Improved Model Poisoning Attacks and Defenses in Federated Learning with Clustering

by

Xinda Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Federated Learning (FL) allows multiple participants to collaboratively train a deep learning model without sharing their private training data. However, due to its distributive nature, FL is vulnerable to various poisoning attacks. An adversary can submit malicious model updates that aim to degrade the joint model's utility. In this thesis, we formulate the adversary's goal as an optimization problem and present an effective model poisoning attack using projected gradient descent. Our empirical results show that our attack has a larger impact on the global model's accuracy than previous attacks.

Motivated by this, we design a robust defense algorithm that mitigates existing poisoning attacks. Our defense leverages constraint k-means clustering and uses a small validation dataset for the server to select optimal updates in each FL round. We conduct experiments on three non-iid image classification datasets and demonstrate the robustness of our defense algorithm under various FL settings.

Acknowledgements

First, I would like to express my gratitude to my supervisor Florian Kerschbaum for the invaluable support and guidance in developing this thesis. I look forward to continuing working with him in the future. I also want to thank my committee members, Gautam Kamath and Hongyang Zhang, for reading the thesis and providing great comments on various parts of it. I would like to thank many of my amazing friends, who made me not feel lonely during such a hard time of the pandemic. Special thanks to Jay Chou for his beautiful songs. I am lucky to have been born in the era with his music. Finally, I thank my family for unconditional love, support and encouragement.

This work benefited from the use of the CPI Chippie compute facility at the University of Waterloo.

Dedication

To my loved ones.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Federated Learning (FL) is an emerging framework for collaboratively training a deep learning model among thousands or even millions of participants [19], [24]. In a typical FL setting, a central server maintains a global model and coordinates the training of a joint global model among several clients. Each client trains a model based on the local data, and the global model is learnt by combining the contributions of all clients. Compared to a traditional centralized setting, FL is computational-efficient on the server side due to outsourcing and parallelizing the model training process. Moreover, each client only needs to send local model updates to the central server, and no raw training data is shared between clients and the service provider. FL has been widely applied in many areas. For example, Google has applied FL for next-word prediction on Android Gboard [16], [44]; Apple leverages FL for creating voice recognition models [15].

Although FL seems to be a promising solution for many real-world machine learning settings, FL system is vulnerable to model poisoning attacks [3], [7], [26]. Due to the distributive nature of FL, malicious clients, which could be either compromised normal clients or fake ones inserted by the attacker, can lower the testing accuracy of the global model by manipulating the model updates in the training round. For instance, the FedAvg algorithm [24], which has been widely used in non-adversarial FL settings, is susceptible to model poisoning attacks. FedAvg takes the weighted average of the model parameters as the global model, and it has been shown that the resulting global model can be arbitrarily manipulated even if only a single client is compromised [7].

Many existing works have proposed Byzantine robust aggregation algorithms that aim to mitigate the impact of model poisoning attacks [1], [7], [11], [26], [43], [45]. The idea of these defenses is to detect and remove outliers of model updates before incorporating them into the global model. However, many of these aggregation algorithms assume independent and identically distributed (iid) training data, while non-iid datasets are more common in real-world scenarios. Existing works [14], [35] have shown that these defenses are insufficient to prevent FL models from poisoning attacks, especially in the non-iid case. Moreover, previous defenses assume the number of participating malicious clients is fixed and remove that number of model updates in every training round. Consequently, the aggregation algorithm may remove an excess number of benign updates, which results in the global model suffering an accuracy loss even if no attackers are present.

Attacking FL models: Our first contribution in this thesis is a generic model poisoning attack to FL. Our new model poisoning attack uses a better optimization technique and can be used as a threat assessment framework for any FL systems. Our attack formulates the problem as an optimization similar to previous works [14], [35] but uses a fundamentally different approach to solve the optimization. We leverage the projected gradient descent (PGD) and Dykstra's algorithm [8] to optimize the malicious update such that it can evade the detection of the defender while effectively poisoning the global model. The PGD algorithm moves the malicious update towards an adversarial direction, leading to low model accuracy while ensuring its closeness to other benign updates. Our experiment results show that our attack breaks all existing defenses and decreases the global model testing accuracy by $10 - 70\%$. Our attack outperforms existing state-of-the-art attacks on all evaluated datasets.

Defending against model poisoning attacks: Besides the attack, as a second contribution, we propose a new defense framework using clustering that aims to mitigate untargeted model poisoning attacks. Our defense first clusters the received updates and then applies a two-phase outlier removal before aggregating. Unlike previous defenses, the number of updates we remove is chosen dynamically based on estimating the number of participating malicious clients in each FL round. Specifically, the server collects a small labelled dataset and uses it as a validation dataset for determining the final model aggregate. We show that our defense achieves similar testing accuracy as FedAvg (less than 1% degradation) in non-adversarial settings. Moreover, our defense is robust under state-of-the-art model poisoning attacks and our new proposed attack among three evaluated datasets (with degradation of testing accuracy around 2%). To further evaluate the robustness of our defense, we design an adaptive attack based on our new attack framework. Our adaptive attack assumes the knowledge of the entire procedure of the algorithm and is particularly optimized against the defense by taking the clustering step into consideration.

The remainder of the paper is organized as follows: In Chapter 2 we give brief introductions to Deep Neural Networks and Federated Learning. We describe state-of-the-art attacks and defenses on poisoning attacks in Chapter 3. In Chapter 4, we discuss the threat models and the assumptions for the defender. We give detailed descriptions of our attack and defense algorithm in Chapter 5 and 6. Finally, in Chapter 7, we evaluate our solutions and compare them with existing works.

Chapter 2

Background

2.1 Deep Neural Networks (DNNs)

A deep neural network consists of multiple layers for extracting features from the input. Each layer contains the weight, bias parameters and a (non-linear) activation function. A DNN classifier is a function $f : \mathcal{X} \to \mathcal{Y}$ that assigns a likelihood score to inputs $x \in \mathcal{X} \subseteq \mathbb{R}^d$ for each of $K \in \mathbb{N}$ classes $\mathcal{Y} \subseteq \mathbb{R}^K$. A softmax activation function is applied to the output layer to convert likelihoods into a vector of real values in $[0, 1]$ that sums to 1. Therefore, the output can be interpreted as the probabilities of the input falling in each prediction class.

A differentiable loss function $\mathcal{L}$ is usually defined to measure the difference between the model prediction and the ground-truth labels. The empirical loss on the model parameter ω over the training dataset D is $l(\omega) = \frac{1}{N} \sum_{(x_i, y_i) \in D} \mathcal{L}(f(x_i; \omega), y_i)$. To solve the optimization problem $\arg\min_\omega l(\omega)$, the mini-batch stochastic gradient descent (SGD) algorithm is often used for optimization. At each iteration, the backpropagation algorithm computes the gradient with respect to the weights of the network. The model weights are then updated towards the local minimum.

2.2 Federated Learning (FL)

Federated Learning (FL) [19], [24] is a machine learning setting where multiple data owners collaboratively learn a global machine learning model without the need to share private data. Suppose we have N clients, and each of them has a local training dataset D_i. Furthermore, a local machine learning model is maintained by each client for its data. Specifically, the model parameters are obtained by solving the optimization problem $\arg\min_\omega l(\omega)$ on D_i for some loss function $\mathcal{L}$. A server maintains a global model via aggregating local models from these clients. In the t^{th} FL round, the server selects a subset of n out of N clients and broadcasts the current global model weights ω_g^t to them. Then each of the chosen clients fine-tunes ω_g^t on its local dataset for a fixed E epochs and obtains a new local model ω_k^t. After the training process completes, each of the n clients computes the model update $\nabla_k^t = \omega_k^t - \omega_g^t$ and sends ∇_k^t back to the server, which then applies an aggregation algorithm f_{agg} on all client updates to obtain $\nabla_g^t = f_{agg}(\{\nabla_k^t : k \in [n]\})$. Finally, the server updates the global model in the $(t+1)^{th}$ round with the new weights $\omega_g^{t+1} = \omega_g^t + \nabla_g^t$, and broadcasts it to a new subset of selected clients. This process is repeated until the global model converges.

FL can be classified into two types: cross-silo and cross-device. In cross-silo FL, a set of organizations (e.g., financial or medical) act as the clients to perform local training, and a third-party entity acts as the global server to coordinate the process. All clients are almost always available in each training round. While in cross-device FL, the clients are usually many mobile or loT devices, and only a fraction of clients participate in each training round. In this thesis, we study the poisoning attack on FL and focus on the cross-device setting where the clients are usually highly unreliable.

Chapter 3

State-of-the-art Attacks and Defenses in FL

3.1 Robust Aggregation Algorithms

The vanilla average aggregation (FedAvg) [24] in standard FL has been shown to be non-robust that a single malicious client can manipulate the final global model weight arbitrarily [7]. Existing works have proposed *Byzantine-robust* Aggregation for FL to defend against model poisoning attacks by adversaries.

Krum

Krum [7] selects one of the n updates that has the lowest score based on ℓ_2 distance. The score of the i^{th} update is computed as follows:

$$s_i = \sum_{\nabla_j \in \mathcal{P}} \|\nabla_i - \nabla_j\|_2$$

where m is an estimation of the upper bound on the number of malicious clients and $\mathcal{P}$ represents the set of the $n - m - 2$ neighbouring updates of ∇_i. The intuition behind this is that malicious updates need to be far from benign ones to poison the global model.

Multi-krum

Multi-krum [7] is a variant of Krum which has much higher global model accuracy. Multi-krum iteratively selects an update from the remaining set multiple times using the same algorithm as Krum, resulting in a selection set $\mathcal{S}$ with c updates such that $c < n - 2m - 2$. Finally, Multi-krum returns the dimensional-wise average of $\mathcal{S}$.

Trimmed-mean

Trimmed-mean [45] is a coordinate-wise aggregation algorithm. For a dimension k, trimmed-mean sorts the values in the k^{th} dimension of the n vectors and removes the m smallest and largest values. Then it performs dimensional-wise averaging on the remaining $n - 2m$ vectors. Yin et al. [45] show in theory that trimmed-mean achieves order-optimal statistical error rates for strongly convex losses.

Median

Median [45] is another algorithm that performs aggregation along each dimension. Median aggregates the client updates by computing the median of the values along each dimension. Like the trimmed-mean aggregation rule, median also has an order-optimal error rate when the objective function is strongly convex.

Divide-and-Conquer (DnC)

DnC [35] randomly selects a sorted index set r and constructs sub-sampled updates indexed by r: $\widetilde{\nabla} = \nabla[r]$. Next, DnC computes an outlier score for each update using SVD-based spectral methods and removes m updates with the high scores. The above process runs *niter* number of times to avoid dependence on a single r. Finally, DnC returns the average of common updates across *niter* iterations. The theoretical guarantee of DnC relies on the assumption that the malicious updates and the benign ones are sufficiently separated. DnC leverages sub-sampling techniques to achieve lower memory and computational cost compared to the standard SVD-based defenses [41].

3.2 Poisoning Attacks

Existing works have shown that Federated learning is vulnerable to various poisoning attacks [14], [6], [35], [28], [18], [40]. There are two categories of attacks based on the adversary's goal: targeted and untargeted. For targeted attacks, the adversary aims to decrease the model accuracy on some specific inputs. In contrast, the goal for untargeted attacks is to minimize the model accuracy on any input test data. Backdoor attacks [3] are a subset of targeted attacks where the attacker changes the model's behavior only on some attacker-chosen inputs (backdoor triggers) while maintaining accuracy on the main FL learning task. Although targeted and backdoor attacks cause the model to misclassify a set of specific input samples, untargeted attacks are designed to decrease the utility of the global model and all client models. As Fang et al. [14] point out, many existing Byzantine-robust aggregation algorithms can only asymptotically bound the error rates of the global model up to a constant factor. Such guarantees do not imply empirical robustness of the global model. For instance, as shown in Section 7.2, a successful untargeted attack can lower the global model accuracy on CIFAR-10 by more than 20% even when robust aggregation algorithms are applied, which may prevent the use of the model in the future. Therefore, in this thesis we focus on untargeted attacks, which pose a more severe threat to the use of FL. Based on the adversary's capability, there are two types of FL poisoning attacks.

3.2.1 Model Poisoning Attacks

Unlike the centralized setting, the distributed nature of FL introduces another new threat for poisoning attacks. In model poisoning attacks, the adversary directly manipulates the model updates sent from compromised clients to the server. Due to the stronger capability of the attacker, model poisoning attacks can achieve a higher impact on the model accuracy.

Multiple model poisoning attacks have been proposed in the literature [4], [14], [35]. Little Is Enough (LIE) attack [4] adds a small perturbation noise to each dimension of the averaged benign updates. The attack assumes the knowledge of the benign updates. Specifically, the attacker first computes the dimensional-wise average of available benign updates (∇^b) and adds calibrated noises onto it based on the standard deviation (σ) of the benign updates. The poisoned update is constructed as $\nabla^m = \nabla^b + z\sigma$ where z is a scalar depending on the number of malicious and total clients. Baruch et al. [4] show that LIE evades the detection of many robust aggregation algorithms and effectively poisons the global model.

Fang [14] and Shejwalkar et al. [35] formulate the tailored attack as an optimization problem. The tailored attack assumes the attacker knows the aggregation algorithm. The attack computes the average of available benign updates as a reference update (∇^b) and set a malicious perturbation direction $\nabla^p \leftarrow sign(\nabla^b)$. The malicious update ∇^m is a perturbed version of the reference update towards the malicious perturbation direction ∇^p. The objective of the optimization is to maximize the distance between the final update after aggregation and the reference update, formally shown as follows:

$$\underset{\gamma}{\text{argmax}}\ \|\nabla^b - f_{agg}(\nabla^m_{\{k\in[m]\}} \cup \nabla_{\{k\in[m+1,n]\}})\|, \tag{3.1}$$
$$\nabla^m_{\{k\in[m]\}} = \nabla^b + \gamma\nabla^p,\ \nabla^b = Avg(\nabla_{\{k\in[n]\}})$$

where $\nabla_{\{k\in[n]\}}$ is the benign update set available to the adversary. The goal is to find an optimal scalar γ that circumvents the aggregation algorithm. This attack assumes the server's aggregation algorithm is known to the attacker, who can easily check whether the tailored malicious update is selected by f_{agg}. Since f_{agg} is usually non-differentiable, a line search is used for the optimization. The algorithm first initializes γ to a large value and gradually decreases γ until ∇^m satisfies the adversarial objective. Shejwalkar et al. [35] improve the previous attack by utilizing better line search techniques and considering other choices of perturbation vectors ∇^p (e.g., normalized version of ∇^b).

In case the underlying aggregation algorithm is hidden from the adversary, the adversary cannot compute the result of the aggregation since f_{agg} is unknown. Shejwalkar et al. [35] propose two *Agg-agnostic* attacks, called Min-max and Min-sum. The construction of the malicious vector ∇^m is the same as the tailored attack above. The general idea of the attack is to ensure the malicious updates lie close to the clique of the benign updates. Min-max tries to maximize the scalar γ while ensuring that the maximum distance between the malicious update and any other updates is upper bounded by the maximum distance between any two benign updates. That is, $\max_{i\in[m+1,n]} \|\nabla^m - \nabla_i\|_2 \leq \max_{i,j\in[m+1,n]} \|\nabla_i - \nabla_j\|_2$. While Min-sum ensures the sum of the distances between the malicious update and all the benign gradients is upper bounded by the sum of distances between any benign gradient and the other benign gradients. That is, $\sum_{i\in[m+1,n]} \|\nabla^m - \nabla_i\|_2 \leq \sum_{i,j\in[m+1,n]} \|\nabla_i - \nabla_j\|_2$. Shejwalkar[35] empirically shows that Min-max and Min-sum attack have a stronger impact on various of defenses than the LIE attack.

3.2.2 Data Poisoning Attacks

Data poisoning attacks have been primarily explored in the centralized setting of machine learning [28], [18]. In data poisoning attacks, the adversary indirectly manipulates the model updates by inserting poisoning data to the local training dataset on compromised clients. Fang et al. [14] apply the simple label flipping attack to FL settings. In this attack, the attacker flips the labels of the local data on compromised clients to some other false labels.

Similar to (3.1), data poisoning attack can be formulated as an optimization problem as follows,

$$\begin{gathered}\underset{D^m_{\{k\in[m]\}}\subset\mathcal{D}}{\operatorname{argmax}} \;\|\nabla^b - f_{agg}(\nabla^m_{\{k\in[m]\}} \cup \nabla_{\{k\in[m+1,n]\}})\| \\ \nabla^m_{k\in[m]} = \mathcal{T}(\omega_g, D^m_{\{k\in[m]\}}) - \omega_g,\; \nabla^b = Avg(\nabla_{\{k\in[n]\}})\end{gathered}$$

where $\mathcal{T}$ is a training algorithm (SGD) that uses malicious input data $D^m \subset \mathcal{D}$ to fine-tune the current model weight ω_g and outputs the new weight. Fang et al. [14] show that using gradient-based optimization to compute D^m is time-consuming and not effective. Shejwalkar et al. [36] propose to use label-flipped data to replace the compromised client's dataset and optimize the number of the data carefully based on the server aggregation algorithm.

We notice that data poisoning attacks can be transferred to model poisoning attacks. For example, we can train the local model on a poisoned dataset and treat the changes of model weights as malicious model updates. Moreover, the literature [14], [6], [35] have suggested that model poisoning attacks are more effective than data poisoning attacks in many FL settings. Therefore, in this thesis we study untargeted model poisoning attacks.

Chapter 4

Attack and Defense Model

In this chapter, we discuss possible threat models of poisoning attacks in FL as well as the assumptions and goals of the defense model.

4.1 Attack Model

Attacker's capability: For untargeted model poisoning attacks, the goal of the adversary is to reduce the global model's accuracy on any test input by crafting malicious updates on compromised participating clients. Following previous works [3], [4], [6], [14], [35], we assume the attacker controls P out of N total clients. Those clients can be either normal clients compromised by the attacker or fake clients injected into the FL system. We assume the *malicious fraction* $q = \frac{P}{N}$ is less than 50%; otherwise no Byzantine robust aggregation algorithm can withstand poisoning attacks. Following previous studies of model poisoning attacks, we assume the adversary can arbitrarily manipulate the updates sent from the P compromised clients to the central server in every FL round.

Attacker's knowledge: Previous works [10], [14], [35] assume the attacker has the following basic knowledge about the FL system: local training code, training dataset, and model updates of the compromised clients. Besides, similar to [14], [35], we consider two other dimensions in adversarial FL settings: the knowledge of the aggregation algorithm and the knowledge of other benign model updates. Both of them are relatively strong assumptions for the adversary. However, they can be used to understand the severity of the threat to FL models and evaluate the robustness of the defense mechanism. Previous tailored attacks [14], [35] assume full knowledge of the aggregation algorithm while the LIE algorithm [4] assumes full knowledge of the benign updates. [35] proposes Min-max and Min-sum attacks that do not require knowledge of the aggregation rule and shows they are more effective when the benign updates are known. In this thesis, for the design our attack we consider the more practical scenario where the defense algorithm is not public and consider both cases whether the benign updates are known or not.

4.2 Defense Model

Defender's goal: The goal of the defender is two-fold. 1) The global model should be robust under reasonable threat models as described above, i.e. the model should maintain high accuracy on the main task; 2) The defense mechanism should not sacrifice model accuracy when there is no presence of attackers, which is known as *fidelity*. Many existing defense algorithms only aim to address robustness but overlook fidelity. For instance, our experiment results in Section 7.3 show using Krum [7] causes the testing accuracy to drop by more than 20% on the MNIST dataset which makes the model useless. We further discuss the common weaknesses of the existing defenses in Chapter 6. Therefore, when designing defense mechanisms, it is necessary to compare the model accuracy under no attacks with a baseline algorithm such as FedAvg [24]. For robustness, we design our defense under a powerful adversarial setting where the benign updates are completely known to the attacker. Furthermore, we propose an adaptive attack when the defense algorithm is revealed to the attacker and evaluate our defense against it.

Defender's knowledge: The defender does not know any client-side information including local training data. Unlike previous work, we assume the server also does not need to know the upper bound of the *malicious fraction* q. The server can only access the model updates it receives from participating clients in each round. To ensure both fidelity and robustness, we assume the server has a tiny clean labelled dataset. The dataset is from a similar domain but may not follow the same distribution as the training data. Moreover, the server's dataset does not have to be iid and may be skewed. We show in Section 7.3 that a very small dataset (100 samples for MNIST and FEMNIST) is sufficient to satisfy our goal. Such a small dataset can often be collected from the public domain and labelled manually.

Chapter 5

Our Attack

In this section, we describe the formulation of our new model poisoning attacks on FL, followed by an algorithm that we can use as a generic framework for attacking any FL defense. In our attack, we assume the attacker does not know the defender's algorithm, which is a practical setting since the aggregation algorithm of a FL platform is not public in many cases. We also assume the attacker knows the benign updates in our derivation. In the case that the benign updates are unknown, the attacker can obtain an estimation of them by performing clean training on the compromised clients' local datasets.

5.1 Model Poisoning Attack Formulation

In Federated Learning, the global server aggregates the client model updates it receives in each round. In order to poison the global model, the goal of the adversary is to send malicious updates such that the global model's accuracy is minimized with the new model weights after the aggregation. We formulate it as an optimization problem as follows.

Let $f(x; \omega)$ be the global model output on an input x with model parameter ω. We craft the malicious updates for the m clients we control, denoted $\nabla_{\{k \in [m]\}}$ (without loss of generality, we assume the first m clients are malicious). Let ω_g be the global model weights from the last iteration, and n be the number of total selected clients in each iteration, respectively. Recall that $\mathcal{L}$ is the loss function that computes the loss between the model output $f(x)$ and the true label y. Then the attacker aims to solve the following optimization problem:

$$\underset{\nabla_{\{k\in[m]\}}}{\operatorname{argmax}}\ \mathcal{L}(f(x;\ \overline{\omega}),\ y)$$
$$\text{where } \overline{\omega} = \omega_g + f_{agg}(\nabla_{\{k\in[m]\}} \cup \nabla_{\{k\in[m+1,n]\}})$$

We note that the above optimization problem depends on f and f_{agg} and does not have a closed-form solution most of the time. Moreover, since f_{agg} is not differentiable in many cases, gradient-based optimization techniques are not applicable here as well.

5.2 Solving the Optimization

Next, we show how we tackle the challenge by altering the optimization goal. We observe that many of the existing defenses (e.g., Krum, Multi-krum and DnC) follow a two-step procedure: 1) applying an outlier detection algorithm to remove some outlier updates; 2) averaging the remaining updates to ensure convergence. Therefore, the poisoned updates have to avoid the aggregation algorithm's detection and point to a malicious direction to decrease the model utility. In our attack, we force the malicious update into a feasible set chosen heuristically. We ensure the distance between the malicious update and each of the other benign update is within a threshold radius r, which is a tunable hyperparameter. Our intuition is that it restricts the malicious updates to be close enough to other benign updates in order to be selected by the defense algorithm. Since the choice of r has to depend on the scale of other updates that might vary between FL rounds, we compute r dynamically for each round to be the maximum distance of any other two benign updates. For simplicity, we set the model update for every selected malicious client to the same vector ∇^m. Therefore the optimization becomes:

$$\begin{aligned} &\underset{\nabla^m}{\operatorname{argmax}}\ \mathcal{L}(f(x;\ \overline{\omega}),\ y) \\ &\qquad \text{where } \overline{\omega} = \omega_g + \frac{1}{n}(m \cdot \nabla^m + \sum_{k\in[m+1,n]} \nabla_k) \\ &\qquad \text{subject to } \|\nabla^m - \nabla_j\|_2 \leq r,\ \text{ for } j \in [m+1, n] \end{aligned} \tag{5.1}$$

Applying change of variable, we optimize $\overline{\omega}$ instead. Then $\nabla^m = (n(\overline{\omega} - \omega_g) - \sum_{k\in[m+1,n]} \nabla_k)/m$. Thus, (5.1) becomes:

$$\begin{aligned}&\underset{\overline{\omega}}{\operatorname{argmax}}\ \mathcal{L}(f(x;\ \overline{\omega}),\ y)\\&\qquad\text{subject to } \|\overline{\omega}-(\omega_g+\frac{m\cdot\nabla_j+\sum_{k\in[m+1,n]}\nabla_k}{n})\|_2\leq\frac{m}{n}r,\\&\qquad\text{for } j\in[m+1,n]\end{aligned}$$

We observe that the feasible set is the intersection of $(n-m)$ ℓ_2 balls centered at $\omega_g+\frac{m\cdot\nabla_j+\sum_{k\in[m+1,n]}\nabla_k}{n}$ $(j\in[m+1,n])$ with radius $\frac{m}{n}\cdot r$, which is a convex set. Therefore, the problem is converted to finding $\overline{\omega}$ such that the model attains maximal empirical loss on the attacker's dataset while maintaining $\overline{\omega}$ in the constrained set. To solve the optimization problem, we use the projected gradient descent (PGD) algorithm on the negative loss. To project onto the feasible set, we apply Dykstra's algorithm [8]. Dykstra's algorithm is an iterative algorithm that calculates the projection onto intersections of given convex sets. The algorithm is developed on the premise that the projection onto each convex set (projection onto an ℓ_2 ball in our case) can be calculated cheaply. Finally, we present the overall procedure of our attack in Algorithm 1.

To perform the gradient descent on the model parameter $\overline{\omega}$, we assume the attacker has access to a labelled dataset. The adversary can either use a public-accessible dataset from a similar domain or simply combine the local datasets from all compromised clients. In this thesis, we evaluate our attack under the second setting.

We remark that our attack does not depend on the knowledge of the server's aggregation algorithm, and therefore can be used as a generic framework against any defense.

Algorithm 1: PGD poisoning attack

```
Input: Current model parameter $\omega_g$, attacker's dataset $(\mathcal{X}, \mathcal{Y})$, benign updates
       $\{\nabla_k\}(k \in [m, n))$, learning rate $\eta_t$, number of rounds $T$
Initialize $\overline{\omega}_0$ randomly
Compute centers and radius for the $n - m$ balls
centers $\leftarrow \omega_g + \frac{m \cdot \nabla_j + \sum_{k \in [m+1, n]} \nabla_k}{n},\ j \in [m, n)$
radius $\leftarrow \frac{m}{n} \cdot \max_{\{\nabla_x, \nabla_y\}} \|\nabla_x - \nabla_y\|_2$
for $t \in [T]$ do
    Compute gradient
    $\mathbf{g}_t \leftarrow \nabla_{\overline{\omega}_t} \sum_{(x_i, y_i) \in \mathcal{X}, \mathcal{Y}} \mathcal{L}(f(x_i;\ \overline{\omega}_t),\ y_i)$
    Descent
    $\overline{\omega}_{t+1} \leftarrow \overline{\omega}_t - \eta_t(-\mathbf{g}_t)$
    Projection
    $\overline{\omega}_{t+1} \leftarrow Dykstra(\overline{\omega}_{t+1}$, centers, radius)
end
Reconstruct $\nabla^m$
$\nabla^m = \frac{n(\overline{\omega}_{t+1} - \omega_g) - \sum_{k \in [m+1, n]} \nabla_k}{m}$
Output: The malicious update $\nabla^m$
```

Chapter 6

Our Defense

The strong impact of our attack motivates us to design a more robust defense algorithm against untargeted model poisoning attacks in FL. In this section, we identify a common pattern of existing attacks and point out a weakness of previous defenses. Based on the lessons we learned, we propose a novel clustering-based defense that addresses the two issues.

6.1 Success of Existing Attacks

State-of-the-art attacks are based on solving an optimization problem. The goal of the attacker is to maximize the malicious perturbation while keeping the malicious updates not too far from the benign ones in order to maximize the chances of being selected by the aggregation algorithm. Previous attacks assign identical updates or updates that are close to each other (by adding small noises) for every malicious client, which moves the barycenter of all update vectors towards the malicious direction. In case the training data is non-iid, which is a more realistic setting, such an approach becomes even more effective since the benign updates themselves are not concentrated and do not point to a single direction. Therefore, it becomes harder for the defender to distinguish malicious updates from benign ones. For example, in the Multi-krum case, keeping the malicious updates close to each other results in a small neighbouring distance for those updates, which will increase the possibility of being selected by the aggregation algorithm.

Table 6.1: Results of the one-shot attack for several server aggregation algorithms. The accuracy is in percentage.

	MNIST		FEMNIST	
Agg	Acc	Round	Acc	Round
Multi-krum	10.09	8	4.97	5
Trimmed-mean	9.8	2	4.97	2
DnC	10.28	3	4.84	2

6.2 Weakness of Existing Defenses

In the cross-device FL setting, the server randomly selects a subset of participating clients in each round. Existing defenses assume the knowledge of the (upper bound of) number of total malicious clients P and remove a fixed number of updates m in each FL round based on P. Suppose the total number of clients is N, and the number of participating clients in each round is n, then a common estimation in previous defenses of m is $n \cdot q$, where $q = P/N$ is the *malicious fraction*. However, we note that the number of selected malicious clients is a random variable M following a hypergeometric distribution with p.m.f. $Pr(M = m) = \binom{P}{m} \cdot \binom{N-P}{n-m} / \binom{N}{n}$. Therefore, even if all P, N, n are fixed and known, the defender does not know the exact number of malicious clients involved in a particular training round. By removing only a fixed number of updates, the server suffers from overestimating or underestimating M in each round, causing the removal of an excess number of benign updates or an insufficient number of malicious updates.

One-shot attack. Based on the above observation, we present a simple yet effective attack, called the one-shot attack. We show that the existing defenses that use a fixed estimation of M are vulnerable to such an attack. The idea of the attack is that the adversary greedily sends a sufficiently large model update for every malicious client in *every* round. Instead of trying to evade the defender's detection, the goal of the adversary is to send unstealthy but powerful updates that can destroy the global model after being selected once. Our intuition is that a single significantly large update can lead to a large model aggregate, which causes gradient explosion problems and other numerical instability issues. This may result in a global model (full of *NaN*s) that behaves like random guessing. Even worse, we notice that the model cannot be recovered even if all updates in the future rounds are benign. Table 6.1 gives a demonstration of our one-shot attack. We use the same FL parameters as other experiments in Chapter 7. The table shows the testing accuracy of applying the one-shot attack on three existing aggregation algorithms (Agg) for the MNIST and FEMNIST datasets. The *Round* column shows the round number at which the attack destroys the global model. As shown from the results, our one-shot attack is effective against Multi-krum [7], Trimmed-mean [45] and DnC [35]. Under our attack, the global model behaves like random guessing after only a few FL rounds. We note that our one-shot attack may not be practical since it is not stealthy and can be mitigated by other techniques (e.g., norm-clipping) or by inspecting the clients' model updates manually. However, it motivates us to design aggregation algorithms that estimate m in a dynamic fashion in each round.

6.3 Our Design

The design of our defense is based on the lessons we learned in Section 6.1 and Section 6.2. In order to limit the attacker's impact, we first cluster the updates that the server receives in each round. Our intuition for clustering is that if the attacker assigns similar updates for all malicious clients, these updates will fall into a single cluster. A single aggregate vector can then be used to represent similar updates for each cluster. We note that an attacker might exploit this by submitting different malicious updates, and this may cause each cluster to contain at least one malicious update due to the pigeonhole principle. Therefore, we use a two-phase outlier removal which removes suspicious updates within each cluster and among clusters. In phase one, we use a distance-based aggregation algorithm for each cluster and compute a cluster center for each of them. The outlier removal algorithm we use is similar to Multi-krum [7], where for each update we calculate the sum of Euclidean distance to its neighbouring updates and filter out u of them. In phase two, we apply the outlier removal algorithm again among all cluster centers and remove v of them. The final aggregate is the average of remaining cluster centers. Our two-phase outlier removal creates a dilemma for the adversary. It either chooses to greedily pull the model aggregate towards a malicious direction by submitting similar updates or spread the updates into multiple clusters, which results in a much weaker impact on the final aggregate. These two cases are handled by phase two and phase one of our defense algorithm, respectively. For the clustering algorithm, we apply the constrained k-means algorithm [5]. Constrained k-means is a variant of the standard k-means clustering algorithm that avoids local solutions with empty clusters or clusters having very few points. The algorithm ensures that every cluster contains at least a given number of points by imposing constraints onto the underlying clustering optimization problem. Since the constrained k-means algorithm ensures that each cluster has the same number of updates, our algorithm has a similar convergence guarantee with FedAvg [24].

To address the issue of the one-shot attack, instead of fixing the number of updates to remove, we dynamically optimize u and v in each round by assuming the defender has access to a small validation dataset D. Our intuition is that the defender cannot estimate the exact number of malicious clients in a training round only based on the update values, especially in the non-iid case. Therefore, we leverage a small dataset to help us determine the number of updates to remove. Let U and V be the domain set of u and v, respectively. U and V are determined by the number of updates in each cluster and the number of clusters. Then we optimize u and v by selecting the pair $(u, v) \in U \times V$ based on the validation loss on the dataset D.

```
Algorithm 2: Clustering-based Aggregation
   Input: Model updates $\nabla_1, ...\nabla_n$, validation dataset $D$, $U$, $V$, outlier removal
          algorithm $\mathcal{R}$, current model weight $\omega_g$
 1 $min\_loss \leftarrow$ inf
 2 $\overline{\nabla}_{best} \leftarrow \mathbf{0}$
 3 $C_1, ..., C_k = Cluster(\nabla_1, ..., \nabla_n)$
 4 for $(u, v) \in U \times V$ do
 5     for $i = 1, 2, ..., k$ do
 6         $agg_i \leftarrow \mathcal{R}(C_i;\ u)$
 7     end
 8     $\overline{\nabla} \leftarrow \mathcal{R}(agg_1, ..., agg_k;\ v)$
 9     $\overline{\omega} \leftarrow \omega_g + \overline{\nabla}$
10     $loss \leftarrow \sum_{(x_i, y_i) \in D} \mathcal{L}(f(x_i;\ \overline{\omega}),\ y_i)$
11     if $loss < min\_loss$ then
12         $min\_loss \leftarrow loss$
13         $\overline{\nabla}_{best} \leftarrow \overline{\nabla}$
14     end
15 end
   Output: The model aggregate $\overline{\nabla}_{best}$
```

Algorithm 2 describes the details of our defense algorithm. First, the server uses the constrained k-means to cluster the received updates into K clusters with each cluster having a minimum $\lfloor \frac{n}{K} \rfloor$ number of updates. Next, for each pair of $(u, v) \in U \times V$ we calculate the aggregate update $\overline{\nabla}$ by performing a two-phase outlier removal (Lines 4-8). Then we set the global model parameters to the new weight $\overline{\omega}$ and compute the loss on the server's validation dataset D. Finally, we iterate through all pairs of (u, v) and return the best model aggregate that has the smallest validation loss.

Although our defense algorithm requires the server to have a labelled validation dataset, our experiment results in Section 7.3 show that we only need a relatively small dataset size and do not require the data to be iid.

6.4 Adaptive Attacks

An attacker can launch a tailored attack against our defense when the attacker knows our aggregation algorithm. In this section, we design an adaptive attack for our defense by modifying our attack in Chapter 5 to tailor the aggregation algorithm.

We assume the attacker knows the server's aggregation algorithm, including the parameters for clustering but does not have access to the server's validation dataset. Since our defense algorithm uses constrained k-means to cluster the updates before removing outliers, the updates will likely be in a single cluster if the attacker submits similar updates for every compromised client. This may decrease the impact of the attack. Therefore, our tailored attack leverages the knowledge of the clustering algorithm and optimizes each malicious update individually. Instead of setting all malicious updates to be the same, the attacker spreads out the malicious updates into multiple clusters based on the server's clustering algorithm.

We consider the tailored attack as an optimization problem like our PGD poisoning attack. The goal of the attacker is to find updates in some malicious direction such that these updates are spread into different clusters after the server performs clustering while simultaneously maximizing the impact on the global model. However, since the search space of malicious updates is extremely large, it is difficult to solve the optimization problem directly. Our approach is to cluster all benign updates first and use a special instance of our PGD poisoning attack to find malicious updates for each cluster that fall within its boundary. Unlike the previous PGD attack, which forces the malicious update to be within the intersection of ℓ_2 balls of all benign updates, we now only require it to be close to the updates in a specific cluster. This imposes less constraints to the malicious updates and thus generates more effective poisoning updates. Since the adversary does not know the optimal number of clusters k' to distribute the malicious updates, we iterate all possible choices $(1, 2, ..., k)$ and select the best malicious update set that has the highest validation loss on the attacker's dataset.

Algorithm 3 shows our adaptive attack in detail. For a specific k', we randomly select k' clusters and use our PGD attack to generate a malicious update for each of them (Line 6-10). Line 11-13 expands the malicious update set $\mathcal{S}$ to size m by duplicating the items

Algorithm 3: Adaptive attack

```
Input: Attacker's dataset $(\mathcal{X}, \mathcal{Y})$, benign updates $\mathcal{B} = \{\nabla_j\}(j \in [m, n))$, current
       model weight $\omega_g$, server's aggregation algorithm $Agg$.
$max\_loss \leftarrow 0$
$\mathcal{S}_{best} \leftarrow \emptyset$
$C_1, ..., C_k = Cluster(\{\nabla_j\})$
for $k' = 1, 2, ..., k$ do
    $\mathcal{S} \leftarrow \emptyset$
    for $i \in [k']$ do
        Randomly select $C_i$ from $C_1, ..., C_k$ without replacement.
        $\nabla_i^m \leftarrow PGD((\mathcal{X}, \mathcal{Y}),\ C_i)$
        Append $\nabla_i^m$ to $\mathcal{S}$
    end
    for $i \in [k', m)$ do
        Append $\mathcal{S}[i \bmod k']$ to $\mathcal{S}$
    end
    $\overline{\nabla} \leftarrow Agg(\mathcal{S} \cup \mathcal{B})$
    $\overline{\omega} \leftarrow \omega_g + \overline{\nabla}$
    $loss \leftarrow \sum_{(x_i, y_i) \in (\mathcal{X}, \mathcal{Y})} \mathcal{L}(f(x_i;\ \overline{\omega}),\ y_i)$
    if $loss > max\_loss$ then
        $max\_loss \leftarrow loss$
        $\mathcal{S}_{best} \leftarrow \mathcal{S}$
    end
end
Output: The malicious update set $\mathcal{S}_{best}$
```

in it. Then, we set the model parameters to the new weight ω based on the aggregate and compute the validation loss on the attacker's dataset (Line 14-16). We update $\mathcal{S}_{best}$ whenever we achieve a higher loss (Line 17-20). Finally, the best malicious update set $\mathcal{S}_{best}$ is returned by the algorithm.

Chapter 7

Evaluations

7.1 Experimental Setup

7.1.1 Datasets

We follow the literature of poisoning attacks in FL [4], [14], [35], [36], [40] and consider three popular image classification datasets MNIST [22], FEMNIST [12] and CIFAR-10 [20].

MNIST: MNIST [22] is a 10-class greyscale digit image classification dataset with 60,000 training images and 10,000 testing images, each of size 28 × 28. We distribute the training data to 100 clients, each receiving 600 samples.

FEMNIST: FEMNIST [12] is an extended version of MNIST dataset, which consists of 814,255 greyscale handwritten character digits. The dataset is generated by collecting handwritten examples from 3400 individuals, and each example is converted to a 28 × 28 pixel image format. There are 62 classes for the dataset: 52 for upper and lower case letters and 10 for digits.

CIFAR-10: CIFAR-10 [20] is a 10-class color image classification dataset consisting of 50,000 training examples and 10,000 testing examples. Each example is of size 32 × 32.

Local dataset partition: In a real-world setting, especially in the cross-device FL case, clients' local training datasets are usually biased and skewed. In this thesis, we simulate the real-world scenario by synthetically generating non-iid FL datasets where the label distribution skew is different across clients, i.e. $P(y_i)$ is different among parties.

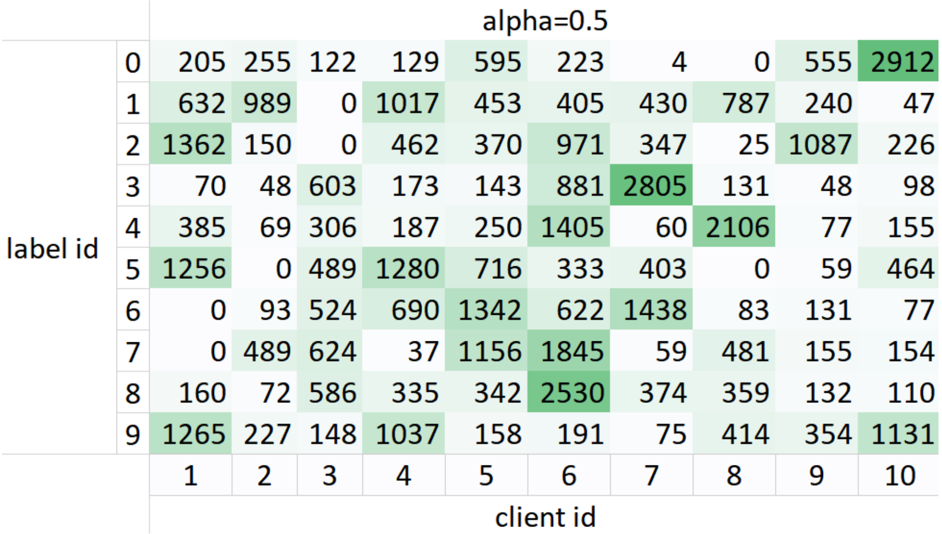
To generate such non-iid datasets, we leverage the Dirichlet distribution [27] to partition the data. This approach has also been widely used in the literature [17], [32], [36], [46]. For each class c, we independently sample $\mathbf{p}_c \sim Dir(\alpha)$ and allocate a fraction $\mathbf{p}_{c,j}$ of examples of class c to client j. Here, α is the concentration parameter of the Dirichlet distribution. We can flexibly change the imbalance level of the dataset by varying α. Decreasing α generates more non-iid datasets. An example of such a partitioning strategy with $\alpha = 0.5$ is shown in Figure 7.1. For small concentration parameters like 0.5, some clients may have no samples for some classes. We note that the amount of data each party owns might also be different using this partitioning method.

We denote the non-iid parameter for the client's local dataset as α_C. We use this approach for FEMNIST and CIFAR-10 dataset with a default $\alpha_C = 1.0$. We explore the effect of other choices of α_C in Section 7.2 and Section 7.3. For MNIST, we observe that for non-iid data generated by this approach ($\alpha_C = 0.5$), there is little accuracy difference from the iid case for the FedAvg [24] algorithm. Therefore, we use a stricter partitioning method as follows.

For MNIST, we follow the partitioning method by McMahan et al. [24] that each client receives digit images corresponding to only l labels. To achieve this, we first randomly assign l different label ids to each client. Then, for all the samples of each label, we partition them equally for the clients that have been assigned to the label. Each client only has examples from l classes in this case. We use this partitioning approach for the MNIST dataset and set $l = 2$.

Attacker and defender's dataset: Our attack performs projected gradient descent on the model weights and therefore requires the attacker to have a labelled dataset. Our intuition is that the quality of the dataset may affect the impact of the attack. An attacker can collect data from elsewhere in order to maximize the impact on the global model. In this thesis, we simply combine the local training data of the P malicious clients that the adversary controls to form the attacker's dataset. Our approach does not require additional effort for the adversary, and our experiment results imply it is sufficient to break existing defenses.

For the defender's validation dataset, by default we use a dataset with the size $S = 100$ for MNIST, FEMNIST and $S = 300$ for CIFAR-10. We study the impact of the size of the validation dataset in Section 7.3. We assume the defender can collect a dataset with a similar distribution as the domain of the learning task. To generate the dataset, by default we uniformly randomly sample data from the combination of the training and testing dataset. We ensure that the clients' local training data, the server's validation data and the testing data do not overlap. We also evaluate the case where the server's validation dataset is biased in terms of label distribution in Section 7.3. To generate non-iid server dataset, we use Dirichlet distribution with different concentration parameters. We sample



alpha=0.5

label id										
0	205	255	122	129	595	223	4	0	555	2912
1	632	989	0	1017	453	405	430	787	240	47
2	1362	150	0	462	370	971	347	25	1087	226
3	70	48	603	173	143	881	2805	131	48	98
4	385	69	306	187	250	1405	60	2106	77	155
5	1256	0	489	1280	716	333	403	0	59	464
6	0	93	524	690	1342	622	1438	83	131	77
7	0	489	624	37	1156	1845	59	481	155	154
8	160	72	586	335	342	2530	374	359	132	110
9	1265	227	148	1037	158	191	75	414	354	1131
	1	2	3	4	5	6	7	8	9	10

client id

Figure 7.1: An example of a non-iid partition for 10 clients on CIFAR-10 dataset according to Dirichlet distribution with $\alpha = 0.5$. Each value represents the number of data samples of a class belonging to a certain client id.

$\mathbf{p} \sim Dir(\alpha_S)$ and assign a fraction $\mathbf{p}_c$ of the server data to be label c. We denote the non-iid parameter for the server's dataset as α_S. Note that $\alpha_S = \infty$ means the dataset is iid, which has an equal amount of data for each class.

7.1.2 FL Parameters and Settings

Model Architectures: We use various types of model architectures on different datasets. Specifically, for MNIST, we train a convolutional neural network (CNN). The architecture of the CNN is shown in Table 7.1. For FEMNIST, we use LeNet [21] architecture. For CIFAR-10, we consider the VGG-11 [38] network. All clients use the same model architecture as the global model.

FL Parameters: We use $N = 100$ clients for the MNIST and CIFAR-10 dataset and randomly select $n = 30$ clients to participate in each FL round. For the larger dataset FEMNIST, we assume $N = 5000$ and select $n = 50$ out of 5000 clients in each round to simulate the cross-device FL setting. By default, we assume the *malicious fraction* $q = 20\%$, but we will also study the impact of other values of q.

We train MNIST and CIFAR-10 with the SGD optimizer with a learning rate initialized to 0.01. We train for a total of $R = 100$ FL epochs and decrease the learning rate after

Table 7.1: The CNN structure for MNIST

Layer	**Size**
Input	$1 \times 28 \times 28$
Conv_1 + Relu	$5 \times 5 \times 10$
Max Pooling	2×2
Conv_2 + Relu	$5 \times 5 \times 20$
Max Pooling	2×2
FC_1 + Relu	50
FC_2	10

Table 7.2: Default FL parameter values for different datasets

Parameter name	MNIST	CIFAR-10	FEMNIST
N (Total # of clients)	100		5000
n (Total # of participants)	30		50
R (# of FL rounds)	100		200
B (Batch size)	10	32	10
q (Malicious fraction)	0.2		
α_C (α for client local data)	1.0		
α_S (α for server data)	∞		
S (Server dataset size)	100	300	100

every 30 epochs. For FEMNIST, we train $R = 200$ FL epochs and set the learning rate to 0.1. We set the local batch size B to 10 for all datasets except for CIFAR-10 where we set it to 32. The number of local epochs is set to 10 for MNIST, CIFAR-10 and 5 for FEMNIST. Table 7.2 lists the FL parameters and the default values that we use unless otherwise mentioned.

7.1.3 Benchmark Attacks and Defenses

We evaluate our attack against multiple defenses from the literature: Krum [7], Multi-krum [7], Trimmed-mean [45], Median [45] and Divide-and-Conquer (DnC) [35]. Details of the defenses can be found in Chapter 3. These defenses have a common parameter which is the estimation of the upper bound of total malicious clients. We set it to $P = N \cdot q$ when evaluating our attack, assuming the defender knows the exact number of the malicious clients. We compare our attack with state-of-the-art *Agg-agnostic* attacks: Min-max and Min-sum. Similar to our attack, these two attacks do not require any knowledge of the aggregation algorithm. We use the default parameter settings for these attacks from the original paper [35].

We consider both fidelity and robustness when evaluating our defense. For fidelity, we compare the testing accuracy of our aggregation algorithm with the FedAvg [24] algorithm under no attack. For robustness, we evaluate our defense against the Min-max, Min-sum attack as well as our new attack. We compare our defense with Krum [7], Multi-krum [7], Trimmed-mean [45], Median [45] and DnC [35]. Moreover, we consider the case where the attacker knows our aggregation algorithm and evaluate our defense against our adaptive attack in Section 6.4. For the clustering algorithm in our defense, we set the number of clusters $k = \lfloor \sqrt{n} \rfloor$ where n is the number of updates selected.

7.1.4 Evaluation Metrics

For untargeted attacks, the attacker's goal is to lower the global model's accuracy on any input data. Therefore, in this thesis we use the difference of the global model testing accuracy before and after attack to measure the *attack impact*.

7.2 Evaluation of Our Attack

7.2.1 Comparison with Other Attacks

In this section, we compare our attack with state-of-the-art model poisoning attacks, Min-max and Min-sum against all defenses in Section 3.1. All three attacks do not require knowledge of the defense algorithm. Table 7.3 shows the results in the case that the benign updates are known to the attacker. From the results, we observe that our attack outperforms Min-max and Min-sum attack in general.

For MNIST, we notice that Min-max performs better than Min-sum in all cases. Our attack has more than 2× as much impact as Min-max for Krum and DnC. For other defenses, our attack causes the global model to drop $\sim 5\%$ more testing accuracy than Min-max.

For FEMNIST, our attack completely destroys the global model for Krum, Multi-krum and Trimmed-mean, i.e. the resulting testing accuracy ($\sim 5.0\%$) is almost the same as random guessing. While Min-max and Min-sum can also break Krum, they are much less impactful for the other two defenses compared to ours. For Median, our attack has a similar impact to Min-max that both drop the testing accuracy by $\sim 11\%$. Min-sum attack only lowers the testing accuracy by 6.6% for Median. The results indicate that DnC seems robust for the FEMNIST dataset. Our attack can only decrease the testing accuracy by 2.0%, while the other attacks have very limited impact on DnC.

For CIFAR-10, our attack's impact is 3× to 5× that of the Min-max and Min-sum attack for Multi-krum, Trimmed-mean and DnC. Our attack is slightly better than Min-max for Median that the impact is 10.8% and 9.3%, respectively. For Krum, our attack lowers the testing accuracy to only 10%.

From the results in Table 7.3, our attack achieves the highest attack impact among all the three evaluated attacks on all datasets. The reason that our attack is superior to Min-max/Min-sum is in two aspects. 1) Our attack moves towards the adversarial direction optimally via gradient ascent while Min-max chooses a perturbation direction heuristically, which might not be optimal. Experiments in [35] show that the perturbation direction affects the impact of the attack significantly. 2) Min-max uses a simple line search to find γ such that the malicious update is close to other benign updates while our attack uses Dykstra's algorithm [8] to project onto the intersection of several Euclidean balls. Moreover, the attacker needs to select a perturbation direction for Min-max/Min-sum attack in advance. In contrast, our attack can be considered as a generic framework and used against any defense algorithms without the need to tune the adversarial direction.

Table 7.3: The testing accuracy of various server aggregation algorithms under three attack settings for three datasets. The values in bold are the *attack impact*. All values are in percentages.

(a) MNIST

	No attack	Min-max	Min-sum	Our attack
Krum	76.5	61.9 **(14.6)**	71.0 **(5.5)**	44.5 **(32.0)**
Multi-krum	94.0	71.1 **(22.9)**	72.9 **(21.1)**	66.3 **(27.7)**
Trimmed-mean	94.6	81.3 **(13.3)**	89.7 **(4.9)**	77.0 **(17.6)**
Median	92.7	81.0 **(11.7)**	82.7 **(10.0)**	76.0 **(16.7)**
DnC	96.9	91.6 **(5.3)**	92.3 **(4.6)**	85.2 **(11.7)**

(b) FEMNIST

	No attack	Min-max	Min-sum	Our attack
Krum	76.1	5.1 **(71.0)**	5.5 **(70.6)**	4.9 **(71.2)**
Multi-krum	81.1	63.5 **(17.6)**	69.0 **(12.1)**	4.9 **(76.2)**
Trimmed-mean	81.0	73.3 **(7.7)**	71.8 **(9.2)**	5.0 **(76.0)**
Median	80.1	69.1 **(11.0)**	73.5 **(6.6)**	69.2 **(10.9)**
DnC	82.2	82.0 **(0.2)**	82.3 **(-0.1)**	80.2 **(2.0)**

(c) CIFAR-10

	No attack	Min-max	Min-sum	Our attack
Krum	54.5	35.4 **(19.1)**	45.6 **(8.9)**	10.0 **(44.5)**
Multi-krum	66.2	60.9 **(5.3)**	62.5 **(3.7)**	40.9 **(25.3)**
Trimmed-mean	69.8	63.5 **(6.3)**	63.3 **(6.5)**	50.0 **(19.8)**
Median	69.1	59.8 **(9.3)**	60.3 **(8.8)**	58.3 **(10.8)**
DnC	70.5	67.7 **(2.8)**	67.1 **(3.4)**	60.1 **(10.4)**

Table 7.4: The impact of our attack in two scenarios: whether the benign updates are known or not.

(a) MNIST

	Known Benign	Unknown Benign
Krum	32.0	48.0
Multi-krum	27.7	28.0
Trimmed-mean	17.6	19.9
Median	16.7	16.2
DnC	11.7	8.6

(b) FEMNIST

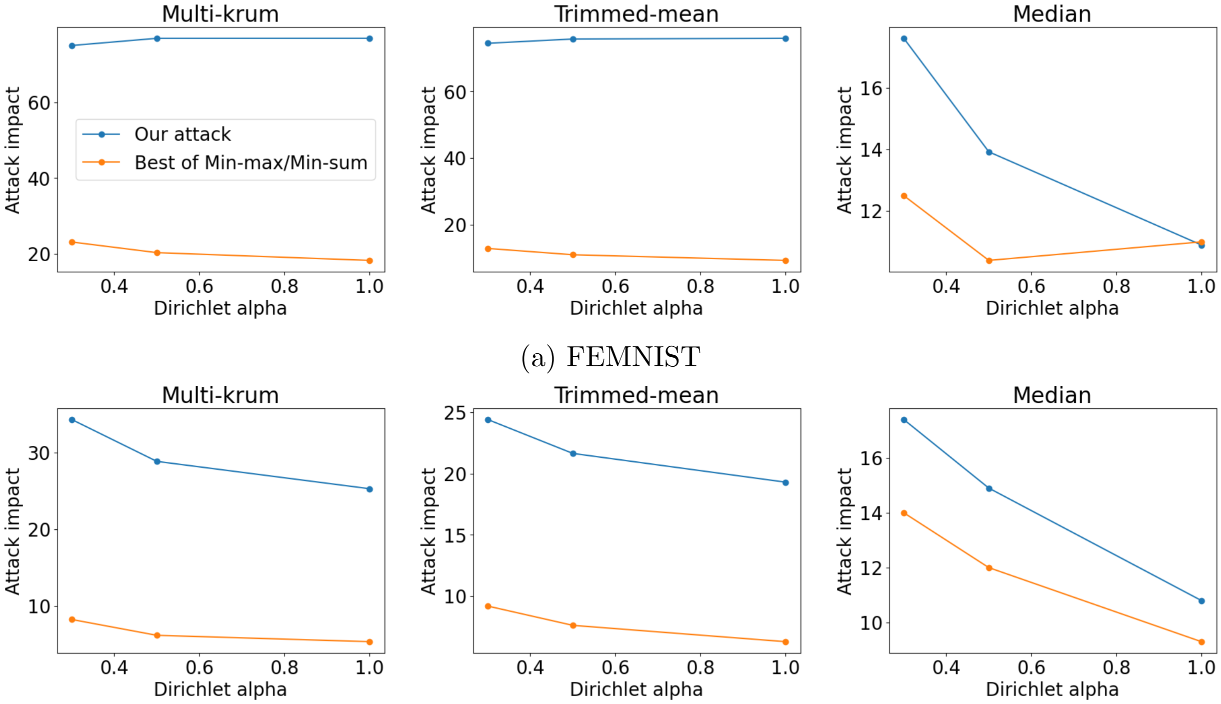
	Known Benign	Unknown Benign
Krum	71.2	71.0
Multi-krum	76.2	76.1
Trimmed-mean	76.0	21.5
Median	10.9	9.7
DnC	2.0	1.2

(c) CIFAR-10

	Known Benign	Unknown Benign
Krum	44.5	44.5
Multi-krum	25.3	16.0
Trimmed-mean	19.8	13.4
Median	10.8	10.0
DnC	10.4	9.7

7.2.2 Unknown Benign Updates

In the case that the benign updates are not available to the attacker, the attacker can train local models on the compromised clients' datasets and compute these updates as an estimation. Table 7.4 shows our attack's impact when the benign updates are unknown. In general, the difference between the cases where the benign updates are known and unknown is little. We note that although we can no longer reduce the testing accuracy close to random guessing for Trimmed-mean on FEMNIST, we are still able to lower the accuracy by 21.5%. Interestingly, for MNIST the unknown benign case has an even higher attack impact for Krum, Multi-krum and Trimmed-mean. We speculate that the reason is that the adversary can still obtain good estimations of benign model updates since the attacker has access to sufficient benign datasets ($N \cdot q = 20$ in this case).



(a) FEMNIST

(b) CIFAR10

Figure 7.2: Effect of α_C under model poisoning attacks.

7.2.3 Effect of the Non-iid Degrees of Local Datasets

We generate client datasets with different non-iid degrees by varying the parameter α_C of the Dirichlet distribution. Figure 7.2 shows the attack impact of our attack and the Min-max/Min-sum attack for different α_C. We note that the impact of all attacks increases as α_C decreases (more non-iid).[1] The reason is that a higher degree of non-iid data leaves more room for the attacker to manipulate the malicious updates. We also note that the gap between our attack and Min-max/Min-sum increases as α_C decreases, indicating our attack becomes more effective when the non-iid degree of the local data increases.

[1] For FEMNIST, our attack reduces the testing accuracy of Multi-krum and Trimmed-mean to random guessing, resulting in little change to it. The small drop is due to the decrease of the testing accuracy with no attack.

Table 7.5: Global model testing accuracy for defense algorithms under no attack ($\alpha_C = 1.0$).

	MNIST	FEMNIST	CIFAR-10
FedAvg	97.6	83.5	72.3
Krum	76.5	76.1	54.5
Multi-krum	94.0	81.1	66.2
Trimmed-mean	94.6	81.0	69.8
Median	92.7	80.1	69.1
DnC	96.9	82.2	70.5
Our defense	97.4	83.0	71.5

7.3 Evaluation of Our Defense

7.3.1 Comparison with Other Defenses

We evaluate our defense on both fidelity and robustness. For fidelity, we compare the defenses with FedAvg [24], a standard aggregation algorithm in non-adversarial FL settings. Table 7.5 shows the global model testing accuracy for multiple defense algorithms when there is no presence of attackers. As shown in the table, our defense has similar testing accuracy to FedAvg ($< 1\%$ difference). However, existing defenses have a much larger testing accuracy drop even if there is no attack. For example, Multi-krum has an accuracy difference of 3.6%, 2.4% and 6.1% for MNIST, FEMNIST and CIFAR-10. For Krum, the accuracy difference is larger than 15% for MNIST and CIFAR-10. Our defense achieves a minimal accuracy loss from FedAvg because we leverage a small validation dataset to remove malicious updates dynamically. In the best case (e.g., non-adversarial setting), our defense might not remove any update since the domain of U and V contains 0.

For robustness, we first consider the setting where the adversary does not have knowledge about the server's aggregation algorithm. We evaluate our defense against our new attack as well as Min-max and Min-sum, which are state-of-the-art *Agg-agnostic* attacks proposed by Shejwalkar et al. [35]. Table 7.6 shows the comparison between our defense and others. We experiment with two choices of α_C for FEMNIST and CIFAR-10. Our defense achieves robustness and has the smallest attack impact among all defenses under all attacks. Specifically, the highest impact on our defense is caused by our attack, which is 1.4%, 2.1% and 2.3% for MNIST, FEMNIST and CIFAR-10 ($\alpha_C = 0.5$). We note that the attack impact on the defense becomes larger in general when α_C decreases. For $\alpha_C = 0.5$, our defense achieves a low accuracy loss of 2% while most other defenses suffer more than 10% of accuracy loss. We remark that although DnC also seems robust for FEMNIST

Table 7.6: The attack impact of various defenses under our attack and Shejwalkar et al.'s attacks [35]. The higher attack impact between Min-max and Min-sum is shown. The number in the brackets is the impact of our adaptive attack tailored to our defense.

(a) MNIST

	Shejwalkar et al.'s	Our attack
Krum	14.6	32.0
Multi-krum	22.9	27.7
Trim-mean	13.3	17.6
Median	11.7	16.7
DnC	5.3	11.7
Our defense	0.5	**1.4 (1.8)**

(b) FEMNIST ($\alpha_C = 1.0$)

	Shejwalkar et al.'s	Our attack
Krum	71.0	71.2
Multi-krum	17.6	76.2
Trim-mean	9.2	76.0
Median	11.0	10.9
DnC	0.2	2.0
Our defense	0.3	**0.9 (1.5)**

(c) FEMNIST ($\alpha_C = 0.5$)

	Shejwalkar et al.'s	Our attack
Krum	70.0	71.2
Multi-krum	20.4	77.0
Trim-mean	9.5	76.1
Median	12.1	13.5
DnC	0.2	3.5
Our defense	1.2	**2.1 (2.7)**

(d) CIFAR-10 ($\alpha_C = 1.0$)

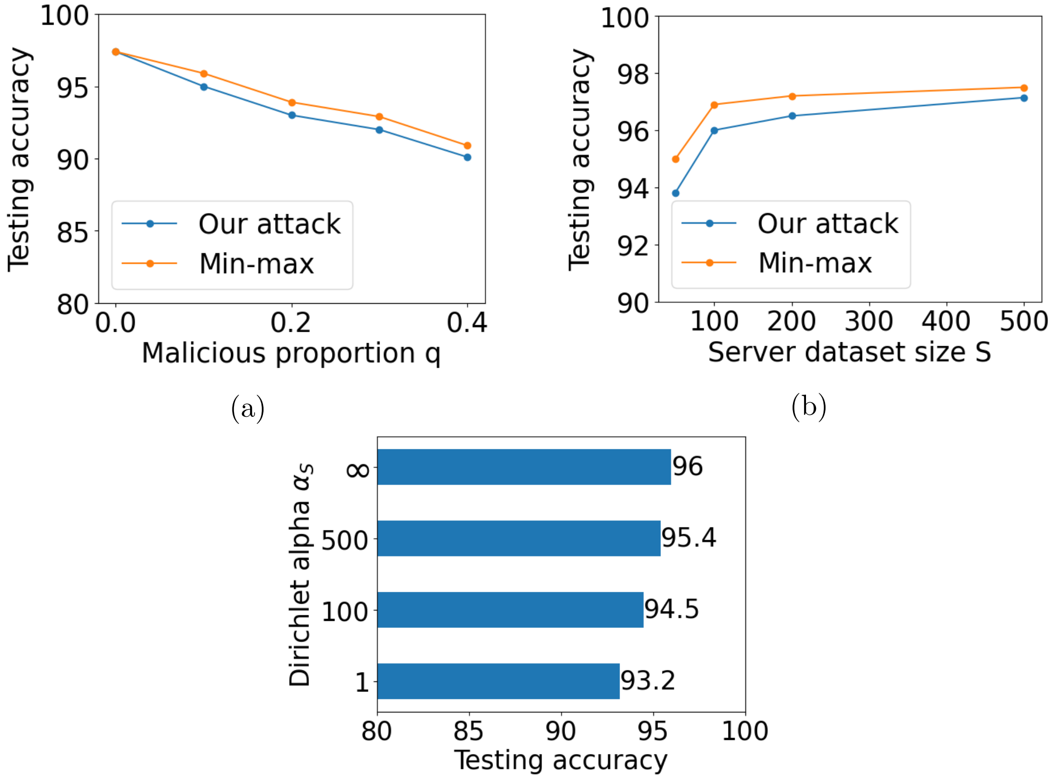
	Shejwalkar et al.'s	Our attack
Krum	19.1	44.5
Multi-krum	5.3	25.3
Trim-mean	6.5	19.8
Median	9.3	10.8
DnC	3.4	10.4
Our defense	0.5	**2.0 (2.5)**

(e) CIFAR-10 ($\alpha_C = 0.5$)

	Shejwalkar et al.'s	Our attack
Krum	23.1	50.4
Multi-krum	5.5	27.0
Trim-mean	8.6	23.0
Median	9.9	11.1
DnC	4.9	11.5
Our defense	1.1	**2.3 (2.9)**

that the impact of our attack is only 3.5% when $\alpha_C = 0.5$, DnC suffers a more than 10% accuracy drop under our attack for the other two datasets, indicating the robustness of DnC might be dataset-dependent.

We then consider the aggregation algorithm is known to the attacker and investigate the robustness of our defense against our adaptive attack. The accuracy impacts on our aggregation algorithm are shown in Table 7.6. Due to a more powerful adversary, we observe that there is a slight increase in the attack impact compared to the previous case. Nevertheless, the accuracy impact on our defense for three datasets is less than 3%, which is much smaller than any other defenses for an even weaker adversary (non-adaptive).



(a)

(b)

(c)

Figure 7.3: Impact of FL parameters on testing accuracy under attacks for MNIST.

7.3.2 Impact of FL Parameters

Impact of malicious fraction q: Figure 7.3a shows the accuracy of our aggregation algorithm under our attack and the Min-max attack for various values of q. The testing accuracy for both attacks decreases as the number of malicious clients increases. We observe that under existing attacks and our new attack, our defense can still achieve $> 90\%$ testing accuracy on the MNIST dataset when 40% of the clients are malicious.

Impact of server dataset size S: Figure 7.3b shows the testing accuracy under attack for different sizes of the server's validation dataset. The testing accuracy under both attacks increases as the server's validation dataset becomes larger. We observe that when we increase the server dataset to 500, we achieve a testing accuracy under attack at 97% similar to FedAvg. Furthermore, we notice that a dataset with 100 data samples is sufficient to defend against strong model poisoning attacks. Specifically, the testing accuracy increase from the size of 50 to 100 is relatively high (around 3%), while the accuracy does not increase much when the server dataset size grows beyond 100.

Impact of distribution of server data α_S: We investigated the case where the server validation dataset is non-iid. We use the partitioning strategy stated in Section 7.1. Figure 7.3c shows the testing accuracy of our defense under our new attack for different α_S. For $\alpha_S = 1$, the server dataset barely contains examples for some classes. In this extreme case, our defense can still achieve a reasonable high accuracy (93.2%) against a strong model poisoning attack.

Chapter 8

Related Work

In this section, we briefly overview other poisoning attacks and defenses in the literature.

Targeted poisoning attacks

Next to the state-of-the-art untargeted model poisoning attacks introduced in Section 3.2, we focus on discussing existing targeted poisoning attacks in this section. Bhagoji et al. [6] present a poisoning attack that changes the prediction labels of a particular sample by using alternating minimization. Bagdasaryan et al. [3] and Sun et al. [39] demonstrate a constrain-and-scale attack that inserts a semantic backdoor into the model trained with FedAvg. Tolpegin et al. [40] studied targeted data poisoning attacks that malicious updates are computed by training on label-flipped dataset on the compromised clients. Xie et al. [42] propose a targeted attack that decomposes the global trigger pattern into multiple local patterns and embeds them into the training set of the malicious clients. Their evaluation shows that the poisoning updates generated in this way are more effective and stealthy.

Other defenses

Fang et al. [14] remove a fixed number of updates that have a large impact on the global model. For each update, the defense computes the global models with and without the update and calculates the error rate difference on a validation dataset. This method is computationally-expensive and is impractical when the number of participants is large. FLTrust [10] also assumes the server has access to a clean validation dataset and assigns a cosine similarity score to each client in each round. Then the server returns a weighted mean of updates based on the score. Since the score is computed by comparing each update with the ground-truth update from the server, it requires a high-quality server dataset. Andreina et al. [2] propose a feedback-based algorithm where in each iteration the server distributes the current aggregated model to a set of clients. Then the selected clients report a verdict indicating a clean or malicious model based on their local data and the history of previously accepted models. The decision is made by monitoring the variations in error rates made by accepted and current models. However, this approach may not work if the data of a fraction of training clients differ from other validation clients as in the non-iid scenario. Li et al. [23] use an encoder-decoder model for anomaly detection. The idea is that after removing noisy features, the embeddings of normal and malicious model updates can be differentiated in low-dimensional latent space. Updates that have a reconstruction error larger than a threshold are removed. Munoz et al. [29] remove a model update if its cosine distance to the aggregated update deviates too much from the median distance. This approach has been shown that suffers a high false-positive rate [33]. TESSERACT [34] assigns a flip-score to all participating clients. The score measures the magnitude of gradients that experience a direction change from the previous global update. The intuition is that a large number of gradients do not flip their directions with large magnitudes in a benign setting. Differential Privacy (DP) [13] can be applied to bound each participant's influence over the joint model. McMahan et al. [25] first apply DP to FL by clipping the model updates and adding Gaussian noises. However, Bagdasaryan et al. [3] show that this approach only mitigates backdoor attacks at the cost of relatively large model accuracy loss.

There are other defenses involving clustering algorithms that aim to mitigate targeted and backdoor attacks. Auror [37] uses k-means to determine indicative features by dividing all clients into two clusters for each parameter of the model. A feature is labelled as indicative if the distance between the cluster centers exceeds a certain threshold. A client model that has too many indicative features is rejected by the algorithm. FLAME [30] is a defense algorithm proposed by Nguyen et al. to mitigate backdoor attacks. FLAME uses an outlier-based clustering to remove potential adversarial updates combined with weight clipping and noise-adding to limit the attacker's impact. FLAME leverages HDBSCAN clustering algorithm [9] which labels updates as outliers if they do not fit into any cluster. However, simply relying on HDBSCAN to remove outliers may not work in the non-iid case since many benign updates might be labelled as outliers and removed by the algorithm. Deepsight [33] defines *Threshold Exceeding* based on the homogeneity of the training data and uses it to label each model update as benign or suspicious. Then it clusters all updates and removes an entire cluster if more than 1/3 of the updates in the cluster are suspicious. This approach relies on the assumption that all updates in the same cluster are trained from similar iid training data and, therefore, receive the same label. Furthermore, the method discards the entire cluster even if only 1/3 of the updates in it are malicious. Therefore, the attacker may exploit this and make the server remove more benign updates by carefully distributing the malicious updates to multiple clusters.

Chapter 9

Conclusion

In this work, we proposed a new untargeted poisoning attack that can be used to assess the robustness of FL aggregation algorithms. Our attack does not require knowledge of the defense algorithm and can be used against any FL system. The evaluation shows that our attack achieves a much larger attack impact compared to previous attacks. Furthermore, we presented a clustering-based defense algorithm that leverages a small dataset to remove malicious updates dynamically. Our defense addresses common weaknesses of existing Byzantine-robust algorithms and achieves robustness under strong attacks.

References

[1] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[2] Sébastien Andreina, Giorgia Azzurra Marson, Helen Möllering, and Ghassan Karame. Baffle: Backdoor detection via feedback-based federated learning. *CoRR*, abs/2011.02167, 2020.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *CoRR*, abs/1807.00459, 2018.

[4] Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[5] K.P. Bennett, P.S. Bradley, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, May 2000.

[6] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 634–643. PMLR, 09–15 Jun 2019.

[7] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 118–128, Red Hook, NY, USA, 2017. Curran Associates Inc.

[8] James P. Boyle and Richard L. Dykstra. A method for finding projections onto the intersection of convex sets in hilbert spaces. 1986.

[9] Ricardo Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. volume 7819, pages 160–172, 04 2013.

[10] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, 2021.

[11] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. Draco: Byzantine-resilient distributed training via redundant gradients, 2018.

[12] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926, 2017.

[13] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[14] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. *Local Model Poisoning Attacks to Byzantine-Robust Federated Learning.* USENIX Association, USA, 2020.

[15] Filip Granqvist, Matt Seigel, Rogier van Dalen, Áine Cahill, Stephen Shum, and Matthias Paulik. Improving on-device speaker verification using federated learning with privacy. *arXiv preprint arXiv:2008.02651*, 2020.

[16] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018.

[17] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *CoRR*, abs/1909.06335, 2019.

[18] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. *CoRR*, abs/1804.00308, 2018.

[19] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *CoRR*, abs/1610.05492, 2016.

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[22] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[23] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. Learning to detect malicious clients for robust federated learning. *CoRR*, abs/2002.00211, 2020.

[24] H. B. McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[25] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[26] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The Hidden Vulnerability of Distributed Learning in Byzantium. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3518–3527. PMLR, 2018.

[27] Thomas P. Minka. Estimating a dirichlet distribution. Technical report, 2000.

[28] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C. Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. *CoRR*, abs/1708.08689, 2017.

[29] Luis Muñoz-González, Kenneth T. Co, and Emil C. Lupu. Byzantine-robust federated machine learning through adaptive model averaging, 2019.

[30] Thien Duc Nguyen, Phillip Rieger, Huili Chen, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Shaza Zeitouni, Farinaz Koushanfar, Ahmad-Reza Sadeghi, and Thomas Schneider. Flame: Taming backdoors in federated learning, 2021.

[31] Thien Duc Nguyen, Phillip Rieger, Hossein Yalame, Helen Möllering, Hossein Fereidooni, Samuel Marchal, Markus Miettinen, Azalia Mirhoseini, Ahmad-Reza Sadeghi, T. Schneider, and Shaza Zeitouni. Flguard: Secure and private federated learning. *IACR Cryptol. ePrint Arch.*, 2021:25, 2021.

[32] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[33] Phillip Rieger, Thien Duc Nguyen, Markus Miettinen, and Ahmad-Reza Sadeghi. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. *CoRR*, abs/2201.00763, 2022.

[34] Atul Sharma, Wei Chen, Joshua Zhao, Qiang Qiu, Somali Chaterji, and Saurabh Bagchi. Tesseract: Gradient flip score to secure federated learning against model poisoning attacks, 2021.

[35] Virat Shejwalkar and A. Houmansadr. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *NDSS*, 2021.

[36] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on federated learning. *CoRR*, abs/2108.10241, 2021.

[37] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, ACSAC '16, page 508–519, New York, NY, USA, 2016. Association for Computing Machinery.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[39] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. Can you really backdoor federated learning?, 2019.

[40] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data Poisoning Attacks Against Federated Learning Systems. *arXiv e-prints*, page arXiv:2007.08432, July 2020.

[41] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. *CoRR*, abs/1811.00636, 2018.

[42] Chulin Xie, Keli Huang, Pin-Yu Chen, and Bo Li. Dba: Distributed backdoor attacks against federated learning. In *International Conference on Learning Representations*, 2020.

[43] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd, 2018.

[44] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions, 2018.

[45] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5650–5659. PMLR, 10–15 Jul 2018.

[46] Mikhail Yurochkin, Mayank Agarwal, Soumya Shubhra Ghosh, Kristjan H. Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *ICML*, 2019.