Analyzing Federated Learning through an Adversarial Lens

Arjun Nitin Bhagoji*[1], Supriyo Chakraborty[2], Prateek Mittal[1], and Seraphin Calo[2]

[1]Department of Electrical Engineering, Princeton University
[2]I.B.M T.J. Watson Research Center

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 model parameter updates, for iterative aggregation at the server. In this work, we explore the threat of *model poisoning* attacks on federated learning initiated by a single, non-colluding malicious agent where the adversarial objective is to cause the model to mis-classify a set of chosen inputs with high confidence. We explore a number of strategies to carry out this attack, starting with simple *boosting* of the malicious agent's update to overcome the effects of other agents' updates. To increase attack stealth, we propose an alternating minimization strategy, which alternately optimizes for the training loss and the adversarial objective. We follow up by using parameter estimation for the benign agents' updates to improve on attack success. Finally, we use a suite of interpretability techniques to generate visual explanations of model decisions for both benign and malicious models, and show that the explanations are nearly visually indistinguishable. Our results indicate that even a highly constrained adversary can carry out model poisoning attacks while simultaneously maintaining stealth, thus highlighting the vulnerability of the federated learning setting and the need to develop effective defense strategies.

1 Introduction

Federated learning [16] 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 used is usually weighted averaging.

In this work, we exploit this lack of transparency in the agent updates, and explore the possibility of an adversary controlling a small number of malicious agents (usually just 1) performing a *model poisoning attack*. The adversary's objective is to cause the jointly trained global model to misclassify a set of chosen inputs with high confidence, i.e., it seeks to poison the global model in a targeted manner. Since the attack is targeted, the adversary also attempts to ensure that the global model converges to a point with good performance on the test or validation data We note that these inputs are not modified to induce misclassification as in the phenomenon of adversarial examples [5, 26]. Rather, their misclassification is a product of the adversarial manipulations of the training process. We focus on an adversary which directly performs model poisoning instead of data poisoning [3, 21, 17, 27, 17, 13, 7, 11] as the agents' data is never shared with the server. In fact, model poisoning subsumes dirty-label data poisoning in the federated learning setting (see Section 6.1 for a detailed quantitative comparison).

Model poisoning also has a connection to a line of work on defending against Byzantine adversaries which consider a threat model where the malicious agents can send arbitrary gradient updates [4, 8, 18,

*Contact: abhagoji@princeton.edu; Work done while at I.B.M. T.J. Watson Research Center

6, 29] to the server. However, the adversarial goal in these cases is to ensure a distributed implementation of the Stochastic Gradient Descent (SGD) algorithm converges to 'sub-optimal to utterly ineffective models'[18] while the aim of the defenses is to ensure convergence. On the other hand, we consider adversaries aiming to only cause targeted poisoning. In fact, we show that targeted model poisoning is effective even with the use of Byzantine resilient aggregation mechanisms in Section 4. Concurrent and independent work [2] considers both single and multiple agents performing poisoning via model replacement at convergence time. In contrast, our goal is to induce targeted misclassification in the global model even when it is far from convergence while maintaining its accuracy for most tasks.

1.1 Contributions

We design attacks on federated learning that ensure targeted poisoning of the global model while ensuring convergence. Our realistic threat model considers adversaries which only control a small number of malicious agents (usually 1) and have no visibility into the updates that will be provided by the other agents. All of our experiments are on deep neural networks trained on the Fashion-MNIST [28] and Adult Census[1] datasets.

Targeted model poisoning: In each round, the malicious agent generates its update by optimizing for a malicious objective designed to cause targeted misclassification. However, the presence of a multitude of other agents which are simultaneously providing updates makes this challenging. We thus use *explicit boosting* of the malicious agent's which is designed to negate the combined effect of the benign agents. Our evaluation demonstrates that this attack enables an adversary *controlling a single malicious agent* to achieve targeted misclassification at the global model with 100% confidence while ensuring convergence of the global model for deep neural networks trained on both datasets.

Stealthy model poisoning: We introduce notions of stealth for the adversary based on accuracy checking on the test/validation data and weight update statistics and empirically shoe that targeted model poisoning with explicit boosting can be detected in all rounds with the use of these stealth metrics. Accordingly, we modify the malicious objective to account for these stealth metrics to carry out stealthy model poisoning which allows the malicious weight update to avoid detection for a majority of the rounds. Finally, we propose an *alternating minimization* formulation that accounts for both model poisoning and stealth, and enables the malicious weight update to avoid detection in almost all rounds.

Attacking Byzantine-resilient aggregation: We investigate the possibility of model poisoning when the server uses Byzantine-resilient aggregation mechanisms such as Krum [4] and coordinate-wise median [29] instead of weighted averaging. We show that targeted model poisoning of deep neural networks with high confidence is effective even with the use of these aggregation mechanisms.

Connections to data poisoning and interpretability: We show that standard dirty-label data poisoning attacks [7] are not effective in the federated learning setting, even when the number of incorrectly labeled examples is on the order of the local training data held by each agent. Finally, we use a suite of interpretability techniques to generate visual explanations of the decisions made by a global model with and without a targeted backdoor. Interestingly, we observe that the explanations are nearly visually indistinguishable, exposing the fragility of these techniques.

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 server is attempting to train a classifier f with global

[1] https://archive.ics.uci.edu/ml/datasets/adult

parameter vector $\mathbf{w}_G \in \mathbb{R}^n$, where n is the dimensionality of the parameter space. This parameter vector is obtained by distributed training and aggregation over the K agents with the aim of generalizing well over $\mathcal{D}_{\text{test}}$, the test data. Federated learning can handle btoh i.i.d. and non-i.i.d partitioning of training data.

At each time step t, a random subset of k agents is chosen for synchronous aggregation [16]. Every agent $i \in [k]$, *minimizes* [2] *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, any aggregation mechanism can be used. In Section 3, we use weighted averaging based aggregation for our experiments: $\mathbf{w}_G^{t+1} = \mathbf{w}_G^t + \sum_{i\in[k]} \alpha_i \boldsymbol{\delta}_i^{t+1}$, where $\frac{l_i}{l} = \alpha_i$ and $\sum_i \alpha_i = 1$. In Section 4, we study the effect of our attacks on the Byzantine-resilient aggregation mechanisms 'Krum' [4] and coordinate-wise median [29].

2.2 Threat Model: Model Poisoning

Traditional poisoning attacks 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.

Attack Model: We make the following assumptions regarding the adversary: (i) they control exactly one non-colluding, malicious agent with index m (limited effect of malicious updates on the global model); (ii) the data is distributed among the agents in an i.i.d fashion (making it easier to discriminate between benign and possible malicious updates and harder to achieve attack stealth); (iii) 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. Our aim is to *explore the possibility of a successful model poisoning attack even for a highly constrained adversary*.

Adversarial Goals: The adversary's goal is to *ensure the targeted misclassification of the auxiliary data* by the classifier learned at the server. The auxiliary data consists of 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

$$\mathcal{A}(\mathcal{D}_m \cup \mathcal{D}_{\text{aux}}, \mathbf{w}_G^t) = \max_{\mathbf{w}_G^t} \sum_{i=1}^{r} \mathbb{1}[f(\mathbf{x}_i; \mathbf{w}_G^t) = \tau_i]. \tag{1}$$

We note that in contrast to previous threat models considered for Byzantine-resilient learning, the adversary's aim is not to prevent convergence of the global model [29] or to cause it to converge to a bad minimum [18]. Thus, any attack strategy used by the adversary must *ensure that the global model converges to a point with good performance on the test set*. Going beyond the standard federated learning setting, it is plausible that the server may implement measures to detect aberrant models. To bypass such measures, the adversary must also *conform to notions of stealth* that we define and justify next.

2.3 Stealth metrics

Given an update from an agent, there are two critical properties that the server can check. First, the server can verify whether the update, in isolation, would improve or worsen the global model's performance on a validation set. Second, the server can check if that update is very different statistically from other updates. We note that neither of these properties is checked as a part of standard federated learning but we use these to raise the bar for a successful attack.

Accuracy checking: The server checks the validation accuracy of $\mathbf{w}_i^t = \mathbf{w}_G^{t-1} + \boldsymbol{\delta}_i^t$, the model obtained by adding the update from agent i to the current state of the global model. If the resulting model has a validation accuracy much lower than that of the model obtained by aggregating all the other updates, $\mathbf{w}_{G\setminus i}^t = \mathbf{w}_G^{t-1} + \sum_i \boldsymbol{\delta}_i^t$, the server can flag the update as being anomalous. For the malicious agent, this implies that it must satisfy the following in order to be chosen at time step t:

$$\sum_{\{\mathbf{x}_j, y_j\} \in \mathcal{D}_{\text{test}}} \mathbb{1}[f(\mathbf{x}_j; \mathbf{w}_i^t) = y_i] - \mathbb{1}[f(\mathbf{x}_j; \mathbf{w}_{G\setminus i}^t) = y_i] < \gamma_t, \tag{2}$$

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

where γ_t is a threshold the server defines to reject updates. This threshold determines how much performance variation the server can tolerate and can be varied over time. A large threshold will be less effective at identifying anomalous updates but an overly small one could identify benign updates as anomalous, due to natural variation in the data and training process.

Weight update statistics: The range of pairwise distances between a particular update and the rest provides an indication of how different that update is from the rest when using an appropriate distance metric $d(\cdot,\cdot)$. In previous work, pairwise distances were used to define 'Krum' [4] but as we show in Section 4, its reliance on absolute, instead of relative distance values, makes it vulnerable to our attacks. Thus, we rely on the full range which can be computed for all agent updates and for an agent to be flagged as anomalous, their range of distances must differ from the others by a server defined, time-dependent threshold κ_t. In particular, for the malicious agent, we compute the range as $R_m = [\min_{i\in[k]\setminus m} d(\delta_m^t, \delta_i^t), \max_{i\in[k]\setminus m} d(\delta_m^t, \delta_i^t)]$. Let $R^l_{\min,[k\setminus m]}$ and $R^u_{\max,[k\setminus m]}$ be the minimum lower bound and maximum upper bound of the distance ranges for the other benign agents among themselves. Then, for the malicious agent to not be flagged as anomalous, we need that

$$\max\{|R^u_m - R^l_{\min,[k\setminus m]}|, |R^l_m - R^u_{\max,[k\setminus m]}|\} < \kappa_t. \tag{3}$$

This condition ensures that the range of distances for the malicious agent and any other agent is not too different from that for any two benign agents, and also controls the length of R_m. We find that it is also instructive to compare the histogram of weight updates for benign and malicious agents, as these can be very different depending on the attack strategy used. These provide a useful qualitative notion of stealth, which can be used to understand attack behavior.

2.4 Experimental setup

We evaluate our attack strategies using two qualitatively different datasets. The first is an image dataset, Fashion-MNIST [28] which serves as a drop-in replacement for the commonly used MNIST dataset [14], which is not representative of modern computer vision tasks. It consists of 28×28 grayscale images of clothing and footwear items and has 10 output classes. The training set contains 60,000 data samples while the test/validation set has 10,000 samples. For this dataset, we use a 3-layer Convolutional Neural Network (CNN) with dropout as the model architecture. With centralized training, this model achieves 91.7% accuracy on the test set.

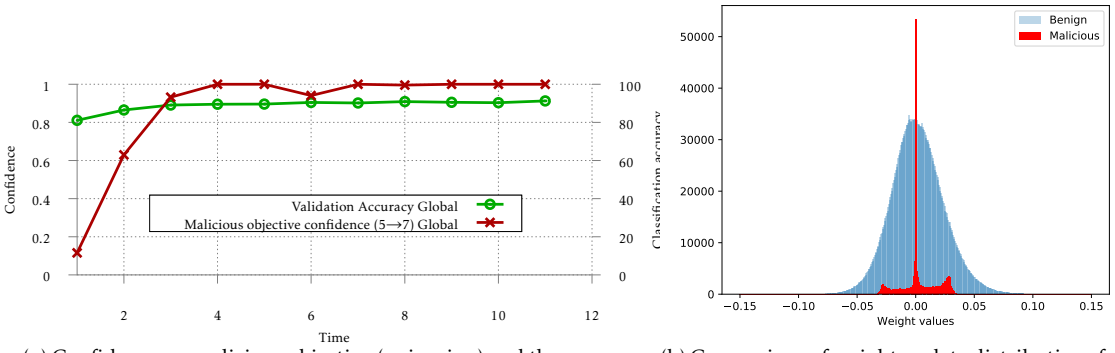
The second dataset is the UCI Adult Census dataset[3] which has over 40,000 samples containing information about adults from the 1994 US Census. The classification problem is to determine if the income for a particular individual is greater (class '0') or less (class '1') than $\$50,000$ a year. For this dataset, we use a fully connected neural network achieving 84.8% accuracy on the test set [9] for the model architecture.

For both datasets, 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. We run federated learning till a pre-specified test accuracy (91% for Fashion MNIST and 84% for the Adult Census data) is reached or the maximum number of time steps have elapsed (40 for $k = 10$ and 50 for $k = 100$). In Section 3, for illustrative purposes, we mostly consider the case where the malicious agent aims to mis-classify a single example in a desired target class ($r = 1$). For the Fashion-MNIST dataset, the example belongs to class '5' (sandal) with the aim of misclassifying it in class '7' (sneaker) and for the Adult dataset it belongs to class '0' with the aim of misclassifying it in class '1'. We also consider the case with $r = 10$ (Appendix)

3 Strategies for Model Poisoning attacks

In this section, we use the adversarial goals laid out in the previous section to formulate the adversarial optimization problem. We then show how explicit boosting can achieve targeted model poisoning. We further explore attack strategies that add stealth and improve convergence.

[3] https://archive.ics.uci.edu/ml/datasets/adult



(a) Confidence on malicious objective (poisoning) and the accuracy on validation data (targeted) on the global model

(b) Comparison of weight update distributions for benign and malicious agents

Figure 1: **Targeted model poisoning attack for CNN on Fashion MNIST data.** The total number of agents is $K = 10$, including the malicious agents. All agents train their local models for 5 epochs with the appropriate objective.

3.1 Adversarial optimization setup

From Eq. 1, two challenges for the adversary are immediately clear. First, the objective represents a difficult combinatorial optimization problem so we relax Eq. 1 in terms of the cross-entropy loss for which automatic differentiation can be used. Second, the adversary does not have access to the global parameter vector $\mathbf{w}_G^t$ for the current iteration and can only influence it though the weight update $\boldsymbol{\delta}_m^t$ it provides to the server S. So, it performs the optimization over $\hat{\mathbf{w}}_G^t$, which is an *estimate* of the value of $\mathbf{w}_G^t$ based on all the information $\mathcal{I}_m^t$ available to the adversary. The objective function for the adversary to achieve targeted model poisoning on the t^{th} iteration is

$$\begin{aligned} \underset{\boldsymbol{\delta}_m^t}{\text{argmin}} \quad & L(\{\mathbf{x}_i, \tau_i\}_{i=1}^r, \hat{\mathbf{w}}_G^t), \\ \text{s.t.} \quad & \hat{\mathbf{w}}_G^t = g(\mathcal{I}_m^t), \end{aligned} \tag{4}$$

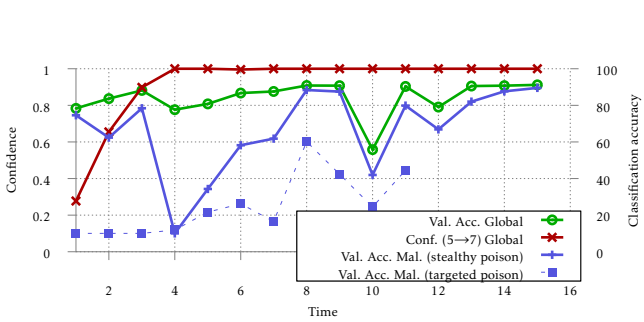
where $g(\cdot)$ is an estimator. For the rest of this section, we use the estimate $\hat{\mathbf{w}}_G^t = \mathbf{w}_G^{t-1} + \alpha_m \boldsymbol{\delta}_m^t$, implying that the malicious agent ignores the updates from the other agents but accounts for scaling at aggregation. This assumption is enough to ensure the attack works in practice.

3.2 Targeted model poisoning for standard federated learning

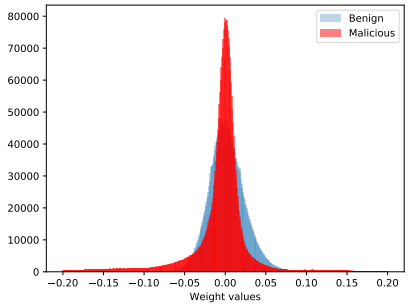
The adversary can directly optimize the adversarial objective $L(\{\mathbf{x}_i, \tau_i\}_{i=1}^r, \hat{\mathbf{w}}_G^t)$ with $\hat{\mathbf{w}}_G^t = \mathbf{w}_G^{t-1} + \alpha_m \boldsymbol{\delta}_m^t$. However, this setup implies that the optimizer has to account for the scaling factor α_m *implicitly*. In practice, we find that when using a gradient-based optimizer such as SGD, *explicit boosting* is much more effective. The rest of the section focuses on explicit boosting and an analysis of implicit boosting is deferred to Section A of the Appendix.

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 with $\hat{\mathbf{w}}_G^t = \mathbf{w}_G^{t-1} + \alpha_m \boldsymbol{\delta}_m^t$ and $\lambda = \frac{1}{\alpha_m}$, then $\hat{\mathbf{w}}_G^t = \mathbf{w}_m^t$, implying that if the estimation was exact, the global weight vector should now satisfy the malicious agent's objective.

Results: In the attack with *explicit boosting*, the malicious agent runs $E_m = 5$ steps of the Adam optimizer [12] to obtain $\tilde{\boldsymbol{\delta}}_m^t$, and then boosts it by $\frac{1}{\alpha_m} = K$. The results for the case with $K = 10$ for the Fashion MNIST



(a) Confidence on malicious objective and accuracy on validation data for $\mathbf{w}_G^t$. Stealth with respect to accuracy checking is also shown for both the stealthy and targeted model poisoning attacks.



(b) Comparison of weight update distributions for benign and malicious agents

Figure 2: **Stealthy model poisoning for CNN on Fashion MNIST**. We use $\lambda = 10$ and $\rho = 1e^{-4}$ for the malicious agent's objective.

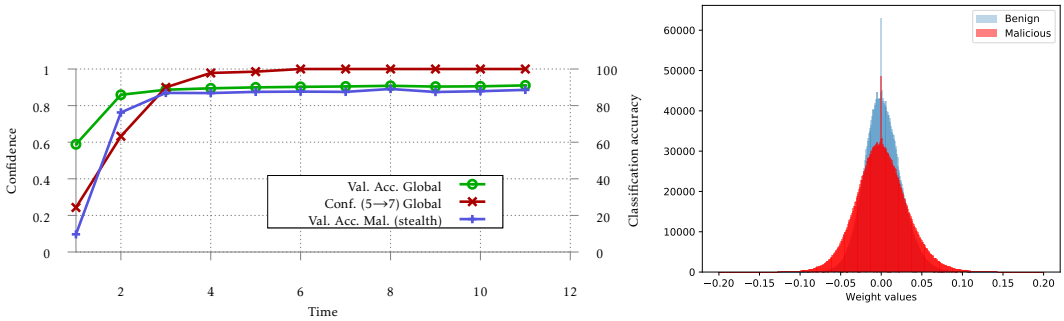
data are shown in Figure 3.1. The attack is clearly successful at causing the global model to classify the chosen example in the target class. In fact, after $t = 3$, the global model is highly confident in its (incorrect) prediction. Further, the global model converges with good performance on the validation set in spite of the targeted poisoning for 1 example. Results for the Adult Census dataset (Section B.1) demonstrate targeted model poisoning is possible across datasets and models. Thus, the explicit boosting attack is able to achieve targeted poisoning in the federated learning setting.

Performance on stealth metrics: While the targeted model poisoning attack using explicit boosting does not take stealth metrics into account, it is instructive to study properties of the model update it generates. Compared to the weight update from a benign agent, the update from the malicious agent is much sparser and has a smaller range (Figure 1b). 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 targeted model poisoning, 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. In Figure 2a the accuracy of the malicious model on the validation data (*Val. Acc. Mal (targeted poison)*) is shown, which is much lower than the global model's accuracy. Thus, both accuracy checking and weight update statistics based detection is possible for the targeted model poisoning attack.

3.3 Stealthy model poisoning

As discussed in Section 2.3, there are two properties which the server can use to detect anomalous updates: accuracy on validation data and weight update statistics. In order to maintain stealth with respect to both of these properties, the adversary can add loss terms corresponding to both of those metrics to the model poisoning objective function from Eq. 4 and improve targeted model poisoning. First, in order to improve the accuracy on validation data, the adversary adds the training loss over the malicious agent's local data shard $\mathcal{D}_m$ ($L(\mathcal{D}_m, \mathbf{w}_G^t)$) to the objective. Since the training data is i.i.d. with the validation data, this will ensure that the malicious agent's update is similar to that of a benign agent in terms of validation loss and will make it challenging for the server to flag the malicious update as anomalous.

Second, the adversary needs to ensure that its update is as close as possible to the benign agents' updates in the appropriate distance metric. For our experiments, we use the ℓ_p norm with $p = 2$. Since the adversary does not have access to the updates for the current time step t that are generated by the other agents, it constrains δ_m^t with respect to $\bar{\delta}_{\text{ben}}^{t-1} = \sum_{i \in [k] \setminus m} \alpha_i \delta_i^{t-1}$, which is the average update from all the other agents for the previous iteration, which the malicious agent has access to. Thus, the adversary adds $\rho \|\delta_m^t - \bar{\delta}_{\text{ben}}^{t-1}\|_2$ to its objective as well. We note that the addition of the training loss term is not sufficient to ensure that the malicious weight update is close to that of the benign agents since there could be multiple local minima



(a) Confidence on malicious objective and accuracy on validation data for $\mathbf{w}_G^t$. Stealth with respect to accuracy checking is also shown.

(b) Comparison of weight update distributions for benign and malicious agents

Figure 3: **Alternating minimization attack with distance constraints for CNN on Fashion MNIST data**. We use $\lambda = 10$ and $\rho = 1e^{-4}$. The number of epochs used by the malicious agent is $E_m = 10$ and it runs 10 steps of the stealth objective for every step of the malicious objective.

with similar loss values. Overall, the adversarial objective then becomes:

$$\underset{\delta_m^t}{\text{argmin}}\, \lambda L(\{\mathbf{x}_i, \tau_i\}_{i=1}^r, \hat{\mathbf{w}}_G^t) + L(\mathcal{D}_m, \mathbf{w}_m^t) + \rho \|\delta_m^t - \bar{\delta}_{\text{ben}}^{t-1}\|_2 \tag{5}$$

Note that for the training loss, the optimization is just performed with respect to $\mathbf{w}_m^t = \mathbf{w}_G^{t-1} + \delta_m^t$, as a benign agent would do. Using explicit boosting, $\hat{\mathbf{w}}_G^t$ is replaced by $\mathbf{w}_m^t$ as well so that only the portion of the loss corresponding to the malicious objective gets boosted by a factor λ.

Results and effect on stealth: From Figure 2a, it is clear that the stealthy model poisoning attack is able to cause targeted poisoning of the global model. We set the accuracy threshold γ_t to be 10% which implies that the malicious model is chosen for 10 iterations out of 15. This is in contrast to the targeted model poisoning attack which never has validation accuracy within 10% of the global model. Further, the weight update distribution for the stealthy poisoning attack (Figure 2b) is similar to that of a benign agent, owing to the additional terms in the loss function. Finally, in Figure 4, we see that the range of ℓ_2 distances for the malicious agent R_m is close, according to Eq. 3, to that between benign agents.

Concurrent work on model poisoning boosts the entire update (instead of just the malicious loss component as we do) when the global model is close to convergence in an attempt to perform model replacement [2] but this strategy is ineffective when the model has not converged.

3.4 Alternating minimization for improved model poisoning

While the stealthy model poisoning attack ensures targeted poisoning of the global model while maintaining stealth according to the two conditions required, it does not ensure that the malicious agent's update is chosen in every iteration. To achieve this, we propose an *alternating minimization attack strategy* which decouples the targeted objective from the stealth objectives, providing finer control over the relative effect of the two objectives. It 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 $\tilde{\delta}_m^{i,t}$. This is then boosted by a factor λ and added to $\mathbf{w}_m^{i-1,t}$. Finally, the stealth objective for that epoch is minimized starting from $\tilde{\mathbf{w}}_m^{i,t} = \mathbf{w}_m^{i-1,t} + \lambda\tilde{\delta}_m^{i,t}$, providing the malicious weight

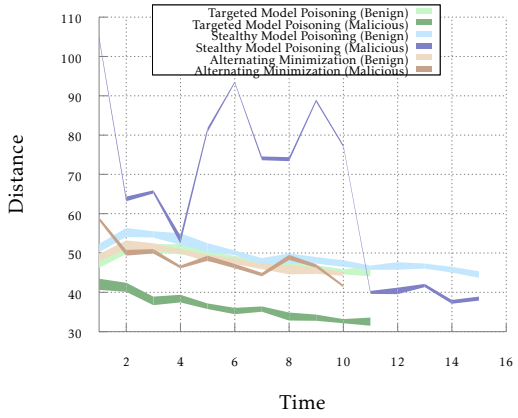


Figure 4: **Range of ℓ_2 distances between all benign agents and between the malicious agent and the benign agents.**

vector $\mathbf{w}_m^{i,t}$ for the next epoch. The malicious agent can run this alternating minimization until both the adversarial and stealth objectives have sufficiently low values. Further, the independent minimization allows for each objective to be optimized for a different number of steps, depending on which is more difficult in achieve. In particular, we find that optimizing the stealth objective for a larger number of steps each epoch compared to the malicious objective leads to better stealth performance while maintaining targeted poisoning.

Results and effect on stealth: The adversarial objective is achieved at the global model with high confidence starting from time step $t = 2$ and the global model converges to a point with good performance on the validation set. This attack can bypass the accuracy checking method as the accuracy on validation data of the malicious model is close to that of the global model.In Figure 4, we can see that the distance spread for this attack closely follows and even overlaps that of benign updates throughout, thus achieving complete stealth with respect to both properties.

4 Attacking Byzantine-resilient aggregation

There has been considerable recent work that has proposed gradient aggregation mechanisms for distributed learning that ensure convergence of the global model [4, 8, 18, 6, 29]. However, the aim of the Byzantine adversaries considered in this line of work is to ensure convergence to ineffective models, i.e. models with poor classification performance. The goal of the adversary we consider is targeted model poisoning, which implies convergence to an effective model on the test data. This difference in objectives leads to the lack of robustness of these Byzantine-resilient aggregation mechanisms against our attacks. We consider the aggregation mechanisms Krum [4] and coordinate-wise median [29] for our evaluation, both of which are provably Byzantine-resilient and converge under appropriate conditions on the loss function. Both aggregation mechanisms are also efficient. Note that in general, these conditions do not hold for neural networks so the guarantees are only empirical.

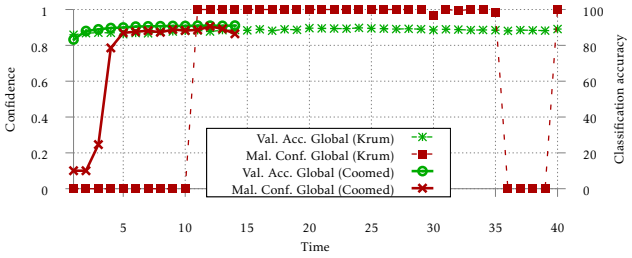


Figure 5: **Model poisoning attacks with Byzantine resilient aggregation mechanisms.** We use targeted model poisoning for coomed and alternating minimization for Krum.

4.1 Krum

Given n agents of which f are Byzantine, Krum requires that $n \geq 2f + 3$. At any time step t, updates $\left(\delta_1^t, \ldots, \delta_n^t\right)$ are received at the server. For each δ_i^t, the $n - f - 2$ closest (in terms of L_p norm) other updates are chosen to form a set C_i and their distances added up to give a score $S(\delta_i^t) = \sum_{\delta \in C_i} \|\delta_i^t - \delta\|$. Krum then chooses $\boldsymbol{\delta}_{\mathbf{krum}} = \delta_i^t$ with the lowest score to add to $\mathbf{w}_i^t$ to give $\mathbf{w}_i^{t+1} = \mathbf{w}_i^t + \boldsymbol{\delta}_{\mathbf{krum}}$. In Figure 5, we see the effect of the alternating minimization attack on Krum with a boosting factor of $\lambda = 2$ for a federated learning setup with 10 agents. Since there is no need to overcome the constant scaling factor α_m, the attack can use a much smaller boosting factor λ than the number of agents to ensure model poisoning. The malicious agent's update is chosen by Krum for 26 of 40 time steps which leads to the malicious objective being met. Further, the global model converges to a point with good performance as the malicious agent has added the training loss to its stealth objective. We note that with the use of targeted model poisoning, we can cause Krum to converge to a model with poor performance as well (see Appendix B.4).

4.2 Coordinate-wise median

Given the set of updates $\{\delta_i^t\}_{i=1}^k$ at time step t, the aggregate update is $\bar{\delta}^t := \text{coomed}\{\{\delta_i^t\}_{i=1}^k\}$, which is a vector with its j^{th} coordinate $\bar{\delta}^t(j) = \text{med}\{\delta_i^t(j)\}$, where med is the 1-dimensional median. Using targeted model poisoning with a boosting factor of $\lambda = 1$, i.e. no boosting, the malicious objective is met with confidence close to 0.9 for 11 of 14 time steps (Figure 5). We note that in this case, unlike with Krum, there is convergence to an effective global model. We believe this occurs due to the fact that coordinate-wise median does not simply pick one of the updates to apply to the global model and does indeed use information from all the agents while computing the new update. Thus, *model poisoning attacks are effective against two completely different Byzantine-resilient aggregation mechanisms*.

5 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.

5.1 Estimation setup

The malicious agent's goal is to choose an appropriate estimate for $\delta_{[k]\setminus m}^t = \sum_{i\in[k]\setminus m} \alpha_i \delta_i^t$, i.e. for the effects of the other agents at time step t. When the malicious agent is chosen at time t, the following information is available to them from the previous time steps they were chosen: i) Global parameter vectors $\mathbf{w}_G^{t_0} \ldots, \mathbf{w}_G^{t-1}$; ii) Malicious weight updates $\delta_m^{t_0} \ldots, \delta_m^t$; 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 $\hat{\delta}_{[k]\setminus m}^t$ which it can use to correct for the effect of other agents in two ways:

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

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

5.2 Estimation strategies and results

When the malicious agent is chosen at time step t [4], information regarding the probable updates from the other agents can be obtained from the previous time steps at which the malicious agent was chosen.

5.2.1 Previous step estimate

In this method, the malicious agent's estimate $\hat{\delta}_{[k]\setminus m}^t$ 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. $\hat{\delta}_{[k]\setminus m}^t = \frac{\mathbf{w}_G^t - \mathbf{w}_G^{t'} - \delta_m^{t'}}{t-t'}$. In the case when the malicious agent is chosen at every time step, this reduces to $\hat{\delta}_{[k]\setminus m}^t = \delta_{[k]\setminus m}^{t-1}$. This estimate can be applied to both the pre- and post-optimization correction methods.

[4] 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

Attack	Targeted Model Poisoning		Alternating Minimization	
Estimation	None	Previous step	None	Previous step
$t = 2$	0.63	0.82	0.17	0.47
$t = 3$	0.93	0.98	0.34	0.89
$t = 4$	0.99	1.0	0.88	1.0

Table 1: Comparison of confidence of targeted misclassification with and without the use of previous step estimation for the targeted model poisoning and alternating minimization attacks.

5.2.2 Results

Attacks using previous step estimation with the pre-optimization correction are more effective at achieving the adversarial objective for both the targeted model poisoning and alternating minimization attacks. In Table 1, we can see that the global model misclassifies the desired sample with a higher confidence when using previous step estimation in the first few iterations. We found that using post-optimization correction was not effective, leading to both lower attack success and affecting global model convergence.

6 Discussion

6.1 Model poisoning vs. data poisoning

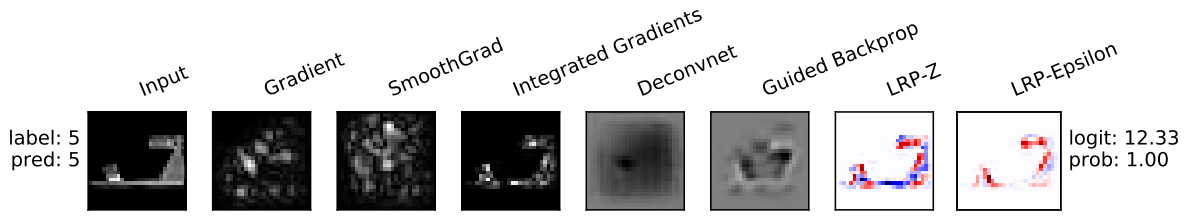
In this section, we elucidate the differences between model poisoning and data poisoning both qualitatively and quantitatively. Data poisoning attacks largely fall in two categories: clean-label [20, 13] and dirty-label [7, 10, 15]. Clean-label attacks assume that the adversary *cannot* change the label of any training data as there is a process by which data is certified as belonging to the correct class and the poisoning of data samples has to be imperceptible. On the other hand, to carry out dirty-label poisoning, the adversary just has to introduce a number of copies of the data sample it wishes to mis-classify with the desired target label into the training set since there is no requirement that a data sample belong to the correct class. Dirty-label data poisoning has been shown to achieve high-confidence targeted misclassification for deep neural networks with the addition of around 50 poisoned samples to the training data [7].

6.1.1 Dirty-label data poisoning in federated learning

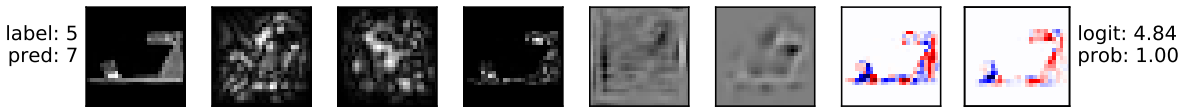
In our comparison with data poisoning, we use the dirty-label data poisoning framework for two reasons. First, federated learning operates under the assumption that data is never shared, only learned models. Thus, the adversary is not concerned with notions of imperceptibility for data certification. Second, clean-label data poisoning assumes access at train time to the global parameter vector, which is absent in the federated learning setting. Using the same experimental setup as before (CNN on Fashion MNIST data, 10 agents chosen every time step), we add copies of the sample that is to be misclassified to the training set of the malicious agent with the appropriate target label. We experiment with two settings. In the first, we add multiple copies of the same sample to the training set. In the second, we add a small amount of random uniform noise to each pixel [7] when generating copies. We observe that even when we add 1000 copies of the sample to the training set, the *data poisoning attack is completely ineffective at causing targeted poisoning in the global model*. This occurs due to the fact that malicious agent's update is scaled, which again underlies the importance of boosting while performing model poisoning. We note also that if the update generated using data poisoning is boosted, it affects the performance of the global model as the entire update is boosted, not just the malicious part. Thus, model poisoning attacks are much more effective than data poisoning in the federated learning setting.

6.2 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 are designed to



(a) Decision visualization for auxiliary data sample on a model with trained with 10 benign agents.



(b) Decision visualization for auxiliary data sample on a model trained with 9 benign agents and a single malicious agent.

Figure 6: **Decision visualizations for benign and malicious models for a CNN on the Fashion MNIST data.**

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 ([19]); (ii) visualizing the association between neuron activations and image features (Guided Backprop ([24]), DeConvNet ([30])); (iii) using gradients for attributing prediction scores to input features (e.g., Integrated Gradients ([25]), or generating sensitivity and saliency maps (SmoothGrad ([23]), Gradient Saliency Maps ([22])) 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 exposes the fragility of interpretability methods [1].

References

[1] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9525–9536, 2018.

[2] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[3] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1807–1814, 2012.

[4] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 2017.

[5] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[6] L. Chen, H. Wang, Z. B. Charles, and D. S. Papailiopoulos. DRACO: byzantine-resilient distributed training via redundant gradients. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, 2018.

[7] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[8] Y. Chen, L. Su, and J. Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proc. ACM Meas. Anal. Comput. Syst.*, 1(2), 2017.

[9] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[10] T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[11] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *IEEE Security and Privacy*, 2018.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[13] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang. Trojaning attack on neural networks. In *NDSS*, 2017.

[16] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[17] S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, 2015.

[18] E. M. E. Mhamdi, R. Guerraoui, and S. Rouault. The hidden vulnerability of distributed learning in byzantium. In *ICML*, 2018.

[19] G. Montavon, S. Bach, A. Binder, W. Samek, and K. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *arXiv preprint arXiv:1512.02479*, 2015.

[20] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. 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.

[21] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. Tygar. Stealthy poisoning attacks on pca-based anomaly detectors. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):73–74, 2009.

[22] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[23] D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[24] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[25] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[27] H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In *ICML*, 2015.

[28] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[29] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. *arXiv preprint arXiv:1803.01498*, 2018.

[30] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision, ECCV 2014 - 13th European Conference, Proceedings*, 2014.

A 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 δ, i.e. $\nabla_\delta L = \alpha_m \nabla_\mathbf{w} L$. Then, initializing δ to some appropriate δ_{ini}, the malicious agent can directly minimize with respect to δ. However, the baseline attack using *implicit boosting* (Figure 7) 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, which is about 4× longer than a benign agent. Since consistently delayed updates from the malicious agent might lead to it being dropped from the system in practice, we focused on explicit boosting attacks throughout.

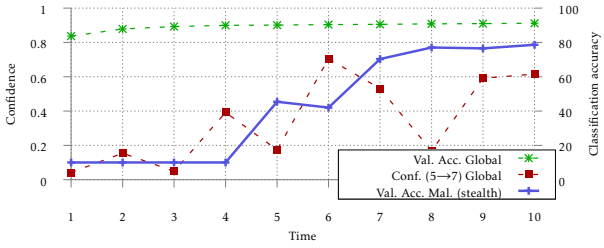


Figure 7: **Targeted model poisoning with implicit boosting.** The number of agents was $k = 10$ with a CNN on Fashion MNIST data.

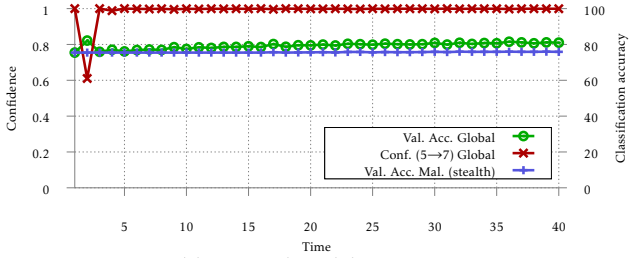
B Further results

B.1 Results on Adult Census dataset

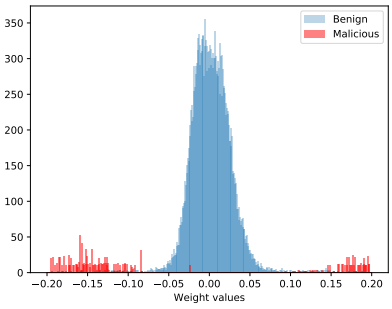
Results for the 3 different attack strategies on the Adult Census dataset (Figure 8) confirm the broad conclusions we derived from the Fashion MNIST data. The baseline attack is able to induce high confidence targeted misclassification for a random test example but affects performance on the benign objective, which drops from 84.8% in the benign case to just around 80%. The alternating minimization attack is able to ensure misclassification with a confidence of around 0.7 while maintaining 84% accuracy on the benign objective.

B.2 Multiple instance poisoning

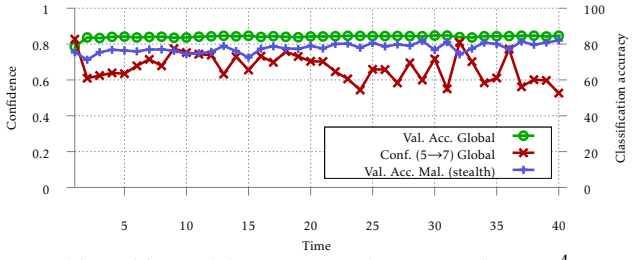
For completeness, we provide results for the case with $r = 10$, i.e. the case when the malicious agent wishes to classify 10 different examples in specific target classes. These results are given Figures 9a (targeted model poisoning) and 9b (Alternating minimization with stealth). While targeted model poisoning is able to induce targeted misclassification, it has an adverse impact on the global model's accuracy. This is countered by the alternating minimization attack, which ensures that the global model converges while still meeting the malicious objective.



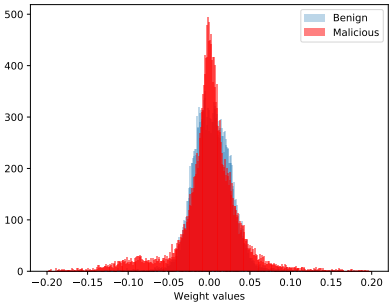
(a) Targeted model poisoning



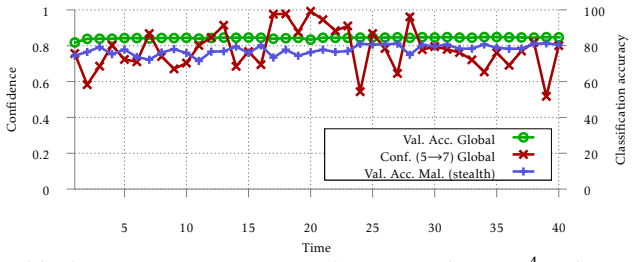
(b) Comparison of weight update distributions for targeted model poisoning



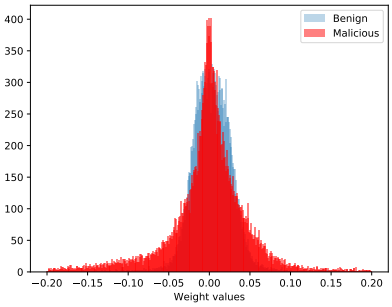
(c) Stealthy model poisoning with $\lambda = 20$ and $\rho = 1e^{-4}$



(d) Comparison of weight update distributions for stealthy model poisoning

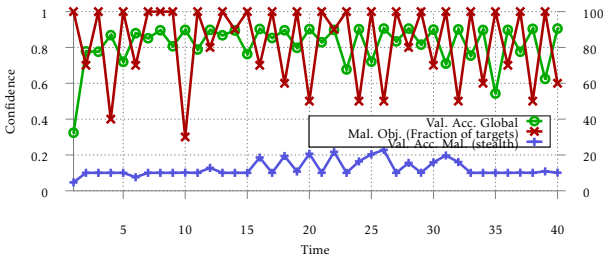


(e) Alternating minimization with $\lambda = 20$ and $\rho = 1e^{-4}$ and 10 epochs for the malicious agent

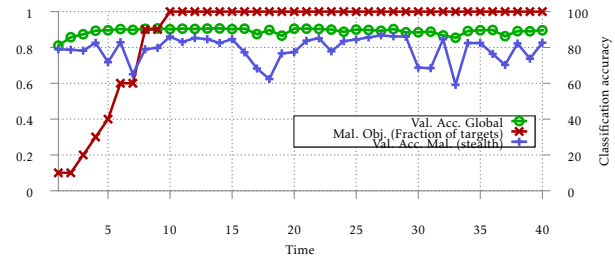


(f) Comparison of weight update distributions for alternating minimization

Figure 8: **Attacks on a fully connected neural network on the Census dataset.**



(a) Targeted model poisoning.



(b) Alternating minimization with 10 epochs for the malicious agent and 10 steps for the stealth objective for every step of the benign objective.

Figure 9: **Attacks with multiple targets ($r = 10$) for a CNN on the Fashion MNIST data.**

B.3 Randomized agent selection

When the number of agents increases to $k = 100$, the malicious agent is not selected in every step. Further, the size of $|\mathcal{D}_m|$ decreases, which makes the benign training step in the alternating minimization attack more challenging. The challenges posed in this setting are reflected in Figure 10a, where although targeted model poisoning is able to introduce a targeted backdoor, it is not present for every step as there are steps where only benign agents provide updates. Nevertheless, targeted model poisoning is effective overall, with the malicious objective achieved along with convergence of the global model at the end of training. The alternating minimization attack strategy with stealth (Figure 10b) is also able to introduce the backdoor, as well as increase the classification accuracy of the malicious model on test data. However, the improvement in performance is limited by the paucity of data for the malicious agent. It is an open question if data augmentation could help improve this accuracy.

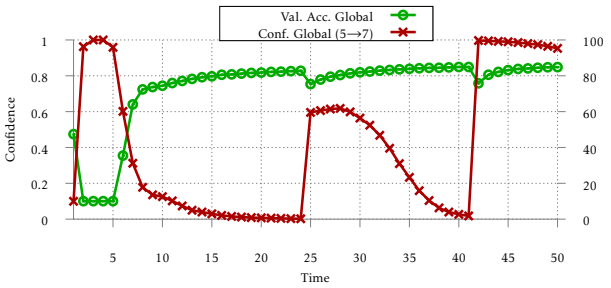
B.4 Bypassing Byzantine-resilient aggregation mechanisms

In Section 4, we presented the results of successful attacks on two different Byzantine resilient aggregation mechanisms: Krum [4] and coordinate-wise median (coomed) [29]. In this section, we present the results for targeted model poisoning when Krum is used (Figure 11a). The attack uses a boosting factor of $\lambda = 2$ with $k = 10$. Since there is no need to overcome the constant scaling factor α_m, the attacks can use a much smaller boosting factor λ to ensure the global model has the targeted backdoor. With the targeted model poisoning attack, the malicious agent's update is the one chosen by Krum for 34 of 40 time steps but this causes the validation accuracy on the global model to be extremely low. Thus, our attack causes Krum to converge to an ineffective model, in contrast to its stated claims of being Byzantine-resilient. However, our attack does not achieve its goal of ensuring that the global model converges to a point with good performance on the test set due to Krum selecting just a single agent at each time step.

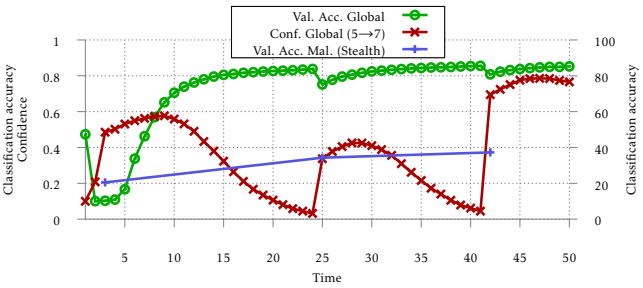
We also consider the effectiveness of the alternating minimization attack strategy when coomed is used for aggregation. While we have shown targeted model poisoning to be effective even when coomed is used, Figure 11b demonstrates that alternating minimization, which ensures that the local model learned at the malicious agent also has high validation accuracy, is not effective.

C Visualization of weight update distributions

Figure C shows the evolution of weight update distributions for the 4 different attack strategies on the CNN trained on the Faishon MNIST dataset. Time slices of this evolution were shown in the main text of the paper. The baseline and concatenated training attacks lead to weight update distributions that differ widely for benign and malicious agents. The alternating minimization attack without distance constraints reduces this qualitative difference somewhat but the closest weight update distributions are obtained with the alternating minimization attack with distance constraints.

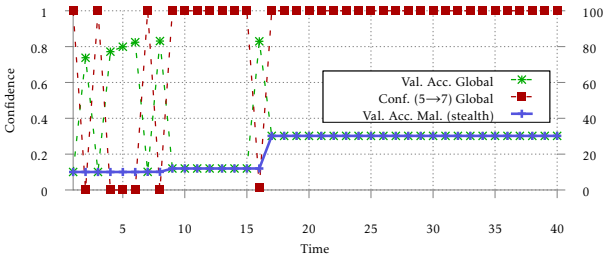


(a) Targeted model poisoning with $\lambda = 100$.

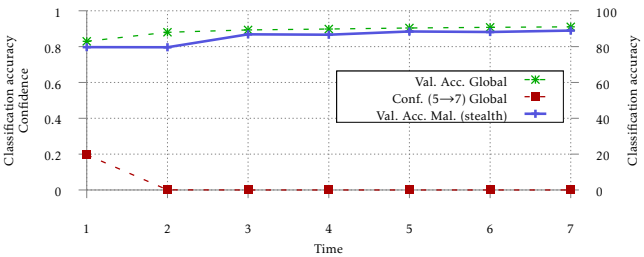


(b) Alternating minimization with $\lambda = 100$, 100 epochs for the malicious agent and 10 steps for the stealth objective for every step of the benign objective.

Figure 10: **Attacks on federated learning in a setting with $K = 100$ and a single malicious agent for a CNN on the Fashion MNIST data.**

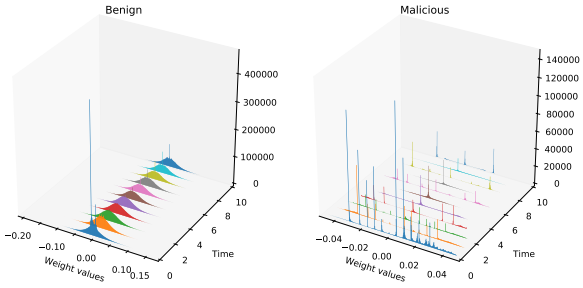


(a) Targeted model poisoning with $\lambda = 2$ against Krum.

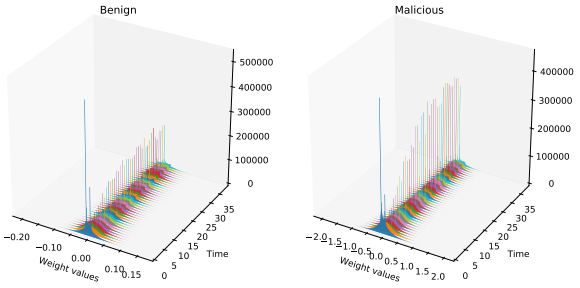


(b) Alternating minimization attack with $\lambda = 2$ against coomed.

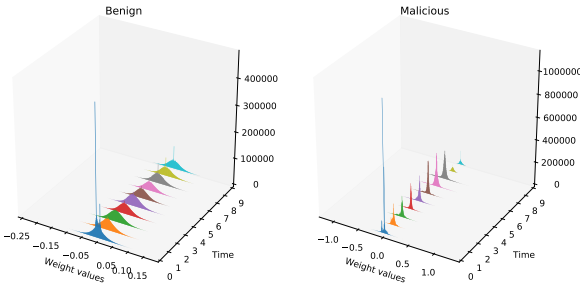
Figure 11: **Additional results for attacks on Byzantine-resilient aggregation mechanisms.**



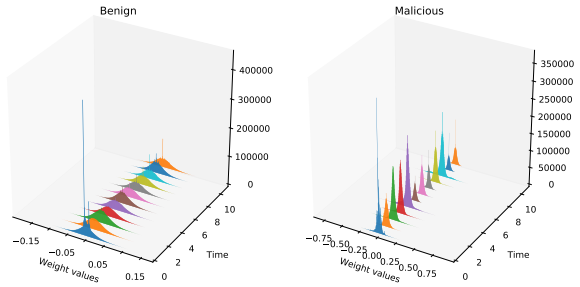
(a) Targeted model poisoning weight update distribution



(b) Stealthy model poisoning weight update distribution



(c) Alternating minimization (only loss stealth) weight update distribution



(d) Alternating minimization (both loss and distance stealth) distance constraints weight update distribution

Figure 12: Weight update distribution evolution over time for all attacks on a CNN for the Fashion MNIST dataset.

D Interpretability for benign inputs

We provide additional interpretability results for global models trained with and without the presence of a malicious agent on benign data in Figures 13 and 14 respectively. These show that the presence of the malicious agent using targeted model poisoning does not significantly affect how the global model makes decisions on benign data.

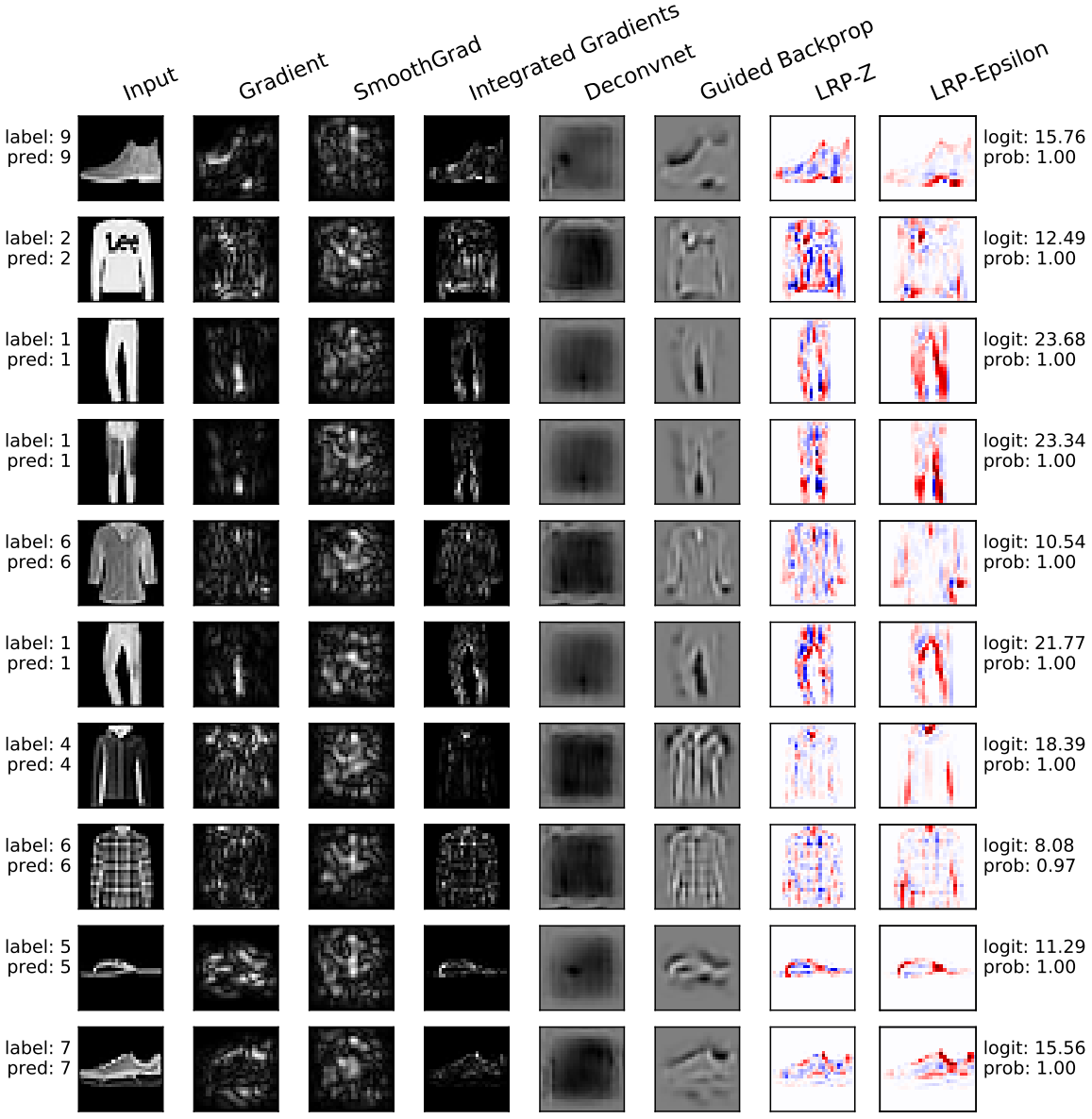


Figure 13: Decision visualizations for global model trained on Fashion MNIST data using only benign agents *on benign data.*

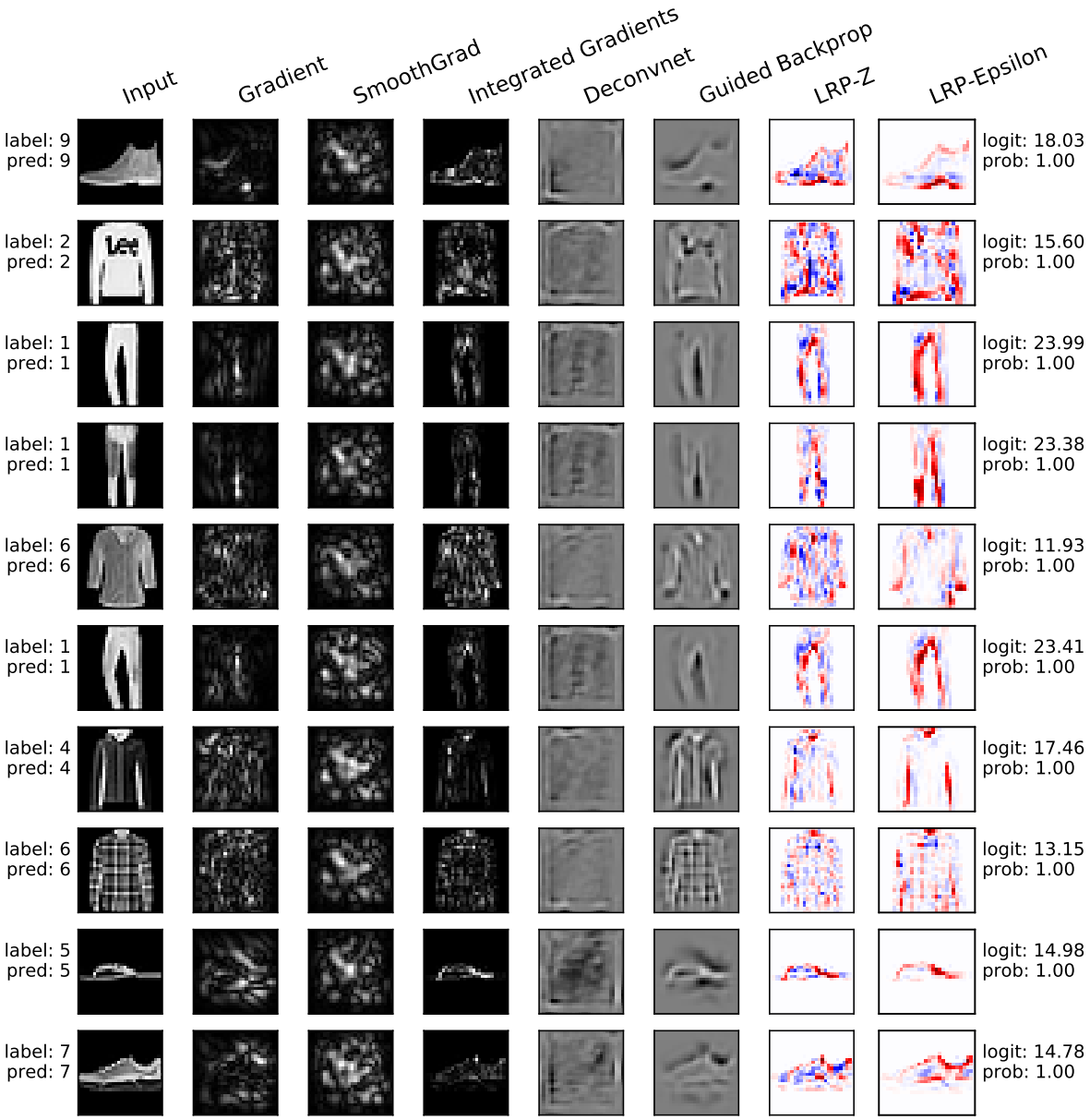


Figure 14: Decision visualizations for global model trained on Fashion MNIST data using 9 benign agents and 1 malicious agent using the baseline attack *on benign data*.