# Defending against Data Poisoning Attacks: From Distributed Learning to Federated Learning

Yuchen Tian[1], Weizhe Zhang[1], Andrew Simpson[2], Yang Liu[1] and Zoe Lin Jiang[1]

[1]*College of Computer Science and Technology, Harbin Institute of Technology(Shenzhen), Shenzhen, 518055, China*
[2]*Department of Computer Science, University of Oxford, Oxford OX1 3QD, UK*
*Email: liu.yang@hit.edu.cn*

**Federated Learning (FL), a variant of Distributed Learning (DL), supports the training of a shared model without accessing private data from different sources. Despite its benefits with regards to privacy preservation, FL's distributed nature and privacy constraints make it vulnerable to data poisoning attacks. Existing defenses, primarily designed for DL, are typically not well adapted to FL. In this paper, we study such attacks and defenses. In doing so, we start from the perspective of DL and then give consideration to a real-world FL scenario, with the aim being to explore the requisites of a desirable defense in FL. Our study shows that: (a) the batch size used in each training round affects the effectiveness of defenses in DL; (b) the defenses investigated are somewhat effective and moderately influenced by batch size in FL settings; and (c) the non-IID data makes it more difficult to defend against data poisoning attacks in FL. Based on the findings, we discuss the key challenges and possible directions in defending against such attacks in FL. In addition, we propose DSPO (Detect and Suppress the Potential Outliers), a defense against data poisoning attacks in FL scenarios. Our results show that DSPO outperforms other defenses in several cases.**

*Keywords: Distributed Learning; Federated Learning; Data Poisoning Attacks; AI Security*

*Received 00 January 2009; revised 00 Month 2009; accepted 00 May 2009*

## 1. INTRODUCTION

The remarkable performance of machine learning in some real-world tasks (e.g., computer vision, personalized recommendation, and speech recognition) has made it a key foundation of the emerging data-driven society. In parallel, the dramatic increase of available data and increasingly complex machine learning models have seen high demands for computation resources in the model training stage. Consequently, there is evidence of a shift towards a distributed learning paradigm that leverages distributed data sources to achieve model training at a large scale [1,2]. Such an approach is termed *Distributed Learning* (DL) [3].

*Federated Learning* (FL) [4], a variant of DL that can address privacy leakage issues during model training, has gained a lot of attention in recent years. FL's main feature (especially for sensitive tasks) involves keeping the raw data (e.g., face images, location information, and medical records) where they are generated (e.g., smartphones, IoT devices, and medical centres). In each training round, different data sources are considered as participants/clients. The central server first broadcasts the global model to a subset of them. Then the selected clients perform model training locally using the raw data. Finally, the optimized local models are sent back to the server to yield an updated global model. These steps are repeated until a desirable global model is constructed. Thus, the model parameters, rather than sensitive data, are transferred between server and clients to achieve a degree of privacy preservation [5].

Although FL is considered to be a promising privacy-preserving learning framework, its inherently distributed nature and its privacy-aware design bring new vulnerabilities [6]. In the FL system, potential malicious participants (adversaries) can report crafted local models to the server. As a consequence, the aggregated global model may suffer from degraded accuracy [7] or misbehave in certain tasks desired by the adversaries [8]. Such attacks are called *poisoning attacks* [9]. In particular, Blanchard *et al.* [10]

show that the global model can be corrupted in the distributed learning environment by only one adversary, which highlights the need for a robust FL system.

In recent years a variety of defense strategies [11–14] have been proposed to defend against poisoning attacks. However, many of these approaches focus on scenarios involving centralized learning or DL, not giving consideration to the unique features of FL. As an example, in DL, where the server has access to the data distributed among a cluster of machines, poisoned local models can be detected by evaluating the performance on some trusted data samples [13, 15]. However, data is preserved in local devices or organizations in FL, thus the server has no visibility to the training data, rendering the defenses inapplicable. Another typical defense in DL, the *robust aggregation rule* [10, 14, 16], tries to remove the potentially malicious update via their statistical properties. The defense assumes the data partitions are independent and identically distributed (IID). However, FL involves non-IID data across different clients. Due to the similarities that exist between DL and FL, some DL defenses have been applied to FL in recent literature [17, 18]. However, their performances and applicability to FL have not been explored fully. The above observations give rise to a key question to be answered: *What happens to current defenses when they are applied to defending against poisoning attacks in FL?*

In this paper we study the vulnerability of FL, with a focus on data poisoning attacks [19] where the adversaries can only manipulate the locally stored data. We evaluate the effectiveness of several defenses, exploring their usability and possible directions for further enhancements. Based on the evaluation results, we argue that the unique features of FL mean that it is important to design desirable defenses against data poisoning attacks. Subsequently, we propose a defense strategy named DSPO (for *Detect and Suppress the Potential Outliers*) that is designed with consideration to FL's features. Finally, we demonstrate its feasibility via the experimental results from two real-world datasets.

Our main contributions are summarized as follows.

- We conduct an empirical study evaluating data poisoning attacks and corresponding defenses in FL, showing the limited effectiveness of the defenses in such a learning scenario.
- We summarize the key challenges and potential directions in defending against data poisoning attacks in the context of FL.
- Based on our findings, we propose DSPO, a defense strategy that curbs the impact of poisoned updates effectively and leads to a negligible accuracy loss in FL setting.

The remainder of the paper is organized as follows. Section 2 gives an overview of related work. In Section 3 we present the required background knowledge for this work. Section 4 provides details about our experimental settings, the problems investigated, and corresponding results and findings. In Section 5 we discuss the key challenges in defending against data poisoning attacks in FL environment. Section 6 introduces the proposed defense and demonstrates its effectiveness in FL. Finally, we draw conclusions and consider possible future work in Section 7.

## 2. RELATED WORK

In this section we discuss the current literature related to our work. We first show the different types of poisoning attacks. We then give consideration to potential defenses that have been proposed in the literature.

### 2.1. Poisoning attacks

Poisoning attacks involve adversaries deliberately injecting malicious information into the model training process, aiming to degrade the model's utility. Many attacks achieve the goal by modifying the original data samples [19, 20] or by inserting crafted ones into the training data [21–23]. These attacks are also termed *data poisoning*.

*Label-flipping attacks* [19, 24, 25] and *backdoor attacks* [17, 18, 26] are two representative instances of data poisoning. Label-flipping attacks decrease the performance of a trained model by manipulating the labels of a training set adversarially. In backdoor attacks, adversaries can embed backdoors into the model, by attaching well-designed features to the data examples. In this way, the model only misbehaves when facing inputs with backdoor triggers, making it hard to be detected. Bagdasaryan *et al.* [18] considered a backdoor attack which replaced the global model with a poisoned local model by scaling up the adversaries' updates. Xie *et al.* [17] proposed a distributed backdoor attack on FL that decomposes the global backdoor pattern into several local ones and embeds them into different adversaries.

The distributed learning environment gives rise to a threat called *model poisoning* [27–30]. Adversaries in a model poisoning attack can modify the updates (usually gradients or local model parameters) sent to the server directly. Fang *et al.* [30] treated the attack process as an optimization problem considering the goal of adversaries. In each iteration, the adversaries sent crafted local models to the server by solving that problem. Bhogoji *et al.* [28] proposed a stealthy model poisoning involving certain normalization into the adversaries' objective; the poisoned local models are then close to the benign ones so that they can bypass several defenses.

Depending on the intention of poisoning attacks, they can also be divided into two classes: *targeted poisoning* and *untargeted poisoning*. Untargeted poisoning aims

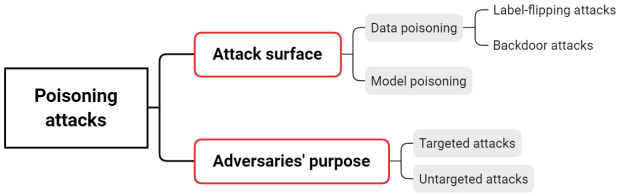



FIGURE 1: Taxonomy of poisoning attacks with different categorization criteria.

to decrease the overall model accuracy or disturb the convergence of the model in the training phase. By contrast, target poisoning only influences the output of certain examples decided by adversaries. For example, a poisoned malware detection model may classify certain categories of malware as benign. Additionally, the prediction of other inputs remains unaffected.

Figure 1 shows a visual depiction of the taxonomy of poisoning attacks.

### 2.2. Defenses

A class of previous works [11, 12, 31], relying on data sanitization methods, is concerned with the identification and removal of poisoned samples from the whole training data. However, these methods are mainly designed for conventional centralized learning or DL, where the server can access the training data directly. This prerequisite hinders the methods' application to FL, where the data is invisible to the server.

To tackle this problem, another class of defenses [13–15], involves measuring the quality of updates sent by clients. These approaches assume that the server maintains a validation set on behalf of the whole training data. Updates showing an undesirable performance are then removed during aggregation. Xie *et al.* [13] ranked gradients from different clients based on the estimated descent of the loss function on the validation set. Pan *et al.* [14] utilized reinforcement learning techniques to help the server learn to identify the poisoned local models. Despite their effectiveness in DL, they are inapplicable to FL, as it is infeasible for the server to construct a desired validation set.

Zhao *et al.* [32] proposed a defense shifting the evaluation to the clients: the server then decides the potentially malicious updates via the reports from them. As FL is communication-sensitive, the approach doubles the communication cost due to the extra information transferred.

Byzantine-resilient defenses [10, 16, 33] have been proposed to cope with the Byzantine failures, where a subsection of clients in the distributed system can send arbitrary results to the server. They can provide robustness by eliminating the malicious effects, with theoretical convergence guarantees during model aggregation. We term these defenses *robust aggregation rules*. However, the guarantee is under the assumption that the data of each client is IID (independent and identically distributed) and the effectiveness (to the best of our knowledge) has yet to be studied in the context of FL featuring non-IID data.

Fung *et al.* [34] proposed a method to distinguish the malicious updates in non-IID setting, based on the assumption that the updates of adversaries are have a high degree of similarity. But this approach adapts poorly to the scenario where there is only one adversary or benign clients with similar data distribution.

### 2.3. Synthesis

Taken together, the above shed light on the vulnerabilities of interest of DL and FL. However, most prior attacks made potentially unrealistic assumptions about the capability of adversaries. Additionally, the defenses failed to fully consider the unique features and constraints in FL. Therefore, in the following, we discuss poisoning attacks and defenses in FL from what we consider to be a more realistic perspective.

## 3. BACKGROUND AND PRELIMINARIES

In this section we present the necessary background information about DL and FL. Additionally, we introduce the label-flipping attack and robust aggregation rules (as representative attack and defenses) in the context our work.

### 3.1. DL and FL

In a distributed environment, the learning process is usually managed by a parameter-server (PS) framework [35], where a central server aggregates the computational results from clients to update the global model. Take the following example as an illustration. Suppose that we have $k$ clients and that we indicate that the $i$th client owns a local training dataset via $D_i$. With their own datasets and the coordination of the server, they collaboratively train a usable model, $w$. The training is a process of solving the optimization problem: $\min_{w\in\mathbb{R}^d}[\frac{1}{k}\sum_{i=1}^{k} L(w, D_i)]$. Here, $L(w, D_i)$ is the loss function measuring how well the model $w$ fits the local training dataset $D_i$.

In DL, the system performs a distributed implementation of mini-batch Stochastic Gradient Descent (mini-batch SGD) to minimize the objective function. Specifically, in training round $t$, the $i$th client randomly samples a batch of data examples $B_i^t$, where $B_i^t \subseteq D_i$ and $b = |B_i^t|$ refers to the batch size. Then, it computes a gradient $g_i^t = \frac{1}{b}\nabla_w L(w^t, B_i^t)$ as an update sent to the server. These operations are conducted in parallel between all clients. Finally, the server averages the updates to adjust the global model, i.e., $w^{t+1} = w^t - \frac{\eta}{k}\sum_{i=1}^{k} g_i^t$, where $\eta$ is the learning rate controlling the step size of gradient descent.

As for FL, if the participants are smartphones or IoT devices, only a subset of them is selected in training

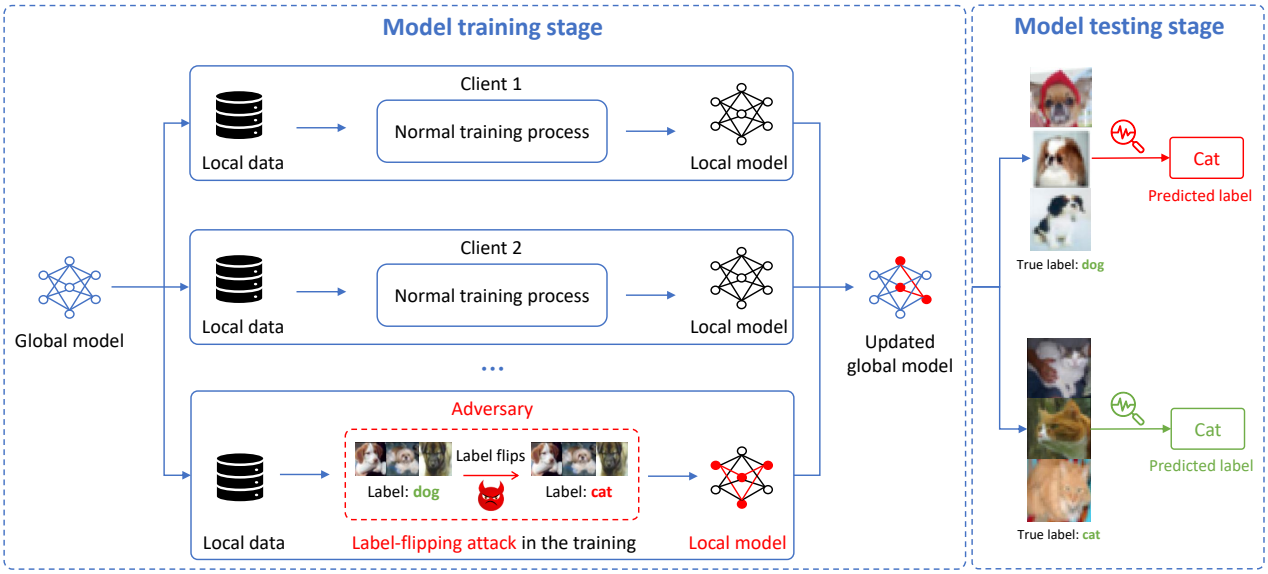



FIGURE 2: Work flow of label-flipping attack.

round $t$ based on their availability. Suppose we have $c$ selected clients, where $1 \leq c \leq k$. FL solves the optimization problem using the FedAvg [4] algorithm, where the $i$th client performs mini-batch SGD locally for $e$ epochs on $D_i$, with every epoch involving multiple iterations of updating its local model via $w_i^t = w_i^t - \frac{\eta}{b}\nabla_w L(w_i^t, B_i)$. The server then averages the updated local models to yield a new global model via $w^{t+1} = \frac{1}{c}\sum_{i=1}^{c} w_i^t$.

### 3.2. Label-flipping attack

A label-flipping attack is an instance of data poisoning attack, whereby adversaries corrupt some data examples by maliciously changing their labels before training. Figure 2 illustrates how label-flipping works in FL. Such an attack can be both targeted and untargeted. Taking a classic classification task as an example, we formulate the attack as follows.

A label-flipping attack is a malicious attempt to flip an original label $l_0 \in \mathcal{L}$ to another label $l \in \mathcal{L}$, where $\mathcal{L}$ is a collection of all labels of the training data. Consider a mapping $\mathcal{A} : \mathcal{L} \to \mathcal{L}$, we use $l = \mathcal{A}(l_0)$ to denote the attack process.

For *targeted label flipping*, the adversaries decide on a source label $l_s$ and a targeted label $l_t$. This can be described as $l = \mathcal{A}(l_0)$, such that $l_0 = l_s, l = l_t$. For convenience we abbreviate this as $\mathcal{A}_{l_s}^{l_t}$. In test time, this can increase the error rate that the model misclassifies examples with label $l_s$ to $l_t$. For example, in a traffic sign classification task, the model may identify stop signs as speed limit signs with high confidence.

As for *untargeted label flipping*, the adversaries randomly change every label to one that is different from its original label: $l = \mathcal{A}(l_0)$, such that $l \neq l_0$. Therefore, the model may suffer from accuracy loss on the overall classification tasks.

Label flipping does not require any prior knowledge about the training, such as the algorithm used in the server [30], the local updates from benign clients [10], or details about model architectures [17]. Moreover, the training data resides locally in FL, making it practical for even non-expert adversaries to launch a label-flipping attack. In this work, we use a label-flipping attack to implement data poisoning in FL.

### 3.3. Robust aggregation rules

Byzantine-resilient defenses, designed for an attack scenario whereby adversaries can send arbitrarily malicious updates to the server, strive to eliminate the adverse impact of malicious updates while providing theoretical convergence guarantees. Instead of simply averaging the updates, such approaches yield robust estimations of the values via certain robust aggregation rules. We provide details of several such rules below.

Let $\mathcal{V} = \{v_i | i \in [k]\}$ be a set of updates, such that the $i$th element $(v_i)$ is a one-dimensional vector transformed from the original local model $w_i$ or gradient $g_i$ for the convenience of further usage in different defenses. Such transformation can unify the analysis in both learning scenarios (i.e., DL and FL).

**Krum [10].** Krum keeps only one update having the lowest score, whose score is the sum of distances to the other closest $k - m - 2$ updates, where $m$ is the number of adversaries and the distance is measured by Euclidean distance. The process can be expressed by:

$$Krum(\mathcal{V}) = v_i, \quad i = \arg\min_{i \in [k]} \sum_{v \in \mathcal{V}_i} ||v_i - v||^2 \qquad (1)$$

Here, $\mathcal{V}_i$ denotes the subset of $\mathcal{V}$, which contains exactly $k-m-2$ updates closest to $v_i$. Krum can theoretically

guarantee the convergence when $m < \frac{k-2}{2}$ under the IID data distribution.

**Median [16].** Median handles each dimension of the updates separately: it sorts parameters on each dimension and choose the median value, recomposing them as the final result. We define an operator $med(\cdot)$, which is able to pick the median value of a given set. If $v_{i(j)}$ be the parameter in $j$th dimension of $v_i$, the median can be formulated as:

$$Median(\mathcal{V}) = v_r, \quad v_{r(j)} = med(\{v_{i(j)} | i \in [k]\}), j \in [d] \tag{2}$$

Here, $d$ is the dimension of $v_i$.

**Trimmed Mean [16].** Similar to Median, this aggregation rule aggregates each parameter separately. Specifically, it first sorts the $j$th parameters of the updates, and then trims off the top $\beta k$ and the last $\beta k$ parameters, where $\beta \in [0, \frac{1}{2})$ denotes the trimmed rate.

We use $\mathcal{U}_j^\beta = \{v_{i(j)} | i \in [(1-2\beta)k]\}$ to denote the trimmed subset on $j$th dimension. The final update can be obtained by averaging the parameters on each dimension(we use $Trmean$ for short):

$$Trmean(\mathcal{V}, \beta) = v_r, \quad v_{r(j)} = \frac{1}{|\mathcal{U}_j^\beta|} \sum_{i \in \mathcal{U}_j^\beta} v_{i(j)}, j \in [d] \tag{3}$$

Although the above defenses are designed for DL, they allow the server to defend adversaries by only inspecting the updates. Moreover, without modification to the original training algorithm, they are formally well-compatible to FL. In the following, we further investigate their applicability and performance in the context of FL.

## 4. DATA POISONING ATTACKS AND DEFENSES IN FL

In this section we present our experiment and our main findings. We first describe the experimental settings, including datasets, threat model, and details of label-flipping attacks. Subsequently, we explain three problems investigated with corresponding results and analysis.

Table 1 summarizes the key notations used in this paper.

TABLE 1: Key notations used in this paper.

| Notation | Description |
|---|---|
| $k$ | The number of clients |
| $m$ | The number of adversaries |
| $w_i^t$ | The local model of the $i$th client in FL on the training round $t$ |
| $g_i^t$ | The gradient computed by $i$th client in DL on the training round $t$ |
| $v_i$ | the update uploaded by $i$th client |
| $v_{i(n)}$ | the parameter of $v_i$ in $n$th dimension |
| $\mathcal{V}$ | a set of updates in one training round |
| $p$ | The degree of non-IID |
| $b$ | The batch size adopted by each client |
| $[k]$ | A set of consecutive integers: $\{1, ..., k\}$ |
| $\mathcal{A}_{l_s}^{l_t}$ | Targeted label-flipping attack, flipping $l_s$ to $l_t$ |
| $\Delta w_{acc}$ | The model accuracy loss |
| $ASR$ | The attack success rate of targeted attack |
| $med(\cdot)$ | Return the median value of a given set |

### 4.1. Experimental set-up

*4.1.1. Datasets and models*

Our experiments involved two image classification datasets: Fashion-MNIST[3] and CIFAR-10[4]. The former, Fashion-MNIST, is composed of a training set with 60,000 examples and a test set with 10,000 examples. Each example is a pair consisting of a 28x28 greyscale image and an associated label. The labels are drawn from 10 classes such as coat, dress, and bag. The latter, CIFAR-10, consists of 60,000 32x32 colour images in 10 classes such as dog, cat, and ship. Of these, 50,000 examples are for training and the other 10,000 ones are for testing.

For Fashion-MNIST, we used a multi-layer perceptron (MLP) with one hidden layer having 256 units. As for CIFAR-10, we chose a CNN model, which is composed of 5 convolutional layers and 2 fully-connected layer. As our goal is to evaluate and compare the effectiveness of the defenses, not to achieve the possible high accuracy of the models on the datasets, we adopted vanilla SGD with a constant learning rate as the optimization method.

*4.1.2. FL settings*

We implemented FL with the PyTorch library, using multi-processing to simulate parallell-computed clients. We considered $k = 20$ clients, with each client locally performing mini-batch SGD for $e$ epochs with default batch size $b = 64$ on the two datasets unless otherwise specified.

As suggested in [4], a large value of $e$ may lead to the model diverging; as such, we set $e = 2$ by default. For MLP and CNN, the learning rate $\eta$ was set to 0.2 and 0.05 respectively. To make our experiments both deterministic and more straightforward, in FedAvg we select all clients to make sure that the adversaries participate in every training round. While these settings do not yield an optimal model with state-of-the-art accuracy, they are sufficient for our purposes. The given experimental results are the average over 3

[3]https://github.com/zalandoresearch/fashion-mnist

[4]https://www.cs.toronto.edu/~kriz/cifar.html

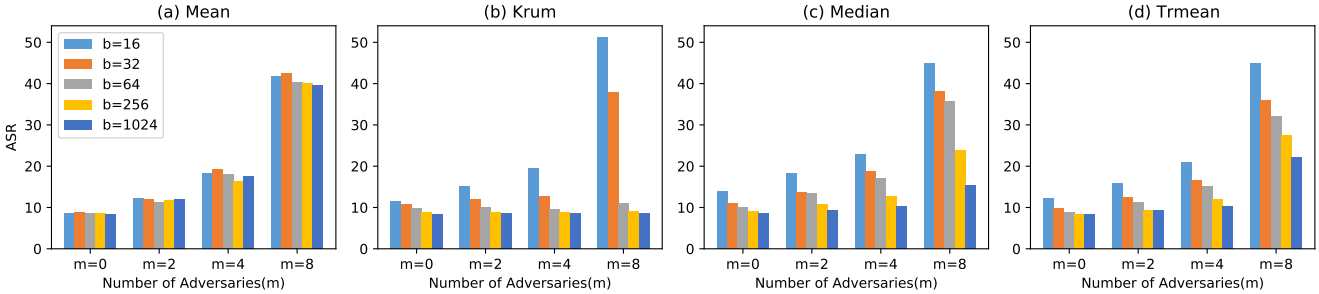



FIGURE 3: *ASR*s of targeted label-flipping attack under different batch sizes using gradient-based aggregation.

runs.

The main feature of FL lies in its non-IID data distribution among all clients. We follow [30] to create such data partitions. Suppose the dataset used for training has $L$ classes. We first divide the clients into $L$ groups, with each group having $\frac{k}{L}$ clients. Then we assign the training samples labelled by $i$ to the $i$th group with probability $p$, and to other groups with probability $\frac{1-p}{L-1}$. Finally, we assign the samples equally to clients within each group. In this procedure, the probability $p$ represents the degree of non-IID, with a larger $p$ indicating that the data differ more. In particular, when $p = \frac{1}{L}$, each training sample is assigned to the groups with equal probability. (This is closest to the IID setting.)

Due to the non-IID property, we argue that it is impractical for the server to maintain a validation set, which is usually a fraction of the training data in centralized learning representing the overall data distribution. Therefore, in our experiments, we only consider FL scenarios where the server has a testing dataset.

#### *4.1.3. Threat model*

We assume a trusted server. We also assume that $m$ of the $k$ clients are adversaries. Since our purpose is to investigate the effectiveness of defenses in FL, we need to ensure the conditions where the defenses claim to be effective. Thus, $m = 8$ (Krum requires $m < \frac{k-2}{2}$) is the maximum number of adversaries we can testify.

The adversaries know their own training data and local model structure, but they are not aware of the training data on other benign clients and the aggregation rules used in the server. The adversaries possess realistic abilities that they can only corrupt their own training data via label flipping; they cannot modify the process of the learning algorithm (e.g., performing more local iterations to enhance the malicious impact [17]).

We assume that the goals of the adversaries are to: (1) mislead the model to report wrong output on certain test examples; and (2) reduce the accuracy of the model on overall tasks.

#### *4.1.4. Data poisoning process*

As mentioned in Section 3.2, we use label-flipping attacks to realise both targeted and untargeted data poisoning. For a targeted attack, in Fashion-MNIST, the adversaries filter out examples whose label is 4 (coat) and change it to 6 (shirt). This is denoted $\mathcal{A}_4^6$. In CIFAR-10, the change is from 5 (dog) to 3 (cat), i.e., $\mathcal{A}_5^3$.

According to the findings of [36], such targeted flipping settings have a higher attack success than other strategies, making the results more obvious. For untargeted attacks, every example's label in the dataset is randomly changed to other labels. Thus the process on the two datasets is identical.

### 4.2. Stability of defenses on gradient-based aggregation

While robust aggregation rules claim to defend adversaries with theoretical guarantees, it is important to understand their performance in realistic tasks. The main idea of such rules is to derive a robust estimation of gradients obtained from clients via statistical features. Thus, the correctness of the gradients contributes to an acceptable final estimation.

For each client, larger batch size $b$ leads to gradients with lower variance computed in different iterations. Consequently, it remains to be answered whether batch size may give rise to the unstable performance of the defenses. As such, we first evaluate the influence of batch size on the effectiveness of such rules to defend against data poisoning attacks.

To explore the internal stability of these rules, we conducted an experiment satisfying all the preconditions required by them. Specifically, we launched targeted label-flipping attack on the Fashion-MNIST dataset under an IID setting.

We set the training rounds to 1,000, and, in each round, the clients update gradients computed from a batch of samples to the server. If the server does not adopt any defense strategy, it simply averages the gradients from the clients.

To make a better comparison, we take the normal operation as *Mean aggregation*. We use attack success

rate ($ASR$) to measure the attack effect:

$$ASR = \frac{n(l_s \to l_t)}{n(l_s)} \times 100\% \quad (4)$$

Here, $n(l_s)$ denotes the number of samples labelled by $ls$ in the test dataset and $n(l_s \to l_t)$ denotes the number of samples misclassified by the model from $l_s$ to $l_t$. The lower ASR is, the better defense ability the rules have. According to our observation, the value of $ASR$ fluctuates during the training, so we take the average of the $ASR$s of the last 200 rounds as the final result.

Figure 3 shows the $ASR$s of the targeted label-flipping attack on the Fashion-MNIST dataset under different numbers of adversaries and varied batch sizes. Interestingly, as shown in Figure 3(a), when $m = 0$, the $ASR$ of compared aggregation rules on Fashion-MNIST is around 8.3%. This can be explained by the highly similar features of the two kinds of data samples, which is hard for MLP to distinguish. Thus, it can be regarded as the best result that one defense can achieve. It is noticeable that the ability of these rules increases as the batch size $b$ grows larger: the larger $b$ contributes to effective defense (as should be the case).

In the worst case, where $m = 8$ (40% clients are adversaries), Krum successfully defends the adversaries, reducing the $ASR$ to 8.6% when $b = 1024$. Median and Trmean, while offering some defense, are less successful.

The results suggest that, in DL where data partition is IID, the studied defenses are effective against targeted data poisoning in the gradient-based aggregation (where the server aggregates gradients computed by clients) — unless we set a large batch size for the training algorithm.

In FL, gradient-based aggregation (e.g., FedSGD [4]) can also be used for the training. However, such robust aggregation rules are not naturally applicable to FL for a number of reasons. First, a large batch size may trap current widely adopted deep learning models into local minimum, giving rise to poor generalization [37]. Second, FL usually involves edge devices with memory and computation power constraints: such limited hardware resources cannot support computation over a large batch of data samples [38]. Third, the high communication cost in distributed environment makes communication-efficient model-based aggregation (e.g., FedAvg) more prevalent in realistic FL.

### 4.3. Feasibility of defenses on model-based aggregation

In FL, a generic model-based aggregation in the server can be summarized as

$$w^{t+1} = \frac{1}{k}\sum_{i=1}^{k} w_t^t \quad (5)$$

In contrast to DL, clients in FL perform more iterations of mini-batch SGD to yield a local model $w_i^t$. We can abstract the effect of multiple model update iterations as $\Delta w_i^t$. As such, the process of local iterations can be formulated as

$$w_i^t = w^t + \Delta w_i^t \quad (6)$$

Here, $w^t$ is the model received from the server at the start of training round $t$. Further, we can understand model-based aggregation as

$$w^{t+1} = \frac{1}{k}\sum_{i=1}^{k}(w^t + \Delta w_i^t) = w^t + \frac{1}{k}\sum_{i=1}^{k}\Delta w_i^t \quad (7)$$

We note that this formulation has a high degree of similarity with gradient-based aggregation in DL (see Section 3.1):

$$w^{t+1} = w^t - \frac{1}{k}\sum_{i=1}^{k}\eta \cdot g_i^t \quad (8)$$

Since $g_i^t$ is computed from a batch of data samples from $D_i$, the poisoned ones may give rise to an incorrect $g_i^t$, whose poisoned features make it distinguishable to the robust aggregation rules. As shown in Section 4.2, a larger batch size represents $D_i$ better, thus yielding a more accurate $g_i^t$. Likewise, $\Delta w_i^t$ is the result of multiple local iterations of mini-batch SGD; it can also reflect the contribution of $D_i$ to $w_i^t$.

We further examine targeted label-flipping attacks on model-based aggregation, to see if such robust aggregation rules are still effective in defending against the attack. The datasets, models and ways to attack are all identical to those of Section 4.2. In particular, we consider a learning scenario closer to FL where FedAvg is used: we set the training rounds to 100 for Fashion-MNIST and CIFAR-10 and average the $ASR$s of last 10 rounds as the final result. The data partition across clients remains IID.

Figure 4 reports the performances of the robust aggregation rules when applying them to FedAvg under the testing conditions as per Section 4.2. When the number of adversaries is less than 8, the rules studied are somewhat effective in defending against targeted label-flipping attacks. Take, for example, the results on Fashion-MNIST when $m = 4$. If we compare Mean aggregation, Krum reduces the $ASR$ from 18.69% to 7.64%; the results for Median and Trmean are 13.35% and 13.56% respectively.

The above are averaged in different batch sizes, as we observe that these rules show more stable defense ability as the batch size varies. However, Median and Trmean fail to reach the best result like they behave in Figure 3. When $m = 8$, an interesting observation is that Krum gives rise to a significantly high $ASR$ when $b = \{16, 32, 64\}$ — which is even higher than that in Mean aggregation (no defense). A possible explanation is that Krum keeps only one local model that happens to be poisoned, which empowers the

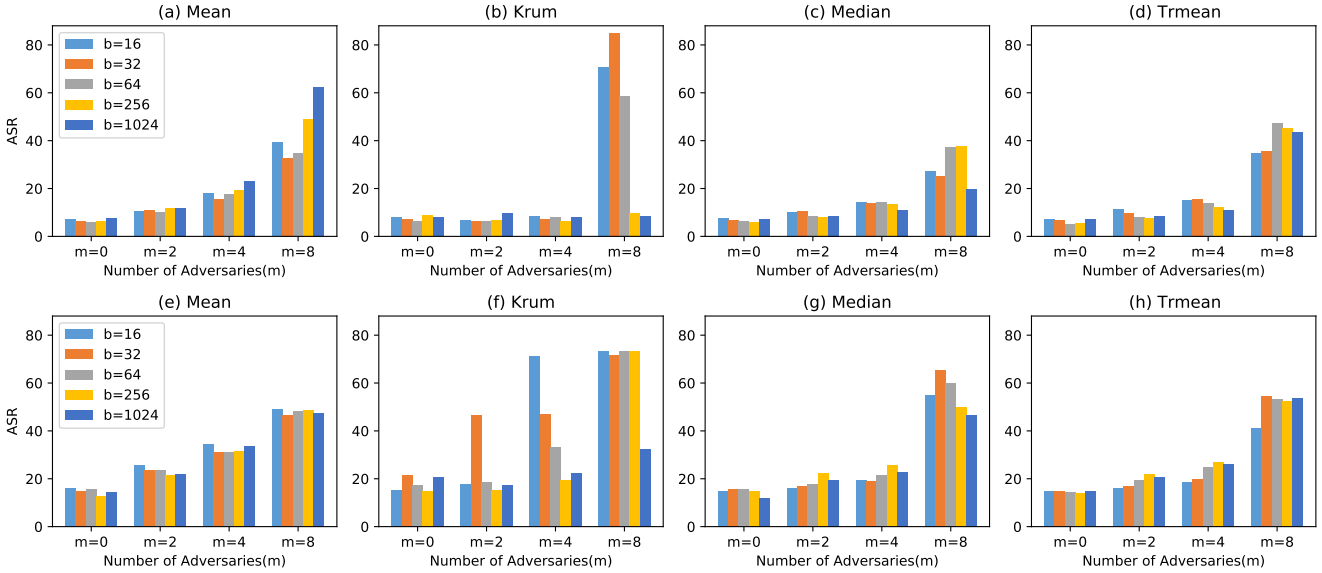



FIGURE 4: *ASR*s of targeted label-flipping attack under different batch sizes using model-based aggregation. (a)-(d): MLP on Fashion-MNIST, (e)-(h): CNN on CIFAR-10.

adversaries. Median and Trmean only show limited contributions in defending against attacks.

As for results on CIFAR-10 (for which a more complex model is used), Krum demonstrates an unstable performance as $b$ varies for different number of attackers. It can be concluded that Krum is prone to the *curse of dimensionality* (from the perspective of defenders, the complex model involves more parameters, resulting in the high dimensions of the updates), which is consistent with the findings of [39]. Median and Trmean behave in a relatively more stable fashion, but, when $m = 8$, they also lead to *ASR*s that are much higher than that in Mean aggregation.

The above observations suggest that robust aggregation rules investigated are still somewhat feasible in defending against targeted label-flipping attacks, even if they are applied to FedAvg — a model-based aggregation for which they were not originally designed. Except for the case where $m = 8$, Krum can achieve the best result without the constraint of large batch size.

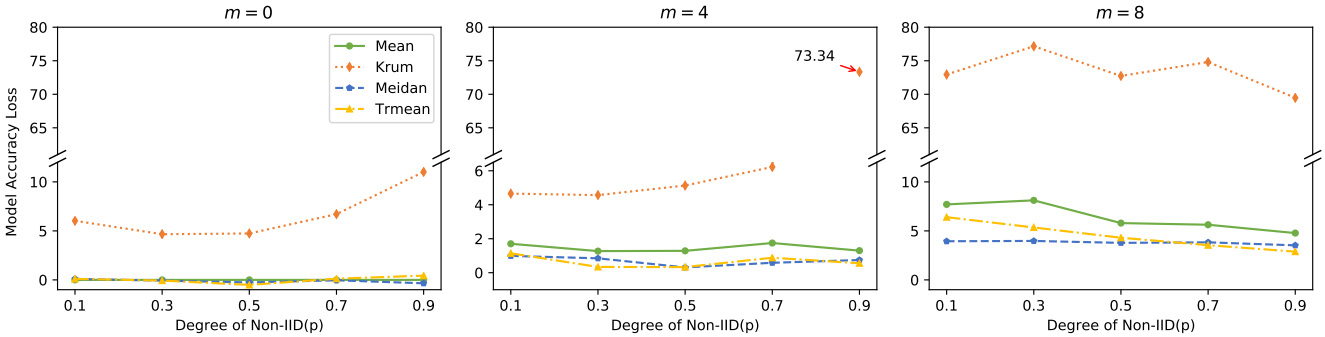
This might indicate that $\Delta w_i^t$ represents the features of $D_i$ better than $g_i^t$: once it contains poisoned data, it may be easier for a defense to distinguish. However, the undesirable performance on $m = 8$ implies that, during the distributed model training phase, with respect to defending against data poisoning attacks, sending gradients is not equivalent to sending local models to the server for aggregation. When we apply such rules in a new scenario (e.g., FL), the theoretical preconditions of these rules may change.

### 4.4. Impact of non-IID data

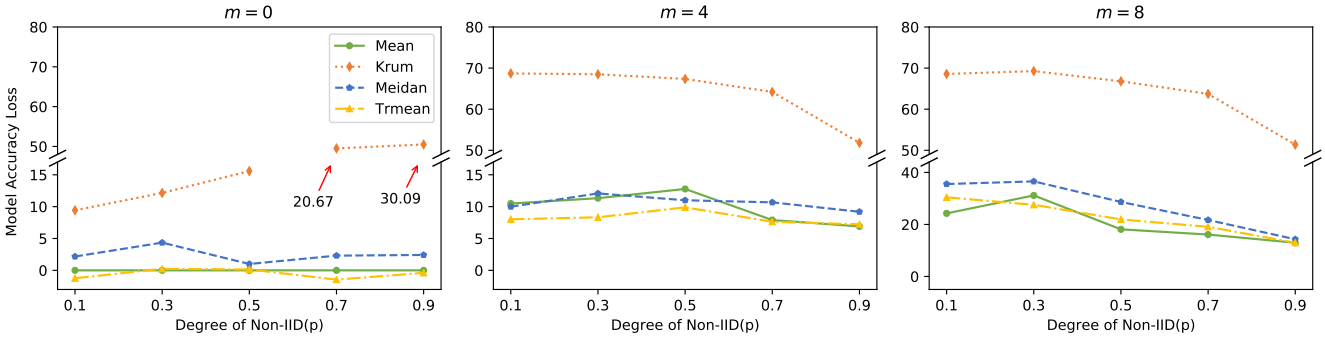
The results of Section 4.3 demonstrate that the robust aggregation rules are applicable to FedAvg under certain conditions. Poisoned data samples result in a discrepancy between benign updates (gradients or local model parameters) and malicious ones. Thus, by measuring the difference between the updates, or by removing the extreme values in each coordination of them, our robust aggregation rules of concern can defend against targeted label-flipping attacks. However, while the observations reported are based on the premise that the data is IID across clients, it is the case that, in realistic FL, non-IID data becomes more common. As mentioned in [40], different data distributions among clients lead the global model to different directions in the aggregation phase. That indicates an inherent divergence even between benign updates in non-IID setting. Thus, we ask *is it the case that the divergence caused by non-IID data may hinder the defense?*

In this experiment, we study the impact of non-IID data on robust aggregation rules when defending against data poisoning attacks. Here, we create non-IID data partitions in different degrees by varying the parameter $p$ (see Section 4.1). We perform FedAvg for 100 training rounds and consider varied numbers of adversaries on the two datasets.

We choose untargeted label-flipping attacks in non-IID setting for the following reason. Targeted poisoning relies on data samples with certain labels. Take $\mathcal{A}_5^3$ on CIFAR-10, for example: the non-IID property causes different clients having different amount of samples labelled 5. As the adversaries in the experiment are randomly assigned at the beginning, adversaries with more samples labelled 5 become more powerful because they can introduce more poisoned data samples. Therefore, it is hard to tell whether




(a) MLP on Fashion-MNIST




(b) CNN on CIFAR-10

FIGURE 5: Accuracy loss for robust aggregation rules as the degree of non-IID increases.

the effectiveness of a defense results from some weak adversaries or the contributions of such rules. To make the result deterministic, we adopt untargeted label-flipping attacks where adversaries have no specific goals. In this way, the capabilities between adversaries differ marginally. We use the model accuracy loss $\Delta w_{acc}$ on overall testing dataset to measure how effective the defense is. The value of $\Delta w_{acc}$ is calculated by $w_{acc} - w'_{acc}$, where $w_{acc}$ is the model accuracy when no adversaries are in the trainin and $w'_{acc}$ denotes the model accuracy when adversaries launch attacks under the defense. The value of $\Delta w_{aac}$ reveals the proportion of misclassified data samples yet to be saved from the adversaries by one defense.

Figure 5 outlines the accuracy loss for different robust aggregation rules as the degree of non-IID increases on the two datasets. Due to the similar tendency between cases where $m = 2$ and $m = 4$, we omit the results of the former case for reasons of conciseness.

For the Fashion-MNIST dataset, when there are no adversaries (i.e., $m = 0$), such rules bring insignificant accuracy loss — except for Krum, whose loss is comparatively higher and tends to increase with $p$ becomes larger. When $m = \{4, 8\}$, the loss of Median and Trmean increases; this increase is relatively low compared with Mean aggregation, indicating the effectiveness of the defense. One exception is Krum, whose accuracy loss is more severe than other robust aggregation rules — even more severe than Mean aggregation where no defense is applied. One possible reason is that the poisoned local model by untargeted label-flipping attack in a non-IID setting is more insidious for Krum to identify. Moreover, for the more challenging CIFAR-10 dataset, the defense effects for other rules are not obvious. It is interesting to note that, in most cases, Median and Trmean have larger accuracy loss than Mean aggregation. This might indicate they are less effective in the high-dimensional context. The decrease of $\Delta w_{acc}$ on Krum, as $p$ increases, is a consequence of the sharp drop of $w_{acc}$ and the marginal increasing $w'_{acc}$ of Krum. It cannot be concluded that Krum performs better as $p$ increases, because the $w'_{acc}$ fluctuates unacceptably (see Table 3: the increase is from 1.79% to 7.22%).

The failure of Krum to defend against untargeted label-flipping attack gives rise to the following inferences. In a non-IID setting, each client has its own (typically unique) data distribution, with the different data distribution bringing its unique contribution to the global model. Removing the suspicious updates from clients, even for the purpose of defense, may hurt the utility of the model. The accuracy loss on the two datasets resulted from Krum when there are no adversaries — as shown in Figure 5 — illustrates this. By removing extreme values in each dimension of the updates, Median and Trmean aim to trim off

the potentially malicious values. The final output is a mixture of parameters on each dimension of different updates. In this way, Median and Trmean comprehensively consider the contributions of different clients to some extent. This may be a possible reason why they outperform Krum in a non-IID setting.

## 5. KEY CHALLENGES IN DEFENDING AGAINST DATA POISONING IN FL

Based on the findings of the previous section and considering the unique features of FL, we now discuss the key challenges in defending against data poisoning attacks.

*It is impractical to maintain a validation set for the purpose of defending against poisoning attacks on the server.* In conventional centralized learning or DL, a validation set — a small fraction of the whole training data — is often used to tune the hyper-parameters of the model. Recent work [13–15] has introduced ways to defend against data poisoning attacks that leverage the validation set. The main idea is to verify the contributions of the updates from different clients. After the verification, the server keeps updates with larger contribution and removes or suppresses ones that contribute less. The contribution reveals to what extent the update improves the model accuracy. As malicious updates from adversaries usually tend to disturb the normal learning process, such defenses can identify updates that are defective. However, due to the potential infeasibility of constructing such a validation set, as well as the conflict with the original intention of FL, we argue this strategy is impractical in FL scenarios. To be specific: owing to the privacy constraints, data is preserved locally on different clients and the server has no direct control over it. Moreover, as FL aims to leverage data from different sources to collaboratively build a global model, the contributions of different updates count — especially when the data is non-IID. However, the verification of updates makes the whole learning process targeted on the validation set, on which the contribution of each update and the adjustment of the global model parameters are both decided. While this may yield a desirable final global model, it ignores the objective effectiveness of each update to the global model.

*From the defender's perspective, sending gradients is not equivalent to sending local models to the server.* The robust aggregation rules investigated in this work were originally designed for the learning algorithm in DL, where the server updates the global model using gradients. The rules only require the inspection of uploaded results from clients. The results of Section 4.2 demonstrate the unstable performance of such rules. Section 4.3 and Section 4.4 further explored the feasibility to apply them into FL scenario where FedAvg is more commonly used. The results suggest that such rules are effective in defending against data poisoning attacks (to some extent) and remove the prerequisite of large batch size. However, the inability in certain cases also implies that the rules need new theoretical guarantees and further improvements — especially for FL. For example, Xiang *et al* [41] first proved the convergence of FedAvg on non-IID data, shedding light on the theoretical understanding of FedAvg in realistic applications. From a defender's perspective, future work on defending against data poisoning attacks may focus more on realistic FL settings (e.g., Fedavg algorithm and non-IID data).

*To distinguish the malicious updates is more difficult for non-IID data.* In FL, data is typically generated in different contexts, leading to an inherent discrepancy in the distribution of data from multiple sources. Gradients or local models, regarded as a representation of local data, diverge from each other due to the non-IID property. The robust aggregation rules work on the observation that poisoned updates look different from those that are computed from 'clean' data samples. For example, Krum measures the similarity of two updates based on the Euclidean distance. However, as mentioned in [39], such measurement is ineffective in revealing the difference between malicious and benign updates. Furthermore, the non-IID property brings natural divergence, facilitating the hiding of malicious updates from inspection. More sophisticated, more suitable measurements, making malicious updates distinguishable in non-IID settings, may be the means to tackle such problem. Additionally, understanding how data poisoning attacks affect the normal updates may also provide us with targeted strategies to defend against them.

*An acceptable defense should have limited impact upon the utility of the model when there are no adversaries.* The server in FL has no visibility to the local data; as such, it cannot be aware whether there are adversaries in the environment or whether clients have been compromised. In addition, the adversaries can launch attacks at a particular frequency [42], making it harder for the server to perceive their existence. Consequently, the server has to defend against potential attacks in each training round to ensure the robustness of the global model. However, striving to accurately preclude malicious updates is prone to give rise to false positives (by which benign updates are classified as malicious ones). Especially in non-IID settings, as shown in Figure 5, removing the updates mistakenly can cause significant accuracy loss. Therefore, how to curb the malicious influence actually caused by adversaries — while maintaining a desirable model accuracy when all clients are benign — matters for defenses in FL.

## 6. DEFENDING AGAINST DATA POISONING ATTACKS: DSPO

To defend against data poisoning attacks in FL, and giving consideration to our findings, we propose **DSPO** (Detect and Suppress the Potential Outliers) — a defense strategy that restricts the impact of considered malicious updates and retains those that are benign in the final aggregation stage. We provide details of the proposed strategy and present evaluation results.

### 6.1. Intuition

The local updates from different clients, which are usually gradients or local models, guide the global model to adjust towards a certain direction with a certain step size. Similar updates bring similar effects to the global model during the model aggregation phase. In the non-IID setting, the global model is finally converged and the convergence curve generally follows the similar trends in the IID setting [40]. Since poisoned updates tend to lead the global model to an ineffective status, it can be speculated that a difference lies between poisoned and benign updates. Krum identifies poisoned updates via Euclidean distance, which reflects the similarity of two updates in overall magnitude. However, as mentioned in [39], updates may differ greatly in certain dimensions while maintaining a high similarity in overall magnitude. Cosine similarity is also used to measure the consistence of the two updates in their orientation [34]. This seems to be more reliable than Euclidean distance as it considers the discrepancy in each dimension. However, cosine similarity is insensitive to the magnitude of the updates: potential poisoned updates may be scaled in each dimension before they are sent to the server. Such scaling impacts little on the cosine similarity but significantly on the global model [10]. Therefore, an ideal measurement needs to consider both the magnitude and the orientation of the updates.

We assume the number of adversaries is less than half of the number of all participants, i.e., $m < \frac{k}{2}$. Although some defenses [13,14] claim to be robust to any number of adversaries, the motivation of FL is to utilize more data from different sources to improve the model utility. We believe that even a successful defense under an adversary-dominated learning environment contributes less to the model utility, as the data from benign clients is only partial. Thus we set $\frac{k}{2}$ as the upper bound of the adversaries ratio.

In our approach, we also detect the poisoned updates but with more sophisticated mechanisms. Based on the observations of Section 4.4, DSPO does not simply discard the potentially poisoned updates. Rather, it suppresses their influences on the global model by assigning them with lower weights. In this way, DSPO can defend against the poisoned updates while maintaining a desirable model accuracy.

### 6.2. Mechanism details

Figure 6 illustrates a general overview of DSPO. Since our defense works in each training round, we omit the explicit declaration of the round notation $t$. In each round of the training, the server receives local updates from different clients. DSPO first determines the pivot of theses updates by picking up the median value in each dimension. DSPO then calculates the pairwise similarity based on the pivot. Based on the pairwise similarities, DSPO assigns varied weights to the updates via a certain weight assignment policy. Finally, a weighted aggregation is performed to yield the final result.

We now explain the mechanism in detail.

**Pivot.** We first go through parameters in each dimension of the updates and take the median value of them. Then we get a pivot $u$ by composing theses medians. In each dimension, the component of $u$ is described as follows.

$$u_{(n)} = med(v_{i(n)} | i \in [k]), n \in [d] \tag{9}$$

**Pairwise similarity.** We calculate the similarity of any two different updates via adjusted cosine similarity.

$$\begin{aligned} sim(v_i, v_j) &= cos(v_i', v_j'), v_i \neq v_j \\ v_i' &= v_i - u \\ v_j' &= v_j - u \end{aligned} \tag{10}$$

Here, $sim(\cdot,\cdot)$ denotes the similarity measurement used in the proposed defense. $sim(\cdot,\cdot)$ reveals a kind of relative similarity of two given updates when referring to the pivot $u$, but not an absolute cosine similarity numerically (as per prior work, such as [34]). Pivot $u$ can be regarded as a general effect on model parameters. The similarity of two updates then reflects on the consistency of their own effects. $sim(\cdot,\cdot) > 0$ indicates that the two updates are more likely have the same effect on the global model.

We offer a 1-dimensional toy example to explain how pairwise similarity works. Suppose we have 5 updates $\mathcal{V} = \{58, 59, 60, 61, 62\}$ with $u = 60$. According to Equation (10), it is easy to calculate: $sim(58, 59) = cos(-2, -1) > 0$, $sim(59, 61) = cos(-1, 1) < 0$. The above calculations mean that $\{58, 59\}$ has a consistent effect on the global model, while $\{59, 61\}$ has an inverse one.

The pairwise similarity can be treated as a mutual evaluation among the updates. In such evaluation, we call the abnormal updates the outliers for short.

**Weight assignment.** We can detect the potential outliers based on the evaluation as they show different effects on the global model. In contrast to Median [16], which keeps a robust estimation in each dimension of the updates, we evaluate the updates on the whole. Further, we assign varied weights to theses updates to suppress the impact of potential outliers.

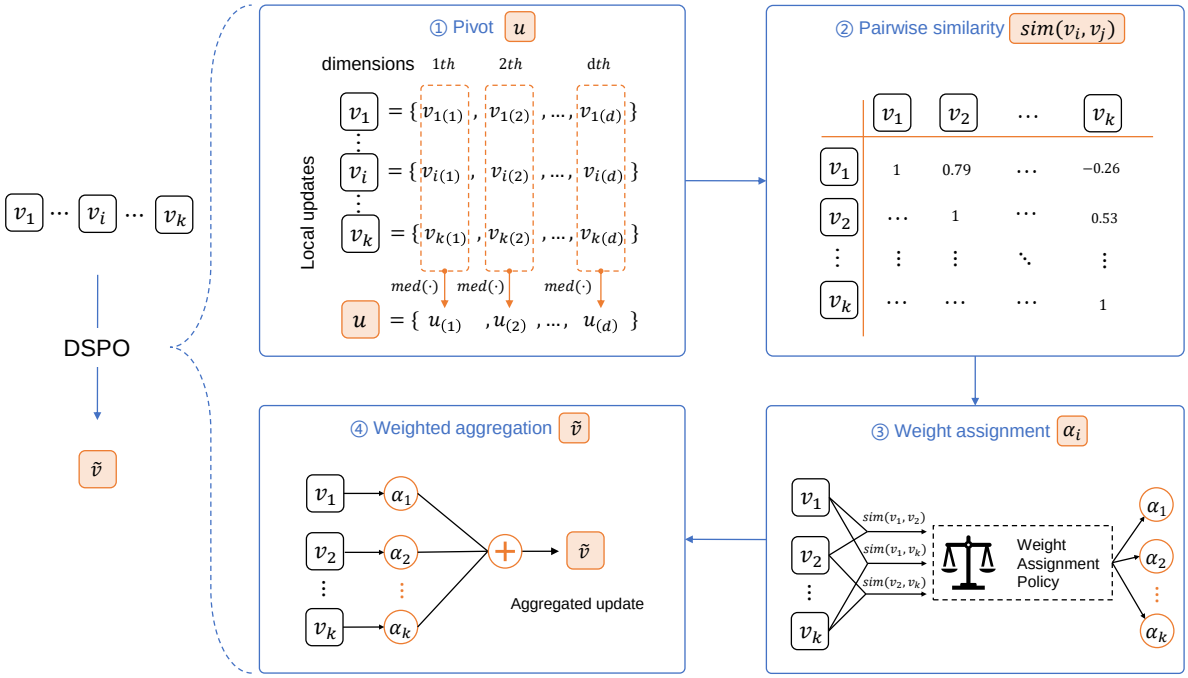



FIGURE 6: The overview of DSPO.

We consider two weight assignment policies: a *score-based* policy and a *vote-based* policy.

*(a) Score-based policy.* We term the pairwise similarity as the score indicating the extent of consistency. Let $s_i$ denotes the sum of the scores of $v_i$, which can be described as Equation 11. An update with higher scores means that it is more consistent with other updates. Otherwise, it is likely to be an outlier as it differs from the majority.

$$s_i = \sum_{v_j \in \mathcal{V}, j \neq i} sim(v_i, v_j) \tag{11}$$

Then, wen use the softmax function to transform $s$ to a corresponding weight $\alpha$ of the update, which is denoted by:

$$\alpha_i = \frac{e^{s_i}}{\sum_{j=1}^{k} e^{s_j}} \tag{12}$$

where $e$ (2.71828...) is Euler's number and $k$ represents the number of updates. In doing so, a higher $s$ produces a larger $\alpha$ and the sum of $\alpha$s equals 1 (i.e., $\sum_{j=1}^{k} \alpha_j = 1$).

Recent work [43] has reported the difficulty of identifying malicious updates in high dimensions. We also note that when $d \gg 1$, the two vectors are close to orthogonal [44, 45]. Therefore, despite the simplicity of the score-based policy, it has a potential that — especially in high dimensions — the scores can not reveal the discrepancy between updates accurately.

*(b) Vote-based policy.* We address the above concern by providing a sign function $vote(\cdot)$, which converts the *score* to a more distinguishable *vote*. The definition of $vote(\cdot)$ is as follows.

$$vote(x) = \begin{cases} 1 & x > 0 \\ -1 & x \leq 0 \end{cases} \qquad x \in \mathbb{R} \tag{13}$$

An update $v_i$ gets 1 vote from $v_j$ if $sim(v_i, v_j) > 0$, otherwise it gets $-1$ vote. Here, we use $s_i$ to denote the sum of the votes of $v_i$ from other updates.

$$s_i = \sum_{v_j \in \mathcal{V}, j \neq i} vote(sim(v_i, v_j)) \tag{14}$$

Since only the minority of the updates are malicious, we are less confident that an update is benign if its votes are less than 0 (more than half the other updates give it a vote of $-1$). We use $c_i$ to denote the confidence of $v_i$ being benign, which accumulates in each training round. Its definition is as follows:

$$c_i = c_i' + conf(s_i) \tag{15}$$

where $c_i'$ represents the old confidence in the last training round and $conf(\cdot)$ is the transition function that transforms the votes to the confidence, which is defined by:

$$conf(s) = \begin{cases} 0.5 & s < -0.1 \cdot k \\ 1 & -0.1 \cdot k \leq s \leq 0 \\ s & s > 0 \end{cases} \tag{16}$$

Here, we make a relaxation about the decision bound as the discrete vote value may be arbitrary. We set a threshold for $s$, which means 10% of the number of updates $(0.1 \cdot k)$. Only when $s_i < -0.1 \cdot k$, we treat

$v_i$ as an outlier and attack it with an low confidence 0.5. The condition $-0.1 \cdot k \leq s_i \leq 0$ is the relaxation area, for $s_i$ in this span: it could be an outlier that happens to be similar to the majority of updates or a benign updates varies in similarity due to the non-IID property mentioned in Section 4.4. Thus we returns a relative higher confidence.

As the training continues, the proportion of accumulated confidence of potential outliers are supposed to get smaller. We get the weight of each update by:

$$\alpha_i = \frac{c_i}{\sum_{j=1}^{k} c_j} \tag{17}$$

**Weighted aggregation.** After obtaining the weight of each update, we use a weighted aggregation to yield the final result:

$$\tilde{v} = \sum_{i=1}^{k} \alpha_i \cdot v_i \tag{18}$$

In the final aggregation stage, the malicious impact from adversaries can be suppressed via a low weight $\alpha$. When there are no adversaries, the whole updates are aggregated, preserving the utility of the global model to the greatest extent.

Algorithm 1 summarizes our defense, Algorithm 2 and Algorithm 3 describe the *score-based* policy and the *vote-based* policy respectively. The time and space complexities of our defense are $\mathcal{O}(k^2 d)$ and $\mathcal{O}(k^2)$ respectively, where $k$ denotes the number of clients in the learning environment and $d$ denotes the dimension of an *update*. To delegate the pairwise similarity computation to the weight assignment policy process, we can reduce the space complexity to $\mathcal{O}(k)$. However, we choose the $\mathcal{O}(k^2)$ version — calculating and store the result of pairwise similarity in advance — for the modularization of the algorithm and the convenience for future improvement (e.g., add more policies).

**Algorithm 1** Workflow of DSPO

**Input:** The number of clients: $k$, a set of updates: $\mathcal{V} = \{v_1, v_2, ..., v_k\}$, the weight assignment policy: *policy*
**Output:** The aggregated final update $\tilde{v}$

$simSet \leftarrow \{\}$
$u \leftarrow 0$
$\tilde{v} \leftarrow 0$
**for** $n = 1 \rightarrow d$ **do**
    $u_{(n)} = med\{v_{1(n)}, v_{2(n)}, ..., v_{k(n)}\}$
**end for**

**for** $i = 1 \rightarrow k$ **do**
    **for** $j = 1 \rightarrow k$ **and** $i \neq j$ **do**
        get $v_i, v_j$ from $\mathcal{V}$
        calculate $sim(v_i, v_j)$ ▷ $u$ is used in $sim()$
        save $sim(v_i, v_j)$ to $simSet$
    **end for**
**end for**

$weightSet \leftarrow \{\}$
**if** *policy* **is** *score* **then**
    $weightSet \leftarrow$ ScoreBasedPolicy($simSet$)
**else if** *policy* **is** *vote* **then**
    $weightSet \leftarrow$ VoteBasedPolicy($simSet$)
**end if**

**for** $i = 1 \rightarrow k$ **do**
    get $\alpha_i$ from $weightSet$
    get $v_i$ from $\mathcal{V}$
    $\tilde{v} \leftarrow \tilde{v} + \alpha_i \times v_i$
**end for**
**return** $\tilde{v}$

### 6.3. Evaluation results

We evaluate our defense on two datasets in a FL setting, giving consideration to an untargeted label-flipping attack. We consider two implementations of DSPO using the two policies discussed above, which we abbreviate as **DSPO (score)** and **DSPO (vote)** respectively.

The experimental settings are described in Section 4.1. We consider two different data partitions: $p = \{0.1, 0.9\}$. The former is closest to the IID setting; the latter is representative of a non-IID setting.

Table 2 summarizes the results of different defenses on Fashion-MNIST. In most of the testing cases, our defenses (both DSPO(score) and DSPO (vote)) show comparable performance with others. Notably, when the number of adversaries increase, our defenses perform well and show less degradation in model accuracy.

Table 3 shows the results on CIFAR-10. Our defenses achieve the highest model accuracy under all threat settings. An interesting observation is that the accuracy of each defense — other DSPO (vote) — is even lower than the Mean aggregation where no defense is applied. This indicates the ineffectiveness of these defenses in such cases. A possible explanation is that the model used for CIFAR-10 is more complicated. Hence, the dimension of the corresponding update during the training is much higher. DSPO (vote) takes this complication into account in the design phase (see Section 6.2) and the experimental results verify the effectiveness of such design.

### 6.4. Discussion

In general, we find that Median, Trmean and our method are all effective in defending against data poisoning attacks in FL scenarios; they are also more robust than Krum in non-IID settings. Median and Trmean discard outliers in each dimension of the local updates, while our method strives to curb the effect of malicious updates and maintains their structure (as

TABLE 2: Model accuracy on Fashion-MNIST with different aggregation rules and attack settings.

| | $p = 0.1$ | | | | $p = 0.9$ | | | |
|---|---|---|---|---|---|---|---|---|
| Methods | $m = 0$ | $m = 2$ | $m = 4$ | $m = 8$ | $m = 0$ | $m = 2$ | $m = 4$ | $m = 8$ |
| Mean | **89.43%** | 88.92% | 87.73% | 81.73% | 86.00% | 84.94% | 84.70% | 81.00% |
| Krum | 83.40% | 84.56% | 84.77% | 16.48% | 74.98% | 75.59% | 12.66% | 16.52% |
| Median | 89.33% | 89.09% | 88.44% | 85.49% | **86.35%** | 85.34% | **85.25%** | 82.48% |
| Trmean | 89.33% | **89.22%** | 88.21% | 83.03% | 85.57% | **85.60%** | 85.04% | 83.12% |
| **DSPO (score)** | 88.86% | 88.69% | **88.60%** | 87.18% | 85.54% | 84.94% | 84.52% | **83.63%** |
| **DSPO (vote)** | 88.71% | 88.76% | 88.57% | **87.98%** | 85.43% | 83.65% | 83.96% | 82.69% |

TABLE 3: Model accuracy on CIFAR-10 with different aggregation rules and attack settings.

| | $p = 0.1$ | | | | $p = 0.9$ | | | |
|---|---|---|---|---|---|---|---|---|
| Methods | $m = 0$ | $m = 2$ | $m = 4$ | $m = 8$ | $m = 0$ | $m = 2$ | $m = 4$ | $m = 8$ |
| Mean | 70.50% | 65.86% | 62.87% | 46.24% | 59.03% | 56.20% | 52.14% | 46.06% |
| Krum | 61.51% | 62.22% | 1.79% | 1.93% | 32.54% | 6.39% | 7.22% | 7.64% |
| Median | 68.33% | 66.27% | 60.51% | 34.95% | 56.62% | 55.74% | 49.84% | 44.73% |
| Trmean | 70.74% | 66.97% | 62.49% | 40.07% | 59.43% | 57.84% | 51.80% | 45.98% |
| **DSPO (score)** | **71.16%** | 67.63% | 57.32% | 1.39% | **60.36%** | **60.29%** | 49.44% | 16.19% |
| **DSPO (vote)** | 70.85% | **69.02%** | **68.23%** | **60.31%** | 59.65% | 58.07% | **52.88%** | **53.48%** |

**Algorithm 2** Workflow of score-based weight assignment policy

**function** SCOREBASEDPOLICY($simSet$)
  $weightSet \leftarrow \{\}$
  $base \leftarrow 0$
  **for** $i = 1 \rightarrow k$ **do**
    $s_i \leftarrow 0$
    **for** $j = 1 \rightarrow k$ **and** $i \neq j$ **do**
      get $sim(v_i, v_j)$ from $simSet$
      $s_i \leftarrow s_i + sim(v_i, v_j)$
    **end for**
    $base \leftarrow base + e^{s_i}$
  **end for**
  **for** $i = 1 \rightarrow k$ **do**
    $\alpha_i = s_i / base$
    save $\alpha_i$ to $weightSet$
  **end for**
  **return** $weightSet$
**end function**

**Algorithm 3** Workflow of vote-based weight assignment policy

**function** VOTEBASEDPOLICY($simSet$)
  $weightSet \leftarrow \{\}$
  $base \leftarrow 0$
  **for** $i = 1 \rightarrow k$ **do**
    $s_i \leftarrow 0$
    $c_i \leftarrow 0$ ▷ $c_i$ is declared as static variable, as it accumulates every time the function is called
    **for** $j = 1 \rightarrow k$ **and** $i \neq j$ **do**
      get $sim(v_i, v_j)$ from $simSet$
      $s_i \leftarrow s_i + vote(sim(v_i, v_j))$
    **end for**
    $c_i \rightarrow c_i + conf(s_i)$
    $base \leftarrow base + c_i$
  **end for**
  **for** $i = 1 \rightarrow k$ **do**
    $\alpha_i = c_i / base$
    save $\alpha_i$ to $weightSet$
  **end for**
  **return** $weightSet$
**end function**

per Krum). Once the data used for training has been poisoned by adversaries, it cannot contribute positively to the global model. Defenders, who can only preclude the malicious updates or alleviate their negative impact, are unable to recover the original data supposed to be in the training set. Therefore, for any defense, it cannot restore model accuracy to the state prior to the attack. The proposed DSPO can defend against the poisoned updates while maintaining a comparably higher model accuracy than other defenses.

However, some limitations should be noted. First, more factors need to be considered. We observe that the model complexity and the number of adversaries affect the effectiveness of the defense, DSPO outperforms other defenses only by adopting a targeted-improved tactic. Quantifying the effect of dimensionality and generalizing such defenses in a high-dimensional context is, therefore, an interesting and potentially promising direction for future work. Second, We only evaluate the

defenses under label-flipping attacks. It is an interesting task to explore the common characteristics of most poisoning attacks. We believe that such common properties is helpful in designing a generic defense.

## 7. CONCLUSION

In this paper, we have given consideration to the task of defending against data poisoning attacks in FL.

We explored the feasibility and effectiveness of some typical defenses. In doing so, our study revealed several key findings. First, we showed that the effectiveness of defenses in DL is influenced by the batch size of data used in each training round. Second, we found that the defenses are somewhat effective, and are slightly affected by batch size in FL setting when the data is IID. Third, we argued that the non-IID data — an important feature of FL — increases the difficulty of successfully defending against data poisoning attacks. We further discussed and summarized the key challenges as well as the insights in designing a desirable defense especially for FL. Based on the findings, we proposed DSPO, a defense strategy that is suitable for FL settings. Our evaluation results on two datasets demonstrated its effectiveness. However, as a generic defense, there is still potential for future improvement.

Our hope is that our findings and the discussions presented in this paper, as well as our proposed method, inspire further research on data poisoning attacks in real-world FL environments. As part of our future work we will consider other kinds of attacks, including backdoor attacks and model poisoning attacks, and study the applicability of current defenses against them.

## ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their helpful and constructive comments. The work described in this paper was supported by the Key-Area Research and Development Program for Guangdong Province, China (2019B010136001), Basic Research Project of Shenzhen, China (JCYJ20190806142601687), Basic Research Project of Shenzhen, China (JCYJ20190806143418198), and Basic Research Project of Shenzhen, China (JCYJ20200109113405927).

## DATA AVAILABILITY STATEMENTS

The data underlying this article will be shared on reasonable request to the corresponding author.

## REFERENCES

[1] Sarathambekai, S. and Umamaheswari, K. (2018) Multi-Objective Optimization Techniques for Task Scheduling Problem in Distributed Systems. *The Computer Journal*, **61**, 248–263.

[2] Shamshirband, S., Fathi, M., Chronopoulos, A. T., Montieri, A., Palumbo, F., and Pescapè, A. (2020) Computational intelligence intrusion detection techniques in mobile cloud computing environments: Review, taxonomy, and open research issues. *Journal of Information Security and Applications*, **55**, 102582.

[3] Langer, M., He, Z., Rahayu, W., and Xue, Y. (2020) Distributed Training of Deep Learning Models: A Taxonomic Perspective. *IEEE Transactions on Parallel and Distributed Systems*, **31**, 2802–2818.

[4] Mcmahan, H. B. and Ramage, D. (2017) Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 20–22 Apr, pp. 1273–1282. PMLR.

[5] Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019) Federated Machine Learning: Concept and Applications. *ACM Transactions on Intelligent Systems and Technology*, **10**, 1–19.

[6] Ma, C., Li, J., Ding, M., Yang, H. H., Shu, F., Quek, T. Q. S., and Poor, H. V. (2020) On safeguarding privacy and security in the framework of federated learning. *IEEE Network*, **34**, 242–248.

[7] Sattler, F., Muller, K.-R., Wiegand, T., and Samek, W. (2020) On the Byzantine Robustness of Clustered Federated Learning. *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 4–8 May, pp. 8861–8865. IEEE.

[8] Nuding, F. and Mayer, R. (2020) Poisoning Attacks in Federated Learning: An Evaluation on Traffic Sign Classification. *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, New York, NY, USA, pp. 168–170. Association for Computing Machinery.

[9] Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E. C., and Roli, F. (2017) Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, New York, NY, USA, pp. 27–38. Association for Computing Machinery.

[10] Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017) Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, Red Hook, NY, USA, pp. 119–129. Curran Associates Inc.

[11] Baracaldo, N., Chen, B., Ludwig, H., and Safavi, J. A. (2017) Mitigating poisoning attacks on machine learning models: A data provenance based approach. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, New York, NY, USA 103–110. Association for Computing Machinery.

[12] Steinhardt, J., Koh, P. W., and Liang, P. (2017) Certified defenses for data poisoning attacks. *Advances in Neural Information Processing Systems*, Red Hook, NY, USA, pp. 3518–3530. Curran Associates Inc.

[13] Xie, C., Koyejo, O., and Gupta, I. (2019) Zeno: Distributed Stochastic Gradient Descent with Suspicion-based Fault-tolerance. *36th International Conference on Machine Learning*, 09–15 Jun, pp. 6893–6901. PMLR.

[14] Pan, X., Zhang, M., Wu, D., Xiao, Q., Ji, S., and Yang, M. (2020) Justinian's GAAvernor: Robust Distributed

Learning with Gradient Aggregation Agent. *29th USENIX Security Symposium (USENIX Security 20)*, Aug, pp. 1641–1658. USENIX Association.

[15] Konstantinov, N. and Lampert, C. H. (2019) Robust learning from untrusted sources. *36th International Conference on Machine Learning, ICML 2019*, 09–15 Jun, pp. 6112–6132. PMLR.

[16] Yin, D., Chen, Y., Ramchandran, K., and Bartlett, P. (2018) Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. *35th International Conference on Machine Learning, ICML 2018*, 10–15 Jul, pp. 5650–5655. PMLR.

[17] Xie, C., Huang, K., Chen, P.-Y., and Li, B. (2020) DBA: Distributed Backdoor Attacks against Federated Learning. *International Conference on Learning Representations 2020*, pp. 1–19.

[18] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. (2020) How to backdoor federated learning. *Proceedings of the 23th International Conference on Artificial Intelligence and Statistics*, 26–28 Aug, pp. 2938–2948. PMLR.

[19] Biggio, B., Nelson, B., and Laskov, P. (2012) Poisoning attacks against support vector machines. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, Madison, WI, USA, pp. 1807–1814.

[20] Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. (2018) Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning. *2018 IEEE Symposium on Security and Privacy (SP)*, 26–28 May, pp. 19–35. IEEE.

[21] Fang, M., Yang, G., Gong, N. Z., and Liu, J. (2018) Poisoning attacks to graph-based recommender systems. *Proceedings of the 34th Annual Computer Security Applications Conference*, New York, NY, USA 381–392. Association for Computing Machinery.

[22] Diakonikolas, I., Kamath, G., Kane, D., Li, J., Steinhardt, J., and Stewart, A. (2019) Sever: A robust meta-algorithm for stochastic optimization. *Proceedings of the 36th International Conference on Machine Learning*, 09–15 Jun, pp. 1596–1606. PMLR.

[23] Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. (2018) Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. *Advances in Neural Information Processing Systems*, Red Hook, NY, USA, pp. 6103–6113. Curran Associates Inc.

[24] Rosenfeld, E., Winston, E., Ravikumar, P., and Kolter, J. Z. (2020) Certified Robustness to Label-Flipping Attacks via Randomized Smoothing. *Proceedings of the 37th International Conference on Machine Learning*, 13–18 Jul, pp. 8230–8241. PMLR.

[25] Paudice, A., Muñoz-González, L., and Lupu, E. C. (2019) Label sanitization against label flipping poisoning attacks. *ECML PKDD 2018 Workshops*, Cham, pp. 5–15. Springer International Publishing.

[26] Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. (2018) Trojaning Attack on Neural Networks. *Proceedings 2018 Network and Distributed System Security Symposium(NDSS)*.

[27] Tomsett, R., Chan, K. S., and Chakraborty, S. (2019) Model poisoning attacks against distributed machine learning systems. *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, Baltimore, Maryland, United States, pp. 481–489. SPIE.

[28] Bhagoji, A. N., Chakraborty, S., Mittal, P., and Calo, S. (2019) Analyzing Federated Learning through an Adversarial Lens. *36th International Conference on Machine Learning, ICML 2019*, 09–15 Jun, pp. 634–643. PMLR.

[29] Xie, C., Koyejo, O., and Gupta, I. (2020) Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, pp. 261–270.

[30] Fang, M., Cao, X., Jia, J., and Gong, N. Z. (2020) Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. *29th USENIX Security Symposium (USENIX Security 20)*, 22–25 Jul, pp. 1605–1622. PMLR.

[31] Shen, Y. and Sanghavi, S. (2019) Learning with bad training data via iterative trimmed loss minimization. *Proceedings of the 36th International Conference on Machine Learning*, 09–15 Jun, pp. 5739–5748. PMLR.

[32] Zhao, L., Hu, S., Wang, Q., Jiang, J., Chao, S., Luo, X., and Hu, P. (2020) Shielding Collaborative Learning: Mitigating Poisoning Attacks through Client-Side Detection. *IEEE Transactions on Dependable and Secure Computing*, **1**, 1–1.

[33] Alistarh, D., Allen-Zhu, Z., and Li, J. (2018) Byzantine Stochastic Gradient Descent. *Advances in Neural Information Processing Systems*, Red Hook, NY, USA, pp. 4613–4623. Curran Associates Inc.

[34] Fung, C., Yoon, C. J. M., and Beschastnikh, I. (2020) The Limitations of Federated Learning in Sybil Settings. *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pp. 301–316.

[35] Li, M., Andersen, D. G., Park, J. W., Ahmed, A., Josifovski, V., Long, J., Shekita, E. J., and Su, B.-Y. (2014) Scaling Distributed Machine Learning with the Parameter Server. *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 583–598.

[36] Tolpegin, V., Truex, S., Gursoy, M. E., and Liu, L. (2020) Data Poisoning Attacks Against Federated Learning Systems. *25th European Symposium on Research in Computer Security, ESORICS 2020*, pp. 480–501. Springer International Publishing.

[37] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017) On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations 2017*.

[38] Zhang, Y., Qu, H., Chen, C., and Metaxas, D. (2019) Taming the noisy gradient: Train deep neural networks with small batch sizes. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 4348–4354.

[39] Mhamdi, E. M. E., Guerraoui, R., and Rouault, S. (2018) The Hidden Vulnerability of Distributed

Learning in Byzantium. *35th International Conference on Machine Learning, ICML 2018*, 10–15 Jul, pp. 5674–5686. PMLR.

[40] Hsieh, K., Phanishayee, A., Mutlu, O., and Gibbons, P. (2020) The non-IID data quagmire of decentralized machine learning. *Proceedings of the 37th International Conference on Machine Learning*, 13–18 Jul, pp. 4387–4398. PMLR.

[41] Xiang, L., Kaixuan, H., Wenhao, Y., Shusen, W., and Zhihua, Z. (2020) On the convergence of fedavg on non-iid data. *International Conference on Learning Representations 2020*.

[42] Sun, Z., Kairouz, P., Suresh, A. T., and McMahan, H. B. (2019) Can You Really Backdoor Federated Learning? *2nd International Workshop on Federated Learning for Data Privacy and Confidentiality in NeurIPS 2019*.

[43] Shejwalkar, V. and Houmansadr, A. (2021) Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. *Proceedings 2021 Network and Distributed System Security Symposium*, Reston, VA. Internet Society.

[44] Cai, T., Fan, J., and Jiang, T. (2013) Distributions of angles in random packing on spheres. *Journal of Machine Learning Research*, **14**, 1532–4435.

[45] Zhang, X., Yu, F. X., Kumar, S., and Chang, S. F. (2017) Learning spread-out local feature descriptors. *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, 22–29 Oct, pp. 4605–4613. IEEE.