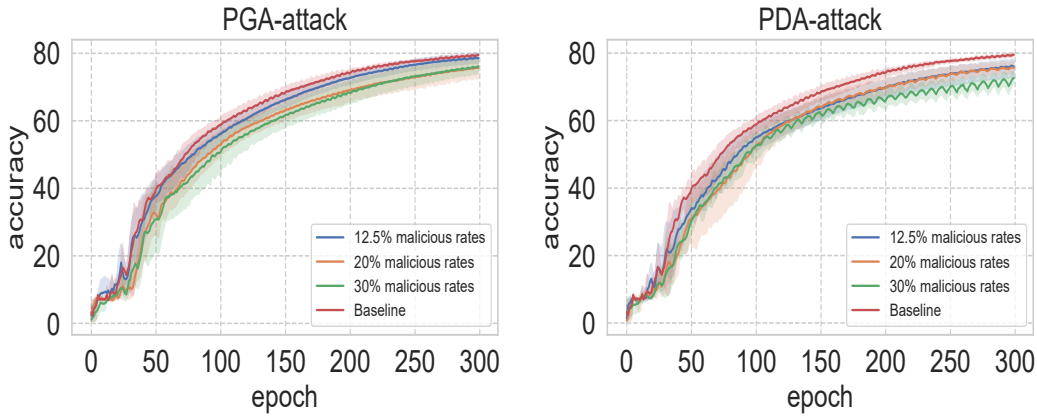


Figure 5.6: Main task Accuracy of FedDet against six designed adaptive untargeted model poisoning attacks with different numbers of malicious clients.

ods did not consider the causes of model parameters' high dimensionality and data heterogeneity. In this work, we proposed a novel Byzantine-robust federated learning, FedDet to solve the problem. FedDet can overcome this issue with high dimensionality and keep the functionality of layers. During the robust aggregation, FedDet normalizes every slice of local models by the median norm value rather than excluding some clients, which can avoid deviation from the optimal aggregated model. In addition, we presented a theoretical security analysis model and conducted an extensive security analysis of FedDet and the state-of-the-art robust aggregation method, Krum. We discussed why the proposed method outperforms the prior method. However, in this work, we do not discuss how FedDet defends targeted model poisoning attacks that can insert backdoors while keeping the trained model's accuracy. Future research will focus on combining the advantages of the proposed method and other defence approaches to defeat untargeted model poisoning and targeted model poisoning attacks.

Chapter 6

Conclusion

In this thesis, first, we proposed a novel approach FLSec for federal learning that can resist backdoor attacks. It analyzes the difference between benign and malicious clients' contributions to the global model and uses new measurements on local model updates to identify malicious updates. Then, we proposed a novel defence system, which is called DeMAC to defend against malicious attacks by measuring the difference in the contribution of benign clients and malicious clients to the global model. We defined a new metric GradScore, to compute the L2-norm of the gradients of the last layer of contributed model updates, which is shown to be effective in detecting updates from malicious clients. Furthermore, we utilized the history records of the contributed model updates to enhance the malicious client detection performance. We evaluated and compared DeMAC with state-of-the-art defence techniques over various attack strategies and datasets. Afterwards, we discuss the vulnerability of FL to adaptive poisoning attacks. We analyze that existing defence methods did not consider the causes of model parameters' high dimensionality and data heterogeneity. So, we proposed a novel Byzantine-robust federated learning, FedDet to solve the problem. FedDet can overcome this issue with high dimensionality and keep the functionality of layers. During the robust aggregation, FedDet normalizes every slice of local models by the median norm value rather than excluding some clients, which can avoid deviation from the optimal aggregated model. In addition, we presented a theoretical security analysis model and conducted an extensive security analysis of FedDet and the state-of-the-art robust aggregation method, Krum. We discussed why the proposed method outperforms the prior method.

6.1 Future work

6.1.1 Defending transferable poisoning attacks

[35] bridges this gap by conducting centralized backdoor attacks and distributed backdoor attacks in federated graph neural networks (GNNs). According to their evaluations, both backdoor attacks can lead to threats to the robustness of federated GNNs. [139] discuss that large language models (LLM) and diffusion models can generate untrustworthy participants and compromise the models deployed by benign clients. Therefore, more robust defence methods should be proposed to address the safety issue introduced by these generative models. Defence methods based on model reconstruction can be used. Model-reconstruction methods aim to modify the infected model directly by model retraining, or neuron-pruning.

6.1.2 Trade-off between robustness and privacy

When designing the defence against model poisoning attacks, with more information about the sharing models, the defender can easily identify the malicious parameters among them. However, the requirement of sharing model parameters leaves the door open for model inference attacks. Thus, it seems hard to keep a balance between robustness and privacy. Sometimes, we need to make a trade-off between robustness and privacy when facing different needs.

6.1.3 Defending Backdoor attacks against Neural Network

In the past few years, deep neural networks (DNNs) have been widely used in various application scenarios, such as face recognition, autonomous driving, and so on. However, DNNs face some security threats, such as backdoor attacks. Similar to backdoor attacks in federated learning, the attackers attempt to insert hidden backdoors in deep neural network models during the training process. The backdoored DNNs behave normally on the main task, whereas they output wrong predictions if hidden backdoors are activated by attacker-specified trigger patterns. To eliminate the backdoor or the trigger in poisoned samples, methods based on trigger elimination or backdoor elimination can be designed. For example, before feeding the samples to DNNs, a pre-processing module can be used to modify the potential contained trigger. Or synthesizing the trigger and then suppressing the trigger effects.

6.1.4 Trustworthy AI for Robots and Autonomous Systems

Robots and autonomous systems (RASs), equipped with artificial intelligence (AI) technologies are attempting to enhance the abilities to perceive complex environments and make decisions quickly by stimulating cognitive and developmental abilities toward human intelligence. However, RASs face some security issues, such as high-visibility data breaches, and cyberattacks. AI techniques have played an important role in data-driven cybersecurity as they can process a large volume of data and provide automation functions. However, AI techniques can be hacked by adversarial behaviours. Hence it is crucial to protect AI models and provide trustworthy AI services for RASs. The classic adversarial defence techniques include adversarial training and adversarial perturbations in the input/feature domain.

Bibliography

[1] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[2] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[3] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE internet of things journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[4] M. Mohammadi and A. Al-Fuqaha, "Enabling cognitive smart cities using big data and machine learning: Approaches and challenges," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 94–101, 2018.

[5] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[7] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 365–11 375, 2021.

[8] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[9] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," *arXiv preprint arXiv:1712.05526*, 2017.

[10] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proceedings of the 34th annual computer security applications conference*, 2018, pp. 381–392.

[11] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *International Conference on Machine Learning*. PMLR, 2019, pp. 634–643.

[12] S. Islam, S. Badsha, I. Khalil, M. Atiquzzaman, and C. Konstantinou, "A triggerless backdoor attack and defense mechanism for intelligent task offloading in multi-uav systems," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5719–5732, 2022.

[13] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[14] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "Dba: Distributed backdoor attacks against federated learning," in *International Conference on Learning Representations*, 2019.

[15] H. Wang, K. Sreenivasan, S. Rajput, H. Vishwakarma, S. Agarwal, J.-y. Sohn, K. Lee, and D. Papailiopoulos, "Attack of the tails: Yes, you really can backdoor federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16 070–16 084, 2020.

[16] Z. Allen-Zhu, F. Ebrahimian, J. Li, and D. Alistarh, "Byzantine-resilient non-convex stochastic gradient descent," *arXiv preprint arXiv:2012.14368*, 2020.

[17] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to {Byzantine-Robust} federated learning," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1605–1622.

[18] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *NDSS*, 2021.

[19] G. Baruch, M. Baruch, and Y. Goldberg, "A little is enough: Circumventing defenses for distributed learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] R. Guerraoui, S. Rouault *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530.

[21] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP)*. IEEE, 2017, pp. 3–18.

[22] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 739–753.

[23] E. M. El Mhamdi, "Robust distributed learning," EPFL, Tech. Rep., 2020.

[24] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.

[25] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 2020, pp. 480–501.

[26] T. D. Nguyen, P. Rieger, M. Miettinen, and A.-R. Sadeghi, "Poisoning attacks on federated learning-based iot intrusion detection system," in *NDSS Workshop on Decentralized IoT Systems and Security*, 2020.

[27] M. Fang, M. Sun, Q. Li, N. Z. Gong, J. Tian, and J. Liu, "Data poisoning attacks and defenses to crowdsourcing systems," in *Proceedings of the web conference 2021*, 2021, pp. 969–980.

[28] X. Cao and N. Z. Gong, "Mpaf: Model poisoning attacks to federated learning based on fake clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3396–3404.

[29] K. Yoo and N. Kwak, "Backdoor attacks in federated learning by rare embeddings and gradient ensembling," *arXiv preprint arXiv:2204.14017*, 2022.

[30] Z. Zhang, A. Panda, L. Song, Y. Yang, M. Mahoney, P. Mittal, R. Kannan, and J. Gonzalez, "Neurotoxin: Durable backdoors in federated learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 26 429–26 446.

[31] Y. Dai and S. Li, "Chameleon: Adapting to peer images for planting durable backdoors in federated learning," *arXiv preprint arXiv:2304.12961*, 2023.

[32] J. Cao and l. Zhu, "A highly efficient, confidential, and continuous federated learning backdoor attack strategy," in *2022 14th International Conference on Machine Learning and Computing (ICMLC)*, 2022, pp. 18–27.

[33] Y. Sun, H. Ochiai, and J. Sakuma, "Semi-targeted model poisoning attack on federated learning via backward error analysis," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 1–8.

[34] S. Datta, G. Lovisotto, I. Martinovic, and N. Shadbolt, "Widen the backdoor to let more attackers in," *arXiv preprint arXiv:2110.04571*, 2021.

[35] J. Xu, R. Wang, S. Koffas, K. Liang, and S. Picek, "More is better (mostly): On the backdoor attacks in federated graph neural networks," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 684–698.

[36] M. Alam, E. Sarkar, and M. Maniatakos, "Perdoor: Persistent non-uniform backdoors in federated learning using adversarial perturbations," *arXiv preprint arXiv:2205.13523*, 2022.

[37] P. Lai, N. Phan, A. Khreishah, I. Khalil, and X. Wu, "Model transferring attacks to backdoor hypernetwork in personalized federated learning," *arXiv preprint arXiv:2201.07063*, 2022.

[38] Y. Fraboni, R. Vidal, and M. Lorenzi, "Free-rider attacks on model aggregation in federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1846–1854.

[39] Y. Liu, Z. Yi, and T. Chen, "Backdoor attacks and defenses in feature-partitioned collaborative learning," *arXiv preprint arXiv:2007.03608*, 2020.

[40] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 603–618.

[41] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "{Updates-Leak}: Data set inference and reconstruction attacks in online learning," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1291–1308.

[42] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 691–706.

[43] E. Bagdasaryan and V. Shmatikov, "Blind backdoors in deep learning models," in *Usenix Security*, 2021.

[44] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[45] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[46] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[47] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 508–519.

[48] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[49] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *ECAI 2012*. IOS Press, 2012, pp. 870–875.

[50] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, "Support vector machines under adversarial label contamination," *Neurocomputing*, vol. 160, pp. 53–62, 2015.

[51] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[52] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[53] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *arXiv preprint arXiv:1912.13445*, 2019.

[54] M. S. Ozdayi, M. Kantarcioglu, and Y. R. Gel, "Defending against backdoors in federated learning with robust learning rate," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9268–9276.

[55] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1544–1551.

[56] Y. Li, A. S. Sani, D. Yuan, and W. Bao, "Enhancing federated learning robustness through clustering non-iid features," in *Proceedings of the Asian Conference on Computer Vision*, 2022, pp. 41–55.

[57] M. Hao, H. Li, G. Xu, H. Chen, and T. Zhang, "Efficient, private and robust federated learning," in *Annual Computer Security Applications Conference*, 2021, pp. 45–60.

[58] M. Fang, J. Liu, N. Z. Gong, and E. S. Bentley, "Aflguard: Byzantine-robust asynchronous federated learning," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 632–646.

[59] L. Muñoz-González, K. T. Co, and E. C. Lupu, "Byzantine-robust federated machine learning through adaptive model averaging," *arXiv preprint arXiv:1909.05125*, 2019.

[60] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[61] L. Chen, H. Wang, Z. Charles, and D. Papailiopoulos, "Draco: Byzantine-resilient distributed training via redundant gradients," in *International Conference on Machine Learning*. PMLR, 2018, pp. 903–912.

[62] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5201–5212.

[63] S. Farhadkhani, R. Guerraoui, N. Gupta, L.-N. Hoang, R. Pinot, and J. Stephan, "Robust collaborative learning with linear gradient overhead," 2023.

[64] C. P. Wan and Q. Chen, "Robust federated learning with attack-adaptive aggregation," *arXiv preprint arXiv:2102.05257*, 2021.

[65] W. Wan, S. Hu, J. Lu, L. Y. Zhang, H. Jin, and Y. He, "Shielding federated learning: Robust aggregation with adaptive client selection," *arXiv preprint arXiv:2204.13256*, 2022.

[66] S. Guo, T. Zhang, H. Yu, X. Xie, L. Ma, T. Xiang, and Y. Liu, "Byzantine-resilient decentralized stochastic gradient descent," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 6, pp. 4096–4106, 2021.

[67] N. M. Jebreel, J. Domingo-Ferrer, A. Blanco-Justicia, and D. Sánchez, "Enhanced security and privacy via fragmented federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[68] C. Xu, Y. Jia, L. Zhu, C. Zhang, G. Jin, and K. Sharif, "Tdfl: Truth discovery based byzantine robust federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4835–4848, 2022.

[69] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020.

[70] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[71] C. Wu, X. Yang, S. Zhu, and P. Mitra, "Mitigating backdoor attacks in federated learning," *arXiv preprint arXiv:2011.01767*, 2020.

[72] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection," *arXiv preprint arXiv:2201.00763*, 2022.

[73] G. F. Cretu, A. Stavrou, M. E. Locasto, S. J. Stolfo, and A. D. Keromytis, "Casting out demons: Sanitizing training data for anomaly sensors," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 81–95.

[74] Y. Mi, J. Guan, and S. Zhou, "Ariba: Towards accurate and robust identification of backdoor attacks in federated learning," *arXiv preprint arXiv:2202.04311*, 2022.

[75] N. Wang, Y. Xiao, Y. Chen, Y. Hu, W. Lou, and Y. T. Hou, "Flare: defending federated learning against model poisoning attacks via latent space representations," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 946–958.

[76] S. Shi, C. Hu, D. Wang, Y. Zhu, and Z. Han, "Federated anomaly analytics for local model poisoning attack," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 2, pp. 596–610, 2021.

[77] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 2545–2555.

[78] T. D. Nguyen, P. Rieger, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, A.-R. Sadeghi, T. Schneider *et al.*, "Flguard: secure and private federated learning," *arXiv preprint arXiv:2101.02281*, 2021.

[79] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" *arXiv preprint arXiv:1911.07963*, 2019.

[80] J. Gao, B. Zhang, X. Guo, T. Baker, M. Li, and Z. Liu, "Secure partial aggregation: Making federated learning more robust for industry 4.0 applications," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6340–6348, 2022.

[81] L. Shen, Y. Zhang, J. Wang, and G. Bai, "Better together: Attaining the triad of byzantine-robust federated learning via local update amplification," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 201–213.

[82] N. Gupta, S. Liu, and N. H. Vaidya, "Byzantine fault-tolerant distributed machine learning using stochastic gradient descent (sgd) and norm-based comparative gradient elimination (cge)," *arXiv preprint arXiv:2008.04699*, 2020.

[83] N. Gupta, S. Liu, and N. Vaidya, "Byzantine fault-tolerant distributed machine learning with norm-based comparative gradient elimination," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2021, pp. 175–181.

[84] Y. Liu, C. Chen, L. Lyu, F. Wu, S. Wu, and G. Chen, "Byzantine-robust learning on heterogeneous data via gradient splitting," 2023.

[85] C. Zhu, S. Roos, and L. Y. Chen, "Leadfl: Client self-defense against model poisoning in federated learning," 2023.

[86] 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," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 613–12 624, 2021.

[87] M. Naseri, J. Hayes, and E. De Cristofaro, "Local and central differential privacy for robustness and privacy in federated learning," *arXiv preprint arXiv:2009.03561*, 2020.

[88] A. Mondal, H. Virk, and D. Gupta, "Beas: Blockchain enabled asynchronous & secure federated machine learning," *arXiv preprint arXiv:2202.02817*, 2022.

[89] S. Andreina, G. A. Marson, H. Möllering, and G. Karame, "Baffle: Backdoor detection via feedback-based federated learning," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 852–863.

[90] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[91] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[92] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni *et al.*, "Flame: Taming backdoors in federated learning," *Cryptology ePrint Archive*, 2021.

[93] Y. Tian, R. Wang, Y. Qiao, E. Panaousis, and K. Liang, "Flvoogd: Robust and privacy preserving federated learning," *arXiv preprint arXiv:2207.00428*, 2022.

[94] X. Cao, J. Jia, Z. Zhang, and N. Z. Gong, "Fedrecover: Recovering from poisoning attacks in federated learning using historical information," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1366–1383.

[95] K. Kumari, P. Rieger, H. Fereidooni, M. Jadliwala, and A.-R. Sadeghi, "Baybfed: Bayesian backdoor defense for federated learning," *arXiv preprint arXiv:2301.09508*, 2023.

[96] J. Park, D.-J. Han, M. Choi, and J. Moon, "Sageflow: Robust federated learning against both stragglers and adversaries," *Advances in neural information processing systems*, vol. 34, pp. 840–851, 2021.

[97] A. Panda, S. Mahloujifar, A. N. Bhagoji, S. Chakraborty, and P. Mittal, "Sparsefed: Mitigating model poisoning attacks in federated learning with sparsification," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 7587–7624.

[98] C. Xie, M. Chen, P.-Y. Chen, and B. Li, "Crfl: Certifiably robust federated learning against backdoor attacks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 372–11 382.

[99] X. Cao, J. Jia, and N. Z. Gong, "Provably secure federated learning against malicious clients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 6885–6893.

[100] X. Cao, Z. Zhang, J. Jia, and N. Z. Gong, "Flcert: Provably secure federated learning against poisoning attacks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3691–3705, 2022.

[101] K. Zhang, G. Tao, Q. Xu, S. Cheng, S. An, Y. Liu, S. Feng, G. Shen, P.-Y. Chen, S. Ma *et al.*, "Flip: A provable defense framework for backdoor mitigation in federated learning," *arXiv preprint arXiv:2210.12873*, 2022.

[102] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[103] M. Naseri, Y. Han, E. Mariconti, Y. Shen, G. Stringhini, and E. De Cristofaro, "Cerberus: exploring federated prediction of security events," in *Proceedings of the 2022*

ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 2337–2351.

[104] H. Mozaffari, V. Shejwalkar, and A. Houmansadr, "Frl: Federated rank learning," *arXiv preprint arXiv:2110.04350*, 2021.

[105] Y. Zhao, K. Xu, H. Wang, B. Li, and R. Jia, "Stability-based analysis and defense against backdoor attacks on edge computing services," *IEEE Network*, vol. 35, no. 1, pp. 163–169, 2021.

[106] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574–4588, 2021.

[107] Y. Miao, Z. Liu, H. Li, K.-K. R. Choo, and R. H. Deng, "Privacy-preserving byzantine-robust federated learning via blockchain systems," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2848–2861, 2022.

[108] Z. Ma, J. Ma, Y. Miao, Y. Li, and R. H. Deng, "Shieldfl: Mitigating model poisoning attacks in privacy-preserving federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1639–1654, 2022.

[109] Z. Zhang, L. Wu, C. Ma, J. Li, J. Wang, Q. Wang, and S. Yu, "Lsfl: A lightweight and secure federated learning scheme for edge computing," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 365–379, 2022.

[110] B. Hou, J. Gao, X. Guo, T. Baker, Y. Zhang, Y. Wen, and Z. Liu, "Mitigating the backdoor attack by federated filters for industrial iot applications," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3562–3571, 2021.

[111] Z. Hu, K. Shaloudegi, G. Zhang, and Y. Yu, "Federated learning meets multi-objective optimization," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2039–2051, 2022.

[112] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," *Advances in neural information processing systems*, vol. 27, 2014.

[113] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964.

[114] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[115] T. D. Nguyen, P. Rieger, R. De Viti, H. Chen, B. B. Brandenburg, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen *et al.*, "{FLAME}: Taming backdoors in federated learning," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1415–1432.

[116] R. Thibaux and M. I. Jordan, "Hierarchical beta processes and the indian buffet process," in *Artificial intelligence and statistics*. PMLR, 2007, pp. 564–571.

[117] D. M. Blei and P. I. Frazier, "Distance dependent chinese restaurant processes." *Journal of Machine Learning Research*, vol. 12, no. 8, 2011.

[118] R. Socher, A. Maas, and C. Manning, "Spectral chinese restaurant processes: Nonparametric clustering based on similarities," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 698–706.

[119] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022.

[120] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *arXiv preprint arXiv:1708.06733*, 2017.

[121] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[122] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[123] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang *et al.*, "Large scale distributed deep networks," *Advances in neural information processing systems*, vol. 25, 2012.

[124] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[125] P. Regulation, "General data protection regulation," *Intouch*, vol. 25, 2018.

[126] "Federated learning: Collaborative machine learning without centralized training data. [online]." https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, 2017.

[127] "Utilization of fate in risk management of credit in small and micro enterprises. [online]." https://www.fedai.org/cases/utilization-of-fate-in-risk-management-of-credit-in-small-and-micro-enterprises/.

[128] "Machine learning ledger orchestration for drug discovery (melloddy). [online]." https://www.melloddy.eu/, 2017.

[129] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "Detox: A redundancy-based framework for faster and more robust gradient aggregation," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[130] C. Xie, S. Koyejo, and I. Gupta, "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6893–6901.

[131] Z. Yang and W. U. Bajwa, "Byrdie: Byzantine-resilient distributed coordinate descent for decentralized learning," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 611–627, 2019.

[132] T. Minka, "Estimating a dirichlet distribution," 2000.

[133] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, "Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1354–1371.

[134] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[135] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[136] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," *arXiv preprint arXiv:1807.04457*, 2018.

[137] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[138] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[139] E. Bagdasaryan, "(un) trustworthy machine learning," Ph.D. dissertation, Cornell University, 2023.

Appendix A

Appendix

A.1 Proofs of Theorem 2

proof 9 *We have*

$$
\begin{aligned}
& G_i^{'} - G_i \\
& = \frac{1}{n}\{(\frac{med(L_i^{'})}{\|\mathbf{w}_{1,i}^{'}\|}\mathbf{w}_{1,i}^{'} + \frac{med(L_i^{'})}{\|\mathbf{w}_{2,i}^{'}\|}\mathbf{w}_{2,i}^{'} + , ..., + \frac{med(L_i^{'})}{\|\mathbf{w}_{n,i}\|}\mathbf{w}_{n,i}) \\
& - (\frac{med(L_i)}{\|\mathbf{w}_{1,i}\|}\boldsymbol{w}_{1,i} + \frac{med(L_i)}{\|\mathbf{w}_{2,i}\|}\mathbf{w}_{2,i} + , ..., + \frac{med(L_i)}{\|\mathbf{w}_{n,i}\|}\mathbf{w}_{n,i})\} \\
& = \frac{1}{n}(\mathbf{w}_{1,i}^{'} + , ..., + \mathbf{w}_{m,i}^{'} + \sum_{i=m+1}^{n}\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}) \\
& = \frac{1}{n}(\mathbf{w}_{1,i}^{'} - G_{i-1}) + , ..., + \frac{1}{n}(\mathbf{w}_{m,i}^{'} - G_{i-1}) + \frac{m}{n}\cdot G_{i-1} \\
& + \frac{1}{n}(\sum_{i=m+1}^{n}\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}),
\end{aligned}
\tag{A.1}
$$

The optimization problem of maximizing $S_i^T(G_i^{'} - G_i)$ can be equivalently converted to maximize $\left|G_i^{'} - G_i\right|$. Out of convenience, we assume that all malicious clients can collude, so we have $\mathbf{w}_{1,i}^{'} = \mathbf{w}_{2,i}^{'} = ... = \mathbf{w}_{m,i}^{'}$. Then we have

$$
\left|G_i^{'} - G_i\right| \leq \frac{1}{n}\left|m\cdot(\mathbf{w}_{1,i}^{'} - G_{i-1})\right| + \frac{m}{n}\cdot|G_{i-1}| + \frac{1}{n}\left|\sum_{i=m+1}^{n}\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}\right|. \tag{A.2}
$$

According to Assumption 1, we have

$$\frac{1}{c}\left|\mathbf{w}_{1,i}^{'}-\mathbf{w}_{1,i}\right| \leq \left|G_i^{'}-G_i\right|. \tag{A.3}$$

Combing above inequality equations (A.2) and (A.3), we get

$$\frac{1}{c}\left|\mathbf{w}_{1,i}^{'}-\mathbf{w}_{1,i}\right| \leq \left|G_i^{'}-G_i\right| \leq \frac{1}{n}\left|m\cdot(\mathbf{w}_{1,i}^{'}-G_{i-1})\right|+\frac{m}{n}\cdot|G_{i-1}|+\frac{1}{n}\left|\sum_{i=m+1}^{n}\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}-\sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}\right|. \tag{A.4}$$

According to the triangle inequality $\left|\mathbf{w}_{1,i}^{'}-\mathbf{w}_{1,i}\right| \geq \left|\mathbf{w}_{1,i}^{'}-G_{i-1}\right|-\left|\mathbf{w}_{1,i}-G_{i-1}\right|$*, we get*

$$(\frac{1}{c}-\frac{m}{n})\left|\mathbf{w}_{1,i}^{'}-G_{i-1}\right| \leq \frac{1}{c}\left|\mathbf{w}_{1,i}-G_{i-1}\right|+\frac{m}{n}\cdot|G_{i-1}|+\frac{1}{n}\left|\sum_{i=m+1}^{n}\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}-\sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}\right|. \tag{A.5}$$

Based on our Definition 2, we have $\left|\mathbf{w}_{1,i}^{'}-G_{i-1}\right|=\rho$*, so we get*

$$\rho \leq \frac{n}{n-cm}\left|\mathbf{w}_{1,i}-G_{i-1}\right|+\frac{cm}{n-cm}|G_{i-1}|+\frac{c}{n-cm}\left|\sum_{i=m+1}^{n}\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}-\sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}\right|. \tag{A.6}$$

As we know $l^{'} \leq \|\mathbf{w}_i\|_{max}$*, so the final upper bound on* ρ *is*

$$\rho \leq \frac{n}{n-cm}\left|\mathbf{w}_{1,i}-G_{i-1}\right|+\frac{cm}{n-cm}|G_{i-1}|+\frac{c}{n-cm}\left|\sum_{i=m+1}^{n}\frac{\|\mathbf{w}_i\|_{max}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}-\sum_{i=1}^{n}\frac{l}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}\right|. \tag{A.7}$$

A.2 Proofs of Theorem 3

proof 10 *We have*

$$
\begin{aligned}
&\|G_i^{'} - G_i\| \\
&= \frac{1}{n}\|(\frac{med(L_i^{'})}{\|\mathbf{w}_{1,i}^{'}\|}\mathbf{w}_{1,i}^{'} + \frac{med(L_i^{'})}{\|\mathbf{w}_{2,i}^{'}\|}\mathbf{w}_{2,i}^{'} + , ..., + \frac{med(L_i^{'})}{\|\mathbf{w}_{n,i}\|}\mathbf{w}_{n,i}) - \sum_{i=1}^{n}\mathbf{w}_{i,i}\| \\
&= \frac{1}{n}\|(\mathbf{w}_{1,i}^{'} + , ..., + \mathbf{w}_{m,i}^{'} + \sum_{i=m+1}^{n}\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i}) - \sum_{i=1}^{n}\mathbf{w}_{i,i}\| \\
&\leq \frac{1}{n}\|\sum_{i=1}^{m}(\mathbf{w}_{i,i}^{'} - \mathbf{w}_{i,i}) + \sum_{i=1}^{n}(\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\| \\
&\leq \frac{1}{n}(m \cdot \|\mathbf{w}_{1,i}^{'} - \mathbf{w}_{1,i}\|) + \frac{1}{n}\cdot\sum_{i=1}^{n}\|(\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\|.
\end{aligned}
\tag{A.8}
$$

According to Assumption 1, we have

$$
\frac{1}{c}\|\mathbf{w}_{1,i}^{'} - \mathbf{w}_{1,i}\| \leq \frac{1}{n}(m \cdot \|\mathbf{w}_{1,i}^{'} - \mathbf{w}_{1,i}\|) + \frac{1}{n}\cdot\sum_{i=1}^{n}\|(\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\|. \tag{A.9}
$$

According to the triangle inequality $\|\mathbf{w}_{1,i}^{'} - \mathbf{w}_{1,i}\| \geq \|\mathbf{w}_{1,i}^{'} - G_{i-1}\| - \|\mathbf{w}_{1,i} - G_{i-1}\|$*, we get*

$$
\frac{1}{c}(\|\mathbf{w}_{1,i}^{'} - G_{i-1}\| - \|\mathbf{w}_{1,i} - G_{i-1}\|) \leq \frac{1}{n}(m \cdot \|\mathbf{w}_{1,i}^{'} - \mathbf{w}_{1,i}\|) + \frac{1}{n}\cdot\sum_{i=1}^{n}\|(\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\|. \tag{A.10}
$$

Given the Definition 2, we get $\|\mathbf{w}_{1,i}^{'} - G_{i-1}\| = \|\gamma\nabla^{p}\| = \rho$ *for DNY-OPT attacks, so we get the final upper bound on* ρ

$$
\rho \leq \|\mathbf{w}_{1,i} - G_{i-1}\| + \frac{c}{n - cm}\cdot\sum_{i=1}^{n}\|(\frac{l^{'}}{\|\mathbf{w}_{i,i}\|}\mathbf{w}_{i,i} - \mathbf{w}_{i,i})\|. \tag{A.11}
$$

A.3 Proofs of Theorem 4 and 5

Proof of theorem 6.

proof 11 *We have the triangle inequality,*

$$
\|\mathbf{w}_{1,i}^{'} - \mathbf{w}_{1,i}\| \geq \|\mathbf{w}_{1,i}^{'} - G_{i-1}\| - \|\mathbf{w}_{1,i} - G_{i-1}\|, \tag{A.12}
$$

then, we get

$$\|\mathbf{w}_{1,i}^{'} - G_{i-1}\| - \|\mathbf{w}_{1,i} - G_{i-1}\| \leq \|\mathbf{w}_{1,i}^{'} - \mathbf{w}_{1,i}\| \leq \max_{i,j\in[m+1,n]} \|\mathbf{w}_i - \mathbf{w}_j\|. \tag{A.13}$$

Based on the Definition 2, we have $\|\mathbf{w}_{1,i}^{'} - G_{i-1}\| = \|\gamma\nabla^p\| = \rho$.

So the upper bound on ρ *is*

$$\rho \leq \|\mathbf{w}_{1,i} - G_{i-1}\| + \max_{i,j\in[m+1,n]} \|\mathbf{w}_i - \mathbf{w}_j\|. \tag{A.14}$$

Proof of theorem 7

proof 12 *We have*

$$\begin{aligned} &(n-m)\|\mathbf{w}_{1,i}^{'} - G_{i-1}\|^2 - \sum_{i\in[m+1,n]} \|\mathbf{w}_{1,i} - G_{i-1}\|^2 \\ &\leq \sum_{i\in[m+1,n]} \|\mathbf{w}_{1,i}^{'} - \mathbf{w}_{1,i}\|^2 \leq max \sum_{i,j\in[m+1,n]} \|\mathbf{w}_i - \mathbf{w}_j\|^2, \end{aligned} \tag{A.15}$$

then we have

$$(n-m)\rho^2 \leq \sum_{i\in[m+1,n]} \|\mathbf{w}_{1,i} - G_{i-1}\|^2 + max \sum_{i,j\in[m+1,n]} \|\mathbf{w}_i - \mathbf{w}_j\|^2. \tag{A.16}$$

So, the proof of the upper bound on ρ *is completed.*

A.4 Proofs of Theorem 6

proof 13 *We assume the compromised local models are the same. Therefore, we have:*

$$\sum_{i\in\Gamma_{\mathbf{w}'}^{n-2m-1}} \|\mathbf{w}^{'} - \mathbf{w}_i\| \leq \min_{j\in[m+1,n]} \sum_{i\in\Gamma_{\mathbf{w}}^{n-m-2}} \|\mathbf{w}_j - \mathbf{w}_i\|, \tag{A.17}$$

then, we get:

$$
\begin{aligned}
&(n-2m-1)\cdot\|\mathbf{w}_{1,i}^{'}-G_{i-1}\| \\
&\leq \min_{j\in[m+1,n]}\sum_{i\in\Gamma_{\mathbf{w}}^{n-m-2}}\|\mathbf{w}_{j,i}-\mathbf{w}_{i,i}\|+\sum_{i\in\Gamma_{\mathbf{w}'}^{n-2m-1}}\|\mathbf{w}_{i,i}-G_{i-1}\| \\
&\leq \min_{j\in[m+1,n]}\sum_{i\in\Gamma_{\mathbf{w}}^{n-m-2}}\|\mathbf{w}_{j,i}-\mathbf{w}_{i,i}\|+(n-2m-1)\cdot\max_{i\in[m+1,n]}\|\mathbf{w}_{i,i}-G_{i-1}\|
\end{aligned}
\tag{A.18}
$$

Since we have $\|\mathbf{w}_{1,i}^{'}-G_{i-1}\|=\rho$. *Then, the proof is completed.*