Local Model Poisoning Attacks to Byzantine-Robust Federated Learning

Minghong Fang*[1], Xiaoyu Cao*[2], Jinyuan Jia[2], Neil Zhenqiang Gong[2]
[1]*CS Department, Iowa State University,* [2]*ECE Department, Duke University*
[1]*myfang@iastate.edu,* [2]*{xiaoyu.cao, jinyuan.jia, neil.gong}@duke.edu*

Abstract

In federated learning, multiple client devices jointly learn a machine learning model: each client device maintains a local model for its local training dataset, while a master device maintains a global model via aggregating the local models from the client devices. The machine learning community recently proposed several federated learning methods that were claimed to be robust against Byzantine failures (e.g., system failures, adversarial manipulations) of certain client devices. In this work, we perform the first systematic study on *local model poisoning attacks* to federated learning. We assume an attacker has compromised some client devices, and the attacker manipulates the local model parameters on the compromised client devices during the learning process such that the global model has a large testing error rate. We formulate our attacks as optimization problems and apply our attacks to four recent Byzantine-robust federated learning methods. Our empirical results on four real-world datasets show that our attacks can substantially increase the error rates of the models learnt by the federated learning methods that were claimed to be robust against Byzantine failures of some client devices. We generalize two defenses for data poisoning attacks to defend against our local model poisoning attacks. Our evaluation results show that one defense can effectively defend against our attacks in some cases, but the defenses are not effective enough in other cases, highlighting the need for new defenses against our local model poisoning attacks to federated learning.

1 Introduction

Byzantine-robust federated learning: In *federated learning* (also known as *collaborative learning*) [32, 39], the training dataset is decentralized among multiple client devices (e.g., desktops, mobile phones, IoT devices), which could belong to different users or organizations. These users/organizations do not want to share their local training datasets, but still desire to jointly learn a model. For instance, multiple hospitals may desire to learn a healthcare model without sharing their sensitive data to each other. Each client device (called *worker device*) maintains a *local model* for its local training dataset. Moreover, the service provider has a *master device* (e.g., cloud server), which maintains a *global model*. Roughly speaking, federated learning repeatedly performs three steps: the master device sends the current global model to worker devices; worker devices update their local models using their local training datasets and the global model, and send the local models to the master device; and the master device computes a new global model via aggregating the local models according to a certain *aggregation rule*.

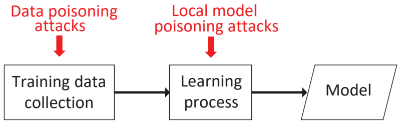


Figure 1: *Data* vs. *local model* poisoning attacks.

For instance, the *mean* aggregation rule that takes the average of the local model parameters as the global model is widely used under non-adversarial settings. However, the global model can be arbitrarily manipulated for mean even if just one worker device is compromised [9, 66]. Therefore, the machine learning community recently proposed multiple aggregation rules (e.g., Krum [9], Bulyan [42], trimmed mean [66], and median [66]), which aimed to be robust against Byzantine failures of certain worker devices.

Existing data poisoning attacks are insufficient: We consider attacks that aim to manipulate the *training phase* of machine learning such that the learnt model (we consider the model to be a classifier) has a high testing error rate indiscriminately for testing examples, which makes the model unusable and eventually leads to denial-of-service attacks. Figure 1 shows the training phase, which includes two components, i.e., *training dataset collection* and *learning process*. The training dataset collection component is to collect a training dataset, while the learning process component produces a model from a given training dataset. Existing attacks mainly

*Equal contribution. Minghong Fang performed this research when he was under the supervision of Neil Zhenqiang Gong.

inject malicious data into the training dataset before the learning process starts, while the learning process is assumed to maintain integrity. Therefore, these attacks are often called *data poisoning attacks* [8, 30, 33, 50, 56, 62]. In federated learning, an attacker could only inject the malicious data into the worker devices that are under the attacker's control. As a result, these data poisoning attacks have limited success to attack Byzantine-robust federated learning (see our experimental results in Section 4.4).

Our work: We perform the first study on *local model poisoning attacks* to Byzantine-robust federated learning. Existing studies [9, 66] only showed local model poisoning attacks to federated learning with the non-robust mean aggregation rule.

Threat model. Unlike existing data poisoning attacks that compromise the integrity of training dataset collection, we aim to compromise the integrity of the learning process in the training phase (see Figure 1). We assume the attacker has control of some worker devices and manipulates the local model parameters sent from these devices to the master device during the learning process. The attacker may or may not know the aggregation rule used by the master device. To contrast with data poisoning attacks, we call our attacks local model poisoning attacks as they directly manipulate the local model parameters.

Local model poisoning attacks. A key challenge of local model poisoning attacks is how to craft the local models sent from the compromised worker devices to the master device. To address this challenge, we formulate crafting local models as solving an optimization problem in each iteration of federated learning. Specifically, the master device could compute a global model in an iteration if there are no attacks, which we call *before-attack* global model. Our goal is to craft the local models on the compromised worker devices such that the global model deviates the most towards the inverse of the direction along which the before-attack global model would change. Our intuition is that the deviations accumulated over multiple iterations would make the learnt global model differ from the before-attack one significantly. We apply our attacks to four recent Byzantine-robust federated learning methods including Krum, Bulyan, trimmed mean, and median.

Our evaluation results on the MNIST, Fashion-MNIST, CH-MNIST, and Breast Cancer Wisconsin (Diagnostic) datasets show that our attacks can substantially increase the error rates of the global models under various settings of federated learning. For instance, when learning a deep neural network classifier for MNIST using Krum, our attack can increase the error rate from 0.11 to 0.75. Moreover, we compare with data poisoning attacks including *label flipping attacks* and *back-gradient optimization based attacks* [43] (state-of-the-art untargeted data poisoning attacks for multi-class classifiers), which poison the local training datasets on the compromised worker devices. We find that these data poisoning attacks have limited success to attack the Byzantine-robust federated learning methods.

Defenses. Existing defenses against data poisoning attacks essentially aim to sanitize the training dataset. One category of defenses [4, 15, 56, 59] detects malicious data based on their negative impact on the error rate of the learnt model. For instance, *Reject on Negative Impact (RONI)* [4] measures the impact of each training example on the error rate of the learnt model and removes the training examples that have large negative impact. Another category of defenses [20, 30, 35] leverages new loss functions, solving which detects malicious data and learns a model simultaneously. For instance, Jagielski et al. [30] proposed TRIM, which aims to jointly find a subset of training dataset with a given size and model parameters that minimize the loss function. The training examples that are not in the selected subset are treated as malicious data. However, these defenses are not directly applicable for our local model poisoning attacks because our attacks do not inject malicious data into the training dataset.

To address the challenge, we generalize RONI and TRIM to defend against our local model poisoning attacks. Both defenses remove the local models that are potentially malicious before computing the global model using a Byzantine-robust aggregation rule in each iteration. One defense removes the local models that have large negative impact on the error rate of the global model (inspired by RONI that removes training examples that have large negative impact on the error rate of the model), while the other defense removes the local models that result in large loss (inspired by TRIM that removes the training examples that have large negative impact on the loss), where the error rate and loss are evaluated on a validation dataset. We call the two defenses *Error Rate based Rejection (ERR)* and *Loss Function based Rejection (LFR)*, respectively. Moreover, we combine ERR and LFR, i.e., we remove the local models that are removed by either ERR or LFR. Our empirical evaluation results show that LFR outperforms ERR; and the combined defense is comparable to LFR in most cases. Moreover, LFR can defend against our attacks in certain cases, but LFR is not effective enough in other cases. For instance, LFR can effectively defend against our attacks that craft local models based on the trimmed mean aggregation rule, but LFR is not effective against our attacks that are based on the Krum aggregation rule. Our results show that we need new defense mechanisms to defend against our local model poisoning attacks.

Our key contributions can be summarized as follows:

- We perform the first systematic study on attacking Byzantine-robust federated learning.
- We propose *local model poisoning attacks* to Byzantine-robust federated learning. Our attacks manipulate the local model parameters on compromised worker devices during the learning process.
- We generalize two defenses for data poisoning attacks to defend against local model poisoning attacks. Our results show that, although one of them is effective in some cases, they have limited success in other cases.

2 Background and Problem Formulation

2.1 Federated Learning

Suppose we have m worker devices and the ith worker device has a local training dataset D_i. The worker devices aim to collaboratively learn a classifier. Specifically, the model parameters $\mathbf{w}$ of the classifier are often obtained via solving the following optimization problem: $\min_{\mathbf{w}} \sum_{i=1}^{m} F(\mathbf{w}, D_i)$, where $F(\mathbf{w}, D_i)$ is the objective function for the local training dataset on the ith device and characterizes how well the parameters $\mathbf{w}$ model the local training dataset on the ith device. Different classifiers (e.g., logistic regression, deep neural networks) use different objective functions. In federated learning, each worker device maintains a local model for its local training dataset. Moreover, we have a master device to maintain a global model via aggregating local models from the m worker devices. Specifically, federated learning performs the following three steps in each iteration:

Step I. The master device sends the current global model parameters to all worker devices.

Step II. The worker devices update their local model parameters using the current global model parameters and their local training datasets in parallel. In particular, the ith worker device essentially aims to solve the optimization problem $\min_{\mathbf{w}_i} F(\mathbf{w}_i, D_i)$ with the global model parameters $\mathbf{w}$ as an initialization of the local model parameters $\mathbf{w}_i$. A worker device could use any method to solve the optimization problem, though *stochastic gradient descent* is the most popular one. Specifically, the ith worker device updates its local model parameters $\mathbf{w}_i$ as $\mathbf{w}_i = \mathbf{w} - \alpha \cdot \frac{\partial F(\mathbf{w}, B_i)}{\partial \mathbf{w}}$, where α is the learning rate and B_i is a randomly sampled batch from the local training dataset D_i. Note that a worker device could apply stochastic gradient descent multiple rounds to update its local model. After updating the local models, the worker devices send them to the master device.

Step III. The master device aggregates the local models from the worker devices to obtain a new global model according to a certain *aggregation rule*. Formally, we have $\mathbf{w} = \mathcal{A}(\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m)$.

The master device could also randomly pick a subset of worker devices and send the global model to them; the picked worker devices update their local models and send them to the master device; and the master device aggregates the local models to obtain the new global model [39]. We note that, for the aggregation rules we study in this paper, sending local models to the master device is equivalent to sending gradients to the master device, who aggregates the gradients and uses them to update the global model.

2.2 Byzantine-robust Aggregation Rules

A naive aggregation rule is to average the local model parameters as the global model parameters. This *mean* aggregation rule is widely used under non-adversarial settings [16, 32, 39]. However, mean is not robust under adversarial settings. In particular, an attacker can manipulate the global model parameters arbitrarily for this mean aggregation rule when compromising only one worker device [9, 66]. Therefore, the machine learning community has recently developed multiple aggregation rules that aim to be robust even if certain worker devices exhibit Byzantine failures. Next, we review several such aggregation rules.

Krum [9] and Bulyan [42]: Krum selects one of the m local models that is similar to other models as the global model. The intuition is that even if the selected local model is from a compromised worker device, its impact may be constrained since it is similar to other local models possibly from benign worker devices. Suppose at most c worker devices are compromised. For each local model $\mathbf{w}_i$, the master device computes the $m-c-2$ local models that are the closest to $\mathbf{w}_i$ with respect to Euclidean distance. Moreover, the master device computes the sum of the distances between $\mathbf{w}_i$ and its closest $m-c-2$ local models. Krum selects the local model with the smallest sum of distance as the global model. When $c < \frac{m-2}{2}$, Krum has theoretical guarantees for the convergence for certain objective functions.

Euclidean distance between two local models could be substantially influenced by a single model parameter. Therefore, Krum could be influenced by some abnormal model parameters [42]. To address this issue, Mhamdi et al. [42] proposed Bulyan, which essentially combines Krum and a variant of trimmed mean (trimmed mean will be discussed next). Specifically, Bulyan first iteratively applies Krum to select θ ($\theta \leq m-2c$) local models. Then, Bulyan uses a variant of trimmed mean to aggregate the θ local models. In particular, for each jth model parameter, Bulyan sorts the jth parameters of the θ local models, finds the γ ($\gamma \leq \theta - 2c$) parameters that are the closest to the median, and computes their mean as the jth parameter of the global model. When $c \leq \frac{m-3}{4}$, Bulyan has theoretical guarantees for the convergence under certain assumptions of the objective function.

Since Bulyan is based on Krum, our attacks for Krum can transfer to Bulyan (see Appendix A). Moreover, Bulyan is not scalable because it executes Krum many times in each iteration and Krum computes pairwise distances between local models. Therefore, we will focus on Krum in the paper.

Trimmed mean [66]: This aggregation rule aggregates each model parameter independently. Specifically, for each jth model parameter, the master device sorts the jth parameters of the m local models, i.e., $w_{1j}, w_{2j}, \cdots, w_{mj}$, where w_{ij} is the jth parameter of the ith local model, removes the largest and smallest β of them, and computes the mean of the remaining $m-2\beta$ parameters as the jth parameter of the global model. Suppose at most c worker devices are compromised. This trimmed mean aggregation rule achieves *order-optimal* error rate when $c \leq \beta < \frac{m}{2}$ and the objective function to be minimized is strongly convex. Specifically, the order-optimal error rate is $\tilde{O}(\frac{c}{m\sqrt{n}} + \frac{1}{\sqrt{mn}})$,[1] where n is the number of training data points on a worker device (worker devices are assumed to have the same number of training data points).

Median [66]: In this median aggregation rule, for each jth model parameter, the master device sorts the jth parameters of the m local models and takes the median as the jth parameter of the global model. Note that when m is an even number, median is the mean of the middle two parameters. Like the trimmed mean aggregation rule, the median aggregation rule also achieves an order-optimal error rate when the objective function is strongly convex.

2.3 Problem Definition and Threat Model

Attacker's goal: Like many studies on poisoning attacks [7, 8, 30, 33, 50, 62, 65], we consider an attacker's goal is to manipulate the learnt global model such that it has a high error rate indiscriminately for testing examples. Such attacks are known as *untargeted poisoning attacks*, which make the learnt model unusable and eventually lead to denial-of-service attacks. For instance, an attacker may perform such attacks to its competitor's federated learning system. Some studies also considered other types of poisoning attacks (e.g., *targeted poisoning attacks* [56]), which we will review in Section 6.

We note that the Byzantine-robust aggregation rules discussed above can *asymptotically* bound the error rates of the learnt global model under certain assumptions of the objective functions, and some of them (i.e., trimmed mean and median) even achieve *order-optimal* error rates. These theoretical guarantees seem to imply the difficulty of manipulating the error rates. However, the asymptotic guarantees do not precisely characterize the *practical* performance of the learnt models. Specifically, the asymptotic error rates are quantified using the $\tilde{O}$ notation. The $\tilde{O}$ notation ignores any constant, e.g., $\tilde{O}(\frac{1}{\sqrt{n}})$=$\tilde{O}(\frac{100}{\sqrt{n}})$. However, such constant significantly influences a model's error rate in practice. As we will show, although these asymptotic error rates still hold for our local model poisoning attacks since they hold for Byzantine failures, our attacks can still significantly increase the testing error rates of the learnt models in practice.

Attacker's capability: We assume the attacker has control of c worker devices. Specifically, like Sybil attacks [17] to distributed systems, the attacker could inject c fake worker devices into the federated learning system or compromise c benign worker devices. However, we assume the number of worker devices under the attacker's control is less than 50% (otherwise, it would be easy to manipulate the global models). We assume the attacker can arbitrarily manipulate the local models sent from these worker devices to the master device. For simplicity, we call these worker devices *compromised worker devices* no matter whether they are fake devices or compromised benign ones.

Attacker's background knowledge: The attacker knows the code, local training datasets, and local models on the compromised worker devices. We characterize the attacker's background knowledge along the following two dimensions:

Aggregation rule. We consider two scenarios depending on whether the attacker knows the aggregation rule or not. In particular, the attacker could know the aggregation rule in various scenarios. For instance, the service provider may make the aggregation rule public in order to increase transparency and trust of the federated learning system [39]. When the attacker does not know the aggregation rule, we will craft local model parameters for the compromised worker devices based on a certain aggregation rule. Our empirical results show that such crafted local models could also attack other aggregation rules. In particular, we observe different levels of *transferability* of our local model poisoning attacks between different aggregation rules.

Training data. We consider two cases (*full knowledge* and *partial knowledge*) depending on whether the attacker knows the local training datasets and local models on the benign worker devices. In the full knowledge scenario, the attacker knows the local training dataset and local model on every worker device. We note that the full knowledge scenario has limited applicability in practice for federated learning as the training dataset is decentralized on many worker devices, and we use it to estimate the *upper bound* of our attacks' threats for a given setting of federated learning. In the partial knowledge scenario, the attacker only knows the local training datasets and local models on the compromised worker devices.

Our threat model is inspired by multiple existing studies [30, 47, 48, 56] on adversarial machine learning. For instance, Suciu et al. [56] recently proposed to characterize an attacker's background knowledge and capability for data poisoning attacks with respect to multiple dimensions such as *Feature*, *Algorithm*, and *Instance*. Our aggregation rule and training data dimensions are essentially the Algorithm and Instance dimensions, respectively. We do not consider the Feature dimension because the attacker controls some worker devices and already knows the features in our setting.

Some Byzantine-robust aggregation rules (e.g., Krum [9] and trimmed mean [66]) need to know the upper bound of the number of compromised worker devices in order to set parameters appropriately. For instance, trimmed mean removes the largest and smallest β local model parameters, where β is at least the number of compromised worker devices (otherwise trimmed mean can be easily manipulated). To calculate a lower bound for our attack's threat, we consider a hypothetical, strong service provider who knows the number of compromised worker devices and sets parameters in the aggregation rule accordingly.

[1] $\tilde{O}$ is a variant of the O notation, which ignores the logarithmic terms.

3 Our Local Model Poisoning Attacks

We focus on the case where the aggregation rule is known. When the aggregation rule is unknown, we craft local models based on an assumed one. Our empirical results in Section 4.3 show that our attacks have different levels of transferability between aggregation rules.

3.1 Optimization Problem

Our idea is to manipulate the global model via carefully crafting the local models sent from the compromised worker devices to the master device in each iteration of federated learning. We denote by s_j the changing direction of the jth global model parameter in the current iteration when there are no attacks, where $s_j = 1$ or -1. $s_j = 1$ (or $s_j = -1$) means that the jth global model parameter increases (or decreases) upon the previous iteration. We consider the attacker's goal (we call it *directed deviation goal*) is to deviate a global model parameter the most towards the inverse of the direction along which the global model parameter would change without attacks. Suppose in an iteration, $\mathbf{w}_i$ is the local model that the ith worker device intends to send to the master device when there are no attacks. Without loss of generality, we assume the first c worker devices are compromised. Our directed deviation goal is to craft local models $\mathbf{w}'_1, \mathbf{w}'_2, \cdots, \mathbf{w}'_c$ for the compromised worker devices via solving the following optimization problem in each iteration:

$$\begin{aligned}
&\max_{\mathbf{w}'_1,\cdots,\mathbf{w}'_c} \mathbf{s}^T(\mathbf{w}-\mathbf{w}'),\\
\text{subject to } &\mathbf{w} = \mathcal{A}(\mathbf{w}_1,\cdots,\mathbf{w}_c,\mathbf{w}_{c+1},\cdots,\mathbf{w}_m),\\
&\mathbf{w}' = \mathcal{A}(\mathbf{w}'_1,\cdots,\mathbf{w}'_c,\mathbf{w}_{c+1},\cdots,\mathbf{w}_m), \qquad (1)
\end{aligned}$$

where $\mathbf{s}$ is a column vector of the changing directions of all global model parameters, $\mathbf{w}$ is the before-attack global model, and $\mathbf{w}'$ is the after-attack global model. Note that $\mathbf{s}$, $\mathbf{w}$, and $\mathbf{w}'$ all depend on the iteration number. Since our attacks manipulate the local models in each iteration, we omit the explicit dependency on the iteration number for simplicity.

In our preliminary exploration of formulating poisoning attacks, we also considered a *deviation goal*, which does not consider the global model parameters' changing directions. We empirically find that our attacks based on both the directed deviation goal and the deviation goal achieve high testing error rates for Krum. However, the directed deviation goal substantially outperforms the deviation goal for trimmed mean and median aggregation rules. Appendix B shows our deviation goal and the empirical comparisons between deviation goal and directed deviation goal.

3.2 Attacking Krum

Recall that Krum selects one local model as the global model in each iteration. Suppose $\mathbf{w}$ is the selected local model in the current iteration when there are no attacks. Our goal is to craft the c compromised local models such that the local model selected by Krum has the largest directed deviation from $\mathbf{w}$. Our idea is to make Krum select a certain crafted local model (e.g., $\mathbf{w}'_1$ without loss of generality) via crafting the c compromised local models. Therefore, we aim to solve the optimization problem in Equation 1 with $\mathbf{w}' = \mathbf{w}'_1$ and the aggregation rule is Krum.

Full knowledge: The key challenge of solving the optimization problem is that the constraint of the optimization problem is highly nonlinear and the search space of the local models $\mathbf{w}'_1, \cdots, \mathbf{w}'_c$ is large. To address the challenge, we make two approximations. Our approximations represent suboptimal solutions to the optimization problem, which means that the attacks based on the approximations may have suboptimal performance. However, as we will demonstrate in our experiments, our attacks already substantially increase the error rate of the learnt model.

First, we restrict $\mathbf{w}'_1$ as follows: $\mathbf{w}'_1 = \mathbf{w}_{Re} - \lambda\mathbf{s}$, where $\mathbf{w}_{Re}$ is the global model received from the master device in the current iteration (i.e., the global model obtained in the previous iteration) and $\lambda > 0$. This approximation explicitly models the directed deviation between the crafted local model $\mathbf{w}'_1$ and the received global model. We also explored the approximation $\mathbf{w}'_1 = \mathbf{w} - \lambda\mathbf{s}$, which means that we explicitly model the directed deviation between the crafted local model and the local model selected by Krum before attack. However, we found that our attacks are less effective using this approximation.

Second, to make $\mathbf{w}_1$ more likely to be selected by Krum, we craft the other $c-1$ compromised local models to be close to $\mathbf{w}'_1$. In particular, when the other $c-1$ compromised local models are close to $\mathbf{w}'_1$, $\mathbf{w}'_1$ only needs to have a small distance to $m-2c-1$ benign local models in order to be selected by Krum. In other words, the other $c-1$ compromised local models "support" the crafted local model $\mathbf{w}'_1$. In implementing our attack, we first assume the other $c-1$ compromised local models are the same as $\mathbf{w}'_1$, then we solve $\mathbf{w}'_1$, and finally we randomly sample $c-1$ vectors, whose distance to $\mathbf{w}'_1$ is at most ε, as the other $c-1$ compromised local models. With our two approximations, we transform the optimization problem as follows:

$$\begin{aligned}
&\max_{\lambda} \lambda\\
\text{subject to } &\mathbf{w}'_1 = Krum(\mathbf{w}'_1,\cdots,\mathbf{w}'_c,\mathbf{w}_{(c+1)},\cdots,\mathbf{w}_m),\\
&\mathbf{w}'_1 = \mathbf{w}_{Re} - \lambda\mathbf{s},\\
&\mathbf{w}'_i = \mathbf{w}'_1, \text{ for } i = 2,3,\cdots,c. \qquad (2)
\end{aligned}$$

More precisely, the objective function in the above optimization problem should be $\mathbf{s}^T(\mathbf{w}-\mathbf{w}_{Re}) + \lambda\mathbf{s}^T\mathbf{s}$. However, $\mathbf{s}^T(\mathbf{w}-\mathbf{w}_{Re})$ is a constant and $\mathbf{s}^T\mathbf{s} = d$ where d is the number of parameters in the global model. Therefore, we simplify the objective function to be just λ. After solving λ in the optimization problem, we can obtain the crafted local model $\mathbf{w}'_1$.

Then, we randomly sample $c-1$ vectors whose distance to $\mathbf{w}'_1$ is at most ε as the other $c-1$ compromised local models. We will explore the impact of ε on the effectiveness of our attacks in experiments.

Solving λ. Solving λ in the optimization problem in Equation 2 is key to our attacks. First, we derive an upper bound of the solution λ to the optimization problem. Formally, we have the following theorem.

Theorem 1. *Suppose λ is a solution to the optimization problem in Equation 2. λ is upper bounded as follows:*

$$\lambda \leq \frac{1}{(m-2c-1)\sqrt{d}} \cdot \min_{c+1 \leq i \leq m} \left(\sum_{l \in \tilde{\Gamma}^{m-c-2}_{\mathbf{w}_i}} D(\mathbf{w}_l, \mathbf{w}_i) \right) + \frac{1}{\sqrt{d}} \cdot \max_{c+1 \leq i \leq m} D(\mathbf{w}_i, \mathbf{w}_{Re}), \tag{3}$$

where d is the number of parameters in the global model, $D(\mathbf{w}_l, \mathbf{w}_i)$ is the Euclidean distance between $\mathbf{w}_l$ and $\mathbf{w}_i$, $\tilde{\Gamma}^{m-c-2}_{w_i}$ is the set of $m-c-2$ benign local models that have the smallest Euclidean distance to $\mathbf{w}_i$.

Proof. See Appendix C. □

Given the upper bound, we use a binary search to solve λ. Specifically, we initialize λ as the upper bound and check whether Krum selects $\mathbf{w}'_1$ as the global model; if not, then we half λ; we repeat this process until Krum selects $\mathbf{w}'_1$ or λ is smaller than a certain threshold (this indicates that the optimization problem may not have a solution). In our experiments, we use 1×10^{-5} as the threshold.

Partial knowledge: In the partial knowledge scenario, the attacker does not know the local models on the benign worker devices, i.e., $\mathbf{w}_{(c+1)}, \cdots, \mathbf{w}_m$. As a result, the attacker does not know the changing directions $\mathbf{s}$ and cannot solve the optimization problem in Equation 2. However, the attacker has access to the before-attack local models on the c compromised worker devices. Therefore, we propose to craft compromised local models based on these before-attack local models. First, we compute the mean of the c before-attack local models as $\tilde{\mathbf{w}} = \frac{1}{c}\sum_{i=1}^{c} \mathbf{w}_i$. Second, we estimate the changing directions using the mean local model. Specifically, if the mean of the jth parameter is larger than the jth global model parameter received from the master device in the current iteration, then we estimate the changing direction for the jth parameter to be 1, otherwise we estimate it to be -1. For simplicity, we denote by $\tilde{\mathbf{s}}$ the vector of estimated changing directions.

Third, we treat the before-attack local models on the compromised worker devices as if they were local models on benign worker devices, and we aim to craft local model $\mathbf{w}'_1$ such that, among the crafted local model and the c before-attack local models, Krum selects the crafted local model. Formally, we have the following optimization problem:

$$\begin{aligned} &\max_{\lambda} \lambda \\ \text{subject to } &\mathbf{w}'_1 = Krum(\mathbf{w}'_1, \mathbf{w}_1, \cdots, \mathbf{w}_c), \\ &\mathbf{w}'_1 = \mathbf{w}_{Re} - \lambda \tilde{\mathbf{s}}. \end{aligned} \tag{4}$$

Similar to Theorem 1, we can also derive an upper bound of λ for the optimization problem in Equation 4. Moreover, similar to the full knowledge scenario, we use a binary search to solve λ. However, unlike the full knowledge scenario, if we cannot find a solution λ until λ is smaller than a threshold (i.e., 1×10^{-5}), then we add one more crafted local model $\mathbf{w}'_2$ such that among the crafted local models $\mathbf{w}'_1$, $\mathbf{w}'_2$, and the c before-attack local models, Krum selects the crafted local model $\mathbf{w}'_1$. Specifically, we solve the optimization problem in Equation 4 with $\mathbf{w}'_2$ added into the Krum aggregation rule. Like the full knowledge scenario, we assume $\mathbf{w}'_2 = \mathbf{w}'_1$. If we still cannot find a solution λ until λ is smaller than the threshold, we add another crafted local model. We repeat this process until finding a solution λ. We find that such iterative searching process makes our attack more effective for Krum in the partial knowledge scenario. After solving λ, we obtain the crafted local model $\mathbf{w}'_1$. Then, like the full knowledge scenario, we randomly sample $c-1$ vectors whose distance to $\mathbf{w}'_1$ is at most ε as the other $c-1$ compromised local models.

3.3 Attacking Trimmed Mean

Suppose w_{ij} is the jth *before-attack* local model parameter on the ith worker device and w_j is the jth before-attack global model parameter in the current iteration. We discuss how we craft each local model parameter on the compromised worker devices. We denote by $w_{max,j}$ and $w_{min,j}$ the maximum and minimum of the jth local model parameters on the benign worker devices, i.e., $w_{max,j}$=max$\{w_{(c+1)j}, w_{(c+2)j}, \cdots, w_{mj}\}$ and $w_{min,j}$=min$\{w_{(c+1)j}, w_{(c+2)j}, \cdots, w_{mj}\}$.

Full knowledge: Theoretically, we can show that the following attack can maximize the directed deviations of the global model (i.e., an optimal solution to the optimization problem in Equation 1): if $s_j = -1$, then we use any c numbers that are larger than $w_{max,j}$ as the jth local model parameters on the c compromised worker devices, otherwise we use any c numbers that are smaller than $w_{min,j}$ as the jth local model parameters on the c compromised worker devices.

Intuitively, our attack crafts the compromised local models based on the maximum or minimum benign local model parameters, depending on which one deviates the global model towards the inverse of the direction along which the global model would change without attacks. The sampled c numbers should be close to $w_{max,j}$ or $w_{min,j}$ to avoid being outliers and being detected easily. Therefore, when implementing the attack, if $s_j = -1$, then we randomly sample the c numbers in the interval $[w_{max,j}, b \cdot w_{max,j}]$ (when $w_{max,j} > 0$) or

$[w_{max,j}, w_{max,j}/b]$ (when $w_{max,j} \leq 0$), otherwise we randomly sample the c numbers in the interval $[w_{min,j}/b, w_{min,j}]$ (when $w_{min,j} > 0$) or $[b \cdot w_{min,j}, w_{min,j}]$ (when $w_{min,j} \leq 0$). Our attack does not depend on b once $b > 1$. In our experiments, we set $b = 2$.

Partial knowledge: An attacker faces two challenges in the partial knowledge scenario. First, the attacker does not know the changing direction variable s_j because the attacker does not know the local models on the benign worker devices. Second, for the same reason, the attacker does not know the maximum $w_{max,j}$ and minimum $w_{min,j}$ of the benign local model parameters. Like Krum, to address the first challenge, we estimate the changing direction variables using the local models on the compromised worker devices.

One naive strategy to address the second challenge is to use a very large number as $w_{max,j}$ or a very small number as $w_{min,j}$. However, if we craft the compromised local models based on $w_{max,j}$ or $w_{min,j}$ that are far away from their true values, the crafted local models may be outliers and the master device may detect the compromised local models easily. Therefore, we propose to estimate $w_{max,j}$ and $w_{min,j}$ using the before-attack local model parameters on the compromised worker devices. In particular, the attacker can compute the mean μ_j and standard deviation σ_j of each jth parameter on the compromised worker devices.

Based on the assumption that each jth parameters of the benign worker devices are samples from a Gaussian distribution with mean μ_j and standard deviation σ_j, we can estimate that $w_{max,j}$ is smaller than $\mu_j + 3\sigma_j$ or $\mu_j + 4\sigma_j$ with large probabilities; and $w_{min,j}$ is larger than $\mu_j - 4\sigma_j$ or $\mu_j - 3\sigma_j$ with large probabilities. Therefore, when s_j is estimated to be -1, we sample c numbers from the interval $[\mu_j + 3\sigma_j, \mu_j + 4\sigma_j]$ as the jth parameter of the c compromised local models, which means that the crafted compromised local model parameters are larger than the maximum of the benign local model parameters with a high probability (e.g., 0.898 – 0.998 when $m = 100$ and $c = 20$ under the Gaussian distribution assumption). When s_j is estimated to be 1, we sample c numbers from the interval $[\mu_j - 4\sigma_j, \mu_j - 3\sigma_j]$ as the jth parameter of the c compromised local models, which means that the crafted compromised local model parameters are smaller than the minimum of the benign local model parameters with a high probability. The jth model parameters on the benign worker devices may not accurately follow a Gaussian distribution. However, our attacks are still effective empirically.

3.4 Attacking Median

We use the same attacks for trimmed mean to attack the median aggregation rule. For instance, in the full knowledge scenario, we randomly sample the c numbers in the interval $[w_{max,j}, b \cdot w_{max,j}]$ or $[w_{max,j}, w_{max,j}/b]$ if $s_j = -1$, otherwise we randomly sample the c numbers in the interval $[w_{min,j}/b, w_{min,j}]$ or $[b \cdot w_{min,j}, w_{min,j}]$.

4 Evaluation

We evaluate the effectiveness of our attacks using multiple datasets in different scenarios, e.g., the impact of different parameters and known vs. unknown aggregation rules. Moreover, we compare our attacks with existing attacks.

4.1 Experimental Setup

Datasets: We consider four datasets: MNIST, Fashion-MNIST, CH-MNIST [31][2] and Breast Cancer Wisconsin (Diagnostic) [18]. MNIST and Fashion-MNIST each includes 60,000 training examples and 10,000 testing examples, where each example is an 28×28 grayscale image. Both datasets are 10-class classification problems. The CH-MNIST dataset consists of 5000 images of histology tiles from patients with colorectal cancer. The dataset is an 8-class classification problem. Each image has 64×64 grayscale pixels. We randomly select 4000 images as the training examples and use the remaining 1000 as the testing examples. The Breast Cancer Wisconsin (Diagnostic) dataset is a binary classification problem to diagnose whether a person has breast cancer. The dataset contains 569 examples, each of which has 30 features describing the characteristics of a person's cell nuclei. We randomly select 455 (80%) examples as the training examples, and use the remaining 114 examples as the testing examples.

Machine learning classifiers: We consider the following classifiers.

Multi-class logistic regression (LR). The considered aggregation rules have theoretical guarantees for the error rate of LR classifier.

Deep neural networks (DNN). For MNIST, Fashion-MNIST, and Breast Cancer Wisconsin (Diagnostic), we use a DNN with the architecture described in Table 7a in Appendix. We use ResNet20 [28] for CH-MNIST. Our DNN architecture does not necessarily achieve the smallest error rates for the considered datasets, as our goal is not to search for the best DNN architecture. Our goal is to show that our attacks can increase the testing error rates of the learnt DNN classifiers.

Compared attacks: We compare the following attacks.

Gaussian attack. This attack randomly crafts the local models on the compromised worker devices. Specifically, for each jth model parameter, we estimate a Gaussian distribution using the before-attack local models on all worker devices. Then, for each compromised worker device, we sample a number from the Gaussian distribution and treat it as the jth parameter of the local model on the compromised worker device. We use this Gaussian attack to show that crafting compromised local models randomly can not effectively attack the Byzantine-robust aggregation rules.

[2] We use a pre-processed version from https://www.kaggle.com/kmader/colorectal-histology-mnist#hmnist_64_64_L.csv.

Table 1: Default setting for key parameters.

Parameter	Description	Value
m	Number of worker devices.	100
c	Number of compromised worker devices.	20
p	Degree of Non-IID.	0.5
ε	Distance parameter for Krum attacks.	0.01
β	Parameter of trimmed mean.	c

Label flipping attack. This is a data poisoning attack that does not require knowledge of the training data distribution. On each compromised worker device, this attack flips the label of each training instance. Specifically, we flip a label l as $L-l-1$, where L is the number of classes in the classification problem and $l = 0, 1, \cdots, L-1$.

Back-gradient optimization based attack [43]. This is the state-of-the-art untargeted data poisoning attack for multi-class classifiers. We note that this attack is not scalable and thus we compare our attacks with this attack on a subset of MNIST separately. The results are shown in Section 4.4.

Full knowledge attack or partial knowledge attack. Our attack when the attacker knows the local models on all worker devices or the compromised ones.

Parameter setting: We describe parameter setting for the federated learning algorithms and our attacks. Table 1 summarizes the default setting for key parameters. We use MXNet [12] to implement federated learning and attacks. We repeat each experiment for 50 trials and report the average results. We observed that the variances are very small, so we omit them for simplicity.

Federated learning algorithms. By default, we assume $m = 100$ worker devices; each worker device applies one round of stochastic gradient descent to update its local model; and the master device aggregates local models from all worker devices. One unique characteristic of federated learning is that the local training datasets on different devices may not be *independently and identically distributed* (i.e., *non-IID*) [39]. We simulate federated learning with different non-IID training data distributions. Suppose we have L classes in the classification problem, e.g., $L = 10$ for the MNIST and Fashion-MNIST datasets, and $L = 8$ for the CH-MNIST dataset. We evenly split the worker devices into L groups. We model non-IID federated learning by assigning a training instance with label l to the lth group with probability p, where $p > 0$. A higher p indicates a higher degree of non-IID. For convenience, we call the probability p *degree of non-IID*. Unless otherwise mentioned, we set $p = 0.5$.

We set 500 iterations for the LR classifier on MNIST; we set 2,000 iterations for the DNN classifiers on all four datasets; and we set the batch size to be 32 in stochastic gradient descent, except that we set the batch size to be 64 for Fashion-MNIST as such setting leads to a more accurate model. The trimmed mean aggregation rule prunes the largest and smallest β parameters, where $c \leq \beta < \frac{m}{2}$. Pruning more parameters leads to larger testing error rates without attacks. By default, we consider $\beta = c$ as the authors of trimmed mean did [66].

Table 2: Testing error rates of various attacks.

(a) LR classifier, MNIST

	NoAttack	Gaussian	LabelFlip	Partial	Full
Krum	0.14	0.13	0.13	0.72	0.80
Trimmed mean	0.12	0.11	0.13	0.23	0.52
Median	0.13	0.13	0.15	0.19	0.29

(b) DNN classifier, MNIST

	NoAttack	Gaussian	LabelFlip	Partial	Full
Krum	0.11	0.10	0.10	0.75	0.77
Trimmed mean	0.06	0.07	0.07	0.14	0.23
Median	0.06	0.06	0.16	0.28	0.32

(c) DNN classifier, Fashion-MNIST

	NoAttack	Gaussian	LabelFlip	Partial	Full
Krum	0.16	0.16	0.16	0.90	0.91
Trimmed mean	0.10	0.10	0.12	0.26	0.28
Median	0.09	0.12	0.12	0.21	0.29

(d) DNN classifier, CH-MNIST

	NoAttack	Gaussian	LabelFlip	Partial	Full
Krum	0.29	0.30	0.43	0.73	0.81
Trimmed mean	0.17	0.25	0.37	0.69	0.69
Median	0.17	0.20	0.17	0.57	0.63

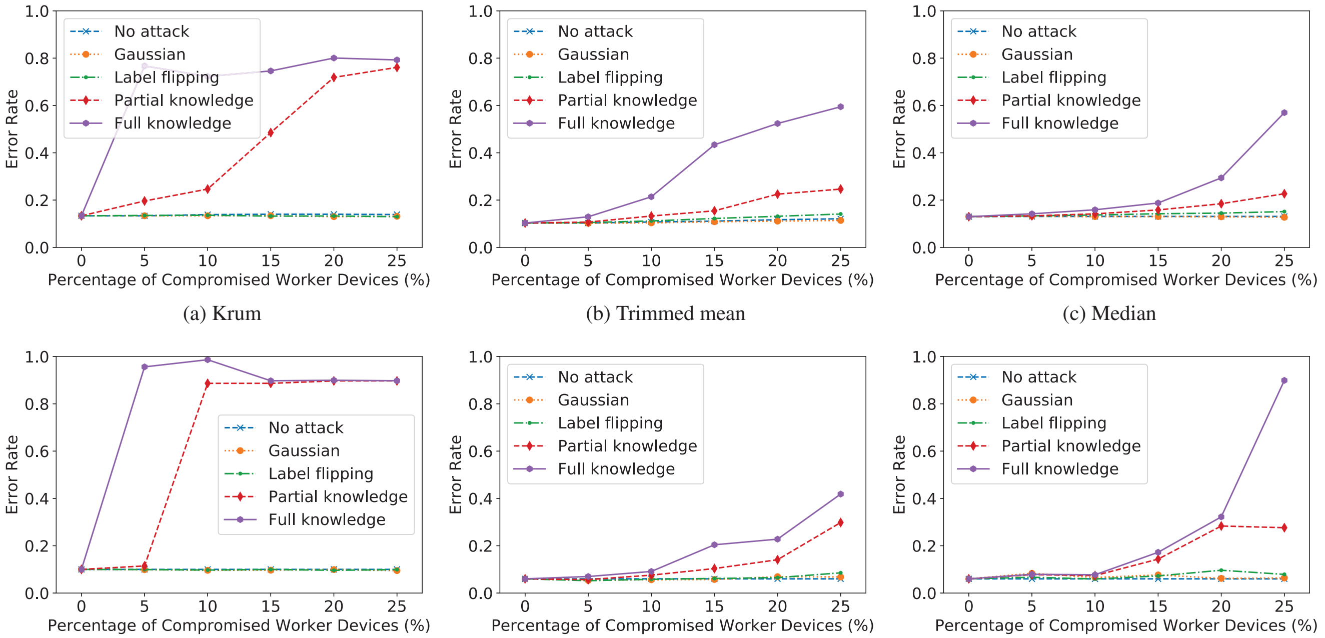
(e) DNN classifier, Breast Cancer Wisconsin (Diagnostic)

	NoAttack	Gaussian	LabelFlip	Partial	Full
Krum	0.03	0.04	0.14	0.17	0.17
Trimmed mean	0.02	0.03	0.05	0.14	0.15
Median	0.03	0.03	0.04	0.17	0.18

Our attacks. Unless otherwise mentioned, we consider 20 worker devices are compromised. Our attacks to Krum have a parameter ε, which is related to the distance between the crafted compromised local models. We set $\varepsilon = 0.01$ (we will study the impact of ε on our attack). We do not set $\varepsilon = 0$ because $\varepsilon = 0$ makes the c compromised local models exactly the same, making the compromised local models easily detected by the master device. Our attacks to trimmed mean and median have a parameter b in the full knowledge scenario, where $b > 1$. Our attacks do not depend on b once $b > 1$. We set $b = 2$. Unless otherwise mentioned, we assume that attacker manipulates the local models on the compromised worker devices in each iteration.

4.2 Results for Known Aggregation Rule

Our attacks are effective: Table 2 shows the testing error rates of the compared attacks on the four datasets. First, these results show that our attacks are effective and substantially outperform existing attacks, i.e., our attacks result in higher er-



(a) Krum (b) Trimmed mean (c) Median

(d) Krum (e) Trimmed mean (f) Median

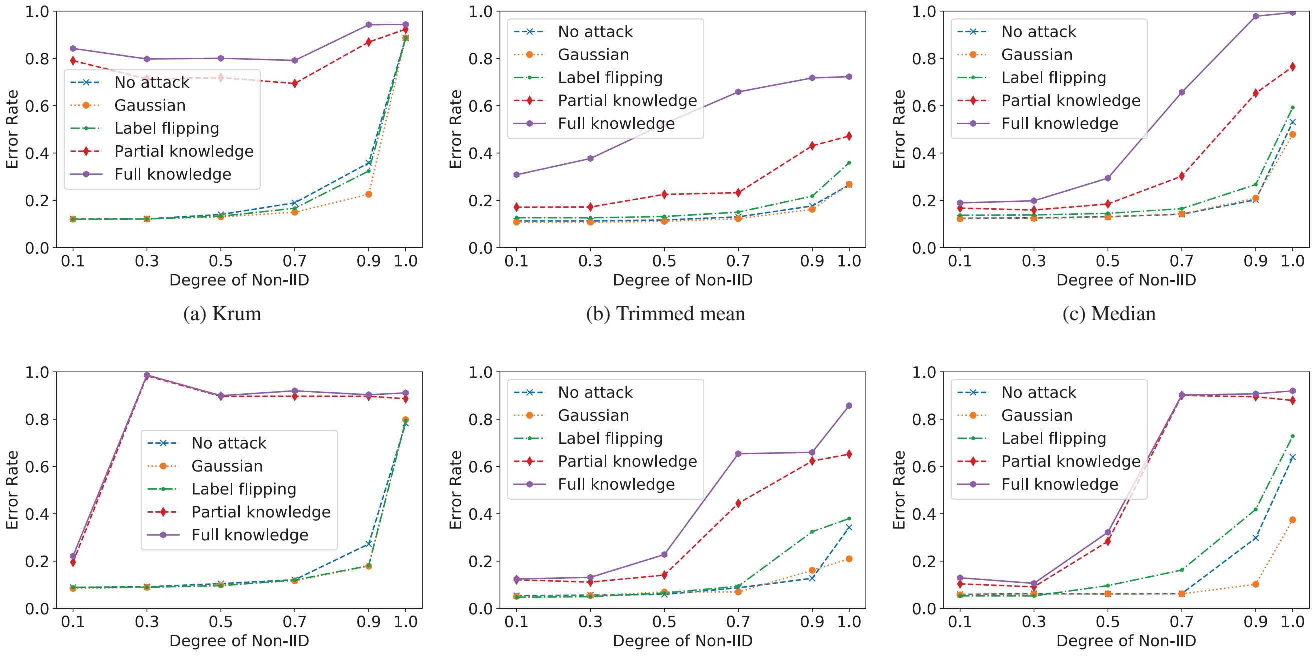
Figure 2: Testing error rates for different attacks as we have more compromised worker devices on MNIST. (a)-(c): LR classifier and (d)-(f): DNN classifier.

ror rates. For instance, when dataset is MNIST, classifier is LR, and aggregation rule is Krum, our partial knowledge attack increases the error rate from 0.14 to 0.72 (around 400% relative increase). Gaussian attacks only increase the error rates in several cases, e.g., median aggregation rule for Fashion-MNIST, and trimmed mean and median for CH-MNIST. Label flipping attacks can increase the error rates for DNN classifiers in some cases but have limited success for LR classifiers.

Second, Krum is less robust to our attacks than trimmed mean and median, except on Breast Cancer Wisconsin (Diagnostic) where Krum is comparable to median. A possible reason why trimmed mean and median outperform Krum is that Krum picks one local model as the global model, while trimmed mean and median aggregate multiple local models to update the global model (the median selects one local model parameter for each model parameter, but the selected parameters may be from different local models). Trimmed mean is more robust to our attacks in some cases while median is more robust in other cases. Third, we observe that the error rates may depend on the data dimension. For instance, MNIST and Fashion-MNIST have 784 dimensions, CH-MNIST has 4096 dimensions, and Breast Cancer Wisconsin (Diagnostic) has 30 dimensions. For the DNN classifiers, the error rates are higher on CH-MNIST than on other datasets in most cases, while the error rates are lower on Breast Cancer Wisconsin (Diagnostic) than on other datasets in most cases.

We note that federated learning may have higher error rate than centralized learning, even if robustness feature is not considered (i.e., mean aggregation rule is used). For instance, the DNN classifiers respectively achieve testing error rates 0.01, 0.08, 0.07, and 0.01 in centralized learning on the four datasets, while they respectively achieve testing error rates 0.04, 0.09, 0.09, and 0.01 in federated learning with the mean aggregation rule on the four datasets. However, in the scenarios where users' training data can only be stored on their edge/mobile devices, e.g., for privacy purposes, centralized learning is not applicable and federated learning may be the only option even though its error rate is higher. Compared to the mean aggregation rule, Byzantine-robust aggregation rule increases the error rate without attacks. However, if Byzantine-robust aggregation rule is not used, a single malicious device can make the learnt global model totally useless [9, 66]. To summarize, in the scenarios where users' training data can only be stored on their edge/mobile devices and there may exist attacks, Byzantine-robust federated learning may be the best option, even if its error rate is higher.

Impact of the percentage of compromised worker devices: Figure 2 shows the error rates of different attacks as the percentage of compromised worker devices increases on MNIST. Our attacks increase the error rates significantly as we compromise more worker devices; label flipping only slightly increases the error rates; and Gaussian attacks have no notable impact on the error rates. Two exceptions are that Krum's error rates decrease when the percentage of compromised worker devices increases from 5% to 10% in Figure 2a and from 10% to 15% in Figure 2d. We suspect the reason is



(a) Krum

(b) Trimmed mean

(c) Median

(d) Krum

(e) Trimmed mean

(f) Median

Figure 3: Testing error rates for different attacks as we increase the degree of non-IID on MNIST. (a)-(c): LR classifier and (d)-(f): DNN classifier.

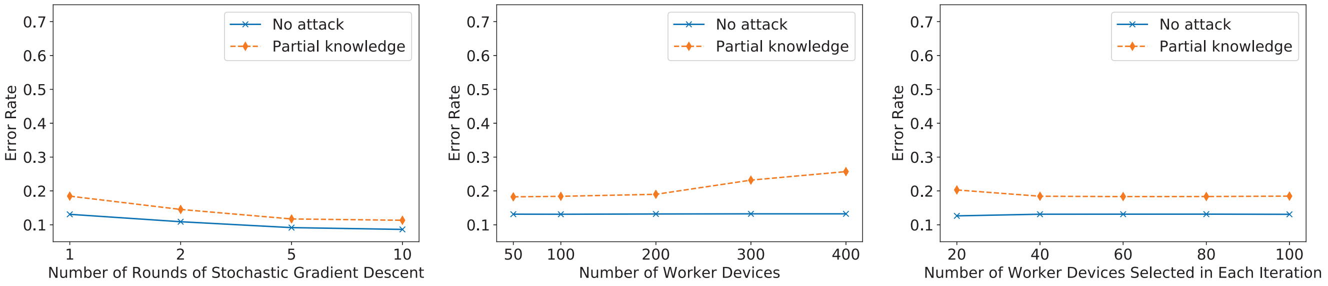
that Krum selects one local model as a global model in each iteration. We have similar observations on the other datasets. Therefore, we omit the corresponding results for simplicity.

Impact of the degree of non-IID in federated learning: Figure 3 shows the error rates for the compared attacks for different degrees of non-IID on MNIST. Error rates of all attacks including no attacks increase as we increase the degree of non-IID, except that the error rates of our attacks to Krum fluctuate as the degree of non-IID increases. A possible reason is that as the local training datasets on different worker devices are more non-IID, the local models are more diverse, leaving more room for attacks. For instance, an extreme example is that if the local models on the benign worker devices are the same, it would be harder to attack the aggregation rules, because their aggregated model would be more likely to depend on the benign local models.

Impact of different parameter settings of federated learning algorithms: We study the impact of various parameters in federated learning including the number of rounds of stochastic gradient descent each worker device performs, number of worker devices, number of worker devices selected to update the global model in each iteration, and β in trimmed mean. In these experiments, we use MNIST and the LR classifier for simplicity. Unless otherwise mentioned, we consider median, as median is more robust than Krum and does not require configuring extra parameters (trimmed mean requires configuring β). Moreover, for simplicity, we consider partial knowledge attacks as they are more practical.

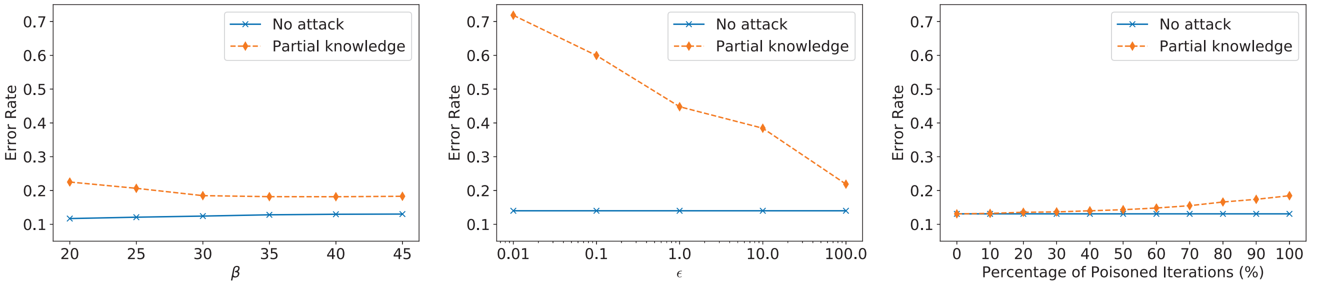
Worker devices can perform multiple rounds of stochastic gradient descent to update their local models. Figure 4a shows the impact of the number of rounds on the testing error rates of our attack. The testing error rates decrease as we use more rounds of stochastic gradient descent for both no attack and our partial knowledge attack. This is because more rounds of stochastic gradient descent lead to more accurate local models, and the local models on different worker devices are less diverse, leaving a smaller attack space. However, our attack still increases the error rates substantially even if we use more rounds. For instance, our attack still increases the error rate by more than 30% when using 10 rounds of stochastic gradient descent. We note that a large number of rounds result in large computational cost for worker devices, which may be unacceptable for resource-constrained devices such as mobile phones and IoT devices.

Figure 4b shows the testing error rates of our attack as the number of worker devices increases, where 20% of worker devices are compromised. Our attack is more effective (i.e., testing error rate is larger) as the federated learning system involves more worker devices. We found a possible reason is that our partial knowledge attacks can more accurately estimate the changing directions with more worker devices. For instance, for trimmed mean of the DNN classifier on MNIST, our partial knowledge attacks can correctly estimate the changing directions of 72% of the global model parameters on average when there are 50 worker devices, and this fraction increases to 76% when there are 100 worker devices.



(a) (b) (c)

Figure 4: (a) Impact of the number of rounds of stochastic gradient descent worker devices use to update their local models in each iteration on our attacks. (b) Impact of the number of worker devices on our attacks. (c) Impact of the number of worker devices selected in each iteration on our attacks. MNIST, LR classifier, and median are used.



(a) (b) (c)

Figure 5: (a) Testing error rates of the trimmed mean aggregation rule when using different β. (b) Testing error rates of the Krum aggregation rule when our attack uses different ε. (c) Testing error rates of the median aggregation rule when our attacks poison a certain fraction of randomly selected iterations of federated learning. MNIST and LR classifier are used.

In federated learning [39], the master device could randomly sample some worker devices and send the global model to them; the sampled worker devices update their local models and send the updated local models to the master device; and the master device updates the global model using the local models from the sampled worker devices. Figure 4c shows the impact of the number of worker devices selected in each iteration on the testing error rates of our attack, where the total number of worker devices is 100. Since the master device randomly selects a subset of worker devices in each iteration, a smaller number of compromised worker devices are selected in some iterations, while a larger number of compromised worker devices are selected in other iterations. On average, among the selected worker devices, $\frac{c}{m}$ of them are compromised ones, where c is the total number of compromised worker devices and m is the total number of worker devices. Our Figure 2 shows that our attacks become effective when $\frac{c}{m}$ is larger than 10%-15%. Note that an attacker can inject a large number of fake devices to a federated learning system, so $\frac{c}{m}$ can be large.

The trimmed mean aggregation rule has a parameter β, which should be at least the number of compromised worker devices. Figure 5a shows the testing error rates of no attack and our partial knowledge attack as β increases. Roughly speaking, our attack is less effective (i.e., testing error rates are smaller) as more local model parameters are trimmed. This is because our crafted local model parameters on the compromised worker devices are more likely to be trimmed when the master device trims more local model parameters. However, the testing error of no attack also slightly increases as β increases. The reason is that more benign local model parameters are trimmed and the mean of the remaining local model parameters becomes less accurate. The master device may be motivated to use a smaller β to guarantee performance when there are no attacks.

Impact of the parameter ε in our attacks to Krum: Figure 5b shows the error rates of the Krum aggregation rule when our attacks use different ε, where MNIST dataset and LR classifier are considered. We observe that our attacks can effectively increase the error rates using a wide range of ε. Moreover, our attacks achieve larger error rates when ε is smaller. This is because when ε is smaller, the distances between the compromised local models are smaller, which makes it more likely for Krum to select the local model crafted by our attack as the global model.

Impact of the number of poisoned iterations: Figure 5c shows the error rates of the median aggregation rule when our attacks poison the local models on the compromised worker

Table 3: Testing error rates of attacks on the DNN classifier for MNIST when the master device chooses the global model with the lowest testing error rate.

	NoAttack	Gaussian	LabelFlip	Partial	Full
Krum	0.10	0.10	0.09	0.69	0.70
Trimmed mean	0.06	0.06	0.07	0.12	0.18
Median	0.06	0.06	0.06	0.11	0.32

devices in a certain fraction of randomly selected iterations of federated learning. Unsurprisingly, the error rate increases when poisoning more iterations.

Alternative training strategy: Each iteration results in a global model. Instead of selecting the last global model as the final model, an alternative training strategy is to select the global model that has the lowest testing error rate.[3] Table 3 shows the testing error rates of various attacks on the DNN classifier for MNIST, when such alternative training strategy is adopted. In these experiments, our attacks attack each iteration of federated learning, and the column "NoAttack" corresponds to the scenarios where no iterations are attacked. Compared to Table 2b, this alternative training strategy is slightly more secure against our attacks. However, our attacks are still effective. For instance, for the Krum, trimmed mean, and median aggregation rules, our partial knowledge attacks still increase the testing error rates by 590%, 100%, and 83%, respectively. Another training strategy is to roll back to a few iterations ago if the master device detects an unusual increase of training error rate. However, such training strategy is not applicable because the training error rates of the global models still decrease until convergence when we perform our attacks in each iteration. In other words, there are no unusual increases of training error rates.

4.3 Results for Unknown Aggregation Rule

We craft local models based on one aggregation rule and show the attack effectiveness for other aggregation rules. Table 4 shows the transferability between aggregation rules, where MNIST and LR classifier are considered. We observe different levels of transferability between aggregation rules. Specifically, Krum based attack can well transfer to trimmed mean and median, e.g., Krum based attack increases the error rate from 0.12 to 0.15 (25% relative increase) for trimmed mean, and from 0.13 to 0.18 (38% relative increase) for median. Trimmed mean based attack does not transfer to Krum but transfers to median well. For instance, trimmed mean based attack increases the error rates from 0.13 to 0.20 (54% relative increase) for median.

Table 4: Transferability between aggregation rules. "Krum attack" and "Trimmed mean attack" mean that we craft the compromised local models based on the Krum and trimmed mean aggregation rules, respectively. Partial knowledge attacks are considered. The numbers are testing error rates.

	Krum	Trimmed mean	Median
No attack	0.14	0.12	0.13
Krum attack	0.70	0.15	0.18
Trimmed mean attack	0.14	0.25	0.20

4.4 Comparing with Back-gradient Optimization based Attack

Back-gradient optimization based attack (BGA) [43] is state-of-the-art untargeted data poisoning attack for multi-class classifiers such as multi-class LR and DNN. BGA formulates a bilevel optimization problem, where the inner optimization is to minimize the training loss on the poisoned training data and the outer optimization is to find poisoning examples that maximize the minimal training loss in the inner optimization. BGA iteratively finds the poisoned examples by alternately solving the inner minimization and outer maximization problems. We implemented BGA and verified that our implementation can reproduce the results reported by the authors. However, BGA is not scalable to the entire MNIST dataset. Therefore, we uniformly sample 6,000 training examples in MNIST, and we learn a 10-class LR classifier. Moreover, we assume 100 worker devices, randomly distribute the 6,000 examples to them, and assume 20 worker devices are compromised.

Generating poisoned data: We assume an attacker has *full knowledge* about the training datasets on all worker devices. Therefore, the attacker can use BGA to generate poisoned data based on the 6,000 examples. In particular, we run the attack for 10 days on a GTX 1080Ti GPU, which generates 240 ($240/6000 = 4\%$) poisoned examples. We verified that these poisoned data can effectively increase the testing error rate if the LR classifier is learnt in a *centralized* environment. In particular, the poisoned data can increase the testing error rate of the LR classifier from 0.10 to 0.16 (60% relative increase) in centralized learning. However, in federated learning, the attacker can only inject the poisoned data to the compromised worker devices. We consider two scenarios on how the attacker distributes the poisoned data to the compromised worker devices:

Single worker. In this scenario, the attacker distributes the poisoned data on a single compromised worker device.

Uniform distribution. In this scenario, the attacker distributes the poisoned data to the compromised worker devices uniformly at random.

We consider the two scenarios because they represent two extremes for distributing data (concentrated or evenly distributed) and we expect one extreme to maximize attack effectiveness. Table 5 compares BGA with our attacks. We observe

[3] We give advantages to the alternative training strategy since we use testing error rate to select the global model.

Table 5: Testing error rates of back-gradient optimization based attacks (SingleWorker and Uniform) and our attacks (Partial and Full).

	NoAttack	SingleWorker	Uniform	Partial	Full
Mean	0.10	0.11	0.15	0.54	0.69
Krum	0.23	0.24	0.25	0.85	0.89
Trimmed mean	0.12	0.12	0.13	0.27	0.32
Median	0.13	0.13	0.14	0.19	0.21

that BGA has limited success at attacking Byzantine-robust aggregation rules, while our attacks can substantially increase the testing error rates. We note that if the federated learning uses the *mean* aggregation rule BGA is still successful. For instance, when the mean aggregation rule is used, BGA can increase the testing error rate by 50% when distributing the poisoned data to the compromised worker devices uniformly at random. However, when applying our attacks for trimmed mean to attack the mean aggregation rule, we can increase the testing error rates substantially more (see the last two cells in the second row of Table 5).

5 Defenses

We generalize RONI [4] and TRIM [30], which were designed to defend against data poisoning attacks, to defend against our local model poisoning attacks. Both generalized defenses remove the local models that are potentially malicious before computing the global model in each iteration of federated learning. One generalized defense removes the local models that have large negative impact on the error rate of the global model (inspired by RONI that removes training examples that have large negative impact on the error rate of the model), while the other defense removes the local models that result in large loss (inspired by TRIM that removes the training examples that have large negative impact on the loss). In both defenses, we assume the master device has a small *validation dataset*. Like existing aggregation rules such as Krum and trimmed mean, we assume the master device knows the upper bound c of the number of compromised worker devices. We note that our defenses make the global model slower to learn and adapt to new data as that data may be identified as from potentially malicious local models.

Error Rate based Rejection (ERR): In this defense, we compute the impact of each local model on the error rate for the validation dataset and remove the local models that have large negative impact on the error rate. Specifically, suppose we have an aggregation rule. For each local model, we use the aggregation rule to compute a global model A when the local model is included and a global model B when the local model is excluded. We compute the error rates of the global models A and B on the validation dataset, which we denote as E_A and E_B, respectively. We define $E_A - E_B$ as the *error rate impact* of a local model. A larger error rate impact indicates that the local model increases the error rate more significantly if we include the local model when updating the global model. We remove the c local models that have the largest error rate impact, and we aggregate the remaining local models to obtain an updated global model.

Table 6: Defense results. The numbers are testing error rates. The columns “Krum” and “Trimmed mean” indicate the attacker’s assumed aggregation rule when performing attacks, while the rows indicate the actual aggregation rules and defenses. Partial knowledge attacks are considered.

	No attack	Krum	Trimmed mean
Krum	0.14	0.72	0.13
Krum + ERR	0.14	0.62	0.13
Krum + LFR	0.14	0.58	0.14
Krum + Union	0.14	0.48	0.14
Trimmed mean	0.12	0.15	0.23
Trimmed mean + ERR	0.12	0.17	0.21
Trimmed mean + LFR	0.12	0.18	0.12
Trimmed mean + Union	0.12	0.18	0.12
Median	0.13	0.17	0.19
Median + ERR	0.13	0.21	0.25
Median + LFR	0.13	0.20	0.13
Median + Union	0.13	0.19	0.14

Loss Function based Rejection (LFR): In this defense, we remove local models based on their impact on the loss instead of error rate for the validation dataset. Specifically, like the error rate based rejection, for each local model, we compute the global models A and B. We compute the cross-entropy loss function values of the models A and B on the validation dataset, which we denote as L_A and L_B, respectively. Moreover, we define $L_A - L_B$ as the *loss impact* of the local model. Like the error rate based rejection, we remove the c local models that have the largest loss impact, and we aggregate the remaining local models to update the global model.

Union (i.e., ERR+LFR): In this defense, we combine ERR and LFR. Specifically, we remove the local models that are removed by either ERR or LFR.

Defense results: Table 6 shows the defense results of ERR, FLR, and Union, where partial knowledge attacks are considered. We use the default parameter setting discussed in Section 4.1, e.g., 100 worker devices, 20% of compromised worker devices, MNIST dataset, and LR classifier. Moreover, we sample 100 testing examples uniformly at random as the validation dataset. Each row of the table corresponds to a defense, e.g., Krum + ERR means that the master device uses ERR to remove the potentially malicious local models and uses Krum as the aggregation rule. Each column indicates the attacker’s assumed aggregation rule when performing attacks, e.g., the column “Krum” corresponds to attacks that are based on Krum. We have several observations.

First, LFR is comparable to ERR or much more effective than ERR, i.e., LFR achieves similar or much smaller testing error rates than ERR. For instance, Trimmed mean + ERR and Trimmed mean + LFR achieve similar testing error rates (0.17 vs. 0.18) when the attacker crafts the compromised local models based on Krum. However, Trimmed mean + LFR achieves a much smaller testing error rate than Trimmed mean + ERR (0.12 vs. 0.21), when the attacker crafts the compromised local models based on trimmed mean. Second, Union is comparable to LFR in most cases, except one case (Krum + LFR vs. Krum and Krum + Union vs. Krum) where Union is more effective.

Third, LFR and Union can effectively defend against our attacks in some cases. For instance, Trimmed mean + LFR (or Trimmed mean + Union) achieves the same testing error rate for both no attack and attack based on trimmed mean. However, our attacks are still effective in other cases even if LFR or Union is adopted. For instance, an attack, which crafts compromised local models based on Krum, still effectively increases the error rate from 0.14 (no attack) to 0.58 (314% relative increase) for Krum + LFR. Fourth, the testing error rate grows in some cases when a defense is deployed. This is because the defenses may remove benign local models, which increases the testing error rate of the global model.

6 Related Work

Security and privacy of federated/collaborative learning are much less explored, compared to centralized machine learning. Recent studies [29, 40, 44] explored privacy risks in federated learning, which are orthogonal to our study.

Poisoning attacks: Poisoning attacks aim to compromise the integrity of the training phase of a machine learning system [5]. The training phase consists of two components, i.e., training dataset collection and learning process. Most existing poisoning attacks compromise the training dataset collection component, e.g., inject malicious data into the training dataset. These attacks are also known as *data poisoning attacks*) [3, 8, 13, 19, 27, 30, 33, 43, 45, 50, 51, 56, 61, 62, 65]. Different from data poisoning attacks, our local model poisoning attacks compromise the learning process.

Depending on the goal of a poisoning attack, we can classify poisoning attacks into two categories, i.e., *untargeted poisoning attacks* [8, 30, 33, 50, 62, 65] and *targeted poisoning attacks* [3, 6, 13, 27, 37, 45, 51, 56]. Untargeted poisoning attacks aim to make the learnt model have a high testing error indiscriminately for testing examples, which eventually result in a denial-of-service attack. In targeted poisoning attacks, the learnt model produces attacker-desired predictions for particular testing examples, e.g., predicting spams as non-spams and predicting attacker-desired labels for testing examples with a particular trojan trigger (these attacks are also known as *backdoor/trojan attacks* [27]). However, the testing error for other testing examples is unaffected. Our local model poisoning attacks are untargeted poisoning attacks. Different from existing untargeted poisoning attacks that focus on centralized machine learning, our attacks are optimized for Byzantine-robust federated learning. We note that Xie et al. [63] proposed inner product manipulation based untargeted poisoning attacks to Byzantine-robust federated learning including Krum and median, which is concurrent to our work.

Defenses: Existing defenses were mainly designed for data poisoning attacks to centralized machine learning. They essentially aim to detect the injected malicious data in the training dataset. One category of defenses [4, 15, 56, 59] detects malicious data based on their (negative) impact on the performance of the learnt model. For instance, Barreno et al. [4] proposed *Reject on Negative Impact (RONI)*, which measures the impact of each training example on the performance of the learnt model and removes the training examples that have large negative impact. Suciu et al. [56] proposed a variant of RONI (called tRONI) for targeted poisoning attacks. In particular, tRONI measures the impact of a training example on only the target classification and excludes training examples that have large impact.

Another category of defenses [20, 30, 35, 55] proposed new loss functions, optimizing which obtains model parameters and detects the injected malicious data simultaneously. For instance, Jagielski et al. [30] proposed TRIM, which aims to jointly find a subset of training dataset with a given size and model parameters that minimize the loss function. The training examples that are not in the selected subset are treated as malicious data. These defenses are not directly applicable for our local model poisoning attacks because our attacks do not inject malicious data into the training dataset.

For federated learning, the machine learning community recently proposed several aggregation rules (e.g., Krum [9], Bulyan [42], trimmed mean [66], median [66], and others [14]) that were claimed to be robust against Byzantine failures of certain worker devices. Our work shows that these defenses are not effective in practice against our optimized local model poisoning attacks that carefully craft local models on the compromised worker devices. Fung et al. [23] proposed to compute weight for each worker device according to historical local models and take the weighted average of the local models to update the global model. However, their method can only defend against label flipping attacks, which can already be defended by existing Byzantine-robust aggregation rules. We propose ERR and LFR, which are respectively generalized from RONI and TRIM, to defend against our local model poisoning attacks. We find that these defenses are not effective enough in some scenarios, highlighting the needs of new defenses against our attacks.

Other security and privacy threats to machine learning: Adversarial examples [5, 57] aim to make a machine learning system predict labels as an attacker desires via adding carefully crafted noise to normal testing examples in the testing phase. Various methods (e.g., [2, 11, 25, 36, 46, 47, 52, 54, 57]) were proposed to generate adversarial examples, and many defenses (e.g., [10, 25, 26, 38, 41, 48, 64]) were explored to mitigate them. Different from poisoning attacks, adversarial examples compromise the testing phase of machine learning. Both poisoning attacks and adversarial examples compromise the integrity of machine learning. An attacker could also compromise the confidentiality of machine learning. Specifically, an attacker could compromise the confidentiality of users' private training or testing data via various attacks such as *model inversion attacks* [21, 22], *membership inference attacks* [40, 49, 53], and *property inference attacks* [1, 24]. Moreover, an attacker could also compromise the confidentiality/intellectual property of a model provider via stealing its model parameters and hyperparameters [34, 58, 60].

7 Conclusion, Limitations, and Future Work

We demonstrate that the federated learning methods, which the machine learning community claimed to be robust against Byzantine failures of some worker devices, are vulnerable to our local model poisoning attacks that manipulate the local models sent from the compromised worker devices to the master device during the learning process. In particular, to increase the error rates of the learnt global models, an attacker can craft the local models on the compromised worker devices such that the aggregated global model deviates the most towards the inverse of the direction along which the global model would change when there are no attacks. Moreover, finding such crafted local models can be formulated as optimization problems. We can generalize existing defenses for data poisoning attacks to defend against our local model poisoning attacks. Such generalized defenses are effective in some cases but are not effective enough in other cases. Our results highlight that we need new defenses to defend against our local model poisoning attacks.

Our work is limited to untargeted poisoning attacks. It would be interesting to study targeted poisoning attacks to federated learning. Moreover, it is valuable future work to design new defenses against our local model poisoning attacks, e.g., new methods to detect compromised local models and new adversarially robust aggregation rules.

8 Acknowledgements

We thank the anonymous reviewers and our shepherd Nikita Borisov for constructive reviews and comments. This work was supported by NSF grant No.1937786.

References

[1] Giuseppe Ateniese, Luigi V Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, and Giovanni Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *International Journal of Security and Networks*, 10(3), 2015.

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *ICML*, 2018.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *arxiv*, 2018.

[4] Marco Barreno, Blaine Nelson, Anthony D Joseph, and JD Tygar. The security of machine learning. *Machine Learning*, 2010.

[5] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *ACM ASIACCS*, 2006.

[6] Arjun Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *ICML*, 2019.

[7] Battista Biggio, Luca Didaci, Giorgio Fumera, and Fabio Roli. Poisoning attacks to compromise face templates. In *IEEE ICB*, 2013.

[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *ICML*, 2012.

[9] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *NIPS*, 2017.

[10] Xiaoyu Cao and Neil Zhenqiang Gong. Mitigating evasion attacks to deep neural networks via region-based classification. In *ACSAC*, 2017.

[11] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *IEEE S & P*, 2017.

[12] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[13] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. In *arxiv*, 2017.

[14] Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. In *POMACS*, 2017.

[15] Gabriela F. Cretu, Angelos Stavrou, Michael E. Locasto, Salvatore J. Stolfo, and Angelos D. Keromytis. Casting out demons: Sanitizing training data for anomaly sensors. In *IEEE S & P*, 2008.

[16] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *NIPS*, 2012.

[17] John R. Douceur. The Sybil attack. In *IPTPS*, 2002.

[18] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[19] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. Poisoning attacks to graph-based recommender systems. In *ACSAC*, 2018.

[20] Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. Robust logistic regression and classification. In *NIPS*, 2014.

[21] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM CCS*, 2015.

[22] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security Symposium*, 2014.

[23] Clement Fung, Chris J.M. Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. In *arxiv*, 2018.

[24] Karan Ganju, Qi Wang, Wei Yang, Carl A. Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *CCS*, 2018.

[25] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv*, 2014.

[26] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. In *arXiv*, 2017.

[27] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. In *Machine Learning and Computer Security Workshop*, 2017.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[29] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: Information leakage from collaborative deep learning. In *CCS*, 2017.

[30] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE S & P*, 2018.

[31] Jakob Nikolas Kather, Cleo-Aron Weis, Francesco Bianconi, Susanne M Melchers, Lothar R Schad, Timo Gaiser, Alexander Marx, and Frank Gerrit Zöllner. Multi-class texture analysis in colorectal cancer histology. *Scientific reports*, 2016.

[32] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[33] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *NIPS*, 2016.

[34] Bin Liang, Miaoqiang Su, Wei You, Wenchang Shi, and Gang Yang. Cracking classifiers for evasion: A case study on the google's phishing pages filter. In *ACM WWW*, 2016.

[35] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. Robust linear regression against training data poisoning. In *AISec*, 2017.

[36] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.

[37] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. Trojaning attack on neural networks. In *NDSS*, 2018.

[38] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

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

[40] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE S & P*, 2019.

[41] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischof. On detecting adversarial perturbations. In *ICLR*, 2017.

[42] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *ICML*, 2018.

[43] 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. In *AISec*, 2017.

[44] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Standalone and federated learning under passive and active white-box inference attacks. In *IEEE S & P*, 2019.

[45] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. P. Rubinstein, U. Saini, C. Sutton, J. D. Tygar, and K. Xia. Exploiting machine learning to subvert your spam filter. In *LEET*, 2008.

[46] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM ASIACCS*, 2017.

[47] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *EuroS&P*, 2016.

[48] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE S & P*, 2016.

[49] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. Knock knock, who's there? membership inference on aggregate location data. In *NDSS*, 2018.

[50] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *ACM IMC*, 2009.

[51] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NIPS*, 2018.

[52] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and K Michael Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *ACM CCS*, 2016.

[53] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *IEEE S & P*, 2017.

[54] Nedim Srndic and Pavel Laskov. Practical evasion of a learning-based classifier: A case study. In *IEEE S & P*, 2014.

[55] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. Certified defenses for data poisoning attacks. In *NIPS*, 2017.

[56] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. When does machine learning fail? generalized transferability for evasion and poisoning attacks. In *Usenix Security Symposium*, 2018.

[57] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv*, 2013.

[58] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security Symposium*, 2016.

[59] Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *NIPS*, 2018.

[60] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *IEEE S & P*, 2018.

[61] Binghui Wang and Neil Zhenqiang Gong. Attacking graph-based classification via manipulating the graph structure. In *CCS*, 2019.

[62] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *ICML*, 2015.

[63] Cong Xie, Sanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *UAI*, 2019.

[64] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[65] Guolei Yang, Neil Zhenqiang Gong, and Ying Cai. Fake co-visitation injection attacks to recommender systems. In *NDSS*, 2017.

[66] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *ICML*, 2018.

Table 7: (a) The DNN architecture (input layer is not shown) used for MNIST and Fashion MNIST. (b) Testing error rates when applying attacks for Krum to attack Bulyan.

(a)

Layer Type	Size
Convolution + ReLU	$3 \times 3 \times 30$
Max Pooling	2×2
Convolution + ReLU	$3 \times 3 \times 50$
Max Pooling	2×2
Fully Connected + ReLU	200
Softmax	10 / 8

(b)

	Bulyan
No attack	0.14
Partial Knowledge	0.36
Full Knowledge	0.38

Table 8: Testing error rates of our attacks based on the deviation goal and directed deviation goal.

	Krum	Trimmed mean	Median
Deviation goal	0.87	0.10	0.12
Directed deviation goal	0.80	0.52	0.29

A Attacking Bulyan

Bulyan is based on Krum. We apply our attacks for Krum to attack Bulyan. Table 7b shows results of attacking Bulyan. The dataset is MNIST, the classifier is logistic regression, $m = 100$, $c = 20$, $\theta = m - 2c$ (Bulyan selects θ local models using Krum), and $\gamma = \theta - 2c$ (Bulyan takes the mean of γ parameters). Our results show that our attacks to Krum can transfer to Bulyan. Specifically, our partial knowledge attack increases the error rate by around 150%, while our full knowledge attack increases the error rate by 165%.

B Deviation Goal

The deviation goal is to craft local models $\mathbf{w}'_1, \mathbf{w}'_2, \cdots, \mathbf{w}'_c$ for the compromised worker devices via solving the following optimization problem in each iteration:

$$\begin{aligned} &\max_{\mathbf{w}'_1, \cdots, \mathbf{w}'_c} ||\mathbf{w} - \mathbf{w}'||_1, \\ \text{subject to } &\mathbf{w} = \mathcal{A}(\mathbf{w}_1, \cdots, \mathbf{w}_c, \mathbf{w}_{c+1}, \cdots, \mathbf{w}_m), \\ &\mathbf{w}' = \mathcal{A}(\mathbf{w}'_1, \cdots, \mathbf{w}'_c, \mathbf{w}_{c+1}, \cdots, \mathbf{w}_m), \end{aligned} \tag{5}$$

where $|| \cdot ||_1$ is L_1 norm. We can adapt our attacks based on the directed deviation goal to the deviation goal. For simplicity, we focus on the full knowledge scenario.

Krum: Similar to the directed deviation goal, we make two approximations, i.e., $\mathbf{w}'_1 = \mathbf{w}_{Re} - \lambda$ and the c compromised local models are the same. Then, we formulate an optimization problem similar to Equation 2, except that $\mathbf{w}'_1 = \mathbf{w}_{Re} - \lambda \mathbf{s}$ is changed to $\mathbf{w}'_1 = \mathbf{w}_{Re} - \lambda$. Like Theorem 1, we can derive an upper bound of λ, given which we use binary search to solve λ. After solving λ, we obtain $\mathbf{w}'_1$. Then, we randomly sample $c - 1$ vectors whose Euclidean distances to $\mathbf{w}'_1$ are smaller than ε as the other $c - 1$ compromised local models.

Trimmed mean: Theoretically, we can show that the following attack can maximize the deviation of the global model: we use any c numbers that are larger than $w_{max,j}$ or smaller than $w_{min,j}$, depending on which one makes the deviation larger, as the jth local model parameters on the c compromised worker devices. Like the directed deviation goal, when implementing the attack, we randomly sample the c numbers in the interval $[w_{max,j}, b \cdot w_{max,j}]$ (when $w_{max,j} > 0$) or $[w_{max,j}, w_{max,j}/b]$ (when $w_{max,j} \leq 0$), or in the interval $[w_{min,j}/b, w_{min,j}]$ (when $w_{min,j} > 0$) or $[b \cdot w_{min,j}, w_{min,j}]$ (when $w_{min,j} \leq 0$), depending on which one makes the deviation larger.

Median: We apply the attack for trimmed mean to median.

Experimental results: Table 8 empirically compares the deviation goal and directed deviation goal, where MNIST and LR classifier are used. For Krum, both goals achieve high testing error rates. However, for trimmed mean and median, the directed deviation goal achieves significantly higher testing error rates than the deviation goal.

C Proof of Theorem 1

We denote by $\Gamma^a_{\mathbf{w}}$ the set of a local models among the crafted c compromised local models and $m - c$ benign local models that are the closest to the local model $\mathbf{w}$ with respect to Euclidean distance. Moreover, we denote by $\tilde{\Gamma}^a_{\mathbf{w}}$ the set of a benign local models that are the closest to $\mathbf{w}$ with respect to Euclidean distance. Since $\mathbf{w}'_1$ is chosen by Krum, we have the following:

$$\sum_{l \in \Gamma^{m-c-2}_{\mathbf{w}'_1}} D(\mathbf{w}_l, \mathbf{w}'_1) \leq \min_{c+1 \leq i \leq m} \sum_{l \in \Gamma^{m-c-2}_{\mathbf{w}_i}} D(\mathbf{w}_l, \mathbf{w}_i), \tag{6}$$

where $D(\cdot, \cdot)$ represents Euclidean distance. The distance between $\mathbf{w}'_1$ and the other $c - 1$ compromised local models is 0, since we assume they are the same in the optimization problem in Equation 2 when finding $\mathbf{w}'_1$. Therefore, we have:

$$\sum_{l \in \tilde{\Gamma}^{m-2c-1}_{\mathbf{w}'_1}} D(\mathbf{w}_l, \mathbf{w}'_1) \leq \min_{c+1 \leq i \leq m} \sum_{l \in \Gamma^{m-c-2}_{\mathbf{w}_i}} D(\mathbf{w}_l, \mathbf{w}_i). \tag{7}$$

According to the triangle inequality $D(\mathbf{w}_l, \mathbf{w}'_1) \geq D(\mathbf{w}'_1, \mathbf{w}_{Re}) - D(\mathbf{w}_l, \mathbf{w}_{Re})$, we get:

$$\begin{aligned} &(m - 2c - 1) \cdot D(\mathbf{w}'_1, \mathbf{w}_{Re}) \\ \leq &\min_{c+1 \leq i \leq m} \sum_{l \in \Gamma^{m-c-2}_{\mathbf{w}_i}} D(\mathbf{w}_l, \mathbf{w}_i) + \sum_{l \in \tilde{\Gamma}^{m-2c-1}_{\mathbf{w}'_1}} D(\mathbf{w}_l, \mathbf{w}_{Re}) \\ \leq &\min_{c+1 \leq i \leq m} \sum_{l \in \tilde{\Gamma}^{m-c-2}_{\mathbf{w}_i}} D(\mathbf{w}_l, \mathbf{w}_i) + (m - 2c - 1) \cdot \max_{c+1 \leq i \leq m} D(\mathbf{w}_i, \mathbf{w}_{Re}). \end{aligned}$$

Since $D(\mathbf{w}'_1, \mathbf{w}_{Re}) = \|\lambda \cdot \mathbf{s}\|_2 = \sqrt{d} \cdot \lambda$, we have:

$$\begin{aligned} \lambda \leq &\frac{1}{(k - c + 1)\sqrt{d}} \cdot \min_{c+1 \leq i \leq m} \sum_{l \in \tilde{\Gamma}^{k}_{\mathbf{w}_i}} D(\mathbf{w}_l, \mathbf{w}_i) \\ &+ \frac{1}{\sqrt{d}} \cdot \max_{c+1 \leq i \leq m} D(\mathbf{w}_i, \mathbf{w}_{Re}). \end{aligned} \tag{8}$$

The bound only depends on the before-attack local models.