DeFL: Defending against Model Poisoning Attacks in Federated Learning via Critical Learning Periods Awareness

Gang Yan[1], **Hao Wang**[2], **Xu Yuan**[3], **Jian Li**[1]
[1]SUNY-Binghamton University
[2] Louisiana State University
[3]University of Louisiana at Lafayette
gyan2@binghamton.edu, haowang@lsu.edu, xu.yuan@louisiana.edu, lij@binghamton.edu

Abstract

Federated learning (FL) is known to be susceptible to model poisoning attacks in which malicious clients hamper the accuracy of the global model by sending manipulated model updates to the central server during the FL training process. Existing defenses mainly focus on Byzantine-robust FL aggregations, and largely ignore the impact of the underlying deep neural network (DNN) that is used to FL training. Inspired by recent findings on critical learning periods (CLP) in DNNs, where small gradient errors have irrecoverable impact on the final model accuracy, we propose a new defense, called *a CLP-aware defense against poisoning of FL* (`DeFL`). The key idea of `DeFL` is to measure fine-grained differences between DNN model updates via an easy-to-compute federated gradient norm vector (`FGNV`) metric. Using `FGNV`, `DeFL` simultaneously detects malicious clients and identifies CLP, which in turn is leveraged to guide the adaptive removal of detected malicious clients from aggregation. As a result, `DeFL` not only mitigates model poisoning attacks on the global model but also is robust to detection errors. Our extensive experiments on three benchmark datasets demonstrate that `DeFL` produces significant performance gain over conventional defenses against state-of-the-art model poisoning attacks.

Introduction

Federated learning (FL) (McMahan et al. 2017) is an emerging distributed learning paradigm that enables many clients to collaboratively learn a deep neural network (DNN) model (called the *global model*) without sharing their private local training data. This is done through an iterative process where a *central server* repeatedly coordinates dispersed clients via collecting clients' local model updates computed on their local data, aggregating clients' updates using an *aggregation rule*, and finally using the aggregated updates to tune the global model, which is broadcast to a subset of clients at the beginning of each FL training round.

However, due to its distributed nature, FL is vulnerable to model poisoning attacks (Imteaj et al. 2023), which attempt to degrade the global model accuracy by contributing malicious model updates during the training process. Depending on the adversarial goal, model poisoning attacks can be either *untargeted* (Blanchard et al. 2017; El El Mhamdi, Guerraoui, and Rouault 2018; Mahloujifar, Mahmoody, and Mohammed 2019; Baruch, Baruch, and Goldberg 2019; Fang et al. 2020; Xie, Koyejo, and Gupta 2020; Shejwalkar and Houmansadr 2021), where the goal is to minimize the global model accuracy on *any* test input, or *targeted* (Bhagoji et al. 2019; Sun et al. 2019; Bagdasaryan et al. 2020), where the goal is to minimize the accuracy on *specific* test inputs. To this end, untargeted model poisoning attacks can completely cripple the global model and hence pose more severe threats to FL, which is the focus of this paper.

Existing defenses against model poisoning attacks mainly rely on robust methods, where the central server leverages *Byzantine-robust aggregation rules* to reduce the impact of malicious model updates via either detecting and removing *statistical outliers* or limiting their impacts. However, these approaches suffer from several key limitations: i) only robust when there is a smaller number of malicious clients (Blanchard et al. 2017; Yin et al. 2018); ii) prediction based malicious client detection using historical update information is accurate when there is a larger number of malicious clients in each training round (Zhang et al. 2022); and iii) require the central server to have access to a clean validation dataset whose distribution does not diverge too much from the overall training data distribution (Li et al. 2020; Cao et al. 2021). As a result, existing defenses are shown (Bhagoji et al. 2019; Fang et al. 2020) to be still vulnerable to model poisoning attacks, especially to those with high attack impacts.

Exacerbating these limitations is the fact that existing defenses mainly focus on designing Byzantine-robust aggregation rules, and largely ignore the impact of the underlying DNN that is used to training FL models since existing defenses implicitly assume that all FL training phases are equally important. Unfortunately, this assumption is invalid due to the existence of *critical learning periods* (CLP), i.e., the final quality of a DNN model is determined by the first few training rounds, in which deficits such as low quality or quantity of training data will cause irreversible model degradation (Achille, Rovere, and Soatto 2019; Jastrzebski et al. 2019; Golatkar, Achille, and Soatto 2019; Jastrzebski et al. 2021; Yan, Wang, and Li 2022).

To address the above limitations, we propose `DeFL`, a novel CLP-aware defense against model poisoning attacks to FL that seamlessly analyzes the DNN that is used for FL training and leverages its structural information to detect malicious model updates and identify CLP. Specifically, we propose a new easy-to-compute federated gradient norm vector (`FGNV`) metric to analyze the internal structure of the DNN and measure fine-grained differences between model updates. Using `FGNV`, we develop lightweight approaches to not only identify CLP but also reliably detect statistical outliers and consequently exclude them from aggregation in an adaptive manner. Our extensive evaluations using three real-world datasets, CIFAR-10 (Krizhevsky and Hinton 2009), MNIST and Fashion-MNIST (LeCun et al. 1998), show that `DeFL` dramatically mitigates the impacts of state-of-the-art model poisoning attacks (Baruch, Baruch, and Goldberg 2019; Fang et al. 2020; Shejwalkar and Houmansadr 2021). Furthermore, `DeFL` can effectively detect statistical outliers and is also robust to detection errors since our proposed `FGNV` is able to capture the fine-grained differences in the structure of the DNN that is used for FL training.

In summary, we make the following contributions:

- We propose an easy-to-compute federated gradient norm vector metric to analyze the internal structure of the DNN that is used for FL training and measure fine-grained differences between model updates. Using this metric, we design a threshold-based rule to identify the CLP, and build a voting-based statistical outlier detector, being capable to labeling model updates as benign or malicious.
- We propose `DeFL`, a novel CLP-aware defense against model poisoning attacks to FL. `DeFL` adaptively removes malicious clients from model aggregation in each FL training round via holistically combining the CLP and detector. To the best of our knowledge, this is the first work that conducts a deep DNN model inspection on each layer to mitigate model poisoning attacks to FL.
- We empirically evaluate the performance and effectiveness of `DeFL` against several state-of-the-art model poisoning attacks on three benchmark datasets. Our extensive results show that `DeFL` is up to 12× more effective against these model poisoning attacks compared to state-of-the-art defenses.

Background

Federated Learning

FL leverages a large set of clients $\mathcal{N} = \{1, \cdots, N\}$ to collaboratively learn a model with decentralized data under the coordination of a central server. Formally, the goal of FL is to solve the following optimization problem

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) := \sum_{i\in\mathcal{N}} p_i F_i(\mathbf{w}), \tag{1}$$

where $F_i(\mathbf{w}) = \frac{1}{|\mathcal{D}_i|}\sum_{\xi\in\mathcal{D}_i} \ell_i(\mathbf{w};\xi)$ is the local loss function associated with client i's dataset $\mathcal{D}_i$, $p_i = |\mathcal{D}_i|/\sum_i |\mathcal{D}_i|$ is the relative sample size. The training process is orchestrated by repeating the following two steps in each round t:

• **Local training**. The central sever randomly selects a set of clients $\mathcal{N}(t)$ to participate the training in round t. For ease of illustration, let $|\mathcal{N}(t)| = n, \forall t$. Each client $i \in \mathcal{N}(t)$ pulls the latest global model $\mathbf{w}_i(t-1)$ from the central sever, and then performs the local updates $\mathbf{w}_i^k(t) \leftarrow \mathbf{w}_i^{k-1}(t-1) - \eta\mathbf{g}(\mathbf{w}_i^{k-1}(t-1), \mathcal{D}_i)$, where η is the learning rate and $k = 1, \cdots, K$ is the index of local iterations.

• **Model aggregation**. Participants in round t push their local updated models to the central sever, which aggregates local models to obtain a new global model $\mathbf{w}(t)$: $\mathbf{w}(t) \leftarrow \mathcal{H}(\mathbf{w}_1^K(t), \cdots, \mathbf{w}_i^K(t), \cdots, \mathbf{w}_n^K(t))$, where $\mathcal{H}$ is the aggregation rule, e.g., the most widely used federated averaging (FedAvg) (McMahan et al. 2017) performs a weighted average as $\mathbf{w}(t) \leftarrow \sum_{i\in\mathcal{N}(t)} \frac{|\mathcal{D}_i|}{|\cup_{i\in\mathcal{N}(t)}\mathcal{D}_i|}\mathbf{w}_i^K(t)$.

Critical Learning Periods

The first few training epochs—known as *critical learning periods* (CLP)—have been revealed to determine the final quality of a DNN model in both traditional centralized learning (Achille, Rovere, and Soatto 2019; Jastrzebski et al. 2019; Golatkar, Achille, and Soatto 2019; Frankle, Schwab, and Morcos 2020; Jastrzebski et al. 2021) and FL (Yan, Wang, and Li 2022). During the CLP, deficits such as low quality or quantity of training data will cause irreversible model degradation, no matter how much additional training is performed after the period. However, studying critical learning phenomena hinged on costly information metric (e.g., eigenvalues of the Hessian) that emerges after the full training, limiting their practical benefits. We differ from existing works by developing an easy-to-compute metric to identify CLP during the training process in an online manner. Importantly, our new metric measures the fine-grained differences in the structure of the DNN, which can be easily leveraged to simultaneously detect malicious clients and identify CLP, which in turn is used to guide the removal of detected malicious clients from aggregation.

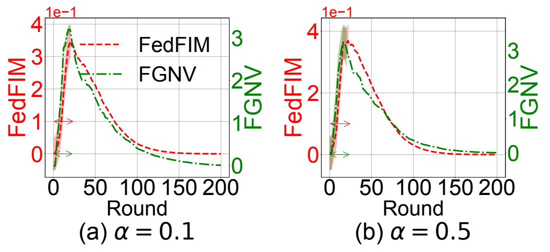
Byzantine-Robust Aggregation Rules

The mean aggregation rule has been widely used in non-adversarial settings (Dean et al. 2012; Konečný et al. 2016; McMahan et al. 2017), which, however, is not robust and can be manipulated by even a single malicious client (Blanchard et al. 2017; Yin et al. 2018; Bhagoji et al. 2019). Therefore, multiple Byzantine-robust aggregation rules have been proposed to defend against poisoning attacks. In the following, we review several representative Byzantine-robust aggregation rules that will be used in this paper.

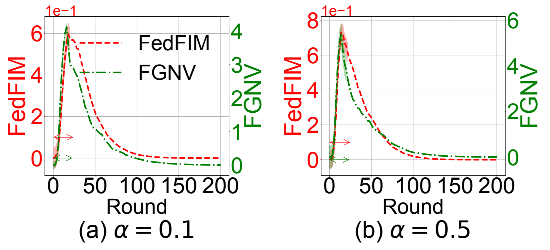
• **FLDetector (Zhang et al. 2022)** defends FL via detecting and removing majority of malicious clients from the aggregation. Specifically, FLDetector detects malicious clients by measuring the consistency between the client's model update and the server's predicted model update based on historical model updates. As a result, a larger number of malicious clients is often needed to build up historical information so as to guarantee the prediction and detection accuracy.

• **FLTrust (Cao et al. 2021)** leverages the validation dataset on the central server to assign a trust score to each clients. Specifically, a local model update has a lower trust score if its update direction deviates more from that of the server model update calculated based on the validation dataset.

• **Adaptive federated average (AFA) (Muñoz-González, Co, and Lupu 2019)** first computes a weighted average of collected gradients in each communication round. Then it



(a) CIFAR-10



(b) Fashion-MNIST

Figure 1: Comparison of detecting CLP in federated settings using FGNV with $\delta = 0.05$ and FedFIM, where the shade and double-arrows indicate identified CLP. The results are conducted using AlexNet on (a) CIFAR-10 and (b) Fashion-MNIST datasets, which are partitioned across 32 clients using Dirichlet distributions $\text{Dir}_{32}(0.1)$ and $\text{Dir}_{32}(0.5)$, respectively.

computes cosine similarities between the weighted average and each of collected gradients. Finally, AFA discards gradients with similarities out of a range, which is a simple function of mean, median and standard deviation of similarities.

• **Multi-krum (Blanchard et al. 2017).** Krum (Blanchard et al. 2017) selects gradients from the set of its input gradients that is close to its $n - m - 2$ neighbor gradients in squared Euclidean norm space with m being an upper bound on the number of malicious clients and n being the number of participated clients in each FL training round. Multi-krum selects a gradient using Krum from a remaining set, adds it to a selection set and removes it from the remaining set.

• **Trimmed-mean (Yin et al. 2018; Xie, Koyejo, and Gupta 2018)** coordinate-wisely aggregates each dimension of input gradients separately. Specifically, for a given dimension j, the j-th parameters of n local models $\{\mathbf{g}_i^j\}_{i=1,\cdots,n}$, Trimmed-mean removes the largest and smallest β of them, and computes the mean of the remaining $n - 2\beta$ parameters as the j-th dimension of the global model.

DeFL

Federated Gradient Norm Vector

Existing defenses for detecting poisoned model updates are based on metrics that treat the DNN as *a black box*, e.g., cosine (Muñoz-González, Co, and Lupu 2019) or L_2-norm (Blanchard et al. 2017; Zhang et al. 2022). However, recent studies (Fung, Yoon, and Beschastnikh 2020; Rieger et al. 2022) showed that different DNN layers exhibit various vulnerabilities to model poisoning attacks and hence may play different roles in defending FL against model poisoning attacks. To this end, we design a new metric, called *Federated Gradient Norm Vector* (FGNV) that allows to analyze the DNN used for FL training, and measures fine-grained differences between model updates.

Specifically, we consider a DNN with L layers for FL training. Let $g_i(\mathbf{w}_i^j;\xi) = \frac{\partial}{\partial \mathbf{w}_i^j}\ell(\mathbf{w}_i^j;\xi)$ be client i's gradient update on layer $\forall j = 1, \cdots, L$ evaluated on ξ. After performing a step SGD on this sample, the training loss or *the global model update difference* of client i on layer j is $\Delta\ell_i^j = \ell(\mathbf{w}_i^j - \eta g_i(\mathbf{w}_i^j;\xi);\xi) - \ell(\mathbf{w}_i^j;\xi)$, which can be approximated by its gradient norm using Taylor expansion, i.e.,

$$\Delta\ell_i^j \approx -\eta\|g_i(\mathbf{w}_i^j;\xi)\|^2. \tag{2}$$

We call this as the $\texttt{FGNV}_i$ of client i on layer j, i.e., $\texttt{FGNV}_i^j := \Delta\ell_i^j$. Denote $\textbf{FGNV}_i = (\texttt{FGNV}_i^1, \cdots, \texttt{FGNV}_i^L)$ as the federated gradient norm vector of client i, which represents the global model update difference of client i over each layer of the DNN. Then the global model update difference over each layer l at round t can be approximated using the weighted average of $\texttt{FGNV}_i^j$ across all selected clients, i.e.,

$$\texttt{FGNV}^j(t) = \sum_{i\in\mathcal{N}(t)} \frac{|\mathcal{D}_i|}{\sum_{i\in\mathcal{N}(t)}|\mathcal{D}_i|}\texttt{FGNV}_i^j(t). \tag{3}$$

Detecting Critical Learning Periods

We develop a simple threshold-based rule to identify the CLP based on the FGNV as follows: if

$$\frac{\sum_{j=1}^L \texttt{FGNV}^j(t) - \sum_{j=1}^L \texttt{FGNV}^j(t-1)}{\sum_{j=1}^L \texttt{FGNV}^j(t-1)} \geq \delta, \tag{4}$$

then the current training round t is in CLP, where δ is the threshold used to declare CLP in federated settings. We set $\delta = 0.05$ as the default value in our experiments and will investigate its impact in Figure 3.

We compare the CLP identified by our FGNV approach with the federated Fisher information (FedFIM) approach in (Yan, Wang, and Li 2022). When training AlexNet on non-IID CIFAR-10 and Fashion-MNIST, we observe that these two approaches yield similar results as shown in Figure 1. However, our FGNV approach is much more computationally efficient (being orders of magnitude faster to compute) and can be easily leveraged for defending FL against model poisoning attacks in each round during the FL training process in an online manner.

Detecting Malicious Clients

In each round t, we identify malicious clients based on the FGNV discussed above. Specifically, we cast the malicious client detection as *a statistical outlier detection problem* (Zhang, Yuan, and Tzeng 2021). Different from existing defenses that consider the DNN as a black box, our FGNV measures fine-grained differences in each layer of the DNN. To this end, we assign the $\textbf{FGNV}_i$ to each client i as its feature vector in our statistical outline detection problem in each training round t, and develop a lightweight voting based method via leveraging a statistical methodology named massive unsupervised outlier detection (MOUD) (Azcorra et al. 2018) to determine if a client is an outlier or not.

The inputs to the MOUD algorithm are the $\texttt{FGNV}_i^j$ over each layer j of each client i. The output of the MOUD algorithm is the outlier in each layer. Generally, MOUD compares the similarity of each client i's global model update difference on layer j, i.e., $\texttt{FGNV}_i^j$ with respect to a reference observed from all participated clients, to determine if client i's update on layer j is an outlier. Specifically, the computation of MOUD in each round t involves the following steps: 1) Based on $\textbf{FGNV}_i, \forall i \in \mathcal{N}(t)$, generate reference vectors on each layer j as $\textbf{FGNV}^j := (\texttt{FGNV}_1^j, \cdots, \texttt{FGNV}_{|\mathcal{N}(t)|}^j)$; 2) Compute $\hat{\beta}_{i'} = \frac{Cov(\textbf{FGNV}_i^j, \texttt{FGNV}_{i'}^j)}{Var(\texttt{FGNV}_{i'}^j)}$ as the estimated slope of a simple linear regression model; 3) Define the index of client i on layer j as $I(\texttt{FGNV}_i^j, \textbf{FGNV}^j) = \frac{1}{|\mathcal{N}(t)|}\sum_{i'=1}^{|\mathcal{N}(t)|}\hat{\beta}_{i'}$ and output layer j of client i as an outlier if its value deviates significantly from others.

Based on the detection of each layer j of each client i, we further develop a simple voting method to determine if client i is an outlier or not. To reduce the detection errors and improve the robustness, we leverage an adaptive threshold (as the number of layers declared as outliers from MOUD) for voting. Specifically, we first set the threshold as L, i.e., if MOUD outputs all L layers of client i as outliers, then client i is declared as an outlier in round t. If no client is detected as malicious, then we set the threshold as $L-1$ and repeat the above process until at least one client is clarified as malicious. We call our detection method that couples a `FGNV` based MOUD with a voting strategy as `MOUD-Vote`.

The Design of **`DeFL`** Defense

Per our discussions on CLP, the final model accuracy will be permanently impaired if the global model is severely poisoned during the early training phase, no matter how much additional training is performed after the period (Achille, Rovere, and Soatto 2019; Yan, Wang, and Li 2022). Therefore, once the CLP is identified, our `DeFL` removes all detected malicious clients from the model aggregation. However, a false positive rate (FPR), i.e., benign clients may be falsely claimed as malicious clients, often occurs in existing detection methods, though our `MOUD-Vote` is shown to have a low FPR, see the Experiments Section. As a result, if `DeFL` strictly removes all detected malicious clients from the model aggregation throughout the training process, it may be of the detriment of the final model accuracy. To this end, we further use a Bayesian model to estimate the clients' probability to provide good model updates based on our detected CLP and `MOUD-Vote`. As inspired by (Muñoz-González, Co, and Lupu 2019), the ability of the clients to provide good model updates can be modeled as a Hidden Markov model. At each round t, the probability $p_i(t)$ that client i provides a good update is given as

$$p_i(t) = \frac{\alpha_i(t)}{\alpha_i(t) + \beta_i(t)}, \qquad (5)$$

Algorithm 1: The `DeFL` defense

```
Input: $\mathbf{w}_i(0), \forall i$;
Initialize $\alpha_i(0) = \beta_i(0) = 1, \forall i$;
 1: for $t = 1, \cdots, T$ do
 2:     //Compute federated gradient norm vector FGNV
 3:     Perform local model updates $\mathbf{w}_i(t)$ and compute the global model update difference of each client $i$ on each layer $j$ as in (2) to generate $\textbf{FGNV}_i(t) = (\texttt{FGNV}_i^1(t), \cdots, \texttt{FGNV}_i^L(t)), \forall i \in \mathcal{N}(t)$;
 4:     //Detect malicious client using MOUD-Vote
 5:     for $i = 1, \cdots, \mathcal{N}(t)$ do
 6:         if MOUD-Vote claims $i$ as a benign client then
 7:             $\alpha_i(t) = \alpha_i(t-1) + 1$;
 8:         else
 9:             $\beta_i(t) = \beta_i(t-1) + 1$;
10:         end if
11:     end for
12:     //Detect CLP and update clients' aggregation weight
13:     if Round $t$ is in CLP then
14:         The aggregation weight for the detected malicious client $i$ is $p_i(t) = 0$;
15:         The aggregation weight for the detected benign client $i$ is $p_i(t) = \frac{\alpha_i(t)}{\alpha_i(t)+\beta_i(t)}$;
16:     end if
17:     //Model aggregation
18:     Obtain a new global model $\mathbf{w}(t)$ via local model update aggregation:
            $$\mathbf{w}(t) \leftarrow \sum_{i \in \mathcal{N}(t)} p_i(t) \frac{|\mathcal{D}_i|}{|\cup_{i \in \mathcal{N}(t)} \mathcal{D}_i|} \mathbf{w}_i(t).$$
19: end for
```

where α_i and β_i are the parameters of a Beta distribution, and $\alpha_i(t) = \alpha_i(t-1) + 1$ if client i is claimed as a benign client (i.e., provides good update) by our `MOUD-Vote` in round t; otherwise, $\beta_i(t) = \beta_i(t-1) + 1$. We summarize our `DeFL` defense in Algorithm 1.

From a high-level perspective, `DeFL` strictly removes all detected malicious clients from the model aggregation during the initial phase of the learning procedure (i.e., the detected CLP) to avoid poisoning attacks on the global model since the initial learning phase plays a critical role in FL performance. However, the malicious client detection may suffer from FPR, and removing too many clients from global model aggregation may degrade the final model accuracy. To address these issues and improve the robustness of `DeFL` against FPR, we augment `DeFL` with a Bayesian model to learn to associate a "good" update probability to each client. Roughly speaking, after the CLP, a smaller aggregation weight is associated with detected malicious clients in the global model aggregation rather than completely removing them. Likewise, a larger aggregation weight is assigned to detected benign clients in the global model aggregation after the CLP. As a result, `DeFL` is able to consistently defend FL against model poisoning attacks and is robust to the FPR of the detection method.

Dataset (Model)	Aggregation Rule	Fang		LIE		Min-Max		Min-Sum	
		Full	Partial	Full	Partial	Full	Partial	Full	Partial
CIFAR-10 (AlexNet)	`DeFL`	**7.36**	**5.37**	**7.97**	**4.62**	**7.52**	**7.48**	**9.83**	**8.51**
	FLDetector	18.34	16.9	13.0	11.27	18.29	17.85	17.51	16.11
	FLTrust	9.37	13.84	11.84	9.39	26.87	15.52	21.57	22.5
	AFA	11.67	18.83	13.55	18.12	35.41	33.4	43.59	29.48
	Multi-krum	33.85	29.38	34.41	32.11	37.28	34.7	40.61	32.25
	Trimmed-mean	38.71	36.11	40.56	34.28	45.04	35.68	44.58	36.36
CIFAR-10 (VGG-11)	`DeFL`	**4.25**	**4.26**	**3.98**	**1.47**	**4.19**	**1.92**	**8.83**	**8.91**
	FLDetector	13.47	13.17	13.81	8.68	7.61	4.88	20.88	13.84
	FLTrust	8.62	8.14	7.72	7.78	24.83	15.73	13.21	14.97
	AFA	13.21	11.31	9.81	19.06	42.43	33.41	34.66	30.07
	Multi-krum	26.41	24.71	30.03	26.42	48.03	41.82	44.23	35.39
	Trimmed-mean	35.35	34.62	38.95	34.76	47.35	43.03	40.71	43.12
MNIST (FC)	`DeFL`	**0.95**	**0.87**	**1.01**	**1.03**	**0.93**	**0.91**	**0.78**	**0.85**
	FLDetector	3.18	2.92	2.98	3.17	2.14	2.17	2.32	2.39
	FLTrust	1.49	1.6	1.57	1.59	1.73	1.71	1.58	1.8
	AFA	3.09	2.33	2.43	2.15	2.99	2.57	2.76	3.18
	Multi-krum	9.99	8.69	13.89	10.6	13.0	7.63	11.6	6.82
	Trimmed-mean	5.5	5.44	10.34	6.24	6.43	4.86	6.32	4.66
Fashion MNIST (AlexNet)	`DeFL`	**1.76**	**0.96**	**2.8**	**1.98**	**3.73**	**4.34**	**1.8**	**1.96**
	FLDetector	10.97	13.27	10.04	13.64	14.28	11.64	15.9	10.52
	FLTrust	15.97	16.77	17.52	16.99	18.99	18.07	18.81	17.82
	AFA	10.89	11.56	11.05	13.3	24.75	19.28	21.81	25.95
	Multi-krum	39.15	38.96	35.77	24.83	52.04	44.42	51.04	38.88
	Trimmed-mean	42.99	43.24	45.95	43.26	53.35	42.0	48.93	44.99

Table 1: Attack impacts of state-of-the-art model poisoning attacks defended by our `DeFL` and state-of-the-art defenses when benign gradients are either known (*Full*) or unknown (*Partial*) to the adversary, using non-IID partitioned datasets with $\alpha = 0.1$.

Experiments

Experimental Setup

Datasets. We use CIFAR-10 (Krizhevsky and Hinton 2009), MNIST and Fashion-MNIST (LeCun et al. 1998) as evaluation datasets, which are widely used in prior works. We simulate the non-identically and independent distributed (non-IID) FL scenario by considering a heterogeneous partition for which the number of data points and class proportions are unbalanced. In particular, we simulate a heterogeneous partition into N clients by sampling $\boldsymbol{p}_i \sim \text{Dir}_N(\alpha)$, where α is the parameter of the Dirichlet distribution. The level of heterogeneity among local datasets across different clients can be reduced when α increases. We choose $\alpha = 0.1$ as the default parameter in our experiments as done in (Fang et al. 2020; Wang et al. 2020a,b; Cao and Gong 2022) and investigate its impact in Figure 5.

Machine learning models. We consider three representative DNN models: AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG-11 (Simonyan and Zisserman 2015) and a fully connected network (FC) with layer sizes $\{784, 512, 10\}$. In particular, we use AlexNet and VGG-11 as the global model architecture for CIFAR-10, FC for MNIST and AlexNet for Fashion-MNIST, respectively.

Baseline defenses and attacks. We consider the aforementioned state-of-the-art defenses: FLDetector, FLTrust, AFA, Multi-krum and Trimmed-mean, and the following four strongest model poisoning attacks in the literature, i.e., Fang (Fang et al. 2020), LIE (Baruch, Baruch, and Goldberg 2019), Min-Sum and Min-Max (Shejwalkar and Houmansadr 2021). Similar to (Baruch, Baruch, and Goldberg 2019; Fang et al. 2020; Shejwalkar and Houmansadr 2021), we consider two settings regarding the adversary's knowledge: (a) ***Full:*** the adversary knows the gradients of benign clients; and (b) ***Partial:*** the adversary is agnostic to the gradient updates shared by benign clients.

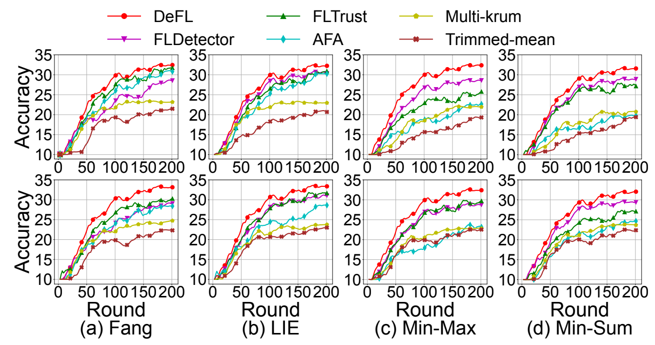


Figure 2: The global model accuracy when state-of-the-art model poisoning attacks are defended by our `DeFL` and state-of-the-art defenses using AlexNet on non-IID partitioned CIFAR-10 under (top) *Full* and (bottom) *Partial* scenarios.

• **Fang** is an optimization based model poisoning attack that can be tailored to aforementioned aggregation rules. Specifically, the adversary computes the average μ of benign gradients that she has access to. Then the adversary computes $-\text{sign}(\mu)$ and a malicious update by solving for a global coefficient λ. The adversary attacks all malicious clients and change their gradient updates' direction based on λ.

• **LIE** adds small amounts of noises to each dimension of the average of benign gradients. The small noises can be sufficiently large to adversely impact the global model and can be sufficiently small to evade detection by the Byzantine-robust aggregation rules. In particular, the adversary computes the average μ and standard deviation σ of benign gradients that she has access to. Furthermore, the adversary computes a coefficient z based on the total number of benign and malicious clients, and hence obtains the malicious update as $\mu + z\sigma$.

• **Min-Sum** ensures that the sum of squared distances of malicious gradients from all benign gradients is upper bounded by the sum of squared distances of any benign gradient from the other benign gradients. All malicious gradients are kept the same for the maximum attack impact.

• **Min-Max** computes malicious gradients such that its maximum distance from any other gradient is upper bounded by the maximum distance between any two benign gradients. As a result, the malicious gradients lie close to the clique of benign gradients.

Parameter settings. We implement our defenses, attacks and FL in PyTorch (Paszke et al. 2017) on Python 3 with three NVIDIA RTX A6000 GPUs. We run each experiments for 100 independent trials and report the average results. For ease of presentation, we omit the variances which are observed to be small in the experiments. By default, we consider a total number of $N = 128$ clients in our experiments. In each round, the FL central server randomly selects $n = 32$ clients to participate in the global model update, in which $m = 4$ are malicious clients. We investigate the impact of the number of malicious clients in Figure 4 and relegate additional experimental results with different number of malicious clients to (Yan et al. 2023).

Each client applies 20 iterations of the stochastic gradient descent to update its local model and the central server aggregates local model updates from all selected clients. We set 200 rounds for all DNN classifiers on all datasets considered in this paper. The local learning rate η is initialized as 0.01 and decayed by a constant factor after each communication round. The batch size is set to be 16. We set the weight decay to be 10^{-4}. The detection threshold δ is tunable parameters. We set $\delta = 0.05$ in all of our experiments and investigate its impact in Figure 3. The Trimmed-mean aggregation rule prunes the largest and smallest β parameters, where $m \leq \beta \leq n/2$. We set $\beta = m$ which is the default setting in Trimmed-mean (Yin et al. 2018).

Experimental Results

Reduced attack impact. We evaluate the performance of our DeFL defense against state-of-the-art model poisoning attacks and compare it with state-of-the-art defenses. The impacts of these attacks when defended by these defenses when benign gradients are either known or unknown to the adversary are summarized in Table 1. Due to space constraints, we only present the testing accuracy using AlexNet on non-IID partitioned CIFAR-10 with $\alpha = 0.1$ in Figure 2. Similar observations can be made in other cases and hence are relegated to (Yan et al. 2023).

Dataset (Model)	Attack	Full		Partial	
		TPR	FPR	TPR	FPR
CIFAR-10 (AlexNet)	Fang	1.0	0.02	0.97	0.02
	LIE	0.99	0.01	0.99	0.03
	Min-Max	0.98	0.03	0.98	0.02
	Min-Sum	0.98	0.02	1.0	0.02
CIFAR-10 (VGG-11)	Fang	0.97	0.08	0.99	0.07
	LIE	0.98	0.07	0.99	0.07
	Min-Max	0.98	0.08	1.0	0.07
	Min-Sum	1.0	0.07	1.0	0.05
MNIST (FC)	Fang	0.99	0.05	0.98	0.09
	LIE	0.99	0.04	0.98	0.08
	Min-Max	0.99	0.05	0.98	0.08
	Min-Sum	0.99	0.05	0.97	0.09
Fashion MNIST (AlexNet)	Fang	0.99	0.02	0.98	0.02
	LIE	1.0	0.04	0.98	0.03
	Min-Max	0.98	0.02	0.97	0.02
	Min-Sum	0.97	0.02	0.98	0.02

Table 2: The TPR and FPR of MOUD-Vote for model poisoning attacks against our DeFL under *Full* and *Partial* scenarios using non-IID partitioned datasets with $\alpha = 0.1$.

It is clear from Table 1 that our DeFL dramatically mitigates the impacts of these strongest model poisoning attacks in the literature, and DeFL is up to 12.04× more effective against these attacks than the best performing defenses in consideration. For example, when running AlexNet on CIFAR-10 with known benign gradients, the Min-Max attack has an attack impact of 7.52 when defended by DeFL, while the attack impact is 18.29 when defended by FLDetector, which is the best performing defense in consideration, i.e., our DeFL is 2.34× more effective. Take the AlexNet on Fashion-MNIST with unknown benign gradients as another example, the Fang attack has an attack impact of 11.56 when defended by AFA, while the attack impact is 0.96 when defended by DeFL, i.e., DeFL is 12.04× more effective.

We note that FLTrust requires the server to have access to a clean validation dataset, which is not the case in a typical FL scenario since the server often does not have such a dataset. FLDetector overcomes this limitation by detecting and removing malicious clients. However, the prediction-based detection requires historical information and hence is accurate when there is a large number of malicious clients participated in each FL training round. However, in practical FL systems, only a small fraction of clients is involved in each training round (Bonawitz et al. 2019). Our DeFL neither requires a validation dataset nor historical malicious client information, but instead leverages the internal structure of DNN and fine-grained model update differences via FGNV to improve the effectiveness of defending FL against model poisoning attacks in practical FL scenarios.

Effectiveness of FGNV. As motivated earlier, our DeFL is a CLP-awareness defense that analyzes the DNN via our proposed FGNV and leverages the structural information encoded in FGNV to detect malicious clients. To this end, we further understand the effectiveness of the information encoded in FGNV on the performance of CLP detection and malicious clients detection. As shown in Figure 1, our proposed threshold-based rule using FGNV is able to detect CLP

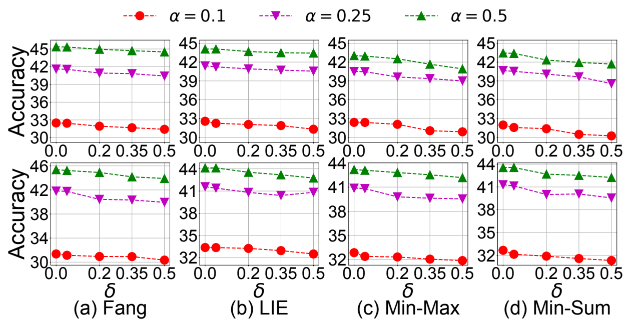


Figure 3: The impact of CLP detection threshold δ on global model accuracy when state-of-the-art attacks defended by DeFL using AlexNet on non-IID partitioned CIFAR-10 under (top) *Full* and (bottom) *Partial* scenarios.

as accurate as that of state-of-the-art approach, however, our FGNV based approach is much more computation efficient.

Table 2 illustrates the *true positive rate* (TPR) and *false positive rate* (FPR) of our MOUD-Vote, which couples FGNV with an unsupervised outlier detection method. TPR (FPR) is the fraction of malicious (benign) clients correctly (falsely) classified as malicious. We observe that our MOUD-Vote can consistently detect majority of malicious clients, and only a small fraction (up to 3%) of malicious clients are not detected, e.g., the TPR under Fang attack using VGG-11 on non-IID partitioned CIFAR-10 is 97%. In addition, MOUD-Vote falsely detects a small fraction of benign clients as malicious, i.e., the FPR of our MOUD-Vote ranges between 0.01 and 0.09. Importantly, our DeFL is robust to these TPR and FPR since DeFL not only removes detected malicious clients from the aggregation during the CLP, but also learns to estimate the aggregation weight for each client. These two techniques together contribute to the improvement of the effectiveness of our DeFL defenses.

Sensitivity of CLP detection threshold. We leverage FGNV to detect CLP via a threshold-based rule in (4). We now evaluate the sensitivity of the threshold value δ. We consider the candidate values of $\{0, 0.05, 0.2, 0.35, 0.5\}$. The global model accuracy when state-of-the-art attacks defended by DeFL using AlexNet on non-IID partitioned CIFAR-10 with different α is shown in Figure 3. As expected, as δ becomes larger, fewer rounds in the initial training phases are declared as CLP by (4). As a result, DeFL only completely removes detected malicious clients in fewer arounds according to Algorithm 1. However, the global model accuracy does not degrade significantly as δ becomes larger, this is due to the fact that DeFL is further augmented with a learning process to associate aggregation weight to malicious clients. For ease of simplicity, we set $\delta = 0.05$ in all of our experiments.

Impact of the number of malicious clients. Figure 4 shows the impact of the number of malicious clients, where we consider the ratio of malicious clients in each round as $\{12.5\%, 18.75\%, 25\%, 31.25\%\}$. We observe that the global model accuracy only drops slightly. This is because DeFL can efficiently detect malicious clients with consistently

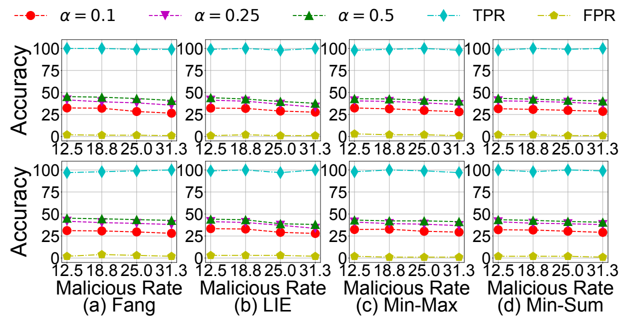


Figure 4: The impact of the number of malicious clients using AlexNet on non-IID partitioned CIFAR-10 under (top) *Full* and (bottom) *Partial* scenarios.

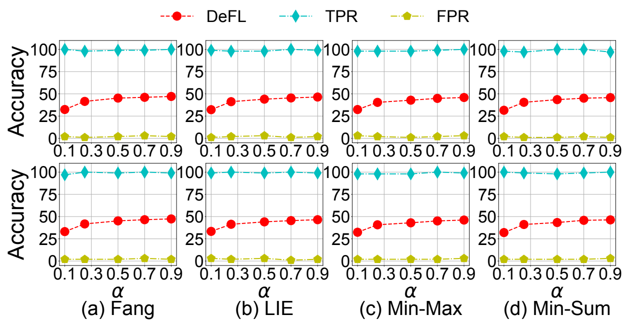


Figure 5: The effect of degree of non-IID nature of data on global model accuracy using AlexNet on CIFAR-10 under (top) *Full* and (bottom) *Partial* scenarios.

large TPR and small FPR as shown in Figure 4, and mitigate their impacts via CLP awareness.

Impact of non-IID degree of data distribution. We simulate a heterogeneous data partition into N clients using the Dirichlet distribution with parameter α. As shown in Figure 5, as the non-IID degree decreases (as α increases), the global model accuracy when state-of-the-art attacks defended by DeFL increases. This is intuitive since a lower degree of non-IID data makes the adversaries easier to be detected and removed from the aggregation. Again we observe that DeFL consistently has a large TPR and small FPR across different non-IID degrees.

Conclusion

In this paper, we proposed, DeFL, a CLP-aware defense against model poisoning attacks to FL. Different from existing defenses that mainly focused on designing robust aggregation rules, DeFL analyzed the underlying DNN used for FL training and measured fine-grained difference between DNN model updates via an easy-to-compute federated gradient norm vector. Using this metric, we designed simple rules to identify CLP and detect malicious clients, which are seamlessly integrated into DeFL to mitigate model poisoning attacks to FL. Our extensive evaluations showed that DeFL outperforms baseline defenses.

Acknowledgements

The work of G. Yan and J. Li was supported in part by the National Science Foundation (NSF) grants CRII-CNS-NeTS-2104880 and RINGS-2148309, and was supported in part by funds from OUSD R&E, NIST, and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program, as well as the DOE DE-EE0009341. The work of H. Wang was supported in part by the NSF grant CRII-OAC-2153502 and the LSU Provost's Fund for Innovation in Research–Faculty Travel Grant Program. The work of X. Yuan was support in part by the NSF grants 1763620, 1948374, and 2146447. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Achille, A.; Rovere, M.; and Soatto, S. 2019. Critical Learning Periods in Deep Networks. In *Proc. of ICLR*.

Azcorra, A.; Chiroque, L. F.; Cuevas, R.; Fernández Anta, A.; Laniado, H.; Lillo, R. E.; Romo, J.; and Sguera, C. 2018. Unsupervised scalable statistical method for identifying influential users in online social networks. *Scientific Reports*, 8(1): 1–7.

Bagdasaryan, E.; Veit, A.; Hua, Y.; Estrin, D.; and Shmatikov, V. 2020. How to backdoor federated learning. In *Proc. of AISTATS*.

Baruch, G.; Baruch, M.; and Goldberg, Y. 2019. A little is enough: Circumventing defenses for distributed learning. In *Proc. of NeurIPS*.

Bhagoji, A. N.; Chakraborty, S.; Mittal, P.; and Calo, S. 2019. Analyzing federated learning through an adversarial lens. In *Proc. of ICML*.

Blanchard, P.; El Mhamdi, E. M.; Guerraoui, R.; and Stainer, J. 2017. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proc. of NeurIPS*.

Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konečnỳ, J.; Mazzocchi, S.; McMahan, B.; et al. 2019. Towards federated learning at scale: System design. In *Proc. of MLSys*.

Cao, X.; Fang, M.; Liu, J.; and Gong, N. Z. 2021. FLTrust: Byzantine-robust Federated Learning via Trust Bootstrapping. In *Proc. of NDSS*.

Cao, X.; and Gong, N. Z. 2022. MPAF: Model Poisoning Attacks to Federated Learning based on Fake Clients. *arXiv preprint arXiv:2203.08669*.

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; Yang, K.; et al. 2012. Large scale distributed deep networks. In *Proc. of NIPS*.

El El Mhamdi, M.; Guerraoui, R.; and Rouault, S. 2018. The hidden vulnerability of distributed learning in byzantium. In *Proc. of ICML*.

Fang, M.; Cao, X.; Jia, J.; and Gong, N. 2020. Local Model Poisoning Attacks to Byzantine-Robust Federated Learning. In *Proc. of USENIX Security*.

Frankle, J.; Schwab, D. J.; and Morcos, A. S. 2020. The Early Phase of Neural Network Training. In *Proc. of ICLR*.

Fung, C.; Yoon, C. J.; and Beschastnikh, I. 2020. The limitations of federated learning in sybil settings. In *Proc. of USENIX RAID*.

Golatkar, A. S.; Achille, A.; and Soatto, S. 2019. Time Matters in Regularizing Deep Networks: Weight Decay and Data Augmentation Affect Early Learning Dynamics, Matter Little Near Convergence. *Proc. of NeurIPS*.

Imteaj, A.; Mamun Ahmed, K.; Thakker, U.; Wang, S.; Li, J.; and Amini, M. H. 2023. Federated Learning for Resource-Constrained IoT Devices: Panoramas and State of the Art. *Federated and Transfer Learning*, 7–27.

Jastrzebski, S.; Arpit, D.; Astrand, O.; Kerg, G. B.; Wang, H.; Xiong, C.; Socher, R.; Cho, K.; and Geras, K. J. 2021. Catastrophic Fisher Explosion: Early Phase Fisher Matrix Impacts Generalization. In *Proc. of ICML*.

Jastrzebski, S.; Kenton, Z.; Ballas, N.; Fischer, A.; Bengio, Y.; and Storkey, A. J. 2019. On the Relation Between the Sharpest Directions of DNN Loss and the SGD Step Length. In *Proc. of ICLR*.

Konečnỳ, J.; McMahan, H. B.; Yu, F. X.; Richtárik, P.; Suresh, A. T.; and Bacon, D. 2016. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv preprint arXiv:1610.05492*.

Krizhevsky, A.; and Hinton, G. 2009. Learning Multiple Layers of Features from Tiny Images. *Technical Report, University of Toronto*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet Classification with Deep Convolutional Neural Networks. *Proc. of NIPS*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, S.; Cheng, Y.; Wang, W.; Liu, Y.; and Chen, T. 2020. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211*.

Mahloujifar, S.; Mahmoody, M.; and Mohammed, A. 2019. Universal multi-party poisoning attacks. In *Proc. of ICML*.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proc. of AISTATS*.

Muñoz-González, L.; Co, K. T.; and Lupu, E. C. 2019. Byzantine-robust federated machine learning through adaptive model averaging. *arXiv preprint arXiv:1909.05125*.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Rieger, P.; Nguyen, T. D.; Miettinen, M.; and Sadeghi, A.-R. 2022. Deepsight: Mitigating backdoor attacks in federated learning through deep model inspection. In *Proc. of NDSS*.

Shejwalkar, V.; and Houmansadr, A. 2021. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In *Proc. of NDSS*.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-scale Image Recognition. In *Proc. of ICLR*.

Sun, Z.; Kairouz, P.; Suresh, A. T.; and McMahan, H. B. 2019. Can you really backdoor federated learning? *arXiv preprint arXiv:1911.07963*.

Wang, H.; Yurochkin, M.; Sun, Y.; Papailiopoulos, D.; and Khazaeni, Y. 2020a. Federated Learning with Matched Averaging. In *Proc. of ICLR*.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020b. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. *Proc. of NeurIPS*.

Xie, C.; Koyejo, O.; and Gupta, I. 2018. Generalized byzantine-tolerant sgd. *arXiv preprint arXiv:1802.10116*.

Xie, C.; Koyejo, O.; and Gupta, I. 2020. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Proc. of UAI*.

Yan, G.; Wang, H.; and Li, J. 2022. Seizing Critical Learning Periods in Federated Learning. In *Proc. of AAAI*.

Yan, G.; Wang, H.; Yuan, X.; and Li, J. 2023. DeFL: Defending Against Model Poisoning Attacks in Federated Learning via Critical Learning Periods Awareness. In https://www.dropbox.com/s/1ng5eoshh4gj2j9/DeFL-AAAI23.pdf?dl=0.

Yin, D.; Chen, Y.; Kannan, R.; and Bartlett, P. 2018. Byzantine-robust distributed learning: Towards optimal statistical rates. In *Proc. of ICML*.

Zhang, Y.; Yuan, X.; and Tzeng, N. 2021. Platform-Oblivious Anti-Spam Gateway. In *Proc. of ACSAC*.

Zhang, Z.; Cao, X.; Jia, J.; and Zhenqiang Gong, N. 2022. FLDetector: Defending Federated Learning Against Model Poisoning Attacks via Detecting Malicious Clients. In *Proc. of ACM SIGKDD*.