# Towards Optimized Model Poisoning Attacks Against Federated Learning

**Virat Shejwalkar and Amir Houmansadr**
College of Information and Computer Science
University of Massachusetts, Amherst
{vshejwalkar, amir}@cs.umass.edu

## Abstract

Federated learning (FL) enables many data owners (e.g., mobile device owners) to train a joint ML model (e.g., a next-word prediction classifier) without the need of sharing their private training data. However, FL is known to be susceptible to model poisoning attacks by malicious participants (e.g., adversary-owned mobile devices), who aim to minimize the accuracy of the jointly trained model by contributing malicious updates during the federated training process. In this paper, we present a general framework for model poisoning attacks on FL. We show that our framework leads to poisoning attacks that are substantially stronger than the state-of-the-art model poisoning attacks. For instance, our attacks result in $1.5\times$ to $16\times$ more reductions in the accuracy of the best joint model obtained using FL compared to the strongest of existing attacks.

## 1 Introduction

Federated learning (FL) is an emerging learning paradigm in which many data owners (called *clients*) collaborate in training a common machine learning model, without sharing their local private training data. In each FL training epoch, a *central server* broadcasts a jointly trained model (called *global model*) to a subset of clients, who then compute gradient updates using a single mini-batch of their local data and the global model. The server aggregates the clients' gradient updates using an aggregation algorithm (AGR), e.g., weighted average [15], and updates the global model using the aggregated updates. This algorithm is called fedSGD algorithm [15], which we consider in our work.

Although effective in many practical settings, FL mechanisms are prone to *poisoning attacks* [16]: *malicious clients* can attempt to degrade the global model accuracy by contributing malicious updates (through their *malicious devices*) during the FL training process. There are two types of poisoning attacks based on the adversary's goal: In *untargeted* poisoning attacks [16, 3, 10, 18], the goal is to minimize the accuracy of the global model on *any* test input, while in *targeted* poisoning attacks [5, 2], the goal is to minimize the accuracy on specific test inputs of adversary's choice and maintain high accuracy on the other test inputs. *Untargeted* attacks are a more severe threat to FL as they can completely cripple FL. There are two types of poisoning attacks based on the adversary's capabilities: In *model* poisoning attacks [10, 5, 2, 3, 16], the adversary directly manipulates the malicious updates she sends to the server, while in *data* poisoning attacks [11, 6], the adversary only manipulates the training data on the malicious (i.e., compromised or owned) clients' devices. Model poisoning is a more sever threat to FL as adversary can arbitrarily manipulate her malicious updates, as discussed and demonstrated in previous works [10, 5, 12].

To defeat such poisoning attacks on FL, a recent line of work has investigated the design of Byzantine-robust FL algorithms, where the central server uses some *robust aggregation algorithm* (AGR) [19, 7, 18, 16] to reduce the impact of malicious updates to preserve the global model utility.

In this work, we propose a general optimization framework to mount *untargeted model poisoning* attacks on robust AGR algorithms and show that these algorithms are significantly more vulnerable than they were previously thought. We consider five state-of-the-art robust AGRs: *Krum* [7], Multi-Krum [7], Bulyan [16], Trimmed-mean [19], and Median [19, 18]. We compare our attacks with state-of-the-art untargeted model poisoning attacks, Fang [10] and LIE [3]. Due to space limitation, we refer interested readers to the original works of attacks and AGRs for the details.

***Our contributions.*** Previous model poisoning works use very strong, impractical advesaries (or threat models). They assume that the adversary knows the server's robust AGR and/or the data on benign client devices, and therefore, also know the benign updates they contribute in each epoch) [10, 3, 16, 18]. On the other hand, we consider two significantly more practical adversaries who do not know the benign updates. Our first adversary, called `agr-tailored`, knows the server's robust AGR and second adversary, called `agr-agnostic`, does not know the robust AGR.

We present a general optimization framework to mount untargeted model poisoning attacks on FL. The high-level approach of our attack is as follows. The adversary computes a reference *benign aggregate* using some benign updates she knows; then she computes a malicious perturbation vector (whose generation will be explained in detail), e.g., a unit vector in the opposite direction of the benign aggregate. Finally, the adversary computes her malicious model update by maximally perturbing the benign reference aggregate in the malicious direction, while also evading detection by robust AGRs. We show that an `agr-tailored` adversary can tailor the objective of the optimization to any robust AGR and propose five *AGR-tailored* attacks for five state-of-the-art robust AGRs. We also show how an `agr-agnostic` adversary can modify the framework to mount an effective model poisoning attack against any robust AGR and propose two *AGR-agnostic* attacks. Our AGR-agnostic attacks exploit the key intuition behing any robust AGR: an update is malicious if it wonders far away from a set of benign updates. Therefore, our AGR-agnostic attacks constrain their search of the most malicious updates to a ball of a small radius around the clique of the benign updates.

Our evaluations using CIFAR10 and FEMNIST datasets demostrate the superiority of our attacks over the state-of-the-art model poisoning attacks. Finally, we note that our attacks can be directly applied to FedAVG algorithm [15], which is a communication efficient version of fedSGD, and that we leave detailed investigation of fedAVG to future work.

## 2 Threat Model of Model Poisoning Attacks

**Adversary's goal.** The goal is to craft malicious gradients (recall that we use fedSGD algorithm) such that when the central server updates the global model using the malicious and benign gradients (benign clients share benign gradients), the accuracy of the resulting global model reduces indiscriminately, i.e., on any test input. This is also known as *untargeted model poisoning attack.*

**Adversary's capabilities.** We assume that the number of malicious clients, $m$, cannot be more than that of benign clients, $n$. More specifically, similar to the state-of-the-art untargeted model poisoning works [10, 3], we use 20% malicious clients in our experiments; we also evaluate our attacks with as low as 5% malicious clients. We assume that the adversary can access and manipulate the gradients on malicious clients' devices.

**Adversary's knowledge.** We consider two adversaries: First is `agr-tailored` adversary, who knows the server's AGR; for transparency reasons, an FL service provider may release the details of her AGR algorithm. Second is `agr-agnostic` adversary, who does not know the server's AGR; for security reasons, an FL service provider may not release the details of her AGR algorithm. We assume that both the adversaries know the benign data only on compromised devices. This is a more practical threat model compared to previous works [10, 3], which assume that the adversary can access the data on benign devices as well.

## 3 Our General Framework for Model Poisoning

In this section, we describe our general framework to mount model poisoning attacks on FL, followed by specific optimizations for different AGRs and threat models, and finally give an algorithm to solve the optimizations.

### 3.1 General optimization formulation

To successfully mount an untargeted attack, our general optimization problem aims to maximize the damage to the global model in each FL epoch. In order to maximize the damage, we craft the malicious gradients, denoted by $\nabla^m_{\{i\in[m]\}}$, such that the aggregate computed by the server is far from a *reference benign aggregate*, denoted by $\nabla^b$. A possible $\nabla^b$ is the average of the $m$ benign gradients that the adversary computes using the benign data on $m$ malicious devices; we denote these gradients by $\nabla_{\{i\in[m]\}}$. The final malicious gradient $\nabla^m$ is a perturbed version of the benign aggregate $\nabla^b$, i.e., $\nabla^m = \nabla^b + \gamma\nabla^{\mathsf{p}}$, where $\nabla^{\mathsf{p}}$ is a *perturbation vector* and $\gamma$ is a *scaling coefficient*. Therefore, the objective for the `agr-tailored` adversary is given by (1).

$$\underset{\gamma,\nabla^{\mathsf{p}}}{\text{argmax}}\ \|\nabla^b - f_{\mathsf{agr}}(\nabla^m_{\{i\in[k]\}} \cup \nabla_{\{i\in[m]\}})\|_2 \quad \ldots \nabla^m_{i\in[k]} = \nabla^b + \gamma\nabla^{\mathsf{p}}, \nabla^b = f_{\mathsf{avg}}(\nabla_{\{i\in[m]\}}) \tag{1}$$

where we chose $k$ such that $\frac{k}{m+k} = \frac{m}{n}$. Note that state-of-the-art robust AGRs [7, 19, 16] are generally not differentiable. Hence, solving (1), i.e., finding the optimal $\gamma$ and $\nabla^{\mathsf{p}}$, using gradient descent based optimizations is not trivial. We overcome this challenge by fixing the perturbation vector $\nabla^{\mathsf{p}}$ and finding the optimial $\gamma$, i.e., solve the modified objective in (2). Algorithm 1 (Section 3.4) describes our algorithm to optimize $\gamma$.

$$\underset{\gamma}{\text{argmax}}\ \|\nabla^b - f_{\mathsf{agr}}(\nabla^m_{\{i\in[k]\}} \cup \nabla_{\{i\in[m]\}})\|_2 \quad \ldots \nabla^m_{i\in[k]}, \nabla^b \text{ as in (1)} \tag{2}$$

**Introducing perturbation vectors.** A perturbation vector is any malicious direction in the space of gradients that the adversary can use to perturb $\nabla^b(i.e., f_{\mathsf{avg}}(\nabla_{\{i\in[m]\}}))$ and find the malicious gradients $\nabla^m_{\{i\in[m]\}}$. In this work, we introduce and study the following three types of $\nabla^{\mathsf{p}}$'s.

*Inverse unit vector* , $\nabla^{\mathsf{p}}_{\mathsf{uv}}$, is $-\frac{\nabla^b}{\|\nabla^b\|_2}$. The intuition here is to compute the malicious gradient by perturbing $\nabla^b$ by a scaled unit vector that points in the opposite direction of $\nabla^b$.

*Inverse standard deviation* , ($\nabla^{\mathsf{p}}_{\mathsf{std}}$), is $-\mathsf{std}(\nabla_{\{i\in[m]\}})$. The intuition here is that the higher the variance of a dimension of benign gradients, the higher the perturbation that the adversary can introduce along the dimension..

*Inverse sign* , ($\nabla^{\mathsf{p}}_{\mathsf{sgn}}$), is $-\mathsf{sign}(f_{\mathsf{avg}}(\nabla_{\{i\in[m]\}}))$. The intuition here is similar to that of ($\nabla^{\mathsf{p}}_{\mathsf{uv}}$), but we observe that $\nabla^{\mathsf{p}}_{\mathsf{sgn}}$ is more effective for some classification tasks, e.g., FEMNIST.

We will show in Section 4.1 that the appropriate choice of perturbation vector $\nabla^{\mathsf{p}}$ is the key to an effective attack. In our explorations, we also tries the perturbation that directly increases the loss of global model on benign data, but we find that using one of the above perturbations works better and significantly outperforms the existing model poisoning attacks. We leave further investigation of the optimal $\nabla^{\mathsf{p}}$ to future work.

## 3.2 AGR-tailored attacks

In this section, we consider `agr-tailored` adversary, who know the server's AGR algorithm and tailors the general attack objective in (2) to the AGR. We provide optimization problems for the five robust AGRs from Section 1.

**Krum.** Krum selects a single gradient from its inputs as its aggregate. Hence, a successful attack forces Krum to select one of its malicious gradients, i.e., $\nabla^m_{i\in[k]} = f_{\mathsf{krum}}(\nabla^m_{\{i\in[k]\}} \cup \nabla_{\{i\in[m]\}})$. Therefore, we modify (2) to (3) for Krum. For each of the input gradients, Krum computes a score that is the sum of distances of $n-m-2$ nearest neighbors of the gradient. Therefore, to maximize the chances of Krum selecting a malicious gradient, we keep all the malicious gradients the same.

$$\underset{\gamma}{\text{argmax}}\ \nabla^m_{i\in[m]} = f_{\mathsf{krum}}(\nabla^m_{\{i\in[k]\}} \cup \nabla_{\{i\in[m]\}}); \quad \ldots \nabla^m_{i\in[k]} = f_{\mathsf{avg}}(\nabla_{\{i\in[m]\}}) + \gamma\nabla^{\mathsf{p}} \tag{3}$$

**Multi-krum and Bulyan.** Multi-krum (Bulyan) uses Krum iteratively to construct a selection set $\mathcal{S}$ and computes average (Trimmed-mean) of the gradients in the selection set as its aggregate. Due to the similarity of their constructions, our attacks on Multi-krum and Bulyan are the same. Our attack on Multi-krum ensures that all of the malicious gradients are in selected $\mathcal{S}$, while maximizing the perturbation $\gamma\nabla^{\mathsf{p}}$ used to compute malicious gradient, $\nabla^m$. Note that, this strategy minimizes the of number benign gradients in $\mathcal{S}$, while maximizing $\gamma\nabla^{\mathsf{p}}$ increases the poisoning impact of $\nabla^m$ on the final aggregate. Therefore, we modify (2) to (4) for Multi-krum; here $|A|$ is the size of $A$.

$$\underset{\gamma}{\text{argmax}}\ k = |\{\nabla \in \nabla^m_{\{i\in[k]\}} | \nabla \in \mathcal{S}\}|; \quad \ldots \nabla^m_{i\in[k]} = f_{\mathsf{avg}}(\nabla_{\{i\in[m]\}}) + \gamma\nabla^{\mathsf{p}} \tag{4}$$

**Trimmed-mean and Median.** For Trimmed-mean and Median, we directly solve the optimization in (2) by fixing the perturbation $\nabla^{\mathsf{p}}$ and keeping all the malicious updates the same. Hence, our objective is to maximize the $L_2$-norm of the distance between the reference benign update $\nabla^b$ and the aggregate computed using Trimmed-mean or Median on the set of benign and malicious updates. This is formalized in (5).

$$\underset{\gamma}{\text{argmax}}\ \|\nabla^b - f_{\mathsf{trmean/median}}(\nabla^m_{\{i\in[k]\}} \cup \nabla_{\{i\in[m]\}})\|_2; \quad \ldots \nabla^m_{i\in[k]} = f_{\mathsf{avg}}(\nabla_{\{i\in[m]\}}) + \gamma\nabla^{\mathsf{p}} \quad (5)$$

Note that in (5), we aim to compute $\gamma$ that maximizes the required $L_2$-norm distance. As we demonstrate in our evaluations, this extremely simple approach of crafting malicious updates outperforms the complex approaches proposed by Fang [10] attacks by very large margins for all the datasets.

### 3.3 AGR-agnostic attacks

Now, we consider the `agr-agnostic` adversary, who do not know the server's AGR algorithm.

**Intuition behind our attacks.** All the robust AGRs for FL tend to remove/attenuate malicious gradients based on one or more of the following criteria: 1) distances from the benign gradients [7, 4, 16, 1, 19], 2) distributional differences with the benign gradients [4, 17], 3) difference in $L_p$-norms of the benign and malicious gradients [17]. Hence, the intuition is as follows. The distance based defenses work by removing the gradients that lie outside of the clique formed by the benign gradients. Therefore, our attacks maximize the distance of a malicious gradient from a reference benign gradient, while ensuring that the malicious gradients lie within the clique of benign gradients. This also ensures that $L_p$-norms of the malicious and benign gradients are similar. To ensure distributional similarity, we use perturbations $\gamma\nabla^p$ with the similar distributions as the benign gradients. Next, we present optimization for two novel AGR-agnostic attacks based on the intuition.

**Attack-1 (Min-Max): Minimize maximum distance attack.** Our first attack ensures that the malicious gradients lie close to the clique of the benign gradients. Hence, we compute the malicious gradient such that its maximum distance from any other gradient is upper bounded by the maximum distance between any two benign gradients. (6) formalizes the corresponding optimization. Note that in order to maximize the impact of our attack, we keep all the malicious gradients the same. Hence, we formulate our attack objective in (6) for a single malicious gradient.

$$\underset{\gamma}{\text{argmax}}\ \max_{i\in[m]} \|\nabla^m - \nabla_i\|_2 \leq \max_{i,j\in[m]} \|\nabla_i - \nabla_j\|_2; \quad \ldots \nabla^m = f_{\mathsf{avg}}(\nabla_{\{i\in[m]\}}) + \gamma\nabla^p \quad (6)$$

**Algorithm 1** Algorithm to optimize $\gamma$

```
1:  Input: $\gamma_{\mathsf{init}}, \tau, \mathcal{O}, \nabla_{\{i\in[m]\}}$
2:  step $\leftarrow \gamma_{\mathsf{init}}/2$, $\gamma \leftarrow \gamma_{\mathsf{init}}$
3:  while $|\gamma_{\mathsf{succ}} - \gamma| > \tau$ do
4:      if $\mathcal{O}(\nabla_{\{i\in[n]\}}, \gamma) ==$ True then
5:          $\gamma_{\mathsf{succ}} \leftarrow \gamma$
6:          $\gamma \leftarrow (\gamma + \mathsf{step}/2)$
7:      else
8:          $\gamma \leftarrow (\gamma - \mathsf{step}/2)$
9:      end if
10:     step $=$ step$/2$
11: end while
12: Output $\gamma_{\mathsf{succ}}$
```

**Attack-2 (Min-Sum): Minimize sum of distances attack.** Our second AGR-agnostic Min-Sum attack ensures that the sum of squared distances of the malicious gradient from *all* the benign gradients is upper bounded by the sum of squared distances of any benign gradient from the other benign gradients. (7) formalizes the corresponding optimization. We keep all malicious gradients the same for maximum attack impact. Hence, we formulate our objective in (7) for a single malicious gradient; where $\nabla^m$'s are the same as in (6).

$$\underset{\gamma}{\text{argmax}} \sum_{i\in[m]} \|\nabla^m - \nabla_i\|_2^2 \leq \max_{i\in[m]} \sum_{j\in[m]} \|\nabla_i - \nabla_j\|_2^2 \quad (7)$$

### 3.4 Solving for the most effective scaling factor $\gamma$

Note that, the final objective of all of our attack optimizations is to search for the optimal scaling coefficient, $\gamma$; Algorithm 1 gives our $\gamma$-search algorithm for any of the optimizations.

For clarity of presentation of Algorithm 1, we assume an oracle $\mathcal{O}$ that takes the available benign gradients, $\nabla_{\{i\in[m]\}}$ and $\gamma$ as inputs, and outputs True if $\gamma$ satisfies the adversarial objective, otherwise outputs False. For instance, for our AGR-tailored attack on Krum, $\mathcal{O}$ outputs True if $f_{\mathsf{krum}}$ selects the malicious gradient computed using given $\gamma$, i.e., if (3) is satisfied.

Table 1: Comparing state-of-the-art model poisoning attacks and our attacks for `agr-tailored` and `agr-agnostic` adversaries. Our attacks outperform existing attacks by large margins. $A_\theta$ is accuracy of model without any attacks and $I_\theta$ is the attack impact. We assume 20% malicious clients; in Figure 1, we show that even with just 5% malicious clients, our attacks have noticeable attack impact.

| Dataset (Model) | AGR | No attack ($A_\theta$) | Attack impacts $I_\theta$ | | | | |
|---|---|---|---|---|---|---|---|
| | | | `agr-tailored` adversary | | `agr-agnostic` adversary | | |
| | | | Fang | Our attacks | LIE | Our attacks | |
| | | | | | | Min-Max | Min-Sum |
| CIFAR10 (Alexnet) | Krum | 53.5 | 19.8 | **43.1** | 18.1 | 13.7 | **30.2** |
| | MKrum | 67.6 | 11.2 | **36.8** | 19.7 | **31.7** | 30.4 |
| | Bulyan | 66.9 | 11.8 | **34.6** | 30.0 | 40.6 | **41.1** |
| | TrMean | 67.7 | 12.9 | **45.0** | 19.4 | **38.7** | 27.9 |
| | Median | 65.5 | 12.6 | **39.1** | 19.7 | 34.1 | **39.5** |
| FEMNIST (CNN) | Krum | 69.3 | 1.9 | **2.9** | 0.2 | 1.1 | **8.0** |
| | MKrum | 86.6 | 30.8 | **57.1** | 10.2 | **79.5** | 61.4 |
| | Bulyan | 86.1 | 35.6 | **40.5** | 20.5 | 18.7 | **30.4** |
| | TrMean | 86.7 | 7.9 | **20.1** | 14.4 | 24.7 | **25.2** |
| | Median | 77.1 | 0.8 | **18.2** | 5.8 | **19.8** | 16.6 |

Now, we describe Algorithm 1. The core idea of our algorithm is as follows: We start with a large $\gamma$ value. We reduce $\gamma$ in steps of size step until $\mathcal{O}$ returns True, e.g., for Krum, we reduce $\gamma$ until a malicious gradient is selected by $f_{\mathsf{krum}}$, i.e., (3) is satisfied for the first time. Our final $\gamma$ is always *greater* than this *minimum* $\gamma$ value that satisfies the objective. We halve the step size each time we update $\gamma$ in order to make the search finer. From the minimum $\gamma$ value, we increase $\gamma$ using updated step sizes step, until $\mathcal{O}$ returns False, i.e., for Krum, we increase $\gamma$ until $f_{\mathsf{krum}}$ *does not* select any malicious gradient, i.e., (3) is no more satisfied. Our final $\gamma$ is always *lower* than this *maximum* $\gamma$ value that satisfies the objective. Then we modify $\gamma$ repeatedly and oscillate between the minimum and maximum $\gamma$ values until the change in $\gamma$ is below a threshold $\tau$.

## 4 Evaluation

**Datasets and model architectures.** Our first dataset is *CIFAR10* [13], which has 10 classes and 60,000, 32 $\times$ 32 RGB images. We use 50 clients each with 1,000 samples and use test data of size 10,000 each. We simulate cross-silo FL for CIFAR10 and select all 50 clients in each FL epoch. We use Alexnet [14] as the global model architectures. Our second dataset is *FEMNIST [8, 9]*, which has 62 classes and 671,585 grayscale images. We use 3,400 clients each with her own data of her handwritten digits and letters. We simulate cross-device FL for FEMNIST and randomly select 60 out of 3400 clients in each FL epoch.

**Measurement metrics.** For a given FL setting, $A_\theta$ denotes the accuracy of the best globel model, over all the FL training epochs, in the benign setting without any attack, while $A^*_\theta$ denotes the accuracy under the given attack. We define *attack impact*, $I_\theta$, as *the reduction in the accuracy of the global model due to the attack*, hence for a given attack, $I_\theta = A_\theta - A^*_\theta$.

### 4.1 Evaluation of Our Attacks

Below, we compare our attacks with state-of-the-art model poisoning attacks, Fang [10] and LIE[1] [3], for all the adversaries from § 2. Table 1 gives the results; 'No attack' column shows accuracy $A_\theta$ of the global model in the benign setting, while the rest of the columns show the 'attack impact', $I_\theta$. For fair comparisons, we compare our AGR-tailored attacks with Fang attacks, which are also AGR-tailored attacks, and our AGR-agnostic attacks with LIE which is an AGR-agnostic attack.

**Comparing AGR-tailored attacks.** Table 1 shows that, *our AGR-tailored attacks outperform Fang attacks for all the FL settings by large margins*. For CIFAR10 with Krum AGR, our attacks are 2.5$\times$ more impactful than Fang, while for the rest of the AGRs, their impact is 3$\times$ to 4$\times$ more than that of Fang. For FEMNIST, the impacts of our attacks on Multi-krum, Trimmed-mean, and Median are respectively 2$\times$, 3$\times$, and 15$\times$, that of Fang attack; for Krum and Bulyan, our attacks have moderately higher impacts than Fang attacks. The superiority of our attacks is due to two reasons: First, our attacks find the most impactful perturbation vector for the given dataset, and second, our attacks fine tune the scaling factor, $\gamma$, to maximize attack impact. For Trimmed-mean and Median, our attacks

[1]We omit suffix "attack" when clear from context.

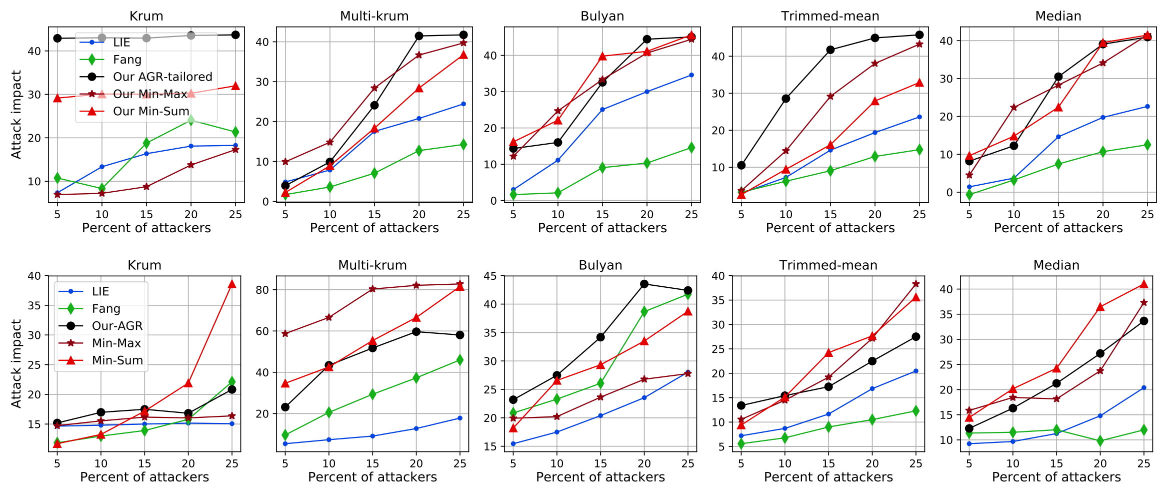

Figure 1: Effect of increasing percentage of malicious clients on the impacts of model poisoning attacks on FL. Upper row gives results for CIFAR10 and lower row for FEMNIST.

take more principled approach than Fang and find malicious gradients that deviate final aggregate maximally.

**Comparing AGR-agnostic attacks.** Table 1 shows that, *both of our AGR-agnostic attacks significantly outperform LIE, the state-of-the-art AGR-agnostic attack* for all the FL settings. We note significantly higher impacts (2× for CIFAR10 and 16× for FEMNIST) of Min-Sum on Krum than that of LIE. For the other AGRs, impacts of our attacks are 1.5× to 2× more for CIFAR10 and 1.5× to 8× for FEMNIST. LIE attack is ineffective, because it adds arbitrary small amounts of noises to compute its malicious gradients, while our AGR-agnostic attacks find the most malicious gradient within a ball formed by the benign gradients. We also note that, except FEMNIST with Krum, *one or more of our AGR-agnostic attacks also outperform AGR-tailored Fang attacks*. Due to the extreme non-iid nature of FEMNIST, the malicious gradients of our AGR-agnostic attacks can be far from benign gradients, which Krum can easily discard. Finally, we note that, attacker can chose any of our AGR-agnostic attacks, as both of them outperform existing attacks; in Appendix A.1, we discuss the reasons for differences in the impacts of Min-Max and Min-Sum attacks.

Table 2: Varying the perturbation vector has significant effect on the impact of our attacks.

| AGR | CIFAR10 | | | FEMNIST | | |
|---|---|---|---|---|---|---|
| | $\nabla^p_{\text{uv}}$ | $\nabla^p_{\text{std}}$ | $\nabla^p_{\text{sign}}$ | $\nabla^p_{\text{uv}}$ | $\nabla^p_{\text{std}}$ | $\nabla^p_{\text{sign}}$ |
| Krum | **43.6** | 12.6 | 14.7 | **7.3** | 1.6 | 6.7 |
| MKrum | 14.4 | **36.8** | 16.3 | 12.0 | 17.3 | **57.1** |
| Bulyan | 31.5 | **45.2** | 13.6 | 29.8 | 25.7 | **41.0** |
| TrMean | 13.5 | **44.8** | 12.9 | 18.0 | 20.9 | **24.0** |
| Median | 9.3 | **39.1** | 7.6 | 17.5 | 14.4 | **18.7** |

**Effect of perturbation vectors.** *For a given dataset, model, and AGR, varying the perturbation $\nabla^p$ significantly changes the impact of our attacks*, as Table 2 shows. In case of CIFAR10, for all AGRs but Krum, standard deviation based $\nabla^p_{\text{std}}$ perturbation has the highest attack impact, while for Krum, $\nabla^p_{\text{uv}}$ has the highest impact. For instance, attacks on CIFAR10 with Multi-krum using $\nabla^p_{\text{std}}$, $\nabla^p_{\text{uv}}$, and $\nabla^p_{\text{sgn}}$ have impacts of 36.8%, 14.4%, and 16.3%, respectively. We observe similar pattern for FEMNIST as well.

Above results show the importance of selecting the appropriate perturbation vector. To select the most effective perturbation vector, we emulate AGR-tailored attacks on FL using the benign data that the adversary has and use the perturbation that is most effective in the emulated FL attacks. In Appendix A.2, we show that impact of perturbations on our attacks effectively transfers from emulated to the real FL settings.

**Effect of the percentage of malicious clients.** Figure 1 shows the impacts of model poisoning attacks when the percentage of malicious clients in FL is varied from 5% to 24% for CIFAR10 and FEMNIST. Even with just 5% malicious clients, our attacks have noticeable impacts: our attacks reduce accuracy of Multi-Krum (Krum) on FEMNIST (CIFAR10) by 60% (43.0%), i.e., final global model accuracy is just 26% (10%). We also note that, all of our attacks outperform the existing attacks for all the FL settings. For CIFAR10 with Krum, our AGR-tailored attack has impact of more than 43%, i.e., it reduces accuracy to random guessing, 10%, even with just 5% clients.

**Acknowledgments.** This work was funded by NSF grant CPS-1739462.

## References

[1] Dan Alistarh, Zeyuan Allen-Zhu, and Jerry Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 4613–4623, 2018.

[2] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[3] Moran Baruch, Baruch Gilad, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. *Advances in Neural Information Processing Systems*, 2019.

[4] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. *arXiv preprint arXiv:1811.12470*, 2018.

[5] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643, 2019.

[6] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. *In Proceedings of 29th International Conference on Machine Learning*, 2012.

[7] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[8] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

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

[10] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Local model poisoning attacks to byzantine-robust federated learning. In *29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, aug 2020. USENIX Association.

[11] Matthew Jagielski, Aline Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures against regression learning. *39th IEEE Symposium on Security and Privacy*, 2018.

[12] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[15] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics*, 2017.

[16] El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3518–3527, 2018.

[17] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H Brendan McMahan. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*, 2019.

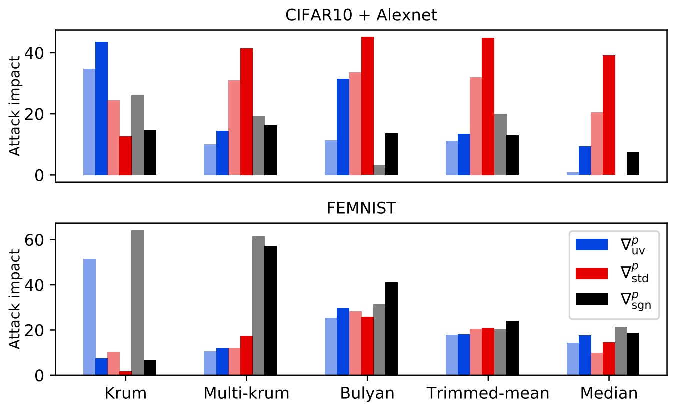

Figure 2: **Selecting an effective perturbation**: As explained in Section A.2, for a given FL setting, if the AGR is known, our adversary *emulates* attacks on the AGR using different $\nabla^p$'s and selects $\nabla^p$ with the highest attack impact (light bars). For unknown AGR, the adversary selects $\nabla^p$ which has the highest impact on the maximum number of AGRs. This selection method is reliable due to the transferability of attack impacts of $\nabla^p$ from the emulated settings (light bars) to actual FL setting (dark bars).

[18] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*, 2018.

[19] Dong Yin, Yudong Chen, Kannan Ramchandran, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *In Proceedings of the 35th International Conference on Machine Learning*, 2018.

## A Appendix

**Note**: This work is currently under submission at NDSS conference.

### A.1 Reasons for the differences in the impacts of Min-Max and Min-Sum attacks

Min-Max finds the malicious gradient whose maximum distance from a benign gradient is less than the maximum distance between any two benign gradient, while Min-Sum finds malicious gradient such that sum of its distances from all other gradients is less than sum of the distances of any benign gradient from other benign gradients. Therefore, the radius of search of malicious gradients of Min-Max is much larger than that of Min-Sum. Therefore, the malicious gradients of Min-Sum more effectively circumvent the filtering of Krum and Bulyan AGRs, and therefore, are more impactful against these AGRs. For the same reason, Multi-krum selects a lesser number of malicious gradients of Min-Max than that of Min-sum. But, as Multi-krum averages the selected gradients, Min-Max, with significantly more malicious gradients, damages the Multi-krum aggregate more effectively than Min-Sum.

### A.2 How to select the most effective perturbation direction?

In section 4.1, we showed that *selecting the appropriate perturbation is the key to an effective model poisoning attack.* However, as the adversary cannot know the end result of using a particular $\nabla^p$, she must decide the $\nabla^p$ to use in each epoch of FL. Below, we provide a simple yet effective method to select $\nabla^p$.

First, consider that the server's AGR is known. We propose that the adversary *emulate* AGR-tailored attacks (Section 3.2) on the given FL settings and select as its final $\nabla^p$ the most effective $\nabla^p$ in the emulated setting. An example of emulated AGR-tailored attack on CIFAR10 with Krum is as follows: For CIFAR10 we assume total of 50 clients including 10 malicious clients. Hence, the adversary emulates an FL setting with 10 benign and 2 malicious clients and mounts the AGR-tailored attack. In Figure 2, for each $\nabla^p$, the lighter bars show the impacts of attacks on the emulated FL settings.

We compare the light and dark bars in Figure 2 and note that, *the relative effects of different perturbations are the same across different AGRs, datasets, and models in both emulated FL settings (light bars) and actual FL (dark bars)*. In other words, the most effective perturbation in an emulated FL setting, is also the most effective in the corresponding actual FL. This transferability allows us to reliably select the most effective perturbation when the AGR is known. Finally, when the AGR is unknown, we simply pick the perturbation with the highest impact across maximum number of AGRs. For instance, we select $\nabla^p_{\mathsf{sgn}}$ for FEMNIST due to its highest attack impact on all AGRs. For CIFAR10, we choose $\nabla^p_{\mathsf{std}}$ as it has the maximum impact on all but Krum AGR.