CONTRA: Defending against Poisoning Attacks in Federated Learning

Sana Awan, Bo Luo, and Fengjun Li

The University of Kansas, Lawrence, KS, USA
{sanaawan, bluo, fli}@ku.edu

Abstract. Federated learning (FL) is an emerging machine learning paradigm. With FL, distributed data owners aggregate their model updates to train a shared deep neural network collaboratively, while keeping the training data locally. However, FL has little control over the local data and the training process. Therefore, it is susceptible to poisoning attacks, in which malicious or compromised clients use malicious training data or local updates as the attack vector to poison the trained global model. Moreover, the performance of existing detection and defense mechanisms drops significantly in a scaled-up FL system with non-iid data distributions. In this paper, we propose a defense scheme named CONTRA to defend against poisoning attacks, e.g., label-flipping and backdoor attacks, in FL systems. CONTRA implements a cosine-similarity-based measure to determine the credibility of local model parameters in each round and a reputation scheme to dynamically promote or penalize individual clients based on their per-round and historical contributions to the global model. With extensive experiments, we show that CONTRA significantly reduces the attack success rate while achieving high accuracy with the global model. Compared with a state-of-the-art (SOTA) defense, CONTRA reduces the attack success rate by 70% and reduces the global model performance degradation by 50%.

1 Introduction

As an emerging machine learning paradigm, federated learning (FL) is considered a promising solution for privacy-preserving distributed learning. In FL, individual clients first train local models with local training data and a shared global model, and then send the updates to an aggregation server to update the global model for the next training iteration [6, 21, 24]. In this way, a shared model is learned over data from multiple clients without sharing the raw data by any means. Besides ensuring data privacy, FL also improves the efficiency and scalability of machine learning tasks by parallelizing the training among multiple clients and reducing communication costs. This new collaborative machine learning trend is adopted in many applications such as mobile keyboards [6, 25] and medical imaging [20]. As a result, federated learning becomes a new target of various adversarial machine learning attacks, such as poisoning attacks [9, 29] and exploratory attacks [3, 15, 32].

In fact, FL is particularly susceptible to poisoning attacks since the clients can fully control the local data and the local training process. In poisoning attacks, malicious clients can poison local model updates by injecting poisoned instances into the training data (i.e., *data poisoning attacks* [14, 34]) or directly manipulating model updates (i.e., *model poisoning attacks* [2, 7, 8, 13, 36]). As a result, the attacker can tamper with the weights of the global model or inject a backdoor into it. While data poisoning is considered a special case of model poisoning [14], the latter is more effective [7] since it can cause the trained model to produce wrong predictions with only a few malicious clients. In general, the impact of poisoning attacks is related to two factors, the ratio of malicious clients among all clients in FL tasks and the amount of poisoned local training data. For example, 3% poisoned data could cause an 11% reduction in test accuracy [33]. Therefore, it is critical to design solutions to defend against poisoning attacks in federated learning.

The goal of the poisoning attacks is to cause the global model to produce attacker-chosen outputs on specific attacker-chosen inputs (i.e., *targeted attacks*) or wrong outputs on all the inputs (i.e., *untargeted attacks*). Since untargeted attacks attempt to reduce the test accuracy of the main task, they deteriorate the benign performance of the aggregated model across all classes and thus could be detected or mitigated by robust aggregation schemes [10, 12, 37] operated on the server. However, the robust aggregation techniques may perform poorly or even fail to work when the number of Byzantine adversaries exceeds certain explicit bound [14]. On the contrary, the adversary in targeted attacks expects the poisoned model to output attacker-chosen predictions for specific attacker-chosen inputs while behaving normally on other inputs. Most of the existing defenses leverage the difference between benign and malicious model updates to distinguish between the benign and potentially malicious client groups using clustering-based [31, 34] or behavior-based [14] model checking schemes. While these approaches demonstrate their effectiveness in detecting or mitigating targeted poisoning attacks, their performance is often evaluated under simplified, less practical settings (discussed in Sec. 3.2). Moreover, all these schemes adopt very specific assumptions about the training data distributions on honest and malicious clients. For example, [10,31] assumed independent and identically distributed training data while [14,27] assumed non-i.i.d. distributions, which indicate that the reported effectiveness were achieved only in specific situations [28].

In this paper, we aim to answer two research questions: ***(i)*** what is the impact of training data distributions and adversary populations on existing defenses against targeted poisoning attacks in FL? And ***(ii)*** how to design a generic and reliable solution against targeted poisoning attacks in FL? To answer the first question, we adopt the Dirichlet distribution [38] to synthesize i.i.d. and non-i.i.d. data distributions in federated learning and investigate the effectiveness of three defense schemes, i.e., Krum [10], PCA-based clustering [34], and FoolsGold [14] under different assumptions about the data distribution and adversarial population. Our results show that they either fail to prevent the attacks when specific assumptions do not hold, become less effective, or cause a reduction in overall

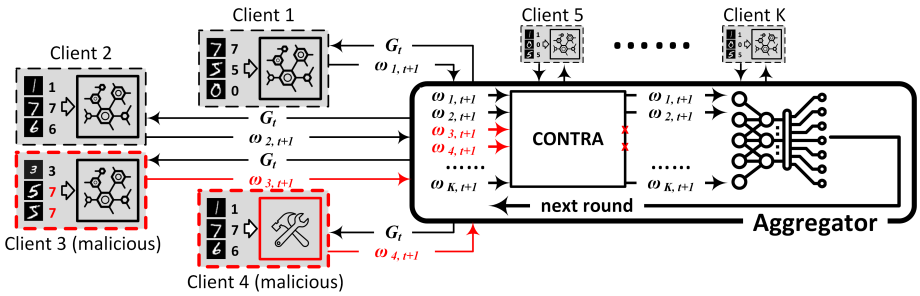

Fig. 1: An overview of FL architecture and the proposed CONTRA approach.

model accuracy. Next, we propose a generic defense scheme, called CONTRA, to thwart poisoning attacks in federated learning. Compared with a state-of-the-art defense, CONTRA reduces the attack success rate by 70% and reduces the global model performance degradation by 50%.

2 Background and Related Work

2.1 Federated Learning

Federated learning (FL) aims to build a global model G over data distributed across multiple clients without sending the raw data to any centralized server. Similar to other modern learning algorithms, FL algorithms rely on stochastic gradient descent (SGD), which minimizes the cost function based on the stochastic estimates of its gradient. Given a dataset $\mathcal{D}$ with n data samples (x_i, y_i), where $y_i \in \mathcal{C}$ is the class label, training a DNN model is to find an optimal set of the parameters $\mathbf{w} = (w^1, ..., w^p)$ that minimizes a chosen loss function $\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\mathbf{w}; x_i, y_i)$. In SGD, $\mathcal{D}$ is divided into multiple batches where each data sample only appears in one batch. In a training round, SGD computes the gradient $g_j = \frac{1}{b}\sum_{(x_i,y_i)\in \mathbf{B_j}} \nabla\mathcal{L}(\mathbf{w}; x_i, y_i)$ for each batch $\mathbf{B_j}$, where $\nabla\mathcal{L}$ is the gradient of the loss function and b is the batch size, and updates $\mathbf{w} = \mathbf{w} - g_j$ iteratively until the DNN model converges.

[24] proposes to combine local SGD on each client with a server that performs model averaging using the *federated averaging* (FedAvg) algorithm. As shown in Figure 1, a FL system consists of K clients and an aggregation server S (called aggregator). In a training round $t \in [1, T]$, a fraction of clients (denoted by the reporting fraction C) are randomly selected to train the global model G_t (with parameters $\mathbf{w}_t$). In particular, a client k trains a local model based on G_t and her local data $\mathcal{D}_k$ as: $\mathbf{w}_{k,t+1} = \mathbf{w}_t - \eta \cdot g_k = \mathbf{w}_t - \eta \cdot b\sum_{j=1}^{n_k/b} g_j$, where η is a fixed local learning rate, and n_k and b denote the size of $\mathcal{D}_k$ and the local mini-batches, respectively. Then, the server S (called aggregator) computes the weighted average of the local model updates from all K clients to update the

parameters of the global model G_{t+1} as: $\mathbf{w}_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w}_{k,t+1}$. The FedAvg algorithm also allows a client to iterate the local training process multiple times (i.e., E training passes in each local epoch) before sending the local update to the aggregator. Finally, when $C = 1$, the algorithm degrades to the baseline *federated SGD* algorithm.

2.2 Poisoning Attacks on Federated Learning

The FL paradigm enlarges the attack surface of the model training process since the clients with full control over local data and local training processes can submit arbitrary updates to change the model. If an attack is to reduce the test accuracy of the model across all classes, it is an untargeted poisoning attack, whereas a targeted attack aims to change the model and cause it to misclassify specific inputs into the target class(es) of the attacker's choice, while not affecting the accuracy of the classes unrelated to the attack. In this paper, we mainly focus on the targeted poisoning attacks, since untargeted attacks can be easily detected or mitigated by state-of-the-art robust aggregation techniques [10, 12, 37].

Data poisoning attacks. FL is vulnerable to data poisoning attacks. An attacker (e.g., client 3 in Fig. 1) can modify its local data by directly flipping the labels of honest training instances of one class (i.e., the source class) to another class (i.e., the target class) while keeping the features of the training data unchanged. It is known as the *label-flipping* attack [9, 14, 34], which can cause substantial drops in global model accuracy and recall even with a small percentage of malicious clients [34]. A *backdoor poisoning attack* was proposed in [17] to break distributed learning schemes based on synchronized SGD, in which the attacker injects backdoored inputs into local data to modify individual features of the training data and embed backdoors into the global model. Recently, Xie et. al. proposed a distributed backdoor attack that split a global trigger pattern into separated local patterns and embedded them into local training data of multiple attackers [36]. Other data poisoning attacks leverage the back-gradient descent to generate adversarial training examples [26].

Model poisoning attacks. In FL, a malicious client (e.g., client 4 in Figure 1) can directly manipulate the local model update to influence the global model, or manipulate the local training algorithm or its parameters to inject poisoning neurons into the global model. For example, the attacker can scale up its poisoning model by a factor of $\frac{n}{\eta}$ to cancel out the model updates from benign clients and replace the global model with its backdoored model [2]. In a subsequent study, Bhagoji et al. proposed an alternating minimization approach to make the attack stealthy [8], which first trained the local model using a benign dataset for the main task and then refined it using a poisoned dataset for the adversarial task. Formulating model poisoning as an optimization problem, [13] proposed to craft local models that force the global model to deviate the most towards the inverse of the direction along which the global model parameter would change without attacks. This attack was effective with synthetic non-i.i.d. datasets, however, it performed poorly on i.i.d. and highly imbalanced non-i.i.d.

datasets, as reported in [30]. Meanwhile, an aggregation-agnostic attack, which consistently applies small changes (i.e. noise) to many parameters to introduce backdoors or perturb the model's convergence, was developed in [5]. It was shown to be robust against statistics-based defenses in the i.i.d. setting.

2.3 Existing Defenses against Poisoning Attacks

Conventional defenses against poisoning attacks in centralized learning involve discovering rare features in the training data that influence the model [19] or small input perturbations that consistently change the outputs [4]. However, these techniques require access to local training data and thus are not applicable in FL. Recently, several detection or defense approaches have been proposed to prevent poisoning attacks in FL, which can be mainly categorised into three directions, *Byzantine robust aggregation*, *clustering-based detection*, and *behavior-based defense*. All of them adopt a common assumption that the attacker population is less than 50%. We will discuss their effectiveness and limitations in Section 3.2.

Byzantine robust aggregation. In SGD-based federated learning algorithms, the aggregator computes the average of the local models to update the weights of the global model. To eliminate incorrect local updates due to Byzantine errors, robust aggregation schemes proposed different aggregation rules to replace the average of the model updates with a robust estimate of the mean, such as median aggregation [12, 37], trimmed mean aggregation [37], or the Krum aggregation that minimizes the Euclidean distances between selected local models [10].

Byzantine robust aggregation schemes demonstrate promising results against untargeted poisoning attacks and targeted model poisoning attacks using boosted learning rates [2], but they are less effective in preventing adaptive poisoning attacks [13], causing a reduction in test accuracy of the global model. Moreover, they work poorly or even fail when a large number of Byzantine adversaries exist in the system [14]. Finally, robust aggregation implicitly assumes the training data is independent and identically distributed. *Our study on Krum shows extremely high attack success rates (e.g.* 70% *to* 90%*) with non-i.i.d. training data.*

Clustering-based detection. In targeted poisoning attacks, a common observation is that the model updates from malicious clients have unique characteristics compared to the ones from honest clients [31, 34]. Therefore, we can re-write the averaging step of FedAvg as:

$$\mathbf{w}_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w}_{k,t+1} = \sum_{\mathcal{D}_i \subset \mathcal{D}_M} \frac{|\mathcal{D}_i|}{n} \mathbf{w}_{i,t+1} + \sum_{\mathcal{D}_j \subset \mathcal{D} - \mathcal{D}_M} \frac{|\mathcal{D}_j|}{n} \mathbf{w}_{j,t+1} \quad (1)$$

where $\mathcal{D}_M$ denotes the union of the training data held by all malicious clients. $\mathbf{w}_{i,t+1}$ and $\mathbf{w}_{j,t+1}$ represent local model updates submitted by malicious and honest client populations, respectively.

The clustering-based approaches propose to check model updates at the aggregator and then cluster them into two groups, for example, using dimensionality reduction techniques such as principal component analysis (PCA) [34],

k-means [31] or DBSCAN [28] clustering algorithms. Thus, the clusters with less than $n/2$ clients are identified as suspicious clusters of malicious clients.

However, these approaches also assume that the training data is independent and identically distributed. *Our study on the PCA-based defense shows that when the data are non-i.i.d., the clustering-based defense schemes fail to distinguish between model updates submitted by malicious clients and honest clients.* Intuitively, this is because the model updates from the honest clients may diverge in most iterations if they have highly imbalanced non-i.i.d. training data.

Behavior-based defense. As honest and malicious clients act differently in targeted poisoning attacks, behavior-based approaches measure the behavioral difference (in terms of local model updates) between the malicious clients and the majority using the Euclidean distance or cosine similarity. For example, adaptive federated averaging (AFA) was proposed in [27], which computed the cosine similarity between the gradients of each local model and the weighted average of all the local models in each round. Based on a range determined by the mean, median, and standard deviation of the calculated cosine similarities, local updates with out-of-range similarities were discarded from the aggregation. However, AFA suffers from the same problem as previous approaches to penalize honest clients with imbalanced non-i.i.d. training data. Recently, Cao et al. proposed to measure the cosine similarity between a local model and the server model trained with a small clean root dataset [11] and assign a trust score to each client. Then, the average of local model updates weighted by the clients' trust scores is used to update the global model. This scheme relies on the root dataset, which has a non-negligible impact on the final global model. Moreover, honest clients whose data distributions are different from the root dataset distribution may be incorrectly penalized.

Another observation about the targeted poisoning attacks is that the groups of honest and malicious clients have distinct contributions to the global model, which would drive the global model towards two different objectives. Moreover, compared with honest clients, malicious clients with the same adversarial objective will produce model updates that are more similar to each other. As pointed out in [14], the distance between the local models submitted by any two malicious clients is smaller than the ones between a malicious client and an honest client. So, FoolsGold [14] proposes to limit contributions of potentially malicious clients with similar model updates to the global model by reducing their learning rates. The FoolsGold scheme shows promising results when the training data is non-i.i.d., since the local models from honest clients may be far from the global model and far from each other too. *However, our study shows that when the training data is i.i.d., it may incorrectly penalize honest clients with similar data distributions and therefore result in substantial drops in test accuracy.*

Besides, recent "structure-based" defenses such as [35] demonstrated promising results in detecting "backdoor neurons" that can be triggered only by backdoored inputs but not clean inputs. However, our work is different from this line of research, which attempt to mitigate backdoor attacks in FL after the training

phase. Moreover, as they target the backdoor attacks, they are less effective in detecting non-backdoor poisoning attacks.

3 The Threat Model and the Problem

3.1 Threat Model and Assumptions

Attacker's goal. In poisoning attacks, the attackers aim to indirectly manipulate the parameters of the learned model by injecting malicious updates to the aggregator. In this paper, we focus on the *label-flipping* and *backdoor data poisoning* attacks [2, 17] in federated classification tasks. In a label-flipping attack, the attacker flips the labels of the training samples from one selected class (i.e., source) to another class (i.e., target), while keeping the features unchanged. Therefore, the attack is independent of the model characteristics, the loss function, or the SGD algorithm. A backdoor attack, on the other hand, embeds special patterns such as patches of pixels or shadows into the original training samples and relabels them with the attacker-chosen label. The patterns act as triggers for the target class, which is exploited by the attacker at test time.

Attacker's capability. We assume that the aggregator is honest while a subset of FL clients is malicious. The proportion of these Sybil-controlled clients among all the participants is denoted by $m\%$. We also assume that each malicious client can manipulate her own training data, but she cannot access or manipulate other clients' data or their learning processes, e.g., the loss function computation, the optimization algorithm, or server's aggregation process. We further assume that the honest clients possess training data that represents every class in the dataset. For an attack to succeed, an attacker must exert more influence on the target class than the total influence from the honest clients. The attacker may target any class by recruiting more malicious clients to outweigh the influence of honest clients. Therefore, we expect a defense to be robust against a significant proportion of malicious clients.

Training data distribution. FL allows us to train deep models with real-world data from distributed, heterogeneous sources, where the statistical characteristics of individual data may differ significantly from each other. Therefore, we assume the training data can be independent and identically distributed or non-identically distributed.

3.2 Factors Impacting Defense Designs against Poisoning Attacks

Existing defense approaches demonstrated promising results in mitigating the poisoning attacks in FL. However, they often made specific assumptions about training data distributions or adopted simplified settings (e.g., small number of clients and extreme synthesis of non-i.i.d. data distributions) when evaluating the performance of the proposed schemes. It remains an open question if these defenses could achieve the same level of effectiveness when some assumptions do not hold. Therefore, in this section, we investigate the factors and limitations

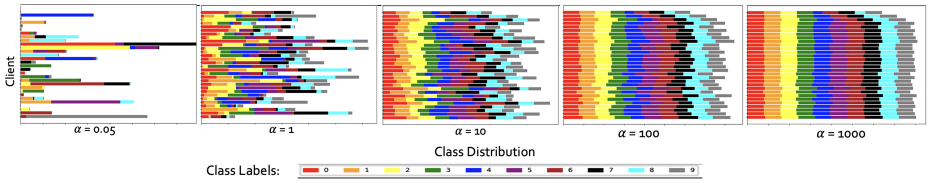

Fig. 2: Synthetic data distributions of 30 clients derived from the Dirichlet distribution with different α, using the MNIST dataset with 10 classes.

that may impact the effectiveness of the defenses. In particular, we implement the label-flipping attack and three representative defenses of robust aggregation (e.g., Krum [10]), clustering-based (e.g., PCA [34]) and behavior-based detection (e.g., FoolsGold [14]) and then compare their effectiveness under different settings with varying malicious client populations and data distributions.

Settings and impacting factors. To obtain a fair understanding of the defenses' performance under a more realistic setting, we adopt a federated learning system with 100 clients and the MNIST dataset with 10 classes, 60K training, and 10K test data samples [23]. We consider four attacker populations to simulate the scenarios with very a few Sybils (i.e. $m = 5\%$ and 10%) to theoretic upper bound of robust aggregation schemes (i.e. $m = 33\%$) and all defenses (i.e. $m = 50\%$). In the label-flipping attack, we randomly select a pair of source and target labels and flip the label of the training samples of the malicious clients who are randomly chosen from the 100 clients.

To evaluate the effectiveness of the defenses, we use two metrics, *model accuracy* (MA) and *attack success rate* (ASR), which assess the final global model performance on the testing data. Detailed definitions are presented in Section 5.1. As a baseline, the accuracy of the global model trained with FEDAVG without and under the label-flipping attack on MNIST is 90% and 76.81%, respectively.

Training and test data distributions. Previous works on FL often assume the training data is independent and identically distributed across the clients [2, 10, 28, 31, 36]. To synthesize a population with i.i.d. data, they usually subsample a population of homogeneous clients with an equal number of samples per class and randomly assign a uniform distribution over all the classes to each client. However, in practice, FL is exposed to statistical heterogeneity such that the training data are unbalanced and usually follow non-representative distribution of the total population. To synthesize a population with non-i.i.d. data, a widely-used approach is "sort-and-partition" [24], which represents a pathologically extreme case of non-identicalness [1]. However, partitions cannot represent complex non-i.i.d. distributions in practical FL scenarios.

In this work, we adopt the *Dirichlet distribution* to synthesize data distributions [2, 18, 36]. We assume that every client's training samples are drawn independently with class labels following a categorical distribution over N classes parameterized by a vector q: ($q_i \geq 0, i \in [1, N]$ and $\|q_1\| = 1$). Specifically, to synthesize a population of non-identical clients, we draw $q \sim Dir(\alpha p)$ from a Dirichlet distribution, where p characterizes a prior class distribution over N

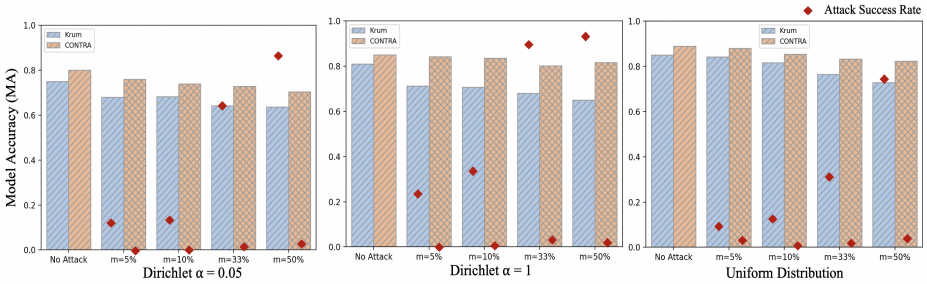

Fig. 3: Model accuracy and attack success rate of Krum and CONTRA under the label-flipping attack with different data distributions and attacker populations.

classes and $\alpha > 0$ is a concentration parameter controlling the identicality among clients. With $\alpha \to \infty$, all clients have identical distributions to the prior; with $\alpha \to 0$, each client holds samples from only one class chosen at random.

For example, we generate unbalanced populations from the training images of the MNIST dataset and distribute them to 100 clients. We set the prior distribution to be uniform across 10 classes. For each client, with a given α, we sample q and assign the corresponding number of images from 10 classes to her. To understand the generated distributions, we visualize the sample distributions drawn from the Dirichlet distribution with different values of the concentration parameter α in Figure 2. Here, we use 30 clients for neat presentation. We can see that the distribution is extremely non-i.i.d. with $\alpha = 0.05$ and i.i.d. with $\alpha = 1000$. With an α between 1 and 10, the distributions are typically non-i.i.d., but when α is larger than 100, they lean towards i.i.d. Therefore, we believe the Dirichlet distribution with an α between 1 and 100 can better represent the real-world data distributions than previous approaches.

The learning process. In each training round, the federated learning schemes randomly select a group of C clients to train the model. C is known as the reporting fraction, which is typically set as 0.1 or 0.2 for scalability purpose. In the local training process, FL applies SGD with a mini-batch size b (e.g., 50-100) and iterates the local training process through E (e.g., 1-5) training passes in a local epoch to accelerate the convergence of the global model. Therefore, we should also consider the impact of these parameters.

Effectiveness and limitations of three defense approaches. Next, we evaluate the effectiveness of Krum [10], PCA-based detection [34], and FoolsGold [14] under the label-flipping attack. We consider settings with different malicious client populations and synthesized data distributions and compare their performance without and under the attack.

Krum. We implement Multi-Krum [24], a variant of Krum. For each local model update, it computes the total Euclidean distance from the $n - f - 2$ nearest neighbors and uses the average of the best k updates to compute the global model, where n and f denote the number of total and malicious clients, respectively. When $k = 1$, multi-Krum is the same as Krum, and it reduces to the basic FedAvg aggregation rule when $k = n$.

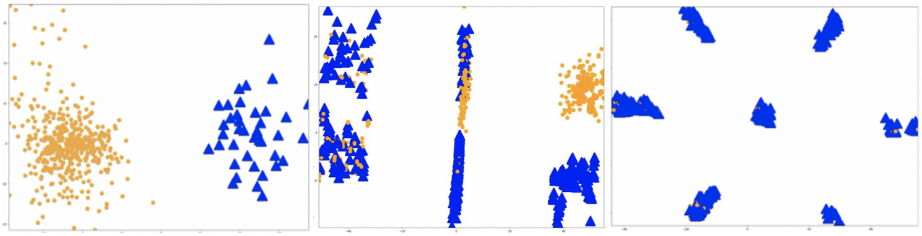

Fig. 4: Two-dimensional PCA plots demonstrate the clusters of honest and malicious gradients (*Left*: $m = 10\%$, uniform distribution; *Middle*: $m = 33\%$, Dirichlet distribution with $\alpha = 1$; *Right*: $m = 50\%$, Dirichlet distribution with $\alpha = 1$).

As shown in Fig. 3, Krum is not effective in defending against the targeted poisoning attacks. Moreover, its performance is strongly influenced by the parameter f and is significantly worse in the non-i.i.d. settings with high malicious clients ratio. Even with a small number of malicious clients (e.g., $m = 10\%$), the attack success rate is 38% and 14% in the non-i.i.d. and i.i.d. settings, respectively. The model accuracy is 81% (the average of 10 runs) in the i.i.d. setting, but it drops to 68% under the the Dirichlet distribution with $\alpha = 0.05$. With more malicious clients (e.g., $m = 33\%$ and 50%), Krum works very poorly in both i.i.d. and non-i.i.d. settings.

PCA-based detection. We implement the scheme in [34]. In each training round, the aggregator computes the difference between a client's model update and the global model (i.e. $\theta_{\Delta,i} \leftarrow \theta_{r,i} - \theta_r$) and extracts the relevant subset of the parameter space using PCA dimensionality reduction. Fig. 4 visualizes the gradients of the local models with two components (i.e. dimensions), where the orange and blue colors denote the gradients of the honest and malicious updates, respectively. As we can see, the honest and malicious models highly overlap in the non-i.i.d. settings. As a result, the two groups cannot be distinguished by any clustering-based detection approach.

FoolsGold. FoolsGold demonstrates a promising result with non-i.i.d. and i.i.d. training data. However, its evaluation adopts over-simplified settings, such as a small set of clients (10 honest clients and 5 malicious clients), non-stochastic SGD (using $C = 1$), and pathological extreme cases to synthesize the i.i.d. and non-i.i.d. distributions. So, we set $C = 0.1$ and evaluate its performance under the more realistic settings used in this work. The results in Tables 2 and 3 show a reduction in model accuracy and an increased attack success rate in all settings.

4 The CONTRA Approach

In this section, we present a new defense mechanism against poisoning attacks in federated learning, called CONTRA. We will first explain the design rationale of CONTRA and then elaborate its building blocks and the algorithm. Finally, we will discuss the strengths and limitations of CONTRA.

4.1 Overview of the CONTRA Design

In federated learning, the local gradients, which are computed to minimize the local objectives, are expected to align with the direction that approaches the optimal of the global objective. However, in each training round, the directions of client i's local objective and the global objective may not align well due to the dynamics in local data distributions, where the angle between the two can be denoted as θ_i. In a targeted attack, malicious clients have a poisoning objective to classify data with a specific feature-space pattern into a target class. Therefore, their local gradients should approach the direction of the poisoning objective but not the one of the global objective. We denote the angle between the directions of a malicious client's gradient and the global objective as δ_j. Intuitively, many clustering-based detection schemes distinguish the clusters of honest and malicious clients, assuming θ_is are similar to each other but different from δ_js.

However, this assumption may not hold, especially when the data are non-i.i.d, as shown in our study on the PCA-based detection scheme (in Sec. 3.2). Motivated by [14], we consider the alignment among the clients' updates in this work. In particular, we consider the angle between the directions of the gradients of two malicious clients (denoted by γ) in each round. Since malicious clients have the same poisoning objective throughout the training process, γ of any two malicious clients should always be smaller than the angle between the gradients of a malicious and an honest clients and the angle between any two honest clients.

Therefore, we propose to measure the alignment level among clients using pairwise cosine similarity. In each round, a higher alignment level means that the client's local objective is closer to the one of another client and thus the client is more suspicious to be malicious. To eliminate the impact of malicious clients to the global model, we introduce a reputation system with two penalty mechanisms based on the alignment level, by reducing the learning rate of a suspicious client and her chance of being selected into the FL process in the future training round. CONTRA enables a high model accuracy and a low attack success rate against targeted poisoning attacks even in cases with a large attacker population (to be discussed in Sec. 5). Meanwhile, as a server-side defense, it is running on the aggregator and does not introduce any changes or overhead at the client.

4.2 The CONTRA Algorithm

Next, we explain the detailed design of CONTRA which consists of three components, alignment level measurement, adaptive learning rates, and reputation-based aggregation. The algorithm of CONTRA is illustrated in Algorithm 1.

Alignment level measurement. CONTRA calculates the cosine similarity between the normalized gradient vectors of two local updates to measure their angular distance, which reflects the similarity of the indicative features in the output layer of the local models [31]. These features are relevant to the correctness of the model and the success of a targeted attack, because they map

directly to the output probabilities. In federated learning, as many clients update the global model collaboratively and iteratively, the gradient of a single client may change drastically over the training rounds. So, CONTRA maintains local updates from each client over several consecutive training rounds and computes the *pairwise similarity between the aggregated historical updates* (Lines 3-8 in Algorithm 1). For malicious clients approaching the poisoning objective, the directions of their aggregated historical updates are more likely to align with each other. Finally, we set the *alignment level* of a client j as the average of its top-k pairwise cosine similarity, denoted as τ_j, where k can be determined based on the estimated attacker population.

Adaptive learning rate. CONTRA employs dynamic learning rate at each local model and adaptively adjusts a client's local learning rate as $lr_j = 1 - \tau_j$ based on her alignment level (Line 20). Local learning rates are typically used to adjust the speed towards global model convergence. So, we reduce the learning rates of suspicious clients to limit their contributions to the global model in each round. The learning rates are further normalized by the maximum learning rate in each round (Lines 23). This is to ensure that if no malicious client exists in the system, the honest clients will not be penalized by our scheme. To avoid incorrectly penalizing honest clients, we also implement the pardon function (Lines 18) proposed in [14] that applies a ratio $min(1, \frac{\tau_j}{\tau_i})$ to each cosine similarity.

Reputation-based aggregation. CONTRA uses a reputation-based aggregation scheme to reduce the chance that a suspicious client is selected to participate in the FL process. In FEDAVG [24], given a reporting fraction $C \in (0, 1]$, $K \times C$ clients are randomly selected to participate in every training round. Therefore, each client has an equal probability $p_i = C$ to be selected. CONTRA employs a reputation-based aggregation scheme that dynamically adjusts a client's probability as $p_i = C + \lambda \cdot r_i$, where r_i is the reputation score of client i. In each round, we rank the clients by p_i and select the top $K \times C$ clients to join the next round. When $\lambda = 0$, the scheme reduces to FEDAVG; otherwise, the client with a higher trust score is more likely to be selected to participate in the next training round. The initial trust scores of all clients are set to 1, so that each client has an equal probability of being selected. In a later round, the trust score of a client j is adjusted by a small Δ if its alignment level τ_j is higher than a preset threshold t (Lines 10-14 in Algorithm 1). The normalized trust scores are then used to adjust the selection probability of the clients. Empirically, we set $\lambda = C(1 - C)$ and use $\Delta = 0.1$ for the image classification tasks and $\Delta = 0.05$ for the Loan dataset (discussed in Sec. 5).

4.3 Discussions

We evaluate the performance of CONTRA in Section 5. It can effectively mitigate targeted poisoning attacks in both i.i.d. and non-i.i.d. settings, with a performance better than existing approaches such as Krum and FoolsGold. However, CONTRA may be less effective or even fail in a special case where there is only one malicious client in the system. For example, under the label-flipping attack,

Algorithm 1 The CONTRA algorithm

Input: Initial model w_0 and local updates $\delta = \{\delta_{i,j}\}$: $\delta_{i,j}$ from client j at iteration i
Set reporting fraction $C = 0.1$; for client j: initial reputation score $r_j = 1$, $G_j = \delta_{1,j}$.
At Server:
for iteration i **do**
  $S_i \leftarrow$ select top-J clients with probability $p_j = C + \lambda \cdot r_j$; $J = K \times C$
  for every client $j \in S_i$ **do**
    $\delta_{i,j} = ClientUpdate(j, w_i)$
    compute historical aggregate: $G_j = G_j + \delta_{i,j}$
    for every client $p \neq j$ **do**
      cosine similarity $cs_{j,p} = \text{dot}(G_j/||G_j||, G_p/||G_p||)$
    end for
    set τ_j as the average of the top-k cosine similarity $cs_{j,p}$ between client j and all other clients
    if $\tau_j > t$ **then**
      $r_j = r_j - \Delta$
    else
      $r_j = r_j + \Delta$
    end if
  end for
  for all client m **do**
    for all client n **do**
      $cs_{m,n} * = min(1, \frac{\tau_m}{\tau_n})$ // re-weighing the cosine similarity
    end for
    $lr_m = (1 - \tau_m)$
    $r_m = r_m / max(r)$ //normalize the reputation score to [0, 1]
  end for
  $lr_m = lr_m / max_m(lr)$ //normalize the learning rate to [0, 1]
  $lr_m = (log[\frac{lr_m}{1-lr_m}] + 0.5)$ //logit function to enlarge the divergence
  $w_{i+1} \leftarrow w_i + \sum_{l=1}^{L} lr_l \times \delta_{i,l}$
end for
At Client: Run *ClientUpdate(j,w)*
B $\leftarrow$ split local dataset into batches of size $|\mathbf{B}|$
for local epoch i from 1 to E **do**
  for all $b \in \mathbf{B}$ **do**
    $w \leftarrow w - \eta \Delta l(w; b)$
    $\delta_j \leftarrow (w_i - w_{init})$
  end for
end for
return δ_j

its accuracy drops from 85.22% (with 20 malicious clients) to 81.23% (with 1 malicious client), while the attack success rate increases from 1.9% to 8.43%. This is because CONTRA relies on similarity (i.e. alignment) among malicious clients with the same poisoning objective to identify suspicious clients in FL. To address this single-attack case, we can integrate CONTRA with an existing robust aggregation scheme such as median aggregation [37] and Krum [10], which

is powerful in detecting outliers. For example, an integrated scheme with Multi-Krum can improve CONTRA's test accuracy to 84.88% and reduce the ASR to 2.7% in the single-attacker case.

Recently, intelligent perturbation-based model poisoning attacks [13] are proposed, which could be extended to defeat CONTRA. For example, malicious clients can submit carefully-crafted updates in pairs with perturbations that cancel out in aggregation, i.e., $\delta_1' = \delta_1 + \varphi$ and $\delta_2' = \delta_2 - \varphi$, where φ is drawn from orthogonal perturbation vectors. Therefore, δ_1' and δ_2' are not similar to each other, but they achieve the same impact on the aggregated global model as δ_1 and δ_2. As pointed out by [14,31], this type of attack is more effective if φ is applied to features that are not important to the model or the attack. Therefore, Fung et al. suggested filtering for indicative features in the model and use a weighted scheme based on the feature importance to mitigate the intelligent attacks [14]. We will explore the solution in this direction in our future work.

5 Experiments

5.1 Datasets, Settings, and Baseline

CONTRA is evaluated on three popular ML datasets: MNIST [23], CIFAR-10 [22], and Loan [16], as summarized in Table 1. Referring to the convergence rate reported in the literature [34, 36], we set the training process to 100, 300, and 200 rounds for the MNIST, CIFAR, and Loan models, respectively.

Data Distribution and Training. For MNIST and CIFAR-10, a Dirichlet distribution is used to divide training images among 100 clients. The distribution hyperparameter α varies from 0.05 to 1000. Meanwhile, the 2,223,300 samples in the Loan dataset are divided into 50 US states, each of whom represents an FL client. FL prototype is implemented in Python using PyTorch. Each experiment is repeated five times on different populations under the same configuration (e.g., α) and the average results are reported. In the experiments, every party uses SGD as optimizer and trains for E local epochs with local learning rate l_r, and a batch size of 50. 10% of the participants are randomly selected in each round to submit locally computed SGD updates for aggregation.

Attack Settings. In the label-flipping attack, a proportion $m = 5\%$, 10%, 20%, 33% and 50% of the clients are malicious. We simulate two types of poisoning attacks: (1) Label-flipping attacks: the adversaries attempt to flip a randomly selected source label (S) of the training samples to a target (adversarial) label (T), while keeping the features unchanged (Section 5.2). And (2) Pixel-pattern backdoor attacks: the adversaries embed trigger patterns to the selected training samples before flipping their labels (Section 5.3).

Metrics. We evaluate the *model accuracy* (MA) and the *attack success rate* (ASR) on the testing data. ASR is defined as: $ASR = N_T/N_S$, where N_S denotes the number of testing samples with source label S and N_T denotes the number of testing samples mislabeled as T. The attack succeeds if the poisoned model outputs the desired target label T for a source label S, otherwise the attack fails.

Table 1: Datasets and performance of unattacked models. Model Accuracy: MA.

	#clients	#classes	train/test	feature	model used	MA	l_r/E
MNIST	100	10	60K/10K	784	1-layer softmax [14]	90%	0.01/1
CIFAR-10	100	10	50K/10K	1024	lightweight Resnet-18	84%	0.1/2
LOAN	50	3	1,778K/444K	127	3 fc	85%	0.001/1

Table 2: FoolsGold [14] performance with varying client data distributions and Dirichlet-distributed non-iid data. A-5 attack; 10 honest and 5 malicious clients.

	shared data [14]			Dirichlet-distributed non-iid data				
	s=0%	s=50%	s=100%	α=0.05	α=1	α=10	α=100	α=1000
MA (%)	81.9	89.1	87.8	75.4	74.2	79.9	79.0	79.5
ASR (%)	0.0	1.7	6.7	1.7	1.8	2.1	2.8	3.1

Baseline. We employ FoolsGold [14] as the baseline. We first implement the A-5 label-flipping attack on MNIST with the same settings as [14]. Table 2 demonstrates FoolsGold's performance with various sharing ratios (s), where s=0% implies that each client owns a single label of samples, while s=100% implies that each client's dataset is uniformly sampled from all classes (iid data). Meanwhile, we also evaluate FoolsGold with more realistic, Dirichlet-distributed non-iid data (when $\alpha \to \infty$, data distributions are close to iid). As shown in Table 2, the model accuracy drops significantly with Dirichlet-distributed data.

5.2 Defense against Label-flipping Attacks

Label-flipping Attack on Dirichlet-distributed Client Data. We evaluate CONTRA and FoolsGold [14] with Dirichlet-distributed non-iid data. We simulate 100 clients for MNIST and CIFAR, and 50 clients for Loan, which is more practical than 15 clients in [14]. The results are shown in Table 3. First, FoolsGold's model accuracy drops significantly with increased clients, as benign inputs are mistakenly penalized. CONTRA outperforms FoolsGold in all settings. FoolsGold achieves an overall average accuracy of 76.3% and ASR of 3.2%, while CONTRA increases model accuracy to 80.8% and reduces ASR to 0.9%. In particular, the average model accuracy degradation (compared with unattacked models) is 9.9% with FoolsGold and 5.5% with CONTRA. We also compare the performance of CONTRA and Krum under the label-flipping attack and show the results in Figure 3. The results indicate that CONTRA is robust and effective, and it outperforms state-of-the-art solutions.

Coordinated Attack. We simulate the scenario that malicious clients coordinate and manipulate their data. Similar to [14], we denote the data sharing rate among the malicious clients as x_s, i.e., x_s% of the samples at each malicious client are uniformly distributed from all classes, while the rest $(1-x_s)$% are attack samples that are exclusive to this client. Data at the honest clients follow Dirichlet distribution with various α. Results on MNIST data (Table 4) indicate that both FoolsGold and CONTRA effectively reduce ASR, while CONTRA out-

Table 3: Performance comparison of CONTRA and FoolsGold [14] under a label-flipping attack: MA and ASR at model convergence.

		MNIST				CIFAR-10				LOAN			
		MA (%)		ASR (%)		MA (%)		ASR (%)		MA (%)		ASR (%)	
m	α	FG	Ours	FG	Ours	FG	Ours	FG	Ours	FG	Ours	FG	Ours
5%	0.05	73.35	75.98	**0.0**	**0.0**	68.23	72.41	**1.3**	**0.0**				
	1	81.86	84.21	1.2	**0.0**	73.63	76.88	1.8	**0.0**				
	10	**84.60**	**89.52**	1.8	**0.0**	78.81	83.27	1.4	**0.0**	80.76	83.11	2.1	0.0
	100	83.72	88.16	1.9	1.1	**81.34**	**84.41**	1.4	**0.0**				
	1000	83.69	88.02	2.1	1.0	80.55	83.22	2.8	0.1				
10%	0.05	70.18	73.92	**1.1**	**0.0**	68.45	71.48	**1.4**	**0.0**				
	1	81.77	83.59	1.3	**0.0**	74.84	77.95	2.2	1.0				
	10	**81.79**	**85.93**	2.3	0.8	78.04	82.19	1.5	1.3	79.34	82.44	3.1	0.0
	100	80.67	85.47	2.6	1.3	78.54	82.14	1.7	0.7				
	1000	80.75	85.44	2.8	**0.0**	**78.80**	**83.47**	3.8	1.2				
20%	0.05	67.98	73.22	4.2	**0.0**	66.29	71.28	2.9	**0.0**				
	1	74.58	82.18	**1.6**	**0.0**	72.21	74.66	1.9	**0.0**				
	10	79.54	**85.65**	2.7	1.1	**78.04**	**83.19**	1.7	1.3	77.35	83.25	3.5	0.9
	100	**80.40**	85.32	3.0	1.1	77.69	82.34	2.9	0.5				
	1000	79.64	85.22	4.3	1.9	77.76	82.63	3.8	1.5				
33%	0.05	69.32	72.82	5.6	1.4	67.82	70.42	3.1	**0.0**				
	1	77.24	80.16	**2.2**	0.9	70.08	72.97	**2.0**	0.4				
	10	**79.20**	83.63	3.1	1.7	73.63	79.87	2.6	1.7	76.42	81.47	5.4	1.8
	100	78.34	**83.86**	3.5	**0.8**	73.86	80.43	5.9	1.3				
	1000	78.55	83.20	3.4	1.7	**74.59**	**81.04**	4.8	1.3				
50%	0.05	66.57	70.42	3.2	1.5	67.75	70.41	3.3	1.8				
	1	75.66	81.64	**1.4**	**0.3**	69.11	73.70	**2.4**	**0.8**				
	10	75.32	81.44	2.9	1.4	71.47	74.62	2.5	1.8	74.88	81.23	6.1	2.3
	100	**76.13**	81.56	4.1	1.5	72.12	**78.41**	6.6	1.9				
	1000	74.55	**82.34**	4.2	2.2	**72.53**	77.38	7.8	2.0				
Mean		77.41	82.15	2.66	0.87	73.85	78.03	2.94	0.82	77.75	82.30	4.04	1

Table 4: CONTRA performance against coordinated attacks on MNIST.

	FoolsGold				Ours			
	$\alpha = 0.05$		$\alpha = 1000$		$\alpha = 0.05$		$\alpha = 1000$	
x_s	MA (%)	ASR (%)	MA (%)	ASR (%)	MA (%)	ASR (%)	MA (%)	ASR (%)
0%	0.6835	0.034	0.8322	0.042	0.7313	0.00	0.8704	0.00
50%	0.6981	0.033	0.81	0.038	0.7287	0.012	0.852	0.011
100%	0.7012	0.00	0.7919	0.056	0.7502	0.00	0.8414	0.021

performs FoolsGold in all settings, as CONTRA's reputation system increases the likelihood of honest clients to be recruited for FL training.

Hyperparameters. We evaluate CONTRA on MNIST with local epoch count $E \in \{1, 5\}$ and reporting fraction $C \in \{0.1, 0.2, 0.4\}$, which corresponds to 10, 20, and 40 clients participating in each round, respectively. Note that C=0.1 is the

Table 5: Model accuracy (%) for different parameter settings. Total client population: 100; malicious clients: 33%; C: reporting fraction.

	local epoch count $E = 1$					local epoch count $E = 5$				
C	α=1000	α=100	$\alpha = 10$	$\alpha = 1$	α=0.05	α=1000	α=100	$\alpha = 10$	$\alpha = 1$	α=0.05
0.40	78.48	76.72	84.19	81.32	76.33	87.92	86.59	84.21	84.06	78.73
0.20	76.85	75.43	85.04	82.20	78.40	88.96	87.43	87.65	83.21	79.14
0.10	83.20	83.86	83.63	80.16	75.38	86.70	85.86	85.41	83.90	75.67

Table 6: Performance comparison of CONTRA (ours) and FoolsGold [14] under backdoor attacks. Malicious clients: 33%; $\alpha = 100$.

MNIST				CIFAR-10				LOAN			
MA (%)		ASR (%)		MA (%)		ASR (%)		MA (%)		ASR (%)	
FG	Ours	FG	Ours	FG	Ours	FG	Ours	FG	Ours	FG	Ours
81.4	87.4	18	2.86	79.6	82	3.85	0.9	81.29	83.2	6.14	1.95

most common setting in the literature, which is used in all previous experiments. Table 5 shows the classification performance. With the increase of C, for similarly distributed client data (larger α), more honest clients are likely to be penalized by the detection scheme resulting in drops in model accuracy. Meanwhile, with smaller α (non-iid data), honest client updates are diverse and differentiated from the sybil updates, therefore, increasing C improves model accuracy by involving more honest clients. Moreover, synchronizing the weights after 5 local epochs significantly improves the model accuracy, especially for larger α.

Computing and Communication Overhead Last, we evaluate the runtime overhead introduced by CONTRA. All experiments are performed on a machine with a 2.6 GHz Intel core i5 processor and 128 GB RAM. The baseline FL system with 100 honest clients takes 18.43 seconds, on average, to run 100 training rounds. With 33% of malicious clients and CONTRA deployed, the average runtime increases to 29.36 seconds. This corresponds to a relative slowdown of approximately 1.60x compared to the baseline, which is very acceptable. The size of the messages exchanged is less than 7 MB in both cases.

5.3 Defense against Backdoor Attacks

For MNIST and CIFAR, adversarial images are selected from a random class. We embed a 4×4 pattern with gray-scale 255 (or 255 in all RGB channels) to the top left corner of each adversarial image, and assign it with an adversarial label. For the Loan dataset, six low-importance features from a random label are chosen. They are replaced with new values that are slightly larger than the maximum value of the feature. To remain stealthy, each attacker's batch has 20 backdoored samples mixed with correctly labeled data.

Experimental results are reported in Table 6. For the lightweight Resnet-18 model on CIFAR-10, the ASR is 3.85% for FoolsGold and under 1% for CONTRA. Meanwhile, with the 1-layer softmax model on MNIST, the ASR is 18% with

FoolsGold, while CONTRA reduces it to 2.86%. The accuracy on Loan does not drop much because of the simple fully-connected model architecture. The ASR is 6.14% and 1.95% with FoolsGold and CONTRA, respectively.

6 Conclusion

Federated learning systems are vulnerable to poisoning attacks, in which adversaries manipulate their local data/label/model and contribute maliciously generated updates to the aggregator, with the intention to degrade the accuracy of the global model or to inject a backdoor into it. We have observed that existing defense mechanisms fall short when the experiment configurations are akin to real world settings, e.g., with larger number of FL clients and non-iid data distributions. In this paper, we present CONTRA, a reputation-based defense against poisoning attacks in federated learning systems. In particular, we identify the different optimization objectives of the honest and adversarial FL participants. In response, we develop a cosine-based similarity measurement for client contributions. We further design a reputation-based approach to dynamically and adaptively limit the contribution of the potentially malicious clients. Through extensive experiments with three popular ML datasets, we demonstrate that CONTRA provides outstanding performance: it reduces the attack success rate to 1%, and maintains significantly higher model accuracy than state-of-the-art defense mechanisms.

Acknowledgements

The authors were sponsored in part by NSF IIS-2014552, DGE-1565570, DGE-1922649, and the Ripple University Blockchain Research Initiative. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Arnold, S., Yesilbas, D.: Demystifying the effects of non-independence in federated learning. arXiv preprint arXiv:2103.11226 (2021)
2. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. In: International Conference on Artificial Intelligence and Statistics. pp. 2938–2948. PMLR (2020)
3. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: ACM Symposium on Info., computer and comm. security (2006)
4. Barreno, M., Nelson, B.N., Joseph, A.D., Tygar, J.: The security of machine learning. In: Machine Learning, 81(2). (2010)
5. Baruch, G., Baruch, M., Goldberg, Y.: A little is enough: Circumventing defenses for distributed learning. Advances in Neural Information Processing Systems **32**, 8635–8645 (2019)

6. Beaufays, F., Rao, K., Mathews, R., Ramaswamy, S.: Federated learning for emoji prediction in a mobile keyboard (2019), https://arxiv.org/abs/1906.04329
7. Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.: Model poisoning attacks in federated learning. In: Workshop on Security in Machine Learning (SecML) (2018)
8. Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.: Analyzing federated learning through an adversarial lens. In: the 36th Intl. Conf. on Machine Learning (2019)
9. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: Proceedings of the 29th International Coference on International Conference on Machine Learning. p. 1467–1474 (2012)
10. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., Stainer, J.: Machine learning with adversaries: Byzantine tolerant gradient descent. In: the 31st International Conference on Neural Information Processing Systems. pp. 118–128 (2017)
11. Cao, X., Fang, M., Liu, J., Gong, N.Z.: FLTrust: Byzantine-robust federated learning via trust bootstrapping (2020)
12. Chen, Y., Su, L., Xu, J.: Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. In: POMACS, 1:44:1–44:25, 2017. (2017)
13. Fang, M., Cao, X., Jia, J., Gong, N.: Local model poisoning attacks to byzantine-robust federated learning. In: 29th USENIX Security Symposium (2020)
14. Fung, C., Yoon, C.J.M., Beschastnikh, I.: The limitations of federated learning in sybil settings. In: 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID). pp. 301–316 (2020)
15. Ganju, K., Wang, Q., Yang, W., Gunter, C.A., Borisov, N.: Property inference attacks on fully connected neural networks using permutation invariant representations. In: ACM Conf. on Computer and Comm. Security. pp. 619–633 (2018)
16. George, N.: Lending club loan data (version 3). https://www.kaggle.com/wordsforthewise/lending-club (April 2019)
17. Gu, T., Dolan-Gavitt, B., Garg, S.: Badnets: Identifying vulnerabilities in the machine learning model supply chain (2019)
18. Hsu, T.M.H., Qi, H., Brown, M.: Measuring the effects of non-identical data distribution for federated visual classification. arXiv preprint arXiv:1909.06335 (2019)
19. Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B.: Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In: the 39th IEEE Symposium on Security and Privacy (2018)
20. Kaissis, G.A., Makowski, M.R., Rückert, D., Braren, R.F.: Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence **2**(6), 305–311 (2020)
21. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: Strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
22. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
23. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
24. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)
25. McMahan, B., Ramage, D.: Federated learning: Collaborative machine learning without centralized training data (2017), https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

26. Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E.C., Roli, F.: Towards poisoning of deep learning algorithms with back-gradient optimization. In: the 10th ACM Workshop on Artificial Intelligence and Security. pp. 27–38 (2017)
27. Muñoz-González, L., Co, K.T., Lupu, E.C.: Byzantine-robust federated machine learning through adaptive model averaging. arXiv preprint arXiv:1909.05125 (2019)
28. Nguyen, T.D., Rieger, P., Yalame, H., Möllering, H., Fereidooni, H., Marchal, S., Miettinen, M., Mirhoseini, A., Sadeghi, A.R., Schneider, T., et al.: FLGUARD: Secure and private federated learning. arXiv preprint arXiv:2101.02281 (2021)
29. Shafahi, A., Huang, W.R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., Goldstein, T.: Poison frogs! targeted clean-label poisoning attacks on neural networks. In: 32nd Intl. Conf. on Neural Info. Processing Systems. pp. 6106–6116 (2018)
30. Shejwalkar, V., Houmansadr, A.: Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In: Network and Distributed Systems Security (NDSS) Symposium, 2021 (2021)
31. Shen, S., Tople, S., Saxena, P.: Auror: defending against poisoning attacks in collaborative deep learning systems. In: In Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, Los Angeles, CA, USA, December 5-9, 2016. 508–519. (2016)
32. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 3–18 (2017)
33. Steinhardt, J., Koh, P.W., Liang, P.: Certified defenses for data poisoning attacks. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 3520–3532. NIPS'17 (2017)
34. Tolpegin, V., Truex, S., Gursoy, M.E., Liu, L.: Data poisoning attacks against federated learning systems. In: European Symposium on Research in Computer Security. pp. 480–501. Springer (2020)
35. Wu, C., Yang, X., Zhu, S., Mitra, P.: Mitigating backdoor attacks in federated learning. arXiv preprint arXiv:2011.01767 (2020)
36. Xie, C., Huang, K., Chen, P.Y., Li, B.: Dba: Distributed backdoor attacks against federated learning. In: Intl. Conf. on Learning Representations (2020)
37. Yin, D., Chen, Y., Kannan, R., Bartlett, P.: Byzantine-robust distributed learning: Towards optimal statistical rates. In: 35th Intl. Conf. on Machine Learning (2018)
38. Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., Khazaeni, Y.: Bayesian nonparametric federated learning of neural networks. In: International Conference on Machine Learning. pp. 7252–7261. PMLR (2019)