ANALYZING FEDERATED LEARNING THROUGH AN ADVERSARIAL LENS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning distributes model training among a multitude of agents, who guided by privacy concerns, perform training using their local data but share only the parameter updates, for iterative aggregation at the server. In this work, we operate within the confines of the federated learning paradigm, and explore the threat of *targeted backdoor attacks* on the global model, via *model poisoning* (as opposed to data poisoning) initiated by a single malicious agent with no collusion. Specifically, we consider a highly constrained adversary that (i) has partial observability into the model parameter space due to lack of knowledge of other agents' updates; (ii) operates in an environment where training data are i.i.d. between the agents (hence spurious updates will easily standout among benign ones); and (iii) has its single malicious update (mostly) cancelled by multiple benign updates. For this highly constrained adversary, we propose a sequence of model poisoning strategies that starting with malicious update boosting, incrementally introduce various forms of regularization, followed by parameter estimation to improve on both attack success and stealth. For each strategy, we analyze its impact on the model parameter space, design possible detection approaches and incorporate the insights for gaining stealth. Finally, we use a suite of interpretability techniques to generate visual explanations of the decision boundary and internal feature representations of a benign and malicious model and show that the explanations are nearly visually indistinguishable. Our evaluation results indicate that even a highly constrained adversary can generate successful model poisoning attacks while simultaneously maintaining stealth, thus highlighting the vulnerability of the federated learning setting and the need for effective defense strategies.

1 INTRODUCTION

Federated learning introduced by McMahan et al. (2017) has recently emerged as a popular implementation of distributed stochastic optimization for large-scale deep neural network training. It is formulated as a multi-round strategy in which the training of a neural network model is distributed between multiple agents. In each round, a random subset of agents, with local data and computational resources, is selected for training. The selected agents perform model training and share only the parameter updates with a centralized parameter server, that facilitates aggregation of the updates. Motivated by privacy concerns, the server is designed to have no visibility into an agents' local data and training process. The aggregation algorithm is agnostic to the data distribution at the agents.

In this work, we exploit this lack of transparency in the agent updates, and explore the possibility of a single malicious agent performing *model poisoning attack* with the goal of introducing *targeted backdoor* into a jointly trained global model. In each round, the malicious agent generates its update by optimizing for a malicious objective different than the training loss for federated learning. Attack stealth, loosely defined as indistinguishability from benign updates, is achieved by factoring in the possible defenses at the server into the malicious objective. Finally, to better understand the inherent resilience of federated learning systems to model poisoning, we restrict the malicious agent to operate in a highly constrained setting while still trying to successfully introduce an undetectable targeted backdoor.

Our Contributions: Towards this end, we propose a sequence of model poisoning attacks. For each strategy we alternately play the role of both an attacker and a defender. We start with mali-

cious update boosting (to negate the combined effect of the benign agents). While this generates a successful attack agnostic of the aggregation algorithm, we design defenses to detect the malicious update. Subsequent schemes employ various optimization techniques to improve the attack by balancing the impact of boosting (for attack success) with training loss (for stealth). Finally, we use a suite of interpretability techniques to generate visual explanations of the decision boundary and internal feature representations of a benign and malicious model. We show that the explanations are nearly visually indistinguishable establishing the stealth of the poisoning attack.

Summary of Results: We demonstrate that constrained adversaries can introduce targeted backdoors into the models held at the server. These backdoors result in highly confident misclassification of certain data examples by the global model. Working with the Fashion-MNIST Xiao et al. (2017) dataset, we introduce these backdoors for settings with 10 and 100 agents. We analyze the weight updates both qualitatively and quantitatively and show that the attacks become more challenging as the server implements different detection mechanisms. We also use previous-step estimation to improve the performance of our attacks.

Related Work: While data poisoning attacks Koh & Liang (2017); Chen et al. (2017a); Jagielski et al. (2018) have been widely studied, model poisoning attacks are relatively less explored. Blanchard et al. (2017); Chen et al. (2017b) consider a federated learning setup with a single Byzantine agent sending arbitrary gradient updates. However, the adversarial goal in both cases is to prevent a distributed implementation of the Stochastic Gradient Descent (SGD) algorithm from converging. Concurrent work by Bagdasaryan et al. (2018) considers multiple colluding agents performing poisoning via model replacement at convergence time. In contrast, our goal is to introduce targeted misclassification in the global model even when it is far from convergence while maintaining its accuracy for most tasks.

For a Byzantine adversary, Chen et al. (2017b) demonstrate the vulnerability of techniques that rely on linear combination of gradient updates and instead propose a geometric median based approach for aggregation. However, in large data settings, median computation dominates the training time rendering it impractical. Furthermore, Bagdasaryan et al. (2018) note that the Krum sampling based defense, proposed by Blanchard et al. (2017) is also ineffective against convergence-time model poisoning attacks. Recently, Chen et al. (2018) used ideas from coding theory to improve resilience of distributed training to Byzantine attacks. Each agent computes redundant gradients, and sends a linear combination of those gradients to the server. A decoder is used to retrieve the sum of the gradients from the updates. However, linear coding approach does not work when data are non-i.i.d. Our attacks are designed to achieve stealth even in the i.i.d. data setting, and hence will transfer to the non-i.i.d. cases. Thus, these defenses will be equally ineffective against our attacks.

2 Federated Learning and Model Poisoning

In this section, we formulate both the learning paradigm and the threat model that we consider throughout the paper. Operating in the federated learning paradigm, where model weights are shared instead of data, gives rise to the *model poisoning* attacks that we investigate.

2.1 Federated Learning

The federated learning setup consists of K agents, each with access to data $\mathcal{D}_i$, where $|\mathcal{D}_i| = l_i$. The total number of samples is $\sum_i l_i = l$. Each agent keeps its share of the data (referred to as a *shard*) private, i.e. $\mathcal{D}_i = \{\mathbf{x}_1^i \cdots \mathbf{x}_{l_i}^i\}$ is not shared with the server S. The objective of the server is to learn a global parameter vector $\mathbf{w}_G \in \mathbb{R}^n$, where n is the dimensionality of the parameter space. This parameter vector minimizes the loss[1] over $\mathcal{D} = \cup_i \mathcal{D}_i$ and the aim is to generalize well over $\mathcal{D}_{\text{test}}$, the test data.

At each time step t, a random subset of k agents is chosen for aggregation. Every agent $i \in [k]$, *minimizes the empirical loss over its own data shard* $\mathcal{D}_i$, by starting from the global weight vector $\mathbf{w}_G^t$ and running an algorithm such as SGD for E epochs with a batch size of B. At the end of its run, each agent obtains a local weight vector $\mathbf{w}_i^{t+1}$ and computes its local update $\boldsymbol{\delta}_i^{t+1} = \mathbf{w}_i^{t+1} - \mathbf{w}_G^t$, which is sent back to the server. To obtain the global weight vector $\mathbf{w}_G^{t+1}$ for the next iteration,

[1] approximately for non-convex loss functions since global minima cannot be guaranteed

any aggregation mechanism can be used. McMahan et al. (2017) assume synchronous training (i.e., server waits till it has received updates from all the agents selected for the time step) and uses weighted averaging based aggregation:

$$\mathbf{w}_G^{t+1} = \mathbf{w}_G^t + \sum_{i \in [k]} \alpha_i \boldsymbol{\delta}_i^{t+1}, \tag{1}$$

where $\frac{l_i}{l} = \alpha_i$ and $\sum_i \alpha_i = 1$. Furthermore, the learning is explicitly designed to handle non-i.i.d partitioning of training data among the different agents.

2.2 Threat Model: Model Poisoning

Traditional poisoning attacks considered in the literature (Koh & Liang, 2017; Chen et al., 2017a; Jagielski et al., 2018) deal with a malicious agent who poisons some fraction of the *data* in order to ensure that the learned model satisfies some adversarial goal. We consider instead an agent who poisons the *model updates* it sends back to the server.

This attack is a plausible threat in the federated learning setting as the model updates from the agents can (i) directly influence the parameters of the global model via the aggregation algorithm; and (ii) display high variability, due to the non-i.i.d local data at the agents, making it harder to isolate the benign updates from the malicious ones.

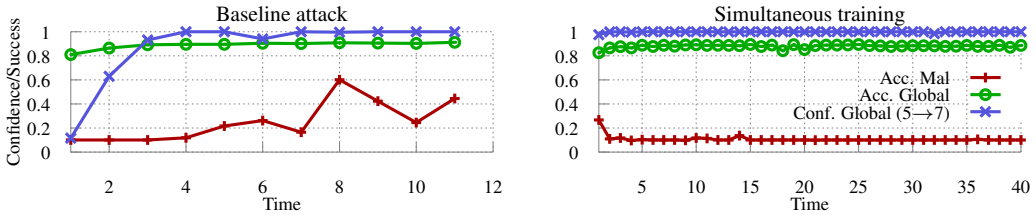
Adversary Model: To explore the extent of vulnerability of the global model in the federated setting we assume the following adversary model: (i) there is exactly one malicious agent and no collusion between agents (the effect of the malicious update on the global could be very limited); (ii) the local data distribution between agents is i.i.d (making it easier to discriminate between benign and possible malicious updates and harder to achieve attack stealth); (iii) limited to no exploitation of the aggregation algorithm (i.e., aggregation agnostic attacks). Our aim is to explore the possibility of a successful model poisoning attack even for a highly constrained adversary.

Let the index of the malicious agent be m. We assume that the malicious agent has access to a subset of the training data $\mathcal{D}_m$ as well as to auxiliary data $\mathcal{D}_{\text{aux}}$ drawn from the same distribution as the training and test data that are part of its adversarial objective. We assume that the server performs synchronous training in each time step.

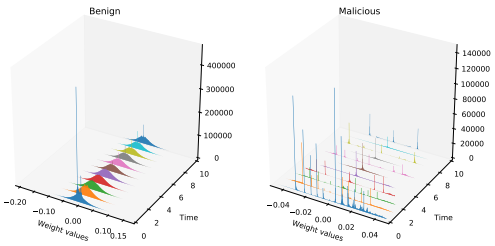
In particular, a malicious agent can have one of two objectives with regard to the loss and/or classification of a data subset at any time step t in the model poisoning setting:
1. Increase the overall loss: In this case, the malicious agent wishes to increase the overall loss on a subset $\mathcal{D}_{\text{aux}} = \{\mathbf{x}_i, y_i\}_{i=1}^r$ of the data. The adversarial objective is in this setting is $\mathcal{A}(\mathcal{D}_m, \{\mathbf{x}_i, y_i\}_{i=1}^r, \mathbf{w}_G^t) = \text{argmax}_{\mathbf{w}_G^t} L(\{\mathbf{x}_i, y_i\}_{i=1}^r, \mathbf{w}_G^t)$, where $L(\cdot, \cdot)$ is an appropriately defined loss function. This objective corresponds to the malicious agent attempting to cause *untargeted misclassification*.
2. Obtain desired classification outcome: The malicious agent has data samples $\{\mathbf{x}_i\}_{i=1}^r$ with true labels $\{y_i\}_{i=1}^r$ that have to be classified as desired target classes $\{\tau_i\}_{i=1}^r$, implying that the adversarial objective is $\mathcal{A}(\mathcal{D}_m, \{\mathbf{x}_i, \tau_i\}_{i=1}^r, \mathbf{w}_G^t) = \text{argmin}_{\mathbf{w}_G^t} L(\{\mathbf{x}_i, \tau_i\}_{i=1}^r, \mathbf{w}_G^t)$. This corresponds to a *targeted misclassification* attempt by the malicious agent.

In this paper, we will focus on malicious agents trying to attain the second objective, i.e. targeted misclassification. At first glance, the problem seems like a simple one for the malicious agent to solve. However, it does not have access to the global parameter vector $\mathbf{w}_G^t$ for the current iteration as is the case in standard poisoning attacks (Muñoz-González et al., 2017; Koh & Liang, 2017) and can only influence it though the weight update $\boldsymbol{\delta}_m^t$ it provides to the server S. The simplest formulation of the optimization problem the malicious agent has to solve such that her objective is achieved on the tth iteration is then

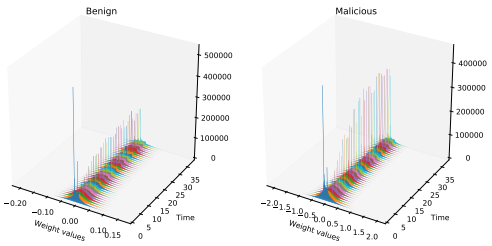
$$\begin{aligned} &\underset{\boldsymbol{\delta}_m^t}{\text{argmin}}\, L(\{\mathbf{x}_i, \tau_i\}_{i=1}^r, \mathbf{w}_G^t), \\ \text{s.t.} \quad &\mathbf{w}_G^t = \mathbf{w}_G^{t-1} + \sum_{i \in [k] \setminus m} \alpha_i \boldsymbol{\delta}_i^t + \alpha_m \boldsymbol{\delta}_m^t. \end{aligned} \tag{2}$$



(a) Metrics of interest for baseline (**left**) and simultaneous training attacks (**right**). Unified legend in **right** plot.



(b) Baseline attack weight updates.



(c) Simultaneous training weight updates.

Figure 1: Metrics of interest and weight update distributions for the *baseline* and *simultaneous training* attacks. Figures 1b and 1c show weight update distributions for both benign (**left**) and malicious agents (**right**).

2.3 Experimental setup

In order to illustrate how our attack strategies work with actual data and models, we use the Fashion-MNIST [2] dataset (Xiao et al., 2017) of 28×28 grayscale images of clothing and footwear items. The training set contains 60,000 data samples while the test set has 10,000 samples. We use a Convoluational Neural Network achieving 91.7% accuracy on the test set for the model architecture. We study the case with the number of agents k set to 10 and 100. When $k = 10$, all the agents are chosen at every iteration, while with $k = 100$, a tenth of the agents are chosen at random every iteration.

For most of our experiments, we consider the case when $r = 1$, which implies that the malicious agent aims to misclassify a single example in a desired target class. For our experiments, we choose a sample from the test set for the Fashion-MNIST dataset at random. The sample belongs to class '5' (sandal) with the aim of misclassifying it in class '7' (sneaker).

3 Strategies for Model Poisoning attacks

We begin by investigating baseline attacks which do not conform to any notion of stealth. We then show how simple detection methods at the server may expose the malicious agent and explore the extent to which modifications to the baseline attack can bypass these methods.

3.1 Limited information poisoning objective

In order to solve the exact optimization problem needed to achieve their objective, the malicious agent needs access to the *current value* of the overall parameter vector $\mathbf{w}_G^t$, which is inaccessible. This occurs due to the nature of the federated learning algorithm, where S computes $\mathbf{w}_G^t$ once it has received updates from all agents. In this case, they have to optimize over an *estimate* of the value of $\mathbf{w}_G^t$:

$$\begin{aligned} &\mathcal{A}(\mathcal{D}_m, \{\mathbf{x}_i, \tau_i\}_{i=1}^r, \hat{\mathbf{w}}_G^t), \\ \text{s.t.} \quad &\hat{\mathbf{w}}_G^t = f(\mathcal{I}_m^t), \end{aligned} \tag{3}$$

[2] Serves as a drop-in replacement for the commonly used MNIST dataset (LeCun et al., 1998), which is not representative of modern computer vision tasks

where $f(\cdot)$ is an estimator for $\hat{\mathbf{w}}_G^t$ based on all the information $\mathcal{I}_m^t$ available to the adversary. We refer to this as the limited information poisoning objective. The problem of choosing a good estimator is deferred to Section 4 and the strategies discussed in the remainder of this section make the assumption that $\hat{\mathbf{w}}_G^t \approx \mathbf{w}_G^{t-1} + \alpha_m \boldsymbol{\delta}_m^t$. In other words, the malicious agent ignores the effects of other agents. As we shall see, this assumption is often enough to ensure the attack works in practice.

3.2 Baseline attack

Using the approximation that $\hat{\mathbf{w}}_G^t \approx \mathbf{w}_G^{t-1} + \alpha_m \boldsymbol{\delta}_m^t$, the malicious agent just has to meet the adversarial objective $\operatorname{argmin}_{\boldsymbol{\delta}_m^t} L(\{\mathbf{x}_i, \tau_i\}_{i=1}^r, \hat{\mathbf{w}}_G^t)$. Depending on the exact structure of the loss, an appropriate optimizer can be chosen. For our experiments, we will rely on gradient-based optimizers such as SGD which work well for neural networks. In order to overcome the effect of scaling by α_m at the server, the final update $\tilde{\boldsymbol{\delta}}_m^t$ that is returned, has to be *boosted*.

Explicit Boosting: Mimicking a benign agent, the malicious agent can run E_m steps of a gradient-based optimizer starting from $\mathbf{w}_G^{t-1}$ to obtain $\tilde{\mathbf{w}}_m^t$ which minimizes the loss over $\{\mathbf{x}_i, \tau_i\}_{i=1}^r$. The malicious agent then obtains an initial update $\tilde{\boldsymbol{\delta}}_m^t = \tilde{\mathbf{w}}_m^t - \mathbf{w}_G^{t-1}$. However, since the malicious agent's update tries to ensure that the model learns labels different from the true labels for the data of its choice ($\mathcal{D}_{\text{aux}}$), it has to overcome the effect of scaling, which would otherwise mostly nullify the desired classification outcomes. This happens because the learning objective for all the other agents is very different from that of the malicious agent, especially in the i.i.d. case. The final weight update sent back by the malicious agent is then $\boldsymbol{\delta}_m^t = \lambda \tilde{\boldsymbol{\delta}}_m^t$, where λ is the factor by which the malicious agent *boosts* the initial update. Note that if the assumption $\hat{\mathbf{w}}_G^t \approx \mathbf{w}_G^{t-1} + \alpha_m \boldsymbol{\delta}_m^t$ holds, and $\lambda = \frac{1}{\alpha_m}$, then $\hat{\mathbf{w}}_G^t \approx \mathbf{w}_m^t$, implying that the global weight vector should now satisfy the malicious agent's objective. This method indirectly accounts for the presence of the other agents when using a boosting factor of $\frac{1}{\alpha_m}$.

Implicit Boosting: While the loss is a function of a weight vector $\mathbf{w}$, we can use the chain rule to obtain the gradient of the loss with respect to the weight update $\boldsymbol{\delta}$, i.e. $\nabla_{\boldsymbol{\delta}} L = \alpha_m \nabla_{\mathbf{w}} L$. Then, initializing $\boldsymbol{\delta}$ to some appropriate $\boldsymbol{\delta}_{\text{ini}}$, the malicious agent can directly minimize with respect to $\boldsymbol{\delta}$.

Results: In the attack with *explicit boosting*, the malicious agent runs $E_m = 5$ steps of the Adam optimizer (Kingma & Ba, 2015) to obtain $\tilde{\boldsymbol{\delta}}_m^t$, and then boosts it by $\frac{1}{\alpha_m} = k$. The results for the case with $k = 10$ are shown in the plot on the *left* in Figure 1a. The attack is clearly successful at causing the global model to classify the chosen example in the target class. In fact, after $t = 3$, the global model is highly confident in its (incorrect) prediction. The baseline attack using *implicit boosting* (Figure 2) is much less successful than the explicit boosting baseline, with the adversarial objective only being achieved in 4 of 10 iterations. Further, it is computationally more expensive, taking an average of 2000 steps to converge at each time step. We focus on explicit boosting attacks for the remainder of the paper and analyze implicit boosting in more detail in the Appendix.

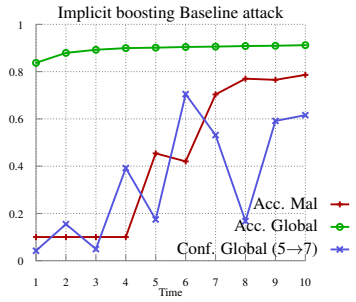


Figure 2: Implicit boosting attack metrics

3.2.1 Measuring attack stealth at server

While the baseline attack is successful at meeting the malicious agent's objective, there are detection methods the server can employ in order to detect if an agent's update is malicious. We now discuss a few possible methods and their implication for the baseline attack. We note that none of these methods are part of the standard federated learning algorithm.

Accuracy checking: When any agent sends a weight update to the server, it can check the validation accuracy of $\mathbf{w}_i^t = \mathbf{w}_G^{t-1} + \boldsymbol{\delta}_i^t$, the model obtained by adding that update to the current state of the global model. If the resulting model has a validation accuracy much lower than that of the other agents, the server may be able to detect that model as coming from a malicious agent. This would be particularly effective in the case where the agents have i.i.d. data. In Figure 1a, the *left* plot shows

the accuracy of the malicious model on the validation data (*Acc. Mal*) at each iteration. This is much lower than the accuracy of the global model (*Acc. Global*) and is no better than random for the first few iterations.

Weight update statistics: There are both qualitative and quantitative methods the server can apply in order to detect weight updates which are malicious, or at the least, different from a majority of the other agents. We investigate the effectiveness of two such methods. The first, qualitative method, is the visualization of weight update distributions for each agent. Since the adversarial objective function is different from the training loss objective used by all the benign agents, we expect the distribution of weight updates to be very different. This is borne out by the weight update distribution observed for the baseline attack in Figure 1b. Compared to the weight update from a benign agent, the update from the malicious agent is much sparser and has a smaller range. This difference is more pronounced for later time steps.

The second, quantitative method uses the spread of pairwise L_p distances between weight update vectors to identify outliers. At each time step, the server computes the pairwise distances between all the weight updates it receives, and flags those weight updates which are either much closer or much farther away than the others. In Figure 4, the spread of L_2 distances between all benign updates and between the malicious update and the benign updates is plotted. For the baseline attack, both the minimum and maximum distance away from any of the benign updates keeps decreasing over time steps, while it remains relatively constant for the other agents. This can enable detection of the malicious agent.

3.3 Attack with simultaneous training

To bypass the two detection methods discussed in the previous section, the malicious agent can try to simultaneously optimize over the adversarial objective and training loss for its local data shard $\mathcal{D}_m$. The resulting objective function is then $\operatorname{argmin}_{\boldsymbol{\delta}_m^t} L(\{\mathbf{x}_i, \tau_i\}_{i=1}^r, \hat{\mathbf{w}}_G^t) + \kappa L(\mathcal{D}_m, \mathbf{w}_m^t)$. Note that for the training loss, the optimization is just performed with respect to $\mathbf{w}_m^t$, as a benign agent would do. When doing explicit boosting, $\hat{\mathbf{w}}_G^t$ is replaced by $\mathbf{w}_m^t$ as well, and the initial weight update $\tilde{\boldsymbol{\delta}}_m^t$ is boosted by λ before being sent to the server. This is the only attack strategy explored in concurrent and independent work by Bagdasaryan et al. (2018).

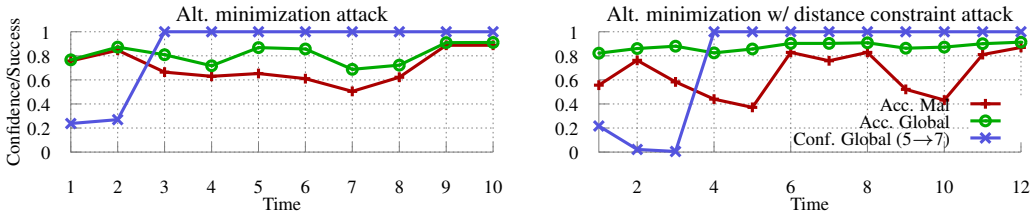
Results: In practice, we optimize over batches of $\mathcal{D}_m$ and concatenate each batch with the single instance $\{\mathbf{x}, \tau\}$ to be misclassified, ensuring that the adversarial objective is satisfied. In fact, as seen in Figure 1 in the plot on the *right*, the adversarial objective is satisfied with high confidence from the first time step $t = 1$.

Effect on stealth: Since the entire weight update corresponding to both adversarial and training objectives is boosted, the accuracy of $\mathbf{w}_m^t$ on the validation is low throughout the federated learning process. Thus, this attack can easily be detected using the accuracy checking method. Further, while the weight update distribution for this attack (Figure 1c) is visually similar to that of benign agents, its range differs, again enabling detection.

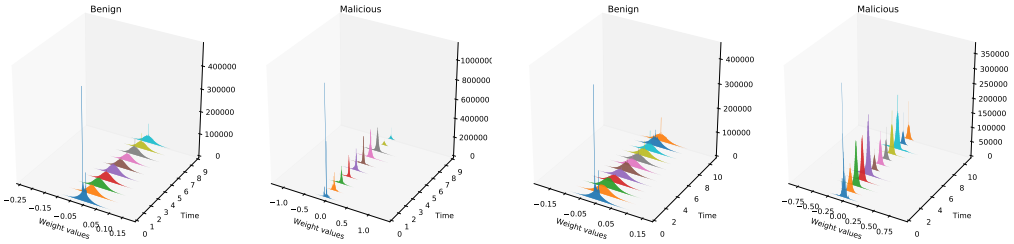
3.4 Alternating Minimization formulation

The malicious agent only needs to boost the part of the weight update that corresponds to the adversarial objective. In the baseline attack, in spite of this being the entire update, the resulting distribution is sparse and of low magnitude compared to a benign agent's updates. This indicates that the weights update needed to meet the adversarial objective could be *hidden* in an update that resembled that of a benign agent. However, as we saw in the previous section, boosting the entire weight update when the training loss is included leads to low validation accuracy. Further, the concatenation strategy does not allow for parts of the update corresponding to the two different objectives to be decoupled.

To overcome this, we propose an *alternating minimization* attack strategy which works as follows for iteration t. For each epoch i, the adversarial objective is first minimized starting from $\mathbf{w}_m^{i-1,t}$, giving an update vector $\tilde{\boldsymbol{\delta}}_m^{i,t}$. This is then boosted by a factor λ and added to $\mathbf{w}_m^{i-1,t}$. Finally, the training loss for that epoch is minimized starting from $\tilde{\mathbf{w}}_m^{i,t} = \mathbf{w}_m^{i-1,t} + \lambda \tilde{\boldsymbol{\delta}}_m^{i,t}$, providing the malicious weight



(a) Metrics of interest for alternating minimization attack without (**left**) and with distance constraints(**right**).



(b) Alternating minimization attack weight updates (c) Alternating minimization attack with distance constraints weight updates

Figure 3: Metrics of interest and weight update distributions for the *alternating minimization* attack with and without distance constraints.

vector $\mathbf{w}_m^{i,t}$ for the next epoch. The malicious agent can run this alternating minimization until both the adversarial objective and training loss have sufficiently low values.

Results: In Figure 3a, the plot on the *left* shows the evolution of the metrics of interest over iterations. The alternating minimization attack is able to achieve its goals as the accuracy of the malicious model closely matches that of the global model even as the adversarial objective is met with high confidence for all time steps starting from $t = 3$.

Effect on stealth: This attack can bypass the accuracy checking method as the validation accuracy of the malicious model is close to that of the global model. Qualitatively, the distribution of the malicious weight update (Figure 3b) is much more similar to that of the benign weights as compared to the baseline attack. Further, in Figure 4, we can see that the spread in distances between the malicious updates and benign updates is close to that between benign agents close to convergence. Thus, this attack is stealthier than the baseline.

3.5 Constraining the weight update

To increase the attack stealth, the malicious agent can also add a distance-based constraint on $\tilde{\mathbf{w}}_m^{i,t}$, which is the intermediate weight vector generated in the alternating minimization strategy. There could be multiple local minima which lead to low training loss, but the malicious agent needs to send back a weight update that is as close as possible (in an appropriate distance metric) to the update they would have sent had they been benign. So, $\mathbf{w}_m^{i,t}$ is constrained with respect to $\mathbf{w}_{m,ben}^t$, obtained by minimizing the training loss over $\mathcal{D}_m$ starting from $\mathbf{w}_G^{t-1}$, i.e. with the malicious agent mimicking a benign one.

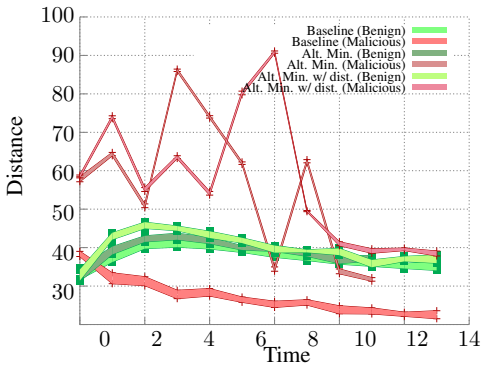


Figure 4: Minimum and maximum L_2 distances between weight updates

For our experiments, we use the L_2 norm as a constraint on $\mathbf{w}_m^{i,t}$, the weight vector obtained at the end of the training loss minimization phase, so

$\rho\|\mathbf{w}^t_{m,ben} - \mathbf{w}^{i,t}_m\|_2$ is added to the loss function. Constraints based on the *empirical distribution of weights* such as the Wasserstein or total variation distances may also be used.

Results and Effect on stealth: The adversarial objective is achieved at the global model with high confidence starting from time step $t = 4$. The weight update distribution for this attack (Figure 3c) is again similar to that of a benign agent. Further, in Figure 4, we can see that the distance spread for this attack closely tracks that of benign updates close to convergence.

4 Improving attack performance through estimation

In this section, we look at how the malicious agent can choose a better estimate for the effect of the other agents' updates at each time step that it is chosen. In the case when the malicious agent is not chosen at every time step, this estimation is made challenging by the fact that it may not have been chosen for many iterations.

4.1 Estimation setup

The malicious agent's goal is to choose an appropriate estimate for $\boldsymbol{\delta}^t_{[k]\setminus m} = \sum_{i\in[k]\setminus m} \alpha_i \boldsymbol{\delta}^t_i$ from Eq. 2. At a time step t when the malicious agent is chosen, the following information is available to them from the previous time steps they were chosen: i) Global parameter vectors $\mathbf{w}^{t_0}_G \ldots, \mathbf{w}^{t-1}_G$; ii) Malicious weight updates $\boldsymbol{\delta}^{t_0}_m \ldots, \boldsymbol{\delta}^t_m$; and iii) Local training data shard $\mathcal{D}_m$, where t_0 is the first time step at which the malicious agent is chosen. Given this information, the malicious agent computes an estimate $\hat{\boldsymbol{\delta}}^t_{[k]\setminus m}$ which it can use to correct for the effect of other agents in two ways:

Post-optimization correction: In this method, once the malicious agent computes its weight update $\boldsymbol{\delta}^t_m$, it subtracts $\lambda\hat{\boldsymbol{\delta}}^t_{[k]\setminus m}$ from it before sending it to the server. If $\hat{\boldsymbol{\delta}}^t_{[k]\setminus m} = \boldsymbol{\delta}^t_{[k]\setminus m}$ and $\lambda = \frac{1}{\alpha_m}$, this will negate the effects of the other agents.

Pre-optimization correction: Here, the malicious agent assumes that $\hat{\mathbf{w}}^t_G = \mathbf{w}^{t-1}_G + \hat{\boldsymbol{\delta}}^t_{[k]\setminus m} + \alpha_m \boldsymbol{\delta}^{T+1}_m$. In other words, the malicious agent optimizes for $\boldsymbol{\delta}^t_m$ assuming it has an accurate estimate of the other agents' updates. For attacks which use explicit boosting, this involves starting from $\mathbf{w}^{t-1}_G + \hat{\boldsymbol{\delta}}^t_{[k]\setminus m}$ instead of just $\mathbf{w}^{t-1}_G$.

4.2 Estimation strategies and results

When the malicious agent is chosen at time step t [3], information regarding the probable updates from the other agents can be obtained from the previous time steps at which the malicious agent was chosen.

Previous step estimate: In this method, the malicious agent's estimate $\hat{\boldsymbol{\delta}}^t_{[k]\setminus m}$ assumes that the other agents' cumulative updates were the same at each step since t' (the last time step at which at the malicious agent was chosen), i.e. $\hat{\boldsymbol{\delta}}^t_{[k]\setminus m} = \frac{\mathbf{w}^t_G - \mathbf{w}^{t'}_G - \boldsymbol{\delta}^{t'}_m}{t-t'}$. In the case when the malicious agent is chosen at every time step, this reduces to $\hat{\boldsymbol{\delta}}^t_{[k]\setminus m} = \boldsymbol{\delta}^{t-1}_{[k]\setminus m}$.

Results: Attacks using previous step estimation are more effective at achieving the adversarial objective in both the baseline and alternating minimization case. In Figure 5, the global model misclassfies the desired sample starting from time step $t = 2$ for the alternating minimization attack, compared to $t = 3$ in the case without estimation. Further, the misclassification occurs with a higher confidence for the baseline attack at $t = 2$.

5 Interpreting poisoned models

Neural networks are often treated as black boxes with little transparency into their internal representation or understanding of the underlying basis for their decisions. Interpretability techniques

[3] If they are chosen at $t = 0$ or t is the first time they are chosen, there is no information available regarding the other agents' updates

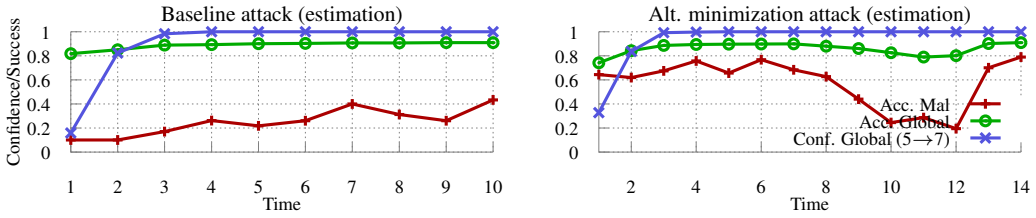


Figure 5: Metrics of interest for the baseline and alternating minimization attacks with explicit boosting and previous step estimation.

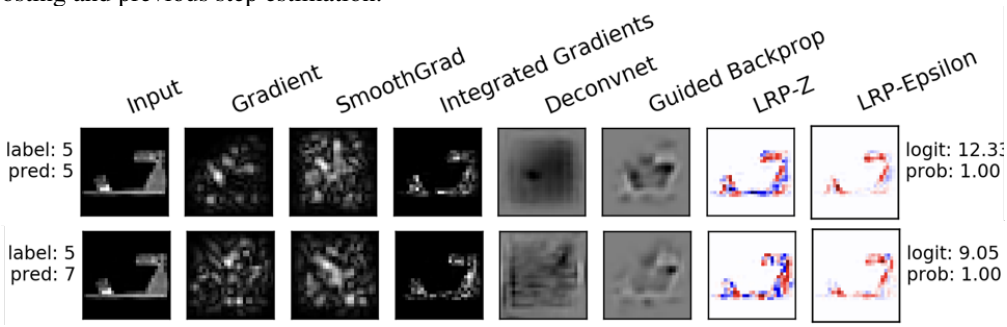


Figure 6: Interpretation of benign ($5 \rightarrow 5$) and malicious ($5 \rightarrow 7$) model decisions via visualization of feature relevance and representations for a randomly chosen auxiliary data sample.

are designed to alleviate these problems by analyzing various aspects of the network. These include (i) identifying the relevant features in the input pixel space for a particular decision via Layerwise Relevance Propagation (LRP) techniques (Montavon et al. (2015)); (ii) visualizing the association between neuron activations and image features (Guided Backprop (Springenberg et al. (2014)), DeConvNet (Zeiler & Fergus (2014))); (iii) using gradients for attributing prediction scores to input features (e.g., Integrated Gradients (Sundararajan et al. (2017)), or generating sensitivity and saliency maps (SmoothGrad (Smilkov et al. (2017)), Gradient Saliency Maps (Simonyan et al. (2013))) and so on. The semantic relevance of the generated visualization, relative to the input, is then used to explain the model decision.

These interpretability techniques, in many ways, provide insights into the internal feature representations and working of a neural network. Therefore, we used a suite of these techniques to try and discriminate between the behavior of a benign global model and one that has been trained to satisfy the adversarial objective of misclassifying a single example. Figure 6 compares the output of the various techniques for both the benign and malicious models on a random auxiliary data sample. Targeted perturbation of the model parameters coupled with tightly bounded noise ensures that the internal representations, and relevant input features used by the two models, for the same input, are almost visually imperceptible. This further reinforces the stealth achieved by our poisoning strategies against various interpretability-based detection techniques.

6 Discussion

In this paper, we have started an exploration of the vulnerability of multi-party machine learning algorithms such as federated learning to *model poisoning* adversaries, who can take advantage of the very privacy these models are designed to provide. In future work, we plan to explore more sophisticated detection strategies at the server, which can provide guarantees against the type of attacker we have considered here. In particular, notions of distances between weight distributions are promising defensive tools. Our attacks in this paper demonstrate that federated learning in its basic form is very vulnerable to model poisoning adversaries. While detection mechanisms can make these attacks more challenging, these can be overcome, demonstrating that multi-party machine learning algorithms robust to attackers of the type considered here must be developed.

REFERENCES

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

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 2017.

Lingjiao Chen, Hongyi Wang, Zachary B. Charles, and Dimitris S. Papailiopoulos. DRACO: byzantine-resilient distributed training via redundant gradients. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, 2018.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017a.

Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(2), 2017b.

Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. *arXiv preprint arXiv:1804.00308*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

Grégoire Montavon, Sebastian Bach, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *arXiv preprint arXiv:1512.02479*, 2015.

Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision, ECCV 2014 - 13th European Conference, Proceedings*, 2014.

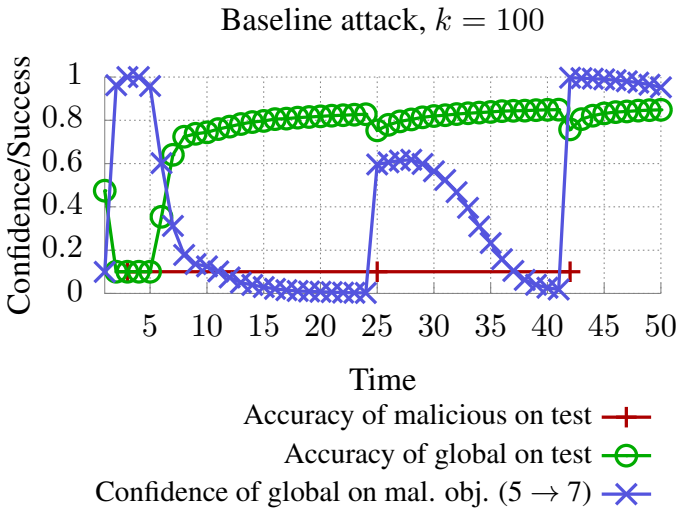


Figure 7: Metrics of interest for the baseline attack with $k = 100$ agents.

A RANDOMIZED SELECTION OF AGENTS

In Figure 7, we can see that the baseline attack is effective at introducing a backdoor even when the malicious agent is not selected in every time step.