Can You Really Backdoor Federated Learning?

Ziteng Sun∗
Cornell University
zs335@cornell.edu

Peter Kairouz
Google
kairouz@google.com

Ananda Theertha Suresh
Google
theertha@google.com

H. Brendan McMahan
Google
mcmahan@google.com

Abstract

The decentralized nature of federated learning makes detecting and defending against adversarial attacks a challenging task. This paper focuses on backdoor attacks in the federated learning setting, where the goal of the adversary is to reduce the performance of the model on targeted tasks while maintaining a good performance on the main task. Unlike existing works, we allow non-malicious clients to have correctly labeled samples from the targeted tasks. We conduct a comprehensive study of backdoor attacks and defenses for the EMNIST dataset, a real-life, user-partitioned, and non-iid dataset. We observe that in the absence of defenses, the performance of the attack largely depends on the fraction of adversaries present and the “complexity” of the targeted task. Moreover, we show that norm clipping and “weak” differential privacy mitigate the attacks without hurting the overall performance. We have implemented the attacks and defenses in TensorFlow Federated (TFF), a TensorFlow framework for federated learning. In open sourcing our code, our goal is to encourage researchers to contribute new attacks and defenses and evaluate them on standard federated datasets.

1 Introduction

Modern machine learning systems can be vulnerable to various kinds of failures, such as bugs in preprocessing pipelines and noisy training labels, as well as attacks that target each step of the system’s training and deployment pipelines. Examples of attacks include data and model update poisoning (5; 21), model evasion (29; 5; 16), model stealing (30), and data inference attacks on users’ private training data (27).

The distributed nature of federated learning (23), particularly when augmented with secure aggregation protocols (7), makes detecting and correcting for these failures and attacks a particularly challenging task. Adversarial attacks can be broadly classified into two types based on the goal of the attack, untargeted or targeted attacks. Under untargeted attacks (6; 15; 12), the goal of the adversary is to corrupt the model in such a way that it does not achieve a near-optimal performance on the main task at hand (e.g., classification) often referred to as the primary task. Under targeted attacks (often referred to as backdoor attacks) (10; 19; 17), the goal of the adversary is to ensure that the learned model behaves differently on certain targeted sub-tasks while maintaining good overall performance on the primary task. For example, in image classification, the attacker may want the model to misclassify some “green cars” as birds while ensuring that other cars are correctly classified.

For both targeted and untargeted attacks, the attacks can be further classified into two types based on the capability of the attacker, model update poisoning or data poisoning. In data poisoning

∗Work done while ZS was an intern at Google.
attacks \((5; 28; 33; 25; 18)\), the attacker can change a subset of all the training samples which is unknown to the learner. In federated learning systems, since the training process is done on local devices, fully compromised clients can change the model update completely, which is called a model poisoning attack \((3; 4)\). Model update poisoning attacks are even harder to counter when secure aggregation (SecAgg) \((7)\), which ensures that the server cannot inspect each user’s update, is deployed in the aggregation of local updates.

Since untargeted attacks reduce the overall performance of the primary task, they are easier to detect. On the other hand, backdoor targeted attacks are harder to detect as the goal of the adversary is often unknown a priori. Hence, following \((3; 4)\), we consider targeted model update poisoning attacks and refer to them as backdoor attacks. Existing approaches against backdoor attacks \((28; 20; 31; 13; 32; 26)\) either require a careful examination of the training data or full control of the training process at the server, which may not apply in the federated learning case. We evaluate various attacks proposed in recent papers and defenses on a medium scale federated learning task with more realistic parameters using TensorFlow Federated \((1)\). Our goal, in open sourcing our code, is to encourage researchers to evaluate new attacks and defenses on standard tasks.

2 Backdoor Attack Scenario

We consider the notations and definitions of federated learning as defined in \((23)\). In particular, let \(K\) be the total number of users. At each round \(t\), the server randomly selects \(C \cdot K\) clients for some \(C < 1\). Let \(S_t\) be this set and \(n_k\) be the number of samples at client \(k\). Denote the model parameters at round \(t\) by \(w_t\). Each selected user computes a model update, denoted by \(\Delta w_t^k\), based on their local data. The server updates its model by aggregating the \(\Delta w_t^k\)'s, i.e.,

\[
w_{t+1} = w_t + \eta \frac{\sum_{k \in S_t} n_k \Delta w_t^k}{\sum_{k \in S_t} n_k}.
\]

where \(\eta\) is the server learning rate. We model the parameters of backdoor attacks as follows.

**Sampling of adversaries.** If \(\epsilon\) fraction of the clients are completely compromised, then each round may contain anywhere between 0 and \(\min(\epsilon \cdot K, C \cdot K)\) adversaries. Under random sampling of clients, the number of adversaries in each round follows a hypergeometric distribution. We refer to this attack model as the random sampling attack. Another model we consider in this work is the fixed frequency attack, where a single adversary appears in every \(f\) rounds \((3; 4)\). For a fair comparison between the two attack models, we set the frequency to be inversely proportional to the number of total number of attackers (i.e., \(f = \frac{1}{(\epsilon \cdot C \cdot K)}\)).

**Backdoor tasks.** Recall that in backdoor attacks, the goal of the adversary is to ensure that the model fails on some targeted tasks. For example, in text classification the backdoor task might be to suggest a particular restaurant’s name after observing the phrase “my favorite restaurant is”. Unlike \((3; 4)\), we allow non-malicious clients to have correctly labeled samples from the targeted backdoor tasks. For instance, if the adversary wants the model to misclassify some green cars as birds, we allow non-malicious clients to have samples from these targeted green cars correctly labeled as cars.

Further, we form the backdoor task by grouping examples from multiple selected “target clients”. Since examples from different target clients follow different distributions, we refer to the number of target clients as the “number of backdoor tasks” and explore its effect on the attack’s success rate. Intuitively, the more backdoor tasks we have, the richer the feature space the attacker is trying to break, and therefore the harder it is for the attacker to successfully backdoor the model without breaking its performance on the main task.

3 Model Update Poisoning Attacks

We focus on model update poisoning attacks based on the model replacement paradigm proposed by \((3; 4)\). When only one attacker is selected in round \(t\) (WLOG assume it is client 1), the attacker attempts to replace the whole model by a backdoored model \(w^*\) by sending

\[
\Delta w_t^1 = \beta (w^* - w_t).
\]

\(^2\)While \((23)\) considers relatively small problems, in more realistic scenarios for mobile devices we might have \(K = 10^2\) or higher, with the number of clients selected \(C \cdot K\) typically constant, say 100 to 1000 per round.
where $\beta = \frac{\sum_{k \in S_t} n_k}{\eta n_k}$ is a boost factor. Then we have

$$\Delta w_{t+1} = w^* + \eta \frac{\sum_{k \in S_t} n_k \Delta w^k}{\sum_{k \in S_t} n_k},$$

which will be in a small neighbourhood of $w^*$ if we assume the model has sufficiently converged and hence the other updates $\Delta w^k$ for $k > 1$ are small. If multiple attackers appear in the same round, we assume that they can coordinate with each other and divide this update evenly.

**Obtaining a backdoored model.** To obtain a backdoored model $w^*$, we assume that the attacker has a set $D_{\text{mal}}$ which describes the backdoor task – for example, different kinds of green cars labeled as birds. We also assume the attacker has a set of training samples generated from the true distribution $D_{\text{trn}}$. Note that for practical applications, such data may be harder for the attacker to obtain.

**Unconstrained boosted backdoor attack.** In this case, the adversary trains a model $w^*$ based on $w_t, D_{\text{mal}}$ and $D_{\text{trn}}$ without any constraints and sends the update based on (1) back to the service provider. One popular training strategy is to initialize with $w_t$ and train the model with $D_{\text{trn}} \cup D_{\text{mal}}$ for a few epoches. This attack generally results in a large update norm and can serve as a baseline.

**Norm bounded backdoor attack.** Unconstrained backdoor attacks can be defended by norm clipping as discussed below. To overcome this, we consider the norm bounded backdoor attack. Here at each round, the model trains on the backdoor task subject to the constraint that the model update is smaller than $M/\beta$. Thus, model update has norm bounded by $M$ after boosted by a factor of $\beta$. This can be done by training the model using multiple rounds of projected gradient descent, where in each round we train the model using the unconstrained training strategy and project it back to the $\ell_2$ ball of size $M/\beta$ around $w_t$.

### 4 Defenses

We consider the following defenses for backdoor attacks.

**Norm thresholding of updates.** Since boosted attacks are likely to produce updates with large norms, a reasonable defense is for the server to simply ignore updates whose norm is above some threshold $M$; in more complex schemes $M$ could even be chosen in randomized fashion. However, in the spirit of investigating what a strong adversary might accomplish, we assume the adversary knows the threshold $M$, and can hence always return malicious updates within this magnitude. Giving this strong advantage to the adversary makes the norm-bounding defense equivalent to the following norm-clipping approach:

$$\Delta w_{t+1} = \sum_{k \in S_t} \frac{\Delta w^k_{t+1}}{\max(1, \|\Delta w^k_{t+1}\|_2/M)}.$$

This model update ensures that the norm of each model update is small and hence less susceptible to the server.

**(Weak) differential privacy.** A mathematically rigorous way for defending against backdoor tasks is to train models with differential privacy (22; 14; 2). These approaches were extended to the federated setting by (24), by first clipping updates (as above) and then adding Gaussian noise. We explore the effect of this method. However, traditionally the amount of noise added to obtain reasonable differential privacy is relatively large. Since our goal is not privacy, but instead preventing attacks, we add a small amount of noise that is empirically sufficient to limit the success of attacks.

### 5 Experiments

In the above backdoor attack framework, we conduct experiments on the EMNIST dataset (11; 9). This dataset is a writer-annotated handwritten digit classification dataset collected from 3383 users with roughly 100 images of digits per user. Each of them has their unique writing style. We train a five-layer convolution neural network with two convolution layers, one max-pooling layer and two dense layers using federated learning in the TensorFlow Federated framework (1). At each round of training, we select $C \cdot K = 30$ clients. Each client trains the model with their own local data for 5 epochs with batch size 20 and client learning rate 0.1. We use a server learning rate of 1.
In the experiment, we consider the backdoor task as classifying 7s from multiple selected “target clients” as 1s. Note that our attack approach does not require 7s from other clients to be classified as 1s. Since 7s coming from different target clients follow different distributions (because they have different writing styles), we refer to the number of target clients as the “number of backdoor tasks”.

**Random sampling vs. fixed frequency attacks.** To begin with, we conduct experiments for the two attack models discussed in Section 2 under different fractions of adversaries. The results are shown in Figure 1 (for unconstrained attack) and Figure 2 (for norm bounded attack). Additional plots are shown in Figure 5 and Figure 6 in the appendix. The figures show that both attack models have similar behaviors, despite fixed frequency attacks being slightly more effective than random sampling attacks. Furthermore, in the fixed frequency attack, it is easier to see if the attack happened in a particular round or not. Hence, to provide additional advantage for the attacker and for ease of interpretability, we focus our analysis on fixed-frequency attacks in the rest of this section.

**Fraction of corrupted users.** In Figure 1 and Figure 2 (also Figure 5 and Figure 6 in the appendix), we consider a malicious task with 30 backdoor tasks (around 300 images). We perform unconstrained attacks and norm-bounded attacks with $\epsilon = 3.3\%, 0.33\%$ fraction of users being malicious. Both fixed-frequency attack (left column) and random sampling (right column) attacks are considered. For fixed-frequency attack, this corresponds to attacking frequency of 1 (attacking every round), and 1/10 (once every ten rounds). From the above experiment, we can infer that the backdoor attack success largely depends on the fraction of adversaries and the performance of backdoor attack degrades as the fraction of fully compromised users falls below 1%.

**Number of backdoor tasks.** The number of backdoor tasks affects the performance in two ways. First, the more backdoor tasks we have, the harder it is to backdoor a fixed-capacity model while maintaining its performance on the main task. Second, since we assume benign users have correct samples from the backdoor task, they can correct the attacked model with these samples. In Figure 3, we consider norm bounded attack with norm bound 10 and 10, 20, 30, 50 backdoor tasks. We can see from the plot that the more backdoor tasks we have, the harder it is to fit a malicious model.
Figure 2: Constrained attack with norm bound 10 for fixed-frequency attacks (left column) and random sampling attack (right column) with different fractions of attackers. Green line is the cumulative mean for the backdoor accuracy.

(a) Attack frequency = 1 ($\epsilon = 3.3\%$)  
(b) Number of attackers = 113 ($\epsilon = 3.3\%$)  
(c) Attack frequency = 1/10 ($\epsilon = 0.33\%$)  
(d) Number of attackers = 11 ($\epsilon = 0.33\%$)

Figure 3: The Effect of Backdoor Size for Constrained Attack with Norm Bound 10.

(a) Backdoor Size = 10  
(b) Backdoor Size = 20  
(c) Backdoor Size = 30  
(d) Backdoor Size = 50
Norm bound for the update. In Figure 4(a), we consider norm bounded update from each user. We assume one attacker appears in every round, which corresponds to $\epsilon = 3.3\%$ corrupted users, and we consider norm bounds of 3, 5, and 10 (the 90 percentile of benign users’ updates are below 2 for most of the rounds), which translates to 0.1, 0.17, 0.33 norm bound for the update before boosting. We can see from the plot that selecting 3 as the norm bound will successfully mitigate the attack with almost no effect on the performance of the main task. Hence we can see that norm bounding may be a valid defense for current backdoor attacks.

![Figure 4(a) Norm clipping bound: Blue - unattacked baseline, Green: 3, Red: 5, Black: 10](image)

![Figure 4(b) Gaussian noise (norm bound = 5). Red: $\sigma = 0$, Green: 3, Red: 5, Black: 10](image)

Figure 4: Effect of norm bounding and Gaussian noise. Dotted: main task. Solid: backdoor task.

Weak differential privacy In Figure 4(b), we consider norm bounding plus adding Gaussian noise. We use norm bound of 5, which itself would not mitigate the attack, and add independent Gaussian noise with variance 0.025 to each coordinate. From the plots, we can see that adding Gaussian noise can also help mitigate the attack beyond norm clipping without hurting the overall performance much. We note that similar to previous works on differential privacy (2), we do not provide a recipe for selecting the norm bound and variance of the Gaussian noise. Rather, we show that some reasonable values motivated by differential privacy literature perform well. Discovering algorithms to learn these bounds and noise values remains an interesting open research direction.

6 Discussion

We studied backdoor attacks and defenses for federated learning under the more realistic EMNIST dataset. In the absence of any defense, we showed that the performance of the adversary largely depends on the fraction of adversaries present. Hence, for reasonable success, there needs to be a large number of adversaries. Perhaps surprisingly norm clipping limits the success of known backdoor attacks considerably. Furthermore, adding a small amount of Gaussian noise, in addition to norm clipping, can help further mitigate the effect of adversaries. This gives rise to several interesting questions.

**Better attacks and defenses.** In the norm bounded case, multiple iterations of “pre-boosted” projected gradient descent may not be the best possible attack in a single round. In fact, the adversary may attempt to directly craft the “worst-case” model update that satisfies the norm bound (without any boosting). Moreover, if the attacker knows they can attack in multiple rounds, there might be better strategies for doing so under a norm bound. Similarly, more advanced defenses should be investigated.

**Effect of model capacity.** Another factor that may affect the performance of backdoor attacks is the model capacity, especially that it is conjectured that backdoor attacks use the spare capacity of the deep network (20). How model capacity interacts with backdoor attacks is an interesting question to consider both from the theoretical and practical sides.

**Interaction of defenses with SecAgg.** Existing approaches on range proofs (e.g. BulletProof (8)) can guarantee this when using secure multiparty computation but how to implement them in a computationally and communication efficient way is still an active research direction. This can also be made compatible with SecAgg if we have an efficient implementation of multi-party range proof.
References


A Additional figures for experiments

Figure 5: Unconstrained attack for fixed-frequency attacks (left column) and random sampling attack (right column) with different fractions of attackers. Green line is the cumulative mean for the backdoor accuracy.

(a) Attack frequency = 1/3 ($\epsilon = 1.1\%$)

(b) Number of attackers = 38 ($\epsilon = 1.1\%$)

(c) Attack frequency = 1/5 ($\epsilon = 0.67\%$)

(d) Number of attackers = 23 ($\epsilon = 0.67\%$)

Figure 6: Constrained attack with norm bound 10 for fixed-frequency attacks (left column) and random sampling attack (right column) with different fractions of attackers. Green line is the cumulative mean for the backdoor accuracy.

(a) Attack frequency = 1/3 ($\epsilon = 1.1\%$)

(b) Number of attackers = 38 ($\epsilon = 1.1\%$)

(c) Attack frequency = 1/5 ($\epsilon = 0.67\%$)

(d) Number of attackers = 23 ($\epsilon = 0.67\%$)