Data Poisoning Attacks Against Federated Learning Systems

Vale Tolpegin, Stacey Truex(✉), Mehmet Emre Gursoy, and Ling Liu

Georgia Institute of Technology, Atlanta, GA 30332, USA
{vtolpegin3,staceytruex,memregursoy}@gatech.edu, ling.liu@cc.gatech.edu

Abstract. Federated learning (FL) is an emerging paradigm for distributed training of large-scale deep neural networks in which participants' data remains on their own devices with only model updates being shared with a central server. However, the distributed nature of FL gives rise to new threats caused by potentially malicious participants. In this paper, we study targeted data poisoning attacks against FL systems in which a malicious subset of the participants aim to poison the global model by sending model updates derived from mislabeled data. We first demonstrate that such data poisoning attacks can cause substantial drops in classification accuracy and recall, even with a small percentage of malicious participants. We additionally show that the attacks can be targeted, i.e., they have a large negative impact only on classes that are under attack. We also study attack longevity in early/late round training, the impact of malicious participant availability, and the relationships between the two. Finally, we propose a defense strategy that can help identify malicious participants in FL to circumvent poisoning attacks, and demonstrate its effectiveness.

Keywords: Federated learning · Adversarial machine learning · Label flipping · Data poisoning · Deep learning

1 Introduction

Machine learning (ML) has become ubiquitous in today's society as a range of industries deploy predictive models into their daily workflows. This environment has not only put a premium on the ML model training and hosting technologies but also on the rich data that companies are collecting about their users to train and inform such models. Companies and users alike are consequently faced with 2 fundamental questions in this reality of ML: (1) How can privacy concerns around such pervasive data collection be moderated without sacrificing the efficacy of ML models? and (2) How can ML models be trusted as accurate predictors?

Federated ML has seen increased adoption in recent years [9,17,40] in response to the growing legislative demand to address user privacy [1,26,38]. Federated learning (FL) allows data to remain at the edge with only model parameters being shared with a central server. Specifically, there is no centralized data curator who collects and verifies an aggregate dataset. Instead, each

L. Chen et al. (Eds.): ESORICS 2020, LNCS 12308, pp. 480–501, 2020.
https://doi.org/10.1007/978-3-030-58951-6_24

data holder (participant) is responsible for conducting training on their local data. In regular intervals participants are then send model parameter values to a central parameter server or aggregator where a global model is created through aggregation of the individual updates. A global model can thus be trained over all participants' data without any individual participant needing to share their private raw data.

While FL systems allow participants to keep their raw data local, a significant vulnerability is introduced at the heart of question (2). Consider the scenario wherein a subset of participants are either malicious or have been compromised by some adversary. This can lead to these participants having mislabeled or poisonous samples in their local training data. With no central authority able to validate data, these malicious participants can consequently poison the trained global model. For example, consider Microsoft's AI chat bot Tay. Tay was released on Twitter with the underlying natural language processing model set to learn from the Twitter users it interacted with. Thanks to malicious users, Tay was quickly manipulated to learn offensive and racist language [41].

In this paper, we study the vulnerability of FL systems to malicious participants seeking to poison the globally trained model. We make minimal assumptions on the capability of a malicious FL participant – each can only manipulate the raw training data on their device. This allows for non-expert malicious participants to achieve poisoning with no knowledge of model type, parameters, and FL process. Under this set of assumptions, label flipping attacks become a feasible strategy to implement data poisoning, attacks which have been shown to be effective against traditional, centralized ML models [5,44,50,52]. We investigate their application to FL systems using complex deep neural network models.

We demonstrate our FL poisoning attacks using two popular image classification datasets: CIFAR-10 and Fashion-MNIST. Our results yield several interesting findings. First, we show that attack effectiveness (decrease in model utility) depends on the percentage of malicious users and the attack is effective even when this percentage is small. Second, we show that attacks can be targeted, i.e., they have large negative impact on the subset of classes that are under attack, but have little to no impact on remaining classes. This is desirable for adversaries who wish to poison a subset of classes while not completely corrupting the global model to avoid easy detection. Third, we evaluate the impact of attack timing (poisoning in early or late rounds of FL training) and the impact of malicious participant availability (whether malicious participants can increase their availability and selection rate to increase effectiveness). Motivated by our finding that the global model may still converge accurately after early-round poisoning stops, we conclude that largest poisoning impact can be achieved if malicious users participate in later rounds and with high availability.

Given the highly effective poisoning threat to FL systems, we then propose a defense strategy for the FL aggregator to identify malicious participants using their model updates. Our defense is based on the insight that updates sent from malicious participants have unique characteristics compared to honest participants' updates. Our defense extracts relevant parameters from the

high-dimensional update vectors and applies PCA for dimensionality reduction. Results on CIFAR-10 and Fashion-MNIST across varying malicious participant rates (2–20%) show that the aggregator can obtain clear separation between malicious and honest participants' respective updates using our defense strategy. This enables the FL aggregator to identify and block malicious participants.

The rest of this paper is organized as follows. In Sect. 2, we introduce the FL setting, threat model, attack strategy, and attack evaluation metrics. In Sect. 3, we demonstrate the effectiveness of FL poisoning attacks and analyze their impact with respect to malicious participant percentage, choice of classes under attack, attack timing, and malicious participant availability. In Sect. 4, we describe and empirically demonstrate our defense strategy. We discuss related work in Sect. 5 and conclude in Sect. 6. Our source code is available[1].

2 Preliminaries and Attack Formulation

2.1 Federated Machine Learning

FL systems allow global model training without the sharing of raw private data. Instead, individual participants only share model parameter updates. Consider a deep neural network (DNN) model. DNNs consist of multiple layers of nodes where each node is a basic functional unit with a corresponding set of parameters. Nodes receive input from the immediately preceding layer and send output to the following layer; with the first layer nodes receiving input from the training data and the final layer nodes generating the predictive result.

In a traditional DNN learning scenario, there exists a training dataset $\mathcal{D} = (x_1, ..., x_n)$ and a loss function $\mathcal{L}$. Each $x_i \in \mathcal{D}$ is defined as a set of features $\mathbf{f}_i$ and a class label $c_i \in \mathcal{C}$ where $\mathcal{C}$ is the set of all possible class values. The final layer of a DNN architecture for such a dataset will consequently contain $|\mathcal{C}|$ nodes, each corresponding to a different class in $\mathcal{C}$. The loss of this DNN given parameters θ on $\mathcal{D}$ is denoted: $\mathcal{L} = \frac{1}{n}\sum_i^n \mathcal{L}(\theta, x_i)$.

When $\mathbf{f}_i$ is fed through the DNN with model parameters θ, the output is a set of predicted probabilities $\mathbf{p}_i$. Each value $p_{c,i} \in \mathbf{p}_i$ is the predicted probability that x_i has a class value c, and $\mathbf{p}_i$ contains a probability $p_{c,i}$ for each class value $c \in \mathcal{C}$. Each predicted probability $p_{c,i}$ is computed by a node n_c in the final layer of the DNN architecture using input received from the preceding layer and n_c's corresponding parameters in θ. The predicted class for instance x_i given a model M with parameters θ then becomes $M_\theta(x_i) = \text{argmax}_{c\in\mathcal{C}} p_{c,i}$. Given a cross entropy loss function, the loss on x_i can consequently can be calculated as $\mathcal{L}(\theta, x_i) = -\sum_{c\in\mathcal{C}} y_{c,i} \log(p_{c,i})$ where $y_{c,i} = 1$ if $c = c_i$ and 0 otherwise. The goal of training a DNN model then becomes to find the parameter values for θ which minimize the chosen loss function $\mathcal{L}$.

The process of minimizing this loss is typically done through an iterative process called stochastic gradient descent (SGD). At each step, the SGD algorithm

[1] https://github.com/git-disl/DataPoisoning_FL.

(1) selects a batch of samples $B \subseteq \mathcal{D}$, (2) computes the corresponding gradient $\mathbf{g}_B = \frac{1}{|B|} \sum_{x \in B} \nabla_\theta \mathcal{L}(\theta, x)$, and (3) then updates θ in the direction $-\mathbf{g}_B$. In practice, $\mathcal{D}$ is shuffled and then evenly divided into $|B|$ sized batches such that each sample occurs in exactly one batch. Applying SGD iteratively to each of the pre-determined batches is then referred to as one epoch.

In FL environments however, the training dataset $\mathcal{D}$ is not wholly available at the aggregator. Instead, N participants $\mathcal{P}$ each hold their own private training dataset $D_1, ..., D_N$. Rather than sharing their private raw data, participants instead execute the SGD training algorithm locally and then upload updated parameters to a centralized server (aggregator). Specifically, in the initialization phase (i.e., round 0), the aggregator generates a DNN architecture with parameters θ_0 which is advertised to all participants. At each global training round r, a subset $\mathcal{P}_r$ consisting of $k \leq N$ participants is selected based on availability. Each participant $P_i \in \mathcal{P}_r$ executes one epoch of SGD locally on D_i to obtain updated parameters $\theta_{r,i}$, which are sent to the aggregator. The aggregator sets the global parameters $\theta_r = \frac{1}{k} \sum_i \theta_{r,i} \ \forall i$ where $P_i \in \mathcal{P}_r$. The global parameters θ_r are then advertised to all N participants. These global parameters at the end of round r are used in the next training round $r + 1$. After R total global training rounds, the model M is finalized with parameters θ_R.

2.2 Threat and Adversary Model

Threat Model: We consider the scenario in which a subset of FL participants are malicious or are controlled by a malicious adversary. We denote the percentage of malicious participants among all participants $\mathcal{P}$ as $m\%$. Malicious participants may be injected to the system by adding adversary-controlled devices, compromising $m\%$ of the benign participants' devices, or incentivizing (bribing) $m\%$ of benign participants to poison the global model for a certain number of FL rounds. We consider the aggregator to be honest and not compromised.

Adversarial Goal: The goal of the adversary is to manipulate the learned parameters such that the final global model M has high errors for particular classes (a subset of $\mathcal{C}$). The adversary is thereby conducting a targeted poisoning attack. This differs from untargeted attacks which instead seek indiscriminate high global model errors across all classes [6,14,51]. Targeted attacks have the desirable property that they decrease the possibility of the poisoning attack being detected by minimizing influence on non-targeted classes.

Adversary Knowledge and Capability: We consider a realistic adversary model with the following constraints. Each malicious participant can manipulate the training data D_i on their own device, but cannot access or manipulate other participants' data or the model learning process, e.g., SGD implementation, loss function, or server aggregation process. The attack is not specific to the DNN architecture, loss function or optimization function being used. It requires training data to be corrupted, but the learning algorithm remains unaltered.

2.3 Label Flipping Attacks in Federated Learning

We use a label flipping attack to implement targeted data poisoning in FL. Given a source class c_{src} and a target class c_{target} from $\mathcal{C}$, each malicious participant P_i modifies their dataset D_i as follows: For all instances in D_i whose class is c_{src}, change their class to c_{target}. We denote this attack by $c_{src} \rightarrow c_{target}$. For example, in CIFAR-10 image classification, airplane $\rightarrow$ bird denotes that images whose original class labels are *airplane* will be poisoned by malicious participants by changing their class to *bird*. The goal of the attack is to make the final global model M more likely to misclassify airplane images as bird images at test time.

Label flipping is a well-known attack in centralized ML [43,44,50,52]. It is also suitable for the FL scenario given the adversarial goal and capabilities above. Unlike other types of poisoning attacks, label flipping does not require the adversary to know the global distribution of $\mathcal{D}$, the DNN architecture, loss function $\mathcal{L}$, etc. It is time and energy-efficient, an attractive feature considering FL is often executed on edge devices . It is also easy to carry out for non-experts and does not require modification or tampering with participant-side FL software.

Table 1. Notations used throughout the paper.

M, M_{NP}	Model, model trained with no poisoning
k	Number of FL participants in each round
R	Total number of rounds of FL training
$\mathcal{P}_r$	FL participants queried at round r, $r \in [1, R]$
$\theta_r, \theta_{r,i}$	Global model parameters after round r and local model parameters at participant P_i after round r
$m\%$	Percentage of malicious participants
c_{src}, c_{target}	Source and target class in label flipping attack
M^{acc}	Global model accuracy
c_i^{recall}	Class recall for class c_i
$m\_cnt_j^i$	Baseline misclassification count from class c_i to class c_j

Attack Evaluation Metrics: At the end of R rounds of FL, the model M is finalized with parameters θ_R. Let $\mathcal{D}_{test}$ denote the test dataset used in evaluating M, where $\mathcal{D}_{test} \cap D_i = \emptyset$ for all participant datasets D_i. In the next sections, we provide a thorough analysis of label flipping attacks in FL. To do so, we use a number of evaluation metrics.

Global Model Accuracy (M^{acc}): The global model accuracy is the percentage of instances $x \in \mathcal{D}_{test}$ where the global model M with final parameters θ_R predicts $M_{\theta_R}(x) = c_i$ and c_i is indeed the true class label of x.

Class Recall (c_i^{recall})*:* For any class $c_i \in \mathcal{C}$, its class recall is the percentage $\frac{TP_i}{TP_i+FN_i} \cdot 100\%$ where TP_i is the number of instances $x \in \mathcal{D}_{test}$ where $M_{\theta_R}(x) = c_i$ **and** c_i is the true class label of x; and FN_i is the number of instances $x \in \mathcal{D}_{test}$ where $M_{\theta_R}(x) \neq c_i$ and the true class label of x is c_i.

Baseline Misclassification Count ($m\_cnt_j^i$)*:* Let M_{NP} be a global model trained for R rounds using FL without any malicious attack. For classes $c_i \neq c_j$, the baseline misclassification count from c_i to c_j, denoted $m\_cnt_j^i$, is defined as the number of instances $x \in \mathcal{D}_{test}$ where $M_{NP}(x) = c_j$ **and** the true class of x is c_i.

Table 1 provides a summary of the notation used in the rest of this paper.

3 Analysis of Label Flipping Attacks in FL

3.1 Experimental Setup

Datasets and DNN Architectures: We conduct our attacks using two popular image classification datasets: CIFAR-10 [22] and Fashion-MNIST [49]. CIFAR-10 consists of 60,000 color images in 10 object classes such as deer, airplane, and dog with 6,000 images included per class. The complete dataset is pre-divided into 50,000 training images and 10,000 test images. Fashion-MNIST consists of a training set of 60,000 images and a test set of 10,000 images. Each image in Fashion-MNIST is gray-scale and associated with one of 10 classes of clothing such as pullover, ankle boot, or bag. In experiments with CIFAR-10, we use a convolutional neural network with six convolutional layers, batch normalization, and two fully connected dense layers. This DNN architecture achieves a test accuracy of 79.90% in the centralized learning scenario, i.e. $N = 1$, without poisoning. In experiments with Fashion-MNIST, we use a two layer convolutional neural network with batch normalization, an architecture which achieves 91.75% test accuracy in the centralized scenario without poisoning. Further details of the datasets and DNN model architectures can be found in Appendix A.

Federated Learning Setup: We implement FL in Python using the PyTorch [35] library. By default, we have $N = 50$ participants, one central aggregator, and $k = 5$. We use an *independent and identically distributed* (*iid*) data distribution, i.e., we assume the total training dataset is uniformly randomly distributed among all participants with each participant receiving a unique subset of the training data. The testing data is used for model evaluation only and is therefore not included in any participant P_i's train dataset D_i. Observing that both DNN models converge after fewer than 200 training rounds, we set our FL experiments to run for $R = 200$ rounds total.

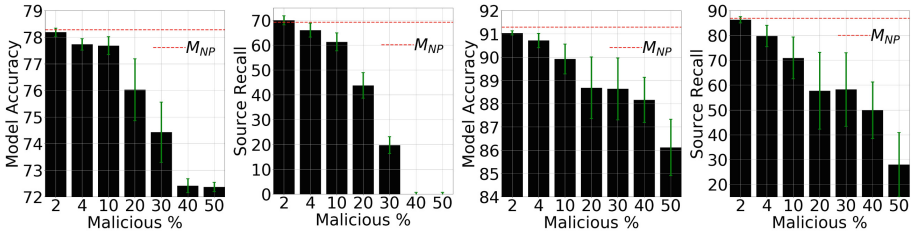
Label Flipping Process: In order to simulate the label flipping attack in a FL system with N participants of which $m\%$ are malicious, at the start of each experiment we randomly designate $N \times m\%$ of the participants from $\mathcal{P}$

as malicious. The rest are honest. To address the impact of random selection of malicious participants, by default we repeat each experiment 10 times and report the average results. Unless otherwise stated, we use $m = 10\%$.

For both datasets we consider three label flipping attack settings representing a diverse set of conditions in which to base adversarial attacks. These conditions include (1) a source class → target class pairing whose source class was very frequently misclassified as the target class in federated, non-poisoned training, (2) a pairing where the source class was very infrequently misclassified as the target class, and (3) a pairing between these two extremes. Specifically, for CIFAR-10 we test (1) 5: dog → 3: cat, (2) 0: airplane → 2: bird, and (3) 1: automobile → 9: truck. For Fashion-MNIST we experiment with (1) 6: shirt → 0: t-shirt/top, (2) 1: trouser → 3: dress, and (3) 4: coat → 6: shirt.

3.2 Label Flipping Attack Feasibility

We start by investigating the feasibility of poisoning FL systems using label flipping attacks. Figure 1 outlines the global model accuracy and source class recall in scenarios with malicious participant percentage m ranging from 2% to 50%. Results demonstrate that as the malicious participant percentage, increases the global model utility (test accuracy) decreases. Even with small m, we observe a decrease in model accuracy compared to a non-poisoned model (denoted by M_{NP} in the graphs), and there is an even larger decrease in source class recall. In experiments with CIFAR-10, once m reaches 40%, the recall of the source class decreases to 0% and the global model accuracy decreases from 78.3% in the non-poisoned setting to 74.4% in the poisoned setting. Experiments conducted on Fashion-MNIST show a similar pattern of utility loss. With $m = 4\%$ source class recall drops by $\sim 10\%$ and with $m = 10\%$ it drops by $\sim 20\%$. It is therefore clear that an adversary who controls even a minor proportion of the total participant population is capable of significantly impacting global model utility.



(a) CIFAR-10 M^{acc} (b) CIFAR-10 c_{src}^{recall} (c) F-MNIST M^{acc} (d) F-MNIST c_{src}^{recall}

Fig. 1. Evaluation of attack feasibility and impact of malicious participant percentage on attack effectiveness. CIFAR-10 experiments are for the 5 → 3 setting while Fashion-MNIST experiments are for the 4 → 6 setting. Results are averaged from 10 runs for each setting of m%. The black bars are mean over the 10 runs and the green error bars denote standard deviation. (Color figure online)

While both datasets are vulnerable to label flipping attacks, the degree of vulnerability varies between datasets with CIFAR-10 demonstrating more vulnerability than Fashion-MNIST. For example, consider the 30% malicious scenario, Figure 1b shows the source class recall for the CIFAR-10 dataset drops to 19.7% while Fig. 1d shows a much lower decrease for the Fashion-MNIST dataset with 58.2% source class recall under the same experimental settings.

Table 2. Loss in source class recall for three source → target class settings with differing baseline misclassification counts in CIFAR-10 and Fashion-MNIST. Loss averaged from 10 runs. Highlighted bold entries are highest loss in each.

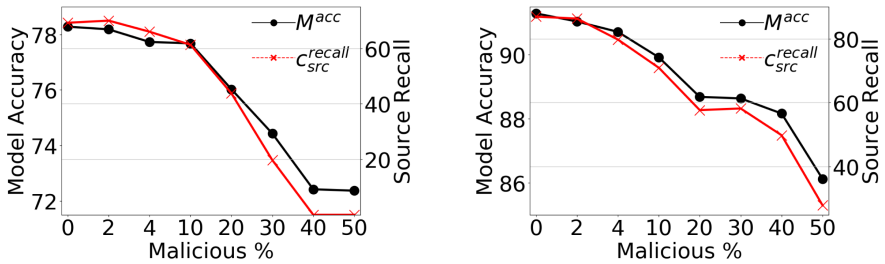
$c_{src} \rightarrow c_{target}$	$m\_cnt^{src}_{target}$	Percentage of Malicious Participants (m%)						
		2	4	10	20	30	40	50
CIFAR-10								
0 → 2	16	**1.42%**	2.93%	**10.2%**	14.1%	48.3%	**73%**	**70.5%**
1 → 9	56	0.69%	**3.75%**	6.04%	15%	36.3%	49.2%	54.7%
5 → 3	200	0%	3.21%	7.92%	**25.4%**	**49.5%**	69.2%	69.2%
Fashion-MNIST								
1 → 3	18	0.12%	0.42%	2.27%	2.41%	**40.3%**	**45.4%**	42%
4 → 6	51	**0.61%**	**7.16%**	**16%**	**29.2%**	28.7%	37.1%	**58.9%**
6 → 0	118	−1%	2.19%	7.34%	9.81%	19.9%	39%	43.4%

On the other hand, vulnerability variation based on source and target class settings is less clear. In Table 2, we report the results of three different combinations of source → target attacks for each dataset. Consider the two extreme settings for the CIFAR-10 dataset: on the low end the 0 → 2 setting has a baseline misclassification count of 16 while the high end count is 200 for the 5 → 3 setting. Because of the DNN's relative challenge in differentiating class 5 from class 3 in the non-poisoned setting, it could be anticipated that conducting a label flipping attack within the 5 → 3 setting would result in the greatest impact on source class recall. However, this was not the case. Table 2 shows that in only two out of the six experimental scenarios did 5 → 3 record the largest drop in source class recall. In fact, four scenarios' results show the 0 → 2 setting, the setting with the lowest baseline misclassification count, as the most effective option for the adversary. Experiments with Fashion-MNIST show a similar trend, with label flipping attacks conducted in the 4 → 6 setting being the most successful rather than the 6 → 0 setting which has more than 2× the number of baseline misclassifications. These results indicate that identifying the most vulnerable source and target class combination may be a non-trivial task for the adversary, and that there is not necessarily a correlation between non-poisoned misclassification performance and attack effectiveness.

We additionally study a desirable feature of the label flipping attack: they appear to be targeted. Specifically, Table 3 reports the following quantities for each source → target flipping scenario: loss in source class recall, loss in target

Table 3. Changes due to poisoning in source class recall, target class recall, and total recall for all remaining classes (non-source, non-target). Results are averaged from 10 runs in each setting. The maximum standard deviation observed was 1.45% in source class recall and 1.13% in target class recall.

$c_{src} \rightarrow c_{target}$	$\Delta\ c_{src}^{recall}$	$\Delta\ c_{target}^{recall}$	$\sum$ all other $\Delta\ c^{recall}$
CIFAR-10			
$0 \rightarrow 2$	-6.28%	1.58%	0.34%
$1 \rightarrow 9$	-6.22%	2.28%	0.16%
$5 \rightarrow 3$	-6.12%	3.00%	0.17%
Fashion-MNIST			
$1 \rightarrow 3$	-2.23%	0.25%	0.01%
$4 \rightarrow 6$	-9.96%	2.40%	0.09%
$6 \rightarrow 0$	-8.87%	2.59%	0.20%



(a) CIFAR-10 (b) Fashion-MNIST

Fig. 2. Relationship between global model accuracy and source class recall across changing percentages of malicious participants for CIFAR-10 and Fashion-MNIST. As each dataset has 10 classes, the scale for M^{acc} vs c_{src}^{recall} is 1:10.

class recall, and loss in recall of all remaining classes. We observe that the attack causes substantial change in source class recall ($> 6\%$ drop in most cases) and target class recall. However, the attack impact on the recall of remaining classes is an order of magnitude smaller. CIFAR-10 experiments show a maximum of 0.34% change in class recalls attributable to non-source and non-target classes and Fashion-MNIST experiments similarly show a maximum change of 0.2% attributable to non-source and non-target classes, both of which are relatively minor compared to source and target classes. Thus, the attack is causing the global model to misclassify instances belonging to c_{src} as c_{target} at test time while other classes remain relatively unimpacted, demonstrating its targeted nature towards c_{src} and c_{target}. Considering the large impact of the attack on source class recall, changes in source class recall therefore make up the vast majority of the decreases in global model accuracy caused by label flipping attacks in FL

systems. This observation can also be seen in Fig. 2 where the change in global model accuracy closely follows the change in source class recall.

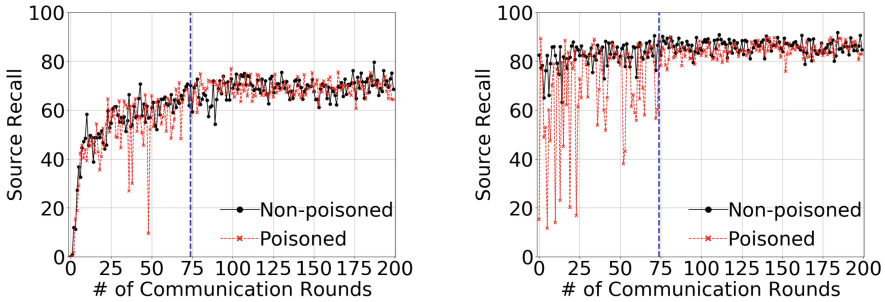
The targeted nature of the label flipping attack allows for adversaries to remain under the radar in many FL systems. Consider systems where the data contain 100 classes or more, as is the case in CIFAR-100 [22] and ImageNet [13]. In such cases, targeted attacks become much more stealthy due to their limited impact to classes other than source and target.

3.3 Attack Timing in Label Flipping Attacks

While label flipping attacks can occur at any point in the learning process and last for arbitrary lengths, it is important to understand the capabilities of adversaries who are available for only part of the training process. For instance, Google's Gboard application of FL requires all participant devices be plugged into power and connected to the internet via WiFi [9]. Such requirements create cyclic conditions where many participants are not available during the day, when phones are not plugged in and are actively in use. Adversaries can take advantage of this design choice, making themselves available at times when honest participants are unable to.

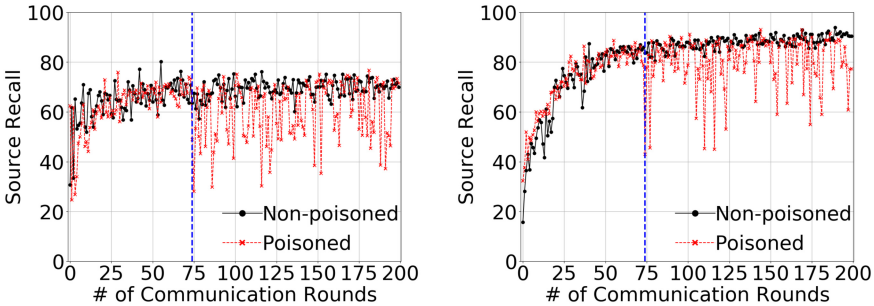
We consider two scenarios in which the adversary is restricted in the time in which they are able to make malicious participants available: one in which the adversary makes malicious participants available only before the 75th training round, and one in which malicious participants are available only after the 75th training round. As the rate of global model accuracy improvement decreases with both datasets by training round 75, we choose this point to highlight how pre-established model stability may effect an adversary's ability to launch an effective label flipping attack. Results for the first scenario are given in Fig. 3 whereas the results for the second scenario are given in Fig. 4.

In Figure 3, we compare source class recall in a non-poisoned setting versus with poisoning only before round 75. Results on both CIFAR-10 and Fashion-MNIST show that while there are observable drops in source class recall during the rounds with poisoning (1–75), the global model is able to recover quickly after poisoning finishes (after round 75). Furthermore, the final convergence of the models (towards the end of training) are not impacted, given the models with and without poisoning are converge with roughly the same recall values. We do note that some CIFAR-10 experiments exhibited delayed convergence by an additional 50–100 training rounds, but these circumstances were rare and still eventually achieved the accuracy and recall levels of a non-poisoned model despite delayed convergence.



(a) CIFAR-10 (b) Fashion-MNIST

Fig. 3. Source class recall by round for experiments with "early round poisoning", i.e., malicious participation only in the first 75 rounds ($r < 75$). The blue line indicates the round at which malicious participation is no longer allowed.



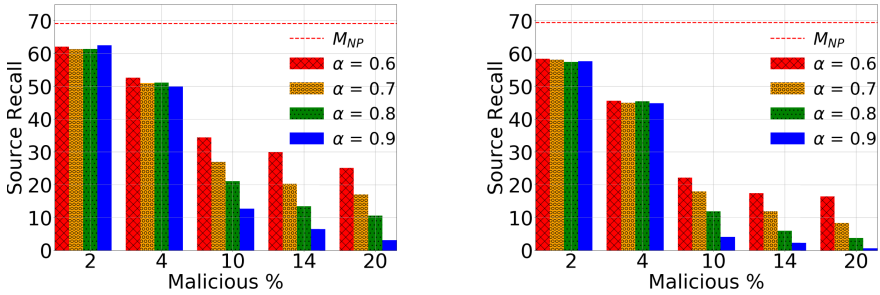
(a) CIFAR-10 (b) Fashion-MNIST

Fig. 4. Source class recall by round for experiments with "late round poisoning", i.e., malicious participation only after round 75 ($r \geq 75$). The blue line indicates the round at which malicious participation starts.

In Fig. 4, we compare source class recall in a non-poisoned setting versus with poisoning limited to the 75th and later training rounds. These results show the impact of such late poisoning demonstrating limited longevity; a phenomena which can be seen in the quick and dramatic changes in source class recall. Specifically, source class recall quickly returns to baseline levels once fewer malicious participants are selected in a training round even immediately

Table 4. Final source class recall when at least one malicious party participates in the final round R versus when all participants in round R are non-malicious. Results averaged for 10 runs for each experimental setting.

$c_{src} \rightarrow c_{target}$	Source Class Recall (c_{src}^{recall})	
	$m\% \in \mathcal{P}_R > 0$	$m\% \in \mathcal{P}_R = 0$
CIFAR-10		
0 → 2	73.90%	82.45%
1 → 9	77.30%	89.40%
5 → 3	57.50%	73.10%
Fashion-MNIST		
1 → 3	84.32%	96.25%
4 → 6	51.50%	89.60%
6 → 0	49.80%	73.15%



(a) CIFAR-10 (b) Fashion-MNIST

Fig. 5. Evaluation of impact from malicious participants' availability α on source class recall. Results are averaged from 3 runs for each setting.

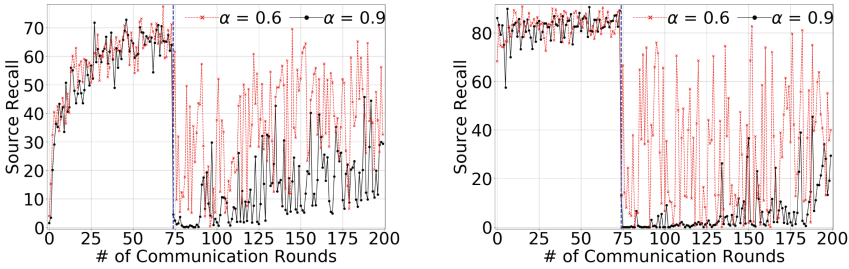
following a round with a large number of malicious participants having caused a dramatic drop. However, the final poisoned model in the late-round poisoning scenario may show substantial difference in accuracy or recall compared to a non-poisoned model. This is evidenced by the CIFAR-10 experiment in Fig. 4, in which the source recall of the poisoned model is ∼10% lower compared to non-poisoned.

Furthermore, we observe that model convergence on both datasets is negatively impacted, as evidenced by the large variances in recall values between consecutive rounds. Consider Table 4 where results are compared when either (1) at least one malicious participant is selected for $\mathcal{P}_R$ or (2) $\mathcal{P}_R$ is made entirely of honest participants. When at least one malicious participant is selected, the final source class recall is, on average, 12.08% lower with the CIFAR-10 dataset and 24.46% lower with the Fashion-MNIST dataset. The utility impact from the label flipping attack is therefore predominantly tied to the number of malicious participants selected in the last few rounds of training.

3.4 Malicious Participant Availability

Given the impact of malicious participation in late training rounds on attack effectiveness, we now introduce a malicious participant availability parameter α. By varying α we can simulate the adversary's ability to control compromised participants' availability (i.e. ensuring connectivity or power access) at various points in training. Specifically, α represents malicious participants' availability and therefore likeliness to be selected relative to honest participants. For example, if $\alpha = 0.6$, when selecting each participant $P_i \in \mathcal{P}_r$ for round r, there is a 0.6 probability that P_i will be one of the malicious participants. Larger α implies higher likeliness of malicious participation. In cases where $k > N \times m\%$, the number of malicious participants in $\mathcal{P}_r$ is bounded by $N \times m\%$.

Figure 5 reports results for varying values of α in late round poisoning, i.e., malicious participation is limited to rounds $r \geq 75$. Specifically, we are interested in studying those scenarios where an adversary boosts the availability of



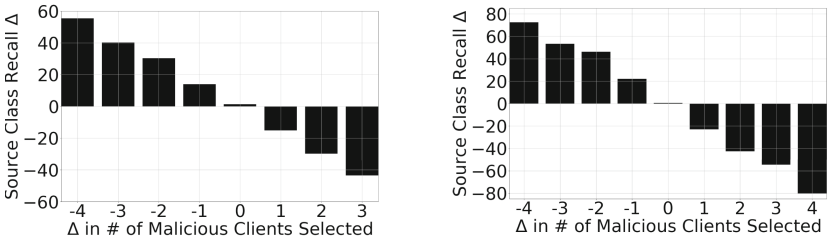
(a) CIFAR-10 (b) Fashion-MNIST

Fig. 6. Source class recall by round when malicious participants' availability is close to that of honest participants ($\alpha = 0.6$) vs significantly increased ($\alpha = 0.9$). The blue line indicates the round in which attack starts.

the malicious participants enough that their selection becomes more likely than the non-malicious participants, hence in Fig. 5 we use $\alpha \geq 0.6$. The reported source class recalls in Fig. 5 are averaged over the last 125 rounds (total 200 rounds minus first 75 rounds) to remove the impact of individual round variability; further, each experiment setting is repeated 3 times and results are averaged. The results show that, when the adversary maintains sufficient representation in the participant pool (i.e. $m \geq 10\%$), manipulating the availability of malicious participants can yield significantly higher impact on the global model utility with source class recall losses in excess of 20%. On both datasets with $m \geq 10\%$, the negative impact on source class recall is highest with $\alpha = 0.9$, which is followed by $\alpha = 0.8$, $\alpha = 0.7$ and $\alpha = 0.6$, i.e., in decreasing order of malicious participant availability. Thus, in order to mount an impactful attack, it is in the best interests of the adversary to perform the attack *with highest malicious participant availability in late rounds*. We note that when k is significantly larger than $N \times m\%$, increasing availability (α) will be insufficient for meaningfully increasing malicious participant selection in individual training rounds. Therefore, experiments where $m < 10\%$ show little variation despite changes in α.

To more acutely demonstrate the impact of α, Fig. 6 reports source class recall by round when $\alpha = 0.6$ and $\alpha = 0.9$ for both the CIFAR-10 and Fashion-MNIST datasets. In both datasets, when malicious participants are available more frequently, the source class recall is effectively shifted lower in the graph, i.e., source class recall values with $\alpha = 0.9$ are often much smaller than those with $\alpha = 0.6$. We note that the high round-by-round variance in both graphs is due to the probabilistic variability in number of malicious participants in individual training rounds. When fewer malicious participants are selected in one training round relative to the previous round, source recall increases. When more malicious participants are selected in an individual round relative to the previous round, source recall falls.

We further explore and illustrate our last remark with respect to the impact of malicious parties' participation in consecutive rounds in Fig. 7. In this figure, the x-axis represents the change in the number of malicious clients participating



(a) CIFAR-10 (b) Fashion-MNIST

Fig. 7. Relationship between change in source class recall in consecutive rounds versus change in number of malicious participants in consecutive rounds. Specifically, $\forall r \geq 75$ the y-axis represents $(c_{src}^{recall}$ @ round $r)$ - $(c_{src}^{recall}$ @ round $r-1)$ while the x-axis represents (# of malicious $\in \mathcal{P}_r$) - (# of malicious $\in \mathcal{P}_{r-1}$).

in consecutive rounds, i.e., (# of malicious $\in \mathcal{P}_r$) – (# of malicious $\in \mathcal{P}_{r-1}$). The y-axis represents the change in source class recall between these consecutive rounds, i.e., $(c_{src}^{recall}$ @ round $r)$ – $(c_{src}^{recall}$ @ round $r-1)$. The reported results are then averaged across multiple runs of FL and all cases in which each participation difference was observed. The results confirm our intuition that, when $\mathcal{P}_r$ contains more malicious participants than $\mathcal{P}_{r-1}$, there is a substantial drop in source class recall. For large differences (such as $+3$ or $+4$), the drop could be as high as 40% or 60%. In contrast, when $\mathcal{P}_r$ contains fewer malicious participants than $\mathcal{P}_{r-1}$, there is a substantial increase in source class recall, which can be as high as 60% or 40% when the difference is -4 or -3. Altogether, this demonstrates the possibility that the DNN could recover significantly even in few rounds of FL training, if a large enough decrease in malicious participation could be achieved.

4 Defending Against Label Flipping Attacks

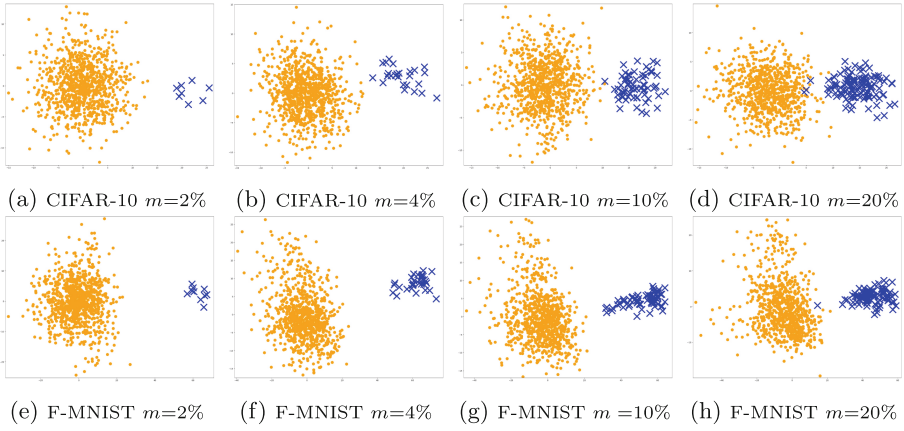
Given a highly effective adversary, how can a FL system defend against the label flipping attacks discussed thus far? To that end, we propose a defense which enables the aggregator to identify malicious participants.

After identifying malicious participants, the aggregator may blacklist them or ignore their updates $\theta_{r,i}$ in future rounds. We showed in Sects. 3.3 and 3.4 that high-utility model convergence can be eventually achieved after eliminating malicious participation. The feasibility of such a recovery from early round attacks supports use of the proposed identification approach as a defense strategy.

Our defense is based on the following insight: The parameter updates sent from malicious participants have unique characteristics compared to honest participants' updates for a subset of the parameter space. However, since DNNs have many parameters (i.e., $\theta_{r,i}$ is extremely high dimensional) it is non-trivial to analyze parameter updates by hand. Thus, we propose an automated strategy for identifying the relevant parameter subset and for studying participant updates using dimensionality reduction (PCA).

Algorithm 1: Identifying Malicious Model Updates in FL

def `evaluate_updates`($\mathcal{R}$: *set of vulnerable train rounds,* $\mathcal{P}$: *participant set*)**:**
- $\mathcal{U} = \emptyset$
- **for** $r \in \mathcal{R}$ **do**
  - $\mathcal{P}_r \leftarrow$ participants $\in \mathcal{P}$ queried in training round r
  - $\theta_{r-1} \leftarrow$ global model parameters after training round $r-1$
  - **for** $P_i \in \mathcal{P}_r$ **do**
    - $\theta_{r,i} \leftarrow$ updated parameters after `train_DNN`(θ_{r-1}, D_i)
    - $\theta_{\Delta,i} \leftarrow \theta_{r,i} - \theta_r$
    - $\theta^{src}_{\Delta,i} \leftarrow$ parameters $\in \theta_{\Delta,i}$ connected to source class output node
    - Add $\theta^{src}_{\Delta,i}$ to $\mathcal{U}$
- $\mathcal{U}' \leftarrow$ `standardize`($\mathcal{U}$)
- $\mathcal{U}'' \leftarrow$ `PCA`($\mathcal{U}'$, *components=2*)
- `plot`($\mathcal{U}''$)



(a) CIFAR-10 m=2% (b) CIFAR-10 m=4% (c) CIFAR-10 m=10% (d) CIFAR-10 m=20%

(e) F-MNIST m=2% (f) F-MNIST m=4% (g) F-MNIST m =10% (h) F-MNIST m=20%

Fig. 8. PCA plots with 2 components demonstrating the ability of Algorithm 1 to identify updates originating from a malicious versus honest participant. Plots represent relevant gradients collected from all training rounds $r > 10$. Blue Xs represent gradients from malicious participants while yellow Os represent gradients from honest participants. (Color figure online)

The description of our defense strategy is given in Algorithm 1. Let $\mathcal{R}$ denote the set of vulnerable FL training rounds and c_{src} be the class that is suspected to be the source class of a poisoning attack. We note that if c_{src} is unknown, the aggregator can defend against potential attacks such that $c_{src} = c\ \forall c \in \mathcal{C}$. We also note that for a given c_{src}, Algorithm 1 considers label flipping for all possible c_{target}. An aggregator therefore will conduct $|\mathcal{C}|$ independent iterations of Algorithm 1, which can be conducted in parallel. For each round $r \in \mathcal{R}$ and participant $P_i \in \mathcal{P}_r$, the aggregator computes the delta in participant's model update compared to the global model, i.e., $\theta_{\Delta,i} \leftarrow \theta_{r,i} - \theta_r$. Recall from Sect. 2.1 that a predicted probability for any given class c is computed by a specific node

n_c in the final layer DNN architecture. Given the aggregator's goal of defending against the label flipping attack from c_{src}, only the subset of the parameters in $\theta_{\Delta,i}$ corresponding to $n_{c_{src}}$ is extracted. The outcome of the extraction is denoted by $\theta^{src}_{\Delta,i}$ and added to a global list $\mathcal{U}$ built by the aggregator. After $\mathcal{U}$ is constructed across multiple rounds and participant deltas, it is standardized by removing the mean and scaling to unit variance. The standardized list $\mathcal{U}'$ is fed into Principal Component Analysis (PCA), which is a popular ML technique used for dimensionality reduction and pattern visualization. For ease of visualization, we use and plot results with two dimensions (two components).

In Fig. 8, we show the results of Algorithm 1 on CIFAR-10 and Fashion-MNIST across varying malicious participation rate m, with $\mathcal{R} = [10, 200]$. Even in scenarios with low m, as is shown in Figs. 8a and 8e, our defense is capable of differentiating between malicious and honest participants. In all graphs, the PCA outcome shows that malicious participants' updates belong to a visibly different cluster compared to honest participants' updates which form their own cluster. Another interesting observation is that our defense does not suffer from the "gradient drift" problem. Gradient drift is a potential challenge in designing a robust defense, since changes in model updates may be caused both by actual DNN learning and convergence (which is desirable) or malicious poisoning attempt (which our defense is trying to identify and prevent). Our results show that, even though the defense is tested with a long period of rounds (190 training rounds since $\mathcal{R} = [10, 200]$), it remains capable of separating malicious and honest participants, demonstrating its robustness to gradient drift.

A FL system aggregator can therefore effectively identify malicious participants, and consequently restrict their participation in mobile training, by conducting such gradient clustering prior to aggregating parameter updates at each round. Clustering model gradients for malicious participant identification presents a strong defense as it does not require access to any public validation dataset, as is required in [3], which is not necessarily possible to acquire.

5 Related Work

Poisoning attacks are highly relevant in domains such as spam filtering [10,32], malware and network anomaly detection [11,24,39], disease diagnosis [29], computer vision [34], and recommender systems [15,54]. Several poisoning attacks were developed for popular ML models including SVM [6,12,44,45,50,52], regression [19], dimensionality reduction [51], linear classifiers [12,23,57], unsupervised learning [7], and more recently, neural networks [12,30,42,45,53,58]. However, most of the existing work is concerned with poisoning ML models in the traditional setting where training data is first collected by a centralized party. In contrast, our work studies poisoning attacks in the context of FL. As a result, many of the poisoning attacks and defenses that were designed for traditional ML are not suitable to FL. For example, attacks that rely on crafting optimal poison instances by observing the training data distribution are inapplicable since the

malicious FL participant may only access and modify the training data s/he holds. Similarly, server-side defenses that rely on filtering and eliminating poison instances through anomaly detection or k-NN [36,37] are inapplicable to FL since the server only observes parameter updates from FL participants, not their individual instances.

The rising popularity of FL has led to the investigation of different attacks in the context of FL, such as backdoor attacks [2,46], gradient leakage attacks [18,27,59] and membership inference attacks [31,47,48]. Most closely related to our work are poisoning attacks in FL. There are two types of poisoning attacks in FL: *data poisoning* and *model poisoning*. Our work falls under the data poisoning category. In data poisoning, a malicious FL participant manipulates their training data, e.g., by adding poison instances or adversarially changing existing instances [16,43]. The local learning process is otherwise not modified. In model poisoning, the malicious FL participant modifies its learning process in order to create adversarial gradients and parameter updates. [4] and [14] demonstrated the possibility of causing high model error rates through targeted and untargeted model poisoning attacks. While model poisoning is also effective, data poisoning may be preferable or more convenient in certain scenarios, since it does not require adversarial tampering of model learning software on participant devices, it is efficient, and it allows for non-expert poisoning participants.

Finally, FL poisoning attacks have connections to the concept of *Byzantine threats*, in which one or more participants in a distributed system fail or misbehave. In FL, Byzantine behavior was shown to lead to sub-optimal models or non-convergence [8,20]. This has spurred a line of work on Byzantine-resilient aggregation for distributed learning, such as Krum [8], Bulyan [28], trimmed mean, and coordinate-wise median [55]. While model poisoning may remain successful despite Byzantine-resilient aggregation [4,14,20], it is unclear whether optimal data poisoning attacks can be found to circumvent an individual Byzantine-resilient scheme, or whether one data poisoning attack may circumvent multiple Byzantine-resilient schemes. We plan to investigate these issues in future work.

6 Conclusion

In this paper we studied data poisoning attacks against FL systems. We demonstrated that FL systems are vulnerable to label flipping poisoning attacks and that these attacks can significantly negatively impact the global model. We also showed that the negative impact on the global model increases as the proportion of malicious participants increases, and that it is possible to achieve targeted poisoning impact. Further, we demonstrated that adversaries can enhance attack effectiveness by increasing the availability of malicious participants in later rounds. Finally, we proposed a defense which helps an FL aggregator separate malicious from honest participants. We showed that our defense is capable of identifying malicious participants and it is robust to gradient drift.

As poisoning attacks against FL systems continue to emerge as important research topics in the security and ML communities [4,14,21,33,56], we plan to continue our work in several ways. First, we will study the impacts of the attack and defense on diverse FL scenarios differing in terms of data size, distribution among FL participants (iid vs non-iid), data type, total number of instances available per class, etc. Second, we will study more complex adversarial behaviors such as each malicious participant changing the labels of only a small portion of source samples or using more sophisticated poisoning strategies to avoid being detected . Third, while we designed and tested our defense against the label flipping attack, we hypothesize the defense will be useful against model poisoning attacks since malicious participants' gradients are often dissimilar to those of honest participants. Since our defense identifies dissimilar or anomalous gradients, we expect the defense to be effective against other types of FL attacks that cause dissimilar or anomalous gradients. In future work, we will study the applicability of our defense against such other FL attacks including model poisoning, untargeted poisoning, and backdoor attacks.

Acknowledgements. This research is partially sponsored by NSF CISE SaTC 1564097 and 2038029. The second author acknowledges an IBM PhD Fellowship Award and the support from the Enterprise AI, Systems & Solutions division led by Sandeep Gopisetty at IBM Almaden Research Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding agencies and companies mentioned above.

A DNN Architectures and Configuration

All NNs were trained using PyTorch version 1.2.0 with random weight initialization. Training and testing was completed using a NVIDIA 980 Ti GPU-accelerator. When necessary, all CUDA tensors were mapped to CPU tensors before exporting to Numpy arrays. Default drivers provided by Ubuntu 19.04 and built-in GPU support in PyTorch was used to accelerate training. Details can be found in our repository: https://github.com/git-disl/DataPoisoning_FL.

Fashion-MNIST: We do not conduct data pre-processing. We use a Convolutional Neural Network with the architecture described in Table 6. In the table, Conv = Convolutional Layer, and Batch Norm = Batch Normalization.

CIFAR-10: We conduct data pre-processing prior to training. Data is normalized with mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225]. Values reflect mean and standard deviation of the ImageNet dataset [13] and are commonplace, even expected when using Torchvision [25] models. We additionally perform data augmentation with random horizontal flipping, random cropping with size 32, and default padding. Our CNN is detailed in Table 5.

Table 5. CIFAR-10 CNN.

Layer type	Size
Conv + ReLu + Batch Norm	$3 \times 3 \times 32$
Conv + ReLu + Batch Norm	$3 \times 32 \times 32$
Max Pooling	2×2
Conv + ReLu + Batch Norm	$3 \times 32 \times 64$
Conv + ReLu + Batch Norm	$3 \times 64 \times 64$
Max Pooling	2×2
Conv + ReLu + Batch Norm	$3 \times 64 \times 128$
Conv + ReLu + Batch Norm	$3 \times 128 \times 128$
Max Pooling	2×2
Fully Connected	2048
Fully Connected + Softmax	128/10

Table 6. Fashion-MNIST CNN.

Layer type	Size
Conv + ReLu + Batch Norm	$5 \times 1 \times 16$
Max Pooling	2×2
Conv + ReLu + Batch Norm	$5 \times 16 \times 32$
Max Pooling	2×2
Fully Connected	1568/10

References

1. An Act: Health insurance portability and accountability act of 1996. Public Law 104-191 (1996)
2. Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., Shmatikov, V.: How to backdoor federated learning. arXiv preprint arXiv:1807.00459 (2018)
3. Baracaldo, N., Chen, B., Ludwig, H., Safavi, J.A.: Mitigating poisoning attacks on machine learning models: a data provenance based approach. In: 10th ACM Workshop on Artificial Intelligence and Security, pp. 103–110 (2017)
4. Bhagoji, A.N., Chakraborty, S., Mittal, P., Calo, S.: Analyzing federated learning through an adversarial lens. In: International Conference on Machine Learning, pp. 634–643 (2019)
5. Biggio, B., Nelson, B., Laskov, P.: Support vector machines under adversarial label noise. In: Asian Conference on Machine Learning, pp. 97–112 (2011)
6. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: Proceedings of the 29th International Conference on International Conference on Machine Learning, pp. 1467–1474 (2012)
7. Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., Roli, F.: Is data clustering in adversarial settings secure? In: Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, pp. 87–98 (2013)
8. Blanchard, P., Guerraoui, R., Stainer, J., et al.: Machine learning with adversaries: byzantine tolerant gradient descent. In: NeurIPS, pp. 119–129 (2017)
9. Bonawitz, K., et al.: Towards federated learning at scale: System design. In: SysML 2019 (2019, to appear). https://arxiv.org/abs/1902.01046
10. Bursztein, E.: Attacks against machine learning - an overview (2018). https://elie.net/blog/ai/attacks-against-machine-learning-an-overview/
11. Chen, S., et al.: Automated poisoning attacks and defenses in malware detection systems: an adversarial machine learning approach. Comput. Secur. **73**, 326–344 (2018)
12. Demontis, A., et al.: Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In: 28th USENIX Security Symposium, pp. 321–338 (2019)
13. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)

14. Fang, M., Cao, X., Jia, J., Gong, N.Z.: Local model poisoning attacks to byzantine-robust federated learning. In: USENIX Security Symposium (2020, to appear)
15. Fang, M., Yang, G., Gong, N.Z., Liu, J.: Poisoning attacks to graph-based recommender systems. In: Proceedings of the 34th Annual Computer Security Applications Conference, pp. 381–392 (2018)
16. Fung, C., Yoon, C.J., Beschastnikh, I.: Mitigating sybils in federated learning poisoning. arXiv preprint arXiv:1808.04866 (2018)
17. Hard, A., et al.: Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604 (2018)
18. Hitaj, B., Ateniese, G., Perez-Cruz, F.: Deep models under the GAN: information leakage from collaborative deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 603–618 (2017)
19. Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., Li, B.: Manipulating machine learning: poisoning attacks and countermeasures for regression learning. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 19–35. IEEE (2018)
20. Kairouz, P., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019)
21. Khazbak, Y., Tan, T., Cao, G.: MLGuard: mitigating poisoning attacks in privacy preserving distributed collaborative learning (2020)
22. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
23. Liu, C., Li, B., Vorobeychik, Y., Oprea, A.: Robust linear regression against training data poisoning. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 91–102 (2017)
24. Maiorca, D., Biggio, B., Giacinto, G.: Towards adversarial malware detection: lessons learned from pdf-based attacks. ACM Comput. Surv. (CSUR) **52**(4), 1–36 (2019)
25. Marcel, S., Rodriguez, Y.: Torchvision the machine-vision package of torch. In: 18th ACM International Conference on Multimedia, pp. 1485–1488 (2010)
26. Mathews, K., Bowman, C.: The California consumer privacy act of 2018 (2018)
27. Melis, L., Song, C., De Cristofaro, E., Shmatikov, V.: Exploiting unintended feature leakage in collaborative learning. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 691–706. IEEE (2019)
28. Mhamdi, E.M.E., Guerraoui, R., Rouault, S.: The hidden vulnerability of distributed learning in byzantium. arXiv preprint arXiv:1802.07927 (2018)
29. Mozaffari-Kermani, M., Sur-Kolay, S., Raghunathan, A., Jha, N.K.: Systematic poisoning attacks on and defenses for machine learning in healthcare. IEEE J. Biomed. Health Inform. **19**(6), 1893–1905 (2014)
30. Muñoz-González, L., et al.: Towards poisoning of deep learning algorithms with back-gradient optimization. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, pp. 27–38 (2017)
31. Nasr, M., Shokri, R., Houmansadr, A.: Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 739–753. IEEE (2019)
32. Nelson, B., et al.: Exploiting machine learning to subvert your spam filter. LEET **8**, 1–9 (2008)
33. Nguyen, T.D., Rieger, P., Miettinen, M., Sadeghi, A.R.: Poisoning attacks on federated learning-based IoT intrusion detection system (2020)

34. Papernot, N., McDaniel, P., Sinha, A., Wellman, M.P.: SoK: security and privacy in machine learning. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 399–414. IEEE (2018)
35. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: NeurIPS, pp. 8024–8035 (2019)
36. Paudice, A., Muñoz-González, L., Gyorgy, A., Lupu, E.C.: Detection of adversarial training examples in poisoning attacks through anomaly detection. arXiv preprint arXiv:1802.03041 (2018)
37. Paudice, A., Muñoz-González, L., Lupu, E.C.: Label sanitization against label flipping poisoning attacks. In: Alzate, C., et al. (eds.) ECML PKDD 2018. LNCS (LNAI), vol. 11329, pp. 5–15. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-13453-2_1
38. General Data Protection Regulation: Regulation (EU) 2016/679 of the European parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. Off. J. Eur. Union (OJ) **59**(1–88), 294 (2016)
39. Rubinstein, B.I., et al.: Antidote: understanding and defending against poisoning of anomaly detectors. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, pp. 1–14 (2009)
40. Ryffel, T., et al.: A generic framework for privacy preserving deep learning. arXiv preprint arXiv:1811.04017 (2018)
41. Schlesinger, A., O'Hara, K.P., Taylor, A.S.: Let's talk about race: identity, chatbots, and AI. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–14 (2018)
42. Shafahi, A., et al.: Poison frogs! Targeted clean-label poisoning attacks on neural networks. In: Advances in Neural Information Processing Systems, pp. 6103–6113 (2018)
43. Shen, S., Tople, S., Saxena, P.: Auror: Defending against poisoning attacks in collaborative deep learning systems. In: Proceedings of the 32nd Annual Conference on Computer Security Applications, pp. 508–519 (2016)
44. Steinhardt, J., Koh, P.W.W., Liang, P.S.: Certified defenses for data poisoning attacks. In: NeurIPS, pp. 3517–3529 (2017)
45. Suciu, O., Marginean, R., Kaya, Y., Daume III, H., Dumitras, T.: When does machine learning fail? Generalized transferability for evasion and poisoning attacks. In: 27th USENIX Security Symposium, pp. 1299–1316 (2018)
46. Sun, Z., Kairouz, P., Suresh, A.T., McMahan, H.B.: Can you really backdoor federated learning? arXiv preprint arXiv:1911.07963 (2019)
47. Truex, S., Liu, L., Gursoy, M.E., Yu, L., Wei, W.: Towards demystifying membership inference attacks. arXiv preprint arXiv:1807.09173 (2018)
48. Truex, S., Liu, L., Gursoy, M.E., Yu, L., Wei, W.: Demystifying membership inference attacks in machine learning as a service. IEEE Trans. Serv. Comput. (2019)
49. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
50. Xiao, H., Xiao, H., Eckert, C.: Adversarial label flips attack on support vector machines. In: ECAI, pp. 870–875 (2012)
51. Xiao, H., Biggio, B., Brown, G., Fumera, G., Eckert, C., Roli, F.: Is feature selection secure against training data poisoning? In: International Conference on Machine Learning, pp. 1689–1698 (2015)
52. Xiao, H., Biggio, B., Nelson, B., Xiao, H., Eckert, C., Roli, F.: Support vector machines under adversarial label contamination. Neurocomputing **160**, 53–62 (2015)

53. Yang, C., Wu, Q., Li, H., Chen, Y.: Generative poisoning attack method against neural networks. arXiv preprint arXiv:1703.01340 (2017)
54. Yang, G., Gong, N.Z., Cai, Y.: Fake co-visitation injection attacks to recommender systems. In: NDSS (2017)
55. Yin, D., Chen, Y., Kannan, R., Bartlett, P.: Byzantine-robust distributed learning: towards optimal statistical rates. In: International Conference on Machine Learning, pp. 5650–5659 (2018)
56. Zhao, L., et al.: Shielding collaborative learning: mitigating poisoning attacks through client-side detection. IEEE Trans. Dependable Secure Comput. (2020)
57. Zhao, M., An, B., Gao, W., Zhang, T.: Efficient label contamination attacks against black-box learning models. In: IJCAI, pp. 3945–3951 (2017)
58. Zhu, C., Huang, W.R., Li, H., Taylor, G., Studer, C., Goldstein, T.: Transferable clean-label poisoning attacks on deep neural nets. In: International Conference on Machine Learning, pp. 7614–7623 (2019)
59. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Advances in Neural Information Processing Systems, pp. 14747–14756 (2019)