# FLAD: Byzantine-robust Federated Learning Based on Gradient Feature Anomaly Detection

Peng Tang, Xiaoyu Zhu, Weidong Qiu, Zheng Huang, Zhenyu Mu and Shujun Li, *Senior Member, IEEE*

***Abstract*—Federated Learning (FL) has gained significant attention due to its ability to jointly train global models by exchanging local gradients instead of raw local datasets. However, poisoning attacks have emerged as a severe threat to FL security, where malicious clients submit crafted gradients to compromise the integrity and availability of the model. Although researchers have worked on countering these attacks to achieve Byzantine-robust FL, it remains challenging to balance high accuracy, robustness, and efficiency simultaneously. We propose FLAD, a novel Byzantine-robust FL approach based on gradient feature anomaly detection, which is the first approach that uses neural networks to adaptively learn gradient features and measure feature similarity to counteract various types of poisoning attacks. Specifically, FLAD employs a small clean dataset to bootstrap trust and trains Feature Extraction Models (FEM). With FEM and DBSCAN clustering, abnormal gradients from malicious clients are detected and eliminated. Extensive experiments on both Non-IID and IID datasets demonstrate that FLAD achieves superior accuracy, robustness, efficiency, and generalizability compared to state-of-the-art approaches. Additionally, we implement privacy-preserving FLAD (PFLAD) using CKKS and Random Permutation techniques to ensure transmitted gradient privacy.**

***Index Terms*—Federated Learning, poisoning attack, byzantine-robust, anomaly detection.**

## I. Introduction

FEDERATED Learning (FL) has rapidly developed as a solution to the data island problem and gained widespread application. FL offers a distributed framework for data processing and machine learning [1]. Training data is distributed across various client devices (e.g., desktops, mobile phones, IoT devices) that aim to collectively learn a model without sharing their local data. Specifically, each client individually trains and updates a local model using its private data, then transmits pertinent local model variables to the server. The server aggregates these variables to maintain a global model and broadcasts the aggregated results to distributed clients. This process is repeated until model convergence [2], enabling collaborative learning across diverse client datasets.

Current researches on FL focus on protocol design [3], and the robustness of FL has not been fully studied. Although FL protects client's private data from direct transmission and shows great development potential, its distributed characteristics and protocol vulnerability also make it vulnerable to various attacks. Model poisoning attack is the common attacks [3]. For model poisoning attack, any malicious client can mislead global model learning process by sending carefully constructed malicious gradients to server, resulting in model classification errors. A poisoned gradient may even control the entire training process of the model, ultimately invalidating the joint learning model [4]. Therefore, a trustworthy FL framework must can effectively prevent model poisoning attacks on global model. Our work will focus on the global model security to propose a Byzantine-robust FL defense against poisoning attacks.

Existing mainstream Byzantine-robust methods are defended by statistical analysis of the data, i.e., the server discriminates malicious gradients based on the statistical similarity of local gradients [5]. However, due to the lack of proper trust bootstrapping in these approaches, the server is prone to identify malicious gradients purposely constructed based on aggregation rules as honest gradients thus misleading the global model learning. FLTrust addresses this deficiency by bootstrapping trust by building server models using trusted datasets on the server [6]. However, FLTrust still suffers from a single way of extracting trust, general and limited features extracted, and modified malicious gradients are still involved in aggregation leading to slow convergence in model training. Moreover, there are other Byzantine-robust FL schemes that center on clipping the magnitude of gradients and adding noise, which also has implications for the accuracy of honest gradients [7], [8].

**Our work.** We propose FLAD, a Byzantine-robust FL framework against model poisoning attacks based on anomaly detection. Following the FLTrust's idea [6], FLAD assumes that server has a small clean dataset (called the server dataset) to bootstrap trust. FLAD uses neural network-based feature extraction models (FEMs) to build an anomaly detector based on a clustering model with the clean server dataset's features as criteria to identify malicious gradients. Honest gradients are recognized since their features (extracted by the FEMs) closely align with server dataset criteria. Conversely, distant features indicate poisoned gradients from malicious clients. Specifically, the server uses FEM trained on server dataset for feature extraction, yielding reduced-dimensional vectors for each client's gradients. Effective differentiation of malicious gradients is achieved by clustering algorithm using Euclidean distance, cosine similarity and relative length difference compared to standard features. Furthermore, we combine CKKS homomorphism [9] and random permutation to ensure data privacy during FL transmission, called PFLAD.

Our main contributions are summarized below.

P. Tang, X. Zhu, W. Qiu, Z. Huang and Z. Mu are School of Cyber Science and Engineering, with Shanghai Jiao Tong University, Shanghai, 200240, China.

S. Li is with School of Computing & Institute of Cyber Security for Society (iCSS), University of Kent, Canterbury, CT2 7NP, United Kingdom.

Corresponding authors: Xiaoyu Zhu (zhux1aoyu@sjtu.edu.cn) and Weidong Qiu (qiuwd@sjtu.edu.cn).

- We propose FLAD, a Byzantine-robust FL framework based on anomaly detection. FLAD is the first defense approach that uses neural networks to adaptively learn gradient features and measure gradient similarity to counteract various types of poisoning attacks. We have open-sourced FLAD[1].
- FLAD rejects malicious gradients, truly achieving aggregation only for honest gradients. Therefore, attackers' number and the type of attacks can change dynamically during training. FLAD is fully applicable in scenarios with a substantial number and even can detect over half of attackers for poisoning attacks.
- FLAD is comprehensively assessed across diverse Non-IID and IID datasets, various attacks and defense methods, etc. Experimental results reveal FLAD's high global accuracy and strong robustness against a range of poisoning attacks, including adaptive attack.
- Furthermore, we implement a privacy-preserving FLAD (PFLAD) model attack defense scheme using CKKS and Random Permutations to ensure transmitted gradient privacy. Therefore, PFLAD achieves both Byzantine-robustness and privacy-preserving, making it suitable for environments with unreliable servers and clients.

## II. Related Work

### A. Poisoning Attacks to FL

Poisoning attacks generally refer to attacking the training phase of the global model and misleading its learning process, thus invalidating the final learned model [3], which are divided into targeted and untargeted attacks.

**Untargeted Attacks.** An untargeted poisoning attack's objective is to arbitrarily compromise the integrity of target model [3]. An *Byzantine attack* is an untargeted poisoning attack that uploads arbitrary carefully constructed malicious gradients to server, which leads to global model training failure. Common untargeted attacks also encompass *Gaussian Attack* [10], *Zero-gradient Attack* [11], *Sign-flipping Attack* [12] and *MPAF Attack* [13]. Moreover, *Adaptive Local Model Untargeted Poisoning Attacks* [14], [15] have been proposed, which treat an attack as an optimization problem, and can cause a higher error rate to the global model.

**Targeted Attacks.** A targeted attack tries to let the attacked model output attacker-specified target labels for specific inputs when one or more triggers are present, while keeping the model's outputs for other clean samples unaffected. Targeted attacks generally can be achieved by poisoning local datasets and extending malicious local gradients to implant distributed or centralized backdoor in global model. Common targeted poisoning attacks encompass *Label-flipping Attack* [16], [17], *Model-replacement Attack* [18], [19] and *Distributed Backdoor Attack* (DBA) [20]. Label-flipping attack is the process of flipping the labels of normal examples to target labels, but the original features of the data remain unchanged. Backdoor poisoning attack requires the attacker to modify individual features or regions of the original dataset in order to implant backdoor triggers in the data so that the final learning model on clean data can function properly, but will constantly predict the attacker's chosen target label when data containing backdoor triggers are present.

### B. Byzantine-robust FL

Currently, state-of-the-art defenses against FL poisoning attacks can be classified into three main categories.

**Robust Aggregation Rules.** Traditional FL adopts FedAvg [21] aggregation, where the server broadcasts weighted averages of local gradients as global model to clients. However, FedAvg is highly susceptible to malicious gradients [14]. Therefore, various Byzantine-robust rules have been proposed, including *Krum* [22], *Trimmed-mean*, *Median* [23], *Bulyan* [24], *DETOX* [25] and *RFA* [26]. While most of them are vulnerable to poisoning attacks in specific scenarios, they incur substantial communication overhead [14], [27]. Moreover, rules such as Krum and Bulyan have explicit restrictions on the proportional threshold of malicious attackers.

**Correction of Local/Global Gradients.** The server's assessment of honesty and malice relies solely on received local gradients during the FL training. Therefore, numerous methods focus on either improving local gradients sent to the server or preprocessing local gradients and clipping global gradients at the server, such as Byrd-SAGA [11], signSGD [28], FLTrust [6], CRFL [7] and Attack-adaptive Aggregation [29]. Byrd-SAGA adjusts local stochastic gradients via averaging before transmission to the server, diminishing noise in honest gradients. In signSGD, clients only need to transmit the sign of their local gradient vectors to the server, and the global model update is determined by majority voting. FLTrust requires server to preprocess the uploaded local gradients by normalization, and then aggregates them according to the weighted average of the trust scores. CRFL achieves the effectiveness and robustness of the global model by clipping and stochastic gradient smoothing of the aggregated global gradients. Attack-adaptive Aggregation dynamically adjusts aggregation weights by training neural networks with attention mechanisms to identify and mitigate potential attacks on federated learning models.

**Malicious Node Detection and Rejection.** Server can avert contamination of global model by rejecting malicious gradients before aggregation. The research [14] has generalized RONI [30] and TRIM [31] to propose ERR (Error Rate based Rejection) and LFR (Loss Function based Rejection) from the actual effect of the global model, which select the local updates that optimize global model performance to aggregate. However, both schemes are not effective in some scenarios [14]. TRIM [31] is a poisoning attack defense for regression learning, which uses a trimmed loss function to determine training data points with minimal loss relative to the model. Although such defenses based on the actual effect of model can bring high accuracy, the high computational overhead incurred during trials cannot be ignored. Moreover, FLDetector [32] is a method designed to detect malicious clients. It primarily leverages the Cauchy Mean Value Theorem and the L-BFGS algorithm to analyze the consistency of

[1][Online]. Available: https://github.com/tp-sh/FLAD

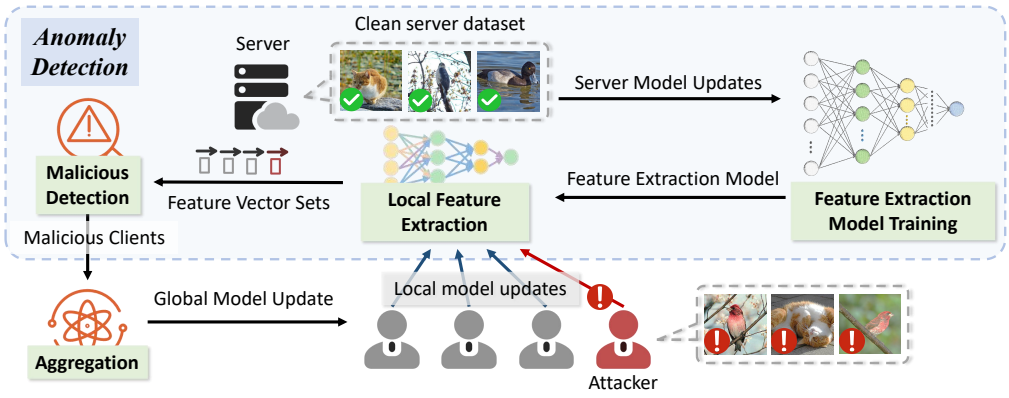



Fig. 1. The framework of FLAD.

clients' model updates, thereby identifying potential malicious clients. AUROR [33] uses the K-means to remove local parameters that exceed the threshold based on the difference between the honest and malicious parameter distributions. Relative Weight Distance (RWD) [34] uses relative distance between local gradients and threshold to detect malicious gradients. FLAME [8] uses cosine similarity-based HDBSCAN to initially filter out malicious updates. DeepSight [35] introduces various metrics to measure similarity between local updates, thus performing poisoning filtering on local models by classification and clustering. However, in such threshold-based defenses, the detection of malicious client is highly dependent on the setting of threshold.

## III. System and Problem Setting

### A. Threat Model

Referring to the related work [6], malicious attackers can fully control poisoned clients, i.e., attackers can control the local dataset and the process of local training. These poisoned clients are not required to follow FL protocols and can collude with each other. Furthermore, honest clients and server perform normal operations required by FL protocols and attackers cannot control any normal operation of them. Adaptive local poisoning attack is stronger in the full-knowledge setting [14]. Therefore, we consider this setting to evaluate FLAD in Section V.

### B. Defense Goals

A reliable Byzantine-robust FL defense must can effectively prevent malicious clients from poisoning global model. Generally, the FL defense should contain four defensive goals:

- **Accuracy**. In non-adversarial settings, the FL defense should achieve a global model's accuracy equivalent to that learned by FedAvg.
- **Robustness**. In the adversarial settings, defense should ensure that the global model is not misled by attackers and global model's accuracy can be similar to FedAvg without attacks, rendering the poisoning attack invalid.
- **Efficiency**. In contrast to FedAvg under non-adversarial settings, defense should avoid increasing client workload and client-server communication overhead significantly.
- **Generalizability**. Defense needs to be applicable to federated systems with different scenarios, such as different scales of client numbers, communication rounds, and varying poisoning ratios, among others.

### C. Defender's Knowledge and Capability

Defense must operates on the server, which lacks access to clients' private datasets, local training processes, and information about the number of malicious clients. However, the server has access to the global model at each interaction. Moreover, following the idea of FLTrust [6], FLAD assumes that server has a small clean dataset $D_s$ (called *server dataset*) to extract honest gradients' features, thus bootstrapping local gradients' trust. Our experiments indicated that FLAD can maintain high model accuracy even if $D_s$ comprises only 0.5% of training data or has a distribution bias of 0.8, as detailed in Section V. Therefore, server can obtain a less demanding $D_s$ with minimal effort, such as via manual curation and labeling [6].

### D. System Setting

We assume that the number of clients is $n$ and the number of malicious attackers is $c$. Without loss of generality, let us assume that the indices of the $n-c$ honest clients are $1 \sim (n-c)$ and those of malicious clients are $(n-c+1) \sim n$. To ensure the model's usability, the number of honest clients should be greater than two, which can be easily achieved in almost all real-world FL applications. In the $t$th iteration, the local gradients received by the server is $\{w_1^t; w_2^t; \cdots; w_n^t\}$ and the global gradients of the aggregated model are $W^t$.

## IV. Our FLAD Method

### A. Overview of FLAD

Given the significant distinctions between gradients from malicious and honest clients [5], the server can construct an anomaly detector using the server dataset to extract features from local gradients and categorize them as honest or malicious. The anomaly detector comprises two components. The first component is for feature extraction, wherein multiple neural networks (NNs) are trained as feature extraction models

(FEMs) on the server dataset. These FEMs are used to extract features from local gradients, and feed such features to the second component that checks the extracted features against benchmark ones extracted from honest gradients in the server dataset. The server evaluates the similarity between feature vectors by DBSCAN [36], enabling effective differentiation and rejection of malicious gradients before aggregation. Figure 1 shows the framework of FLAD.

### B. Anomaly Detector

*1) Feature Extraction:* Before malicious detection on local gradients, the server trains multiple FEMs with the server dataset $D_s$ and uses such models to extract features from local gradients. The server uses $D_s$ to bootstrap trust instead of relying entirely on local gradients.

**Algorithm 1** Training an FEM

**Input:** $W^{t-1}$: Global model gradients at $(t-1)$th round;
$D_s$: The server dataset for bootstrapping trust;
**Output:** $F_{fc}^t$: A feature extraction model in the $t$th round;
eps: Deviation values for honest gradients;
$\mu$: Standard feature for honest gradients;

**Step 1.** Obtain the training set:
Initialize the training set $W_s^t$;
**for** $i = 1$ **to** num **do**
  $w_{C_i}^t \leftarrow W^{t-1}$;
  **for** epoch $e = 1$ **to** $E$ **do**
    $w_{C_i}^t \leftarrow w_{C_i}^t - \alpha \times \nabla l(b_s, w_{C_i}^t)$; % $b_s$: batch size
  **end for**
  $W_s^t[i] \leftarrow w_{C_i}^t$;
**end for**
**Step 2.** Train the FEM:
Initialize the FEM $F_{fc}^t$;
**for** epoch $e = 1$ **to** $E_s$ **do**
  **for** $i = 1$ **to** num **do**
    $w_{C_i}^t \leftarrow W_s^t[i]$; $w_{C_i}^t \cdot \text{label} = [1] \in \mathbb{R}^1$;
    $w_{C_i}^t \leftarrow$Flatten($w_{C_i}^t$);
    $F_{fc}^t \cdot W \leftarrow F_{fc}^t \cdot W - \beta \times \nabla J\left(w_{C_i}^t, F_{fc}^t \cdot W\right)$;
  **end for**
**end for**
eps $\leftarrow \max F_{fc}^t(W_s^t) - \min F_{fc}^t(W_s^t)$;
$\mu \leftarrow \frac{1}{\text{num}} \sum_{i=1}^{\text{num}} F_{fc}^t(w_{C_i}^t)$;
**return** $F_{fc}^t$, eps, $\mu$;

**Feature Extraction Models (FEMs).** We choose a fully connected neural network (FCNN) to build each FEM. The FCNN structure is simple, capable of uncovering patterns within seemingly disordered data like gradient weights. The FEMs' objective is to extract features from confirmed honest and unknown local gradients, enabling the anomaly detector's to correctly distinguish malicious gradients from honest ones. Furthermore, the FEMs' training set from $D_s$ is relatively small, so the training time and computational complexity are acceptable for a powerful server. Note that the FEMs serve solely as feature extraction tools for client gradients and do not function as a binary classification model for these gradients. To ensure the FEMs are able to reflect honest gradients, their training set exclusively comprises clean server gradients, excluding any malicious ones.

In training the FEMs, every class model gradient from $D_s$ is expanded into a one-dimensional vector before being fed into the network. Each FEM's output is a feature value, indicative of the respective input gradient. If the uploaded gradient is expanded with dimension $d$, each FEM's hidden layer dimensions are $\lfloor \frac{d}{10} \rfloor, \lfloor \frac{d}{100} \rfloor, \ldots, 1$, with the input layer dimensionality being $d$.

The server constructs an FEM for one class's gradients as shown in Algorithm 1, which comprises two components: using $D_s$ to obtain benchmark (honest) gradients for building the FEM's training set, and using this set to train the FEM. Specifically, we input $D_s$ into the current global model $W$ for training. The server obtains honest gradients $w_{C_i}$ and assembles training set $W_s = \{w_{C_1}; \cdots; w_{C_{\text{num}}}\}$ for the FEM. Training pairs $\{w_{C_i}, 1\}$, $i \in [1, \text{num}]$, are constructed to train the FEM $F_{fc}$, where num denotes the number of samples in the training set, $\text{num} = \frac{|D_s|}{|b_s|}$. $F_{fc}$ training aims to obtain optimal network parameters $w_i$ and $b_i$, $i \in [1, k]$, i.e., the resolution of the following optimization problem.

$$\text{argmin}_{w_i^*, b_i^*} \sum_{w \in W_s} J\left(\sigma\left(f_k \circ \ldots \circ f_2 \circ f_1(w)\right) - 1\right), \quad (1)$$

where $f_k$ denotes the $k$th FCNN layer, $J$ is the loss function, and $\sigma$ denotes the activation function. The objective in training the FEM is to push the predicted output value as close as possible to 1 for honest gradient inputs $w$. Once the FEM $F_{fc}^t$ is derived for one gradient class, deviation values eps and standard feature extraction output $\mu$ are computed for use in detecting malicious clients.

**Local Gradient Feature Extraction.** The server receives local gradients $\boldsymbol{w}^t = \{w_1^t; w_2^t; \cdots; w_n^t\}$ in the $t$th interaction. The extraction of local gradient features using the FEM $F_{fc}^t$ is shown in Algorithm 2, where $m$ denotes the number of local gradients classes, $w_j^t = \{v_{j1}^t, v_{j2}^t, \cdots, v_{jm}^t\}$, $j \in [1, n]$. Feature models $F_{fc_i}^t$, $i \in [1, m]$ are trained using Algorithm 1 for each class of local gradients. The server inputs the same class of local gradients $\boldsymbol{v}_i^t = \{v_{1i}^t; v_{2i}^t; \cdots; v_{ni}^t\}$ into the corresponding model $F_{fc_i}^t$ to obtain the predicted values $\boldsymbol{p}_i^t = \{p_{1i}^t; p_{2i}^t; \cdots; p_{ni}^t\} \in \mathbb{R}^{n*1}$. $p_{ji}^t$ is the feature extraction value of the $i$th gradient of client $j$ in the $t$th interaction. For each class of local gradients, the above process is used to obtain:

$$\begin{aligned} \mathbf{Pro}^t &= \{\boldsymbol{p}_1^t, \boldsymbol{p}_2^t, \cdots, \boldsymbol{p}_m^t\} \\ &= \left\{\mathbf{Pro}_1^t; \mathbf{Pro}_2^t; \cdots; \mathbf{Pro}_n^t\right\} \in \mathbb{R}^{n*m}, \end{aligned} \quad (2)$$

where $\mathbf{Pro}_j^t = \left\{p_{j1}^t, p_{j2}^t, \cdots, p_{jm}^t\right\} \in \mathbb{R}^{1*m}$ denotes $m$ elementary feature vector extracted from the local gradients of client $j$ in the $t$th interaction. Therefore, the server successfully transforms each client's $m$ high-dimensional gradients into a $1 \times m$ elementary feature vector using Algorithm 2. $\mathbf{Pro}^t \in \mathbb{R}^{n*m}$ represents the collection of $m$ elementary feature vectors for all clients.

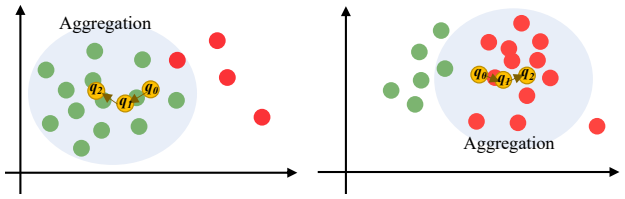
**Remark: Comparison of Neural Network-Based Approaches.** Attack-adaptive Aggregation [29] focuses on distinguishing between honest and malicious gradients in various attack scenarios by training the query encoder $Q$ and key encoder $K$ in the attention mechanism using malicious and honest gradients. In contrast, FLAD emphasizes training on honest data to extract features of honest gradients. Compared to FLAD, Attack-adaptive Aggregation has the following limitations:

**Algorithm 2** Local Gradient Feature Extraction

**Input:** $\boldsymbol{F_{fc}}^t$: Local gradient's FEM at the $t$th interaction, i.e., the output of Algorithm 1. $\boldsymbol{F_{fc}}^t = \left\{F^t_{fc_1}, F^t_{fc_2}, \cdots, F^t_{fc_m}\right\}$, where $F^t_{fc_i}$ denotes the FEM for the $i$th subset of local gradients in the $t$th iteration; $\boldsymbol{w}^t$: Local gradients. $\boldsymbol{w}^t = \left\{w^t_1; w^t_2; \cdots; w^t_n\right\}$;
**Output:** $\mathbf{Pro}^t$: $m$ elementary feature vector sets of all clients;

```
Split w^t into v_i^t, i ∈ {1, 2, ··· , m};
for i = 1 to m do
    p_i^t ← F^t_{fc_i}(v_i^t);
end for
Pro^t = {p_1^t, p_2^t, ··· , p_m^t} ∈ R^{n*m};
return Pro^t = {Pro_1^t; Pro_2^t; ··· ; Pro_n^t};
```

1) *Pretraining Limitations*. Attack-adaptive Aggregation employs self-supervised learning to simulate various attack scenarios and collect malicious and honest gradients for training $Q$ and $K$ encoders. However, it is impossible to simulate all potential attack scenarios, which could lead to new attack methods rendering the pretrained $Q$ and $K$ ineffective.
2) *Aggregation Risks*. The core of Attack-adaptive Aggregation lies in adjusting the aggregation weights of gradients from different clients. However, there remains a risk of malicious gradients being aggregated into the global model parameters, even if their aggregation weights are very small, as shown in Figure 2(a).
3) *Bias in Attention Mechanism*. When the number of malicious clients exceeds 50%, the initial estimation of the attention mechanism $q_0 = \text{median}(\{x_i\})$ may exhibit significant bias, which can result in the alignment scores shifting toward malicious gradients, leading to scenario in Figure 2(b).

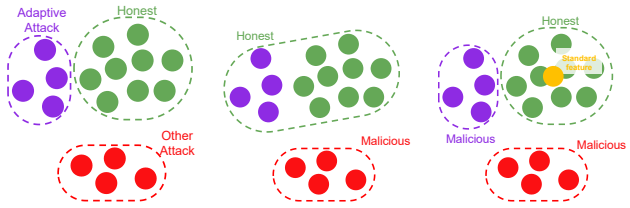



(a) Normal circumstances (b) Malicious clients' proportion $> 50\%$

Fig. 2. Aggregation results of Attack-adaptive Aggregation under different proportions of malicious attacks. Red represent malicious gradients, green represent honest gradients, and yellow represent the iterative aggregation estimates.

*The goal of FLAD is to utilize neural networks to extract features of honest parameters and no attack samples are involved during training.* FLAD focuses solely on identifying honest gradients, treating gradients that deviate from the honest features as malicious gradients, which makes FLAD applicable to a wide range of attack scenarios without requiring assumptions about the number of malicious clients, offering greater scalability and practicality.

*2) Malicious Client Detection: FLAD uses dynamic clustering based on DBSCAN [36] and FEM for malicious clients detection, effectively identifying various attack types, even in scenarios where the proportion of malicious clients exceeds 50%.* The representative clustering-based defenses include Auror [33] and FLAME [8]. Compared with Auror, which uses $k$-means for binary classification, setting cluster number to two introduces the risk of misclassifying malicious gradients as honest. Regarding HDBSCAN-based FLAME, the absence of clustering range restriction makes it vulnerable to adaptive attack and unable to withstand scenarios where attackers exceed 50%. FLAD successfully identifies various attacks using DBSCAN, with adaptive standard feature $\mu$ and deviation value eps as the clustering range restriction, as shown in Figure 3.




(a) Ground truth (b) Auror & FLAME (c) FLAD

Fig. 3. Comparison of different defenses based on clustering, where standard feature in (c) FLAD correspond to the output of Algorithm 1.

DBSCAN, a density-based clustering method, adapts its cluster number according to the specific data. DBSCAN is used to cluster $\mathbf{Pro}^t$ in FLAD, categorizing local gradients into honest and malicious with $\mu$ and eps. Figure 4 illustrates DBSCAN's application in categorizing gradient feature vectors. Feature vectors extracted from honest gradients are highly similar and cluster within a sphere, defined by scan radius eps and $\mu$. The key parameter scan radius in DBSCAN is related to the outputs of FEMs in Algorithm 1. For $m$ classes of model gradients used for detection, the joint deviation value $\text{eps} = ||\mathbf{eps}|| = \sqrt{\sum_{i=1}^{m} \text{eps}_i^2}$ is the scan radius of DBSCAN. Therefore, DBSCAN's scan radius is determined based on the server dataset for each interaction, which avoids false positives caused by artificial parameter settings.

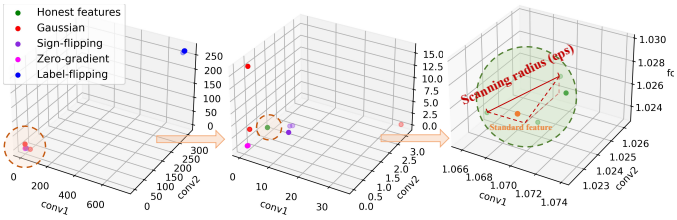



Fig. 4. DBSCAN separates honest and malicious gradient features, with dots representing feature extraction vectors for three classes of local gradients in the 1st interaction of MNIST (0.1) with CNN.

The method for malicious client detection is described in Algorithm 3. The server may classify the local gradients feature vectors into many classes. The method uses the cosine similarity (cos) and the relative length difference (len) to distinguish honest clients from malicious ones. It calculates $\cos_l$ and $\text{len}_l$ between feature vectors $\mathbf{Pro}^t_j$ labeled $l$ and the benchmark feature's mean $\boldsymbol{\mu}$:

$$\cos_l = \cos\left(\boldsymbol{\mu}, \mathbf{Pro}^t_j\right) = \frac{\left\langle \boldsymbol{\mu}, \mathbf{Pro}^t_j \right\rangle}{||\boldsymbol{\mu}|| \cdot ||\mathbf{Pro}^t_j||}, \tag{3}$$

$$\text{len}_l = \text{len}\left(\boldsymbol{\mu}, \mathbf{Pro}_j^t\right) = \left| \frac{||\boldsymbol{\mu}||}{||\mathbf{Pro}_j^t||} - 1 \right|, \tag{4}$$

where $\boldsymbol{\mu} = \{\mu_1, \cdots, \mu_m\}$, which comprises the standard features of $m$ classes of model gradients and is used as honest standard. Considering that the target feature value for FEM training is 1, the honesty similarity of a given label $l$ is defined as $S_l = \alpha(\cos_l - \text{len}_l) + (1-\alpha)||\mathbf{Pro}_j^t||$, $0 < \alpha < 1$. The honest label Key is determined by the maximum $S$, and all remaining labels are categorized as malicious.

**Algorithm 3** Malicious Client Detection

**Input:** $\mathbf{Pro}^t$: $m$ elementary feature vector sets of all clients;
$\mathbf{eps} = \{\text{eps}_1, \cdots, \text{eps}_m\}$: Deviation values for $m$ honest gradients;
$\boldsymbol{\mu} = \{\mu_1, \cdots, \mu_m\}$: Standard features for $m$ honest gradients;
**Output:** Malicious: Malicious client list, $1 \sim n$;

eps $\leftarrow ||\mathbf{eps}||$; **cos** $\leftarrow$ None; **len** $\leftarrow$ None;
$label \in \mathbb{R}^{1*n} \leftarrow$ DBSCAN($\mathbf{Pro}^t$, eps);
**for** $j = 1$ **to** $n$ **do**
  $l \leftarrow j_{\text{label}}$
  **if** $\cos_l ==$ None **then**
    $\cos_l \leftarrow \cos\left(\boldsymbol{\mu}, \mathbf{Pro}_j^t\right)$ % Cosine similarity
    $\text{len}_l \leftarrow \text{len}\left(\boldsymbol{\mu}, \mathbf{Pro}_j^t\right)$ % Relative length difference
  **end if**
**end for**
Take label Key in $\max\left[\alpha(\cos_l - \text{len}_l) + (1-\alpha)||\mathbf{Pro}_j^t||\right]$;
**for** $j = 1$ **to** $n$ **do**
  **if** $j_{\text{label}} \neq$ Key **then**
    Add $j$ to Malicious list;
  **end if**
**end for**
**return** Malicious;

### C. Complete FLAD Algorithm

Algorithm 4 presents the comprehensive Byzantine-robust FL approach FLAD, involving $c$ malicious clients. The server uses FedAvg to aggregate honest gradients after rejecting malicious clients. With malicious clients excluded, FedAvg can effectively merge and learn local gradients from honest clients, facilitating rapid convergence and high accuracy for global model.

**Algorithm 4** FLAD

**Input:** $W^{t-1}$: Global model gradients at $(t-1)$th round;
$D_i$: Local dataset for client $i$, $i \in \{1, 2, \cdots, n\}$;
$D_s$: Server dataset for bootstrapping trust;
**Output:** $W^t$: Global model gradients after the $t$th aggregation;

**Client side:**
**for** $i$ **to** $n$ **in parallel do**
  $\boldsymbol{w_h}^t \leftarrow$ LocalTraining$(W^{t-1}, D_i), i \in [1, n-c]$;
  $\boldsymbol{w_m}^t \leftarrow \boldsymbol{X}, i \in [n-c+1, n]$; % $\boldsymbol{X}$: arbitrary vectors
**end for**
Send $\boldsymbol{w}^t \leftarrow \left\{\boldsymbol{w_h}^t; \boldsymbol{w_m}^t\right\}$ to the server;
**Server side:** $m$ class model gradients for detection
**for** $i = 1$ **to** $m$ **do**
  $F_{fc_i}^t, \text{eps}_i, \mu_i \leftarrow$ FeatureExtractionModel$(W^{t-1}, D_s)$;
**end for**
$\boldsymbol{F_{fc}^t} = \left\{F_{fc_1}^t, \cdots, F_{fc_m}^t\right\}$;
$\mathbf{Pro}^t \leftarrow$ LocalGradientFeature$(\boldsymbol{F_{fc}}^t, \boldsymbol{w}^t)$;
$\mathbf{eps}, \boldsymbol{\mu} = \{\text{eps}_1, \cdots, \text{eps}_m\}, \{\mu_1, \cdots, \mu_m\}$;
Malicious $\leftarrow$ MaliciousClientDetection$(\mathbf{Pro}^t, \mathbf{eps}, \boldsymbol{\mu})$;
$W^t \leftarrow$ Aggregation(Malicious, $\boldsymbol{w}^t$);
Send $W^t$ to all clients;

### D. Proximity Analysis of Tolerance for Malicious Attacks

This section is used to analyze the proximity point of Byzantine attacks tolerated by FLAD. The training objective of FEM is to obtain optimal parameters $\{w_i^*\}_i$ and $\{b_i^*\}_i$ in Eq. (1). We can obtain the feature extraction vectors for num server model gradients:

$$f_{c_i} = \sigma\left(f_k \circ \ldots \circ f_2 \circ f_1(w_{c_i})\right), i \in [1, \text{num}]. \tag{5}$$

DBSCAN's scan radius $eps = ||\max f_{c_i} - \min f_{c_i}||$, where $||\cdot||$ denotes the $L_2$ norm. The average value $\mu = \frac{1}{\text{num}} \sum_{i=1}^{\text{num}} f_{c_i}$ used to measure the standard of honest gradients. Local gradients from clients are fed into the FEM to obtain the corresponding feature extraction $F = \{f_1, \cdots, f_i, \cdots, f_n\}$.

**Assumption 1.** *There are no less than three honest gradients in local gradients, and the client feature vector with the closest $L_2$ norm from the average value $\mu$ is $f_q$:*

$$f_q = \text{argmin}_{f_i \in F} ||f_i - \mu||. \tag{6}$$

*So $f_q$ is considered as the feature extraction vector of the honest client $q$ and $w_q$ is the honest local gradient.*

Since the feature $f_q$ corresponding to $w_q$ is closest to the features of honest gradient $\mu$, it is reasonable to assume that $w_q$ from client $q$ is the honest gradient.

**Definition 1.** *There exists vectors set $F = \{f_1, f_2, \cdots\}$. If there exists no less than three elements $\{f_y, \cdots\}$ in $F$ within $eps$ $L_2$ norm of vector $f_x$, $f_y$ is defined as $1eps - f_x$ directly connected point. If there are also no less than three elements $\{f_z, \cdots\}$ in $F$ within $eps$ $L_2$ norm of vector $f_y$, $f_z$ is defined as $2eps - f_x$ directly connected point, and so on.*

Feature extraction vectors set from local gradients is $\{f_1, \cdots, f_q, \cdots, f_n\}$, and we assume that there exist at most $k\text{eps} - f_q$ directly connected points. According to the principle of DBSCAN, feature vector $f_l$ with the farthest $L_2$ norm from $f_q$ and reachable density with $f_q$ has $||f_l - f_q|| \leq k\text{eps}$. The criticality of the honest and malicious gradients has the following Theorem 1.

**Theorem 1.** *A sufficient condition for the malicious gradient feature $f_m$ to be successfully detected is $||f_m - f_q|| \geq (k+1)eps \geq ||f_l - f_q|| + eps$. A necessary condition for the honest gradient feature $f_h$ to be successfully identified is $||f_h - f_q|| \leq keps$.*

### E. Convergence Analysis of FLAD

We will prove the convergence of the FLAD in this section.

**Assumption 2 (Error Term).** *There exists an error term $\boldsymbol{\epsilon}$ between the malicious gradients and the honest gradients, such that the equation holds: $\sum_{x \in M} \boldsymbol{w'_x} = \sum_{x \in H} \boldsymbol{w_x} + \boldsymbol{\epsilon}$, where $M$ and $H$ denote the sets of malicious and honest clients, respectively. Specifically, $H = [1, n-c]$ and $M = [n-c+1, n]$.*

**proof.** The process of poisoning attack is to find a set of malicious model gradients $\boldsymbol{w'}$ which are different from the honest client's target model $\boldsymbol{w}$, i.e., $\boldsymbol{w} \neq \boldsymbol{w'}$. Let the distance between $\boldsymbol{w'}$ and $\boldsymbol{w}$ be $\overline{\boldsymbol{h}}$, i.e., $\overline{\boldsymbol{h}} = \boldsymbol{w} - \boldsymbol{w'}$. As described in

Section III-A, malicious and honest clients can only achieve their goals by submitting gradients.

We define the classical poisoning attack model. Suppose that the gradients $\{\boldsymbol{w}_{\boldsymbol{x}}, x \in H\}$ of honest clients in round $t$ are independent and identically distributed samples taken from the random variable $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \times \nabla l(b_s, \boldsymbol{w})$, where $E[\boldsymbol{w}] = \overline{\boldsymbol{w}}$ is an unbiased estimate of the gradient. Thus, for any $x \in H$, we have $E[\boldsymbol{w}_{\boldsymbol{x}}] = E[\boldsymbol{w}] = \overline{\boldsymbol{w}}$. In the presence of malicious clients, the malicious gradients received by server are $\left\{\boldsymbol{w}'_{\boldsymbol{x}}, x \in M\right\}$ as follows:

$$\boldsymbol{w}^{\boldsymbol{\tau}}_{\boldsymbol{x}} = \begin{cases} \boldsymbol{w}_{\boldsymbol{x}}, & \text{if the } x\text{th client is honest,} \\ \boldsymbol{w}'_{\boldsymbol{x}}, & \text{if the } x\text{th client is malicious.} \end{cases} \tag{7}$$

An important assumption for convergence of the SGD algorithm is that each honest gradient is an unbiased estimate of actual gradient, which is usually ensured by uniformly randomly sampling the batch data $b_s$. However, the gradients submitted by malicious clients violate the principle of uniform random sampling, making malicious gradients significantly different from honest ones. According to the definition of SGD algorithm, $N$ is the set of all clients, $N \in [1, n]$:

$$\boldsymbol{w}^{\boldsymbol{\tau}} \leftarrow \frac{1}{|N|} \sum_{x \in N} \boldsymbol{w}^{\boldsymbol{\tau}}_{\boldsymbol{x}}, \tag{8}$$

The goal of malicious gradients $\left\{\boldsymbol{w}'_{\boldsymbol{x}}, x \in M\right\}$ is to obtain parameters:

$$\boldsymbol{w}' \leftarrow \frac{1}{|M|} \sum_{x \in M} \boldsymbol{w}'_{\boldsymbol{x}}, \tag{9}$$

Similarly, honest gradients $\{\boldsymbol{w}_{\boldsymbol{x}}, x \in H\}$ is to obtain parameters:

$$\boldsymbol{w} \leftarrow \frac{1}{|H|} \sum_{x \in H} \boldsymbol{w}_{\boldsymbol{x}}, \tag{10}$$

Based on the previously defined parameter distance ($\overline{\boldsymbol{h}} = \boldsymbol{w} - \boldsymbol{w}'$), we have:

$$\begin{aligned} \sum_{x \in M} \boldsymbol{w}'_{\boldsymbol{x}} &= \frac{|M|}{|H|} \sum_{x \in H} \boldsymbol{w}_{\boldsymbol{x}} - |M| \overline{\boldsymbol{h}} \\ &= \left(\frac{|M|}{|H|} - 1\right) \sum_{x \in H} \boldsymbol{w}_{\boldsymbol{x}} - |M| \overline{\boldsymbol{h}} + \sum_{x \in H} \boldsymbol{w}_{\boldsymbol{x}} \\ &= \left(\frac{|M|}{|H|} - 1\right) |H| \overline{\boldsymbol{w}} - |M| \overline{\boldsymbol{h}} + \sum_{x \in H} \boldsymbol{w}_{\boldsymbol{x}}, \end{aligned} \tag{11}$$

Thus, the error term $\boldsymbol{\epsilon}$ between malicious gradient and honest gradient satisfies the following relationship:

$$\boldsymbol{\epsilon} = \left(\frac{|M|}{|H|} - 1\right) |H| \overline{\boldsymbol{w}} - |M| \overline{\boldsymbol{h}}. \tag{12}$$

**Proposition 1 (Convergence).** *According to the detailed steps of FLAD in Section IV, the convergence speed of malicious and/or honest clients is $O\left(1/T^2\right)$, where $T$ is the number of iterations.*

**proof.** After identifying honest clients, FLAD employs the traditional FedAvg aggregation, which follows the basic framework of adaptive learning rate method [37] which has been applied to SGD algorithm and provides convergence guarantees. SGD algorithms with constant learning rate have been proved to converge at a rate of $O\left(1/T^2\right)$.

Assuming an equal amount of local data for each client, the FedAvg aggregation weight of user $U_x$ is determined by function $f(x,t)$ as $\frac{1}{|M^t|+|H^t|}$, $x \in \{M^t, H^t\}$ or 0, $x \notin \{M^t, H^t\}$, where $t$ denotes the current number of iterations, and $M^t$ and $H^t$ denote the clients that are recognized as honest in iteration $t$. $U_x$ may be either a malicious or an honest client, $U_x \in M$ or $H$. In Section III, we assume that the number of honest clients is more than 2, i.e., $|H| > 2$, to ensure that correct data exists in federated training. The convergence property of SGD can be applied to FLAD as long as the training is based on data of honest clients. At this point, FLAD needs to satisfy two conditions:

- Condition 1: for all $x \in M$, $f(x,t) \to 0$,
- Condition 2: for all $x \in H$, $f(x,t) \to 1$.

We consider $\boldsymbol{w}^*_{\boldsymbol{x}}$ as the ideal gradient from initial model $\boldsymbol{w}_{\text{init}}$ to optimized model with respect to local data of $U_x$. According to Assumption 2, $\sum_{x \in M} \boldsymbol{w}^*_{\boldsymbol{x}} = \sum_{x \in H} \boldsymbol{w}^*_{\boldsymbol{x}} + \boldsymbol{\epsilon}$. FLAD can identify anomalous gradients based on differences and reject malicious gradients. As the number of iterations increases, the target of malicious gradients deviate further away from honest gradients and the error term $\boldsymbol{\epsilon}$ will increase. Thus malicious clients $x \in M$ can be more obviously recognized and rejected by FLAD, which will result in malicious gradients tending to have a weight of 0 when globally aggregated, i.e., $|M^t| \to 0$, $|H^t| \to |H|$, $f(x,t) \to 0$. Therefore, malicious gradients have a weight of 0, which satisfies Condition 1. Accordingly, FLAD satisfies for honest clients:

$$\lim_{t \to \infty} \frac{|H|}{|M^t| + |H^t|} = 1. \tag{13}$$

Therefore, the sum of weights of honest clients converges to 1, which satisfies Condition 2. The above proof shows that FLAD satisfies both conditions. Therefore, the convergence rate of FLAD is consistent with that of SGD with adaptive learning rate, i.e., $O\left(1/T^2\right)$.

## V. Performance Evaluation

### A. Experimental Setup

*1) Datasets and FL System Settings:* We have experimented with four datasets: MNIST [38], CIFAR-10 [39], ASSIST2009[2] and FEMNIST [40]. The information about the four datasets and their corresponding global model gradients are shown in Table I, and the specific global model architecture for MNIST is shown in Table II. Following the work [14], in each interaction, the server selects all clients' local gradients for aggregation, except for FEMNIST. The client interacts with the server a total of 20 times in MNIST, CIFAR-10 and FEMNIST and 5 times in ASSIST2009.

**Non-IID and IID data.** For image dataset, we define distribution probability $q$ to measure the distribution difference among different client dataset. A larger $q$ indicates a higher degree of Non-IID. Refer to experimental section in FLTrust [6]

[2]https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data

TABLE I
DATASET AND GLOBAL MODEL SETTINGS.

| DATASET | CLIENT NUMBER | CLIENT DATASET SIZE | FEATURES/QUESTIONS | GLOBAL MODEL | BATCH SIZE | SERVER DATASET |
|---|---|---|---|---|---|---|
| MNIST | 50 | 1,200 | 784 (28*28) | 2 CNN+1 FC | 64 | 300 |
| CIFAR-10 | 50 | 1,000 | 3,072 (3*32*32) | RESNET-18 | 64 | 200 |
| ASSIST2009 | 20 | 1.6K | 26.7K | MAML (2FC) | 128 | 500 |
| FEMNIST | 3,400 (50 PER ROUND) | 2-582 (AVERAGE: 226.83) | 784 (28*28) | VGG-16 | 64 | 300 |

TABLE II
CNN STRUCTURE FOR TRAINING MNIST.

| LAYER | SIZE |
|---|---|
| INPUT | $1 \times 28 \times 28$ |
| CONVOLUTION+RELU | $10 \times 1 \times 5 \times 5$ |
| MAX POOLING | $2 \times 2$ |
| CONVOLUTION+RELU | $20 \times 10 \times 5 \times 5$ |
| MAX POOLING | $2 \times 2$ |
| LINEAR+SOFTMAX | $320 \times 10$ |
| OUTPUT | 10 |

for detailed definitions of $q$. We choose $q = 0.1, 0.8$ for MNIST and $q = 0.1,\ 0.5$ for CIFAR-10. If $q = 0.1$, the client's data is IID. ASSIST2009 dataset is derived from interactions by different students answering different questions, making it inherently Non-IID.

**Server dataset.** In order to highlight the robustness to $D_s$, we borrow the experimental strategy of FLTrust [6] and simulate $D_s$ for the following two scenarios to explore their impact on global accuracy of FLAD.

- Case I: $D_s$ has the same distribution as training dataset.
- Case II: The distribution of $D_s$ differs from the training dataset. Similar to FLTrust [6], we measure the deviation of $D_s$ from the training dataset distribution with bias probability $P$. A higher $P$ corresponds to a greater disparity between $D_s$ and the training dataset. In our experiments, bias labels are assigned as "0" for MNIST and FEMNIST, and as "airplane" for CIFAR-10.

**FLAD Settings.** For MNIST, two convolutional kernels and one fully connected weights are used to construct FEM. For CIFAR-10, we select the convolutional kernels of the first convolutional layer in conv1 and conv2 modules and the fully connected weight of the last layer. For ASSIST2009, we select two fully connected weights. For FEMNIST, we select the fully connected weights of the last three layers. For FEM's training, the loss function $J$ chooses MSE, the optimizer is Adam, the learning rate is $\beta = 1 \times 10^{-3}$, and the total number of training epoch is $E_s = 10$.

*2) Evaluated Defenses:* We choose robust aggregation (Krum [22], Bulyan [24], Median [23]), correction of gradients (FLTrust [6]) and malicious node detection and rejection (RWD [34], FLAME [8], FLDetector[3] [32]) for seven defenses. Moreover, we use FedAvg, which is dominant in the non-adversarial setting, as the experiment baseline. The threshold in RWD is $T = 10$ and the size of FLTrust root dataset is the same as FLAD. Adaptive noising level in FLAME is $\boldsymbol{\sigma} = 0.001$. The remaining parameter settings are the same as in the experimental part of the original paper.

*3) Evaluated Attacks:* For untargeted attacks, we choose Gaussian attack [10], sign-flipping attack [12], zero-gradient attack [11], adaptive model poisoning attack [14], AGR-agnostic attack [15] and MPAF [13]. For targeted attacks, we choose backdoor data poisoning, model-replacement attack [18] and DBA [20]. Furthermore, we assume that these attacks are implemented in the context of full knowledge. Gaussian, sign-flipping and zero-gradient attacks have the same settings as in [11].

**Adaptive Local Model Poisoning Attack.** We use the current state-of-the-art adaptive attack framework [14] to design adaptive attack to explore the tolerance proximity of FLAD for malicious poisoning attacks. The adaptive attack framework is as follows.

$$\begin{aligned} & \max_{w_1', \cdots, w_c'} \mathbf{s}^T (W - W'), \\ \text{subject to} \quad & W = \mathcal{A}(w_1, \cdots, w_c, w_{c+1}, \cdots, w_n), \\ & W' = \mathcal{A}(w_1', \cdots, w_c', w_{c+1}, \cdots, w_n), \end{aligned} \tag{14}$$

where the first $c$ clients are assumed to be malicious. $w_i'$ is the poisoned local gradients uploaded by the $i$th malicious client. $\mathbf{s}$ is the column vector of the direction of change of the global gradients without attack. $W$ and $W_i'$ represent the global model obtained with aggregation rule $\mathcal{A}$ under poisoning and no attack, respectively. In summary, the adaptive attack treats the attack as an optimization problem in which the attackers dynamically optimize the malicious local gradients in each iteration, such that the global model gradient after aggregation changes maximally in the opposite direction as it would have done without attack. Different aggregation rules will lead to different optimization problems. Following the above adaptive local poisoning attack framework, we define $W$ and $W'$:

$$W = \mathcal{A}(w_1, w_2, \cdots, w_{n-c}), \tag{15}$$

$$W' = \mathcal{A}(w_1, w_2, \cdots, w_{n-c}, w_{n-c+1}', \cdots, w_n'). \tag{16}$$

The adaptive model attack can be considered as the optimization problem $\max_{w_{n-c+1}', \cdots, w_n'} \mathbf{s}^T (W - W')$. We assume that the number of local data for each client is equal. Since the values of honest gradients do not affect the results of problem. Therefore, the optimization can be simplified as:

$$\min_{w_{n-c+1}', \cdots, w_n'} \mathbf{s}^T \left( w_{n-c+1}' + \cdots + w_n' \right). \tag{17}$$

Therefore, attacker's goal is to carefully construct malicious gradients $w'$ in the opposite direction of global update without

[3]FLDetector is abbreviated as FLD in our experiments.

attacks as much as possible to confuse FEM in anomaly detectors so that FEM outputs the wrong feature extraction vectors for the malicious gradients, misleading the Algorithm 3 to classify them as honest gradients. The solution to the optimization problem translates into generating holistic confused samples of FEM based on deviating the honest gradients as much as possible.

We borrow FGSM (Fast Gradient Sign Method) [41] to generate adversarial samples to solve the problem (see Eq. (17)) and obtain adaptive attack of FLAD in Algorithm 5. We consider all malicious gradients as a whole and jointly compute their updated gradients on FEM. Honest gradient $w_i$ added to noise $u$ is used as initial value for malicious gradient $w'_j$, avoiding the problem that initial $w'_j$ has a loss function of 0, where $u$ obeys a multivariate Gaussian distribution $\mathcal{N}\left(0, \sigma^2\mathbf{I}\right)$. We use Algorithm 2 to obtain the feature vectors $\mathbf{Pro}'_j$ corresponding to the current $w'_j$, and take the difference between $\mathbf{Pro}'_j$ and $\mathbf{Pro}_i$ as the loss to compute the updated gradient $\nabla loss(w'_j)$ of $w'_j$ in FEM. The new optimized $w'_j$ is obtained with $\epsilon$ as step size, which is 1e-7 in experiments. When $w'_j$ deviates too much to be successfully detected by Algorithm 3, the optimization result $B'_j$ from the previous iteration is returned as the optimal adaptive malicious gradient $w'_j$.

**Algorithm 5** Adaptive Local Model Attack on FLAD

**Input:** $w_i$: Local gradients sent by honest clients, $i \in [1, n-c]$;
$F_{fc}$: Feature extraction model;
$\epsilon$: Step size of optimization per iteration;
**Output:** $w'_j$: Malicious gradients carefully constructed by the malicious clients, $j \in [n-c+1, n]$;

$Pro_i \leftarrow$ LocalGradientFeature$(F_{fc}, w_i)$;
Initialize $w'_j, Pro'_j \leftarrow w_i, Pro_i$;
$w'_j \leftarrow w'_j + \epsilon \times u, u \sim \mathcal{N}\left(0, \sigma^2\mathbf{I}\right)$;
**while** True **do**
  $B'_j \leftarrow w'_j$;
  Pro$'_j \leftarrow$ LocalGradientFeature$(F_{fc}, w'_j)$;
  $loss \leftarrow$ MSE(Pro$'_j$, Pro$_i$);
  Update $w'_j \leftarrow w'_j + \epsilon \times$ sgn$(\nabla$ loss$(w'_j))$;
  $\boldsymbol{w} = \left\{w_i, w'_j\right\}$;
  **if** $F_{fc}(\boldsymbol{w})$ is able to detect malicious clients **then**
    **Break**;
  **end if**
**end while**
**return** $w'_j \leftarrow B'_j$;

**AGR-agnostic Attack.** The research [15] introduced two types of attacks: AGR-tailored attacks targeting specific aggregation algorithms (e.g., Krum, Bulyan) and AGR-agnostic attacks that do not depend on the aggregation algorithm. The core idea of AGR-tailored attacks is similar to the Adaptive Local Model Poisoning Attack [14], and there are no specific attack strategies directly targeting FLAD. Therefore, we select the more challenging-to-detect Min-Sum attack strategy from the AGR-agnostic attacks described in [15]. The specific parameter settings are consistent with [15].

**Model Poisoning Attack based on Fake Clients (MPAF).** The attacker injects multiple fake clients, which participate in training and send malicious local updates. The goal of these malicious updates is to steer the global model's parameters toward a low-performance model chosen by the attacker, thereby significantly reducing the accuracy of the global model [13]. To highlight the superiority of FLAD, we set the proportion of injected fake clients to 20% (10 fake clients in MNIST, CIFAR-10 and FEMNIST), the sample rate $\beta = 1$, the scaling factor $\lambda = 1 \times 10^3$, and the base model $w'$ as a randomly initialized model.

**Backdoor Data Poisoning Attack.** Backdoor data poisoning attack requires the attacker to implant the backdoor trigger in the source training data so that the final model will function properly on clean data, but will constantly predict the target class chosen by the attacker when data containing the trigger is present. In our experiments, 80% of the backdoor data is injected into local dataset of the backdoor client during the training and we inject $8 \times 8$ square pixel block in the top left corner of the malicious client local dataset as backdoor trigger. For MNIST and FEMNIST, we change the target label of the image with $8 \times 8$ black block to "0". For CIFAR-10, we inject pixel blocks corresponding to the channel color on three channels and change the target label to "airplane". After injecting the backdoor to local dataset and changing the target label, malicious client train local model as an honest client.

**Model-replacement Attack.** The backdoor trigger of model-replacement attack is set up as for backdoor data poisoning attack. Following the work [18], in order to mitigate the dilution of the attack effect due to global aggregation, we extend the poisoning local gradients by setting the weight factor $\gamma = n$ and send the extended local gradients to the server [18], where $n$ is the total number of clients.

**Distributed Backdoor Attack (DBA).** We split the $8 \times 8$ centralized trigger into four $2 \times 8$ local triggers and evenly divide malicious clients into four groups to inject different local triggers. During training, the server will receive local gradients containing local triggers from four groups of malicious clients. The complete algorithm is detailed in DBA [20].

### *B. Experiment Results*

*1) Defense Goals:* We define four defensive goals of accuracy, robustness, efficiency and universality in Section III-B. The extensive experimental results show that our FLAD achieves these goals.

**Accuracy.** In the non-adversarial setting, FLAD's global accuracy is similar to FedAvg in Table III. On the MNIST (0.8), FedAvg, FLD and FLAD both achieve an accuracy of 95%, while Krum and Bulyan achieve 90% and 94% respectively. On the CIFAR-10 (0.1), the accuracy of FLAD and FedAvg can reach 69%, while the accuracy of FLTrust and FLAME are 54% and 47% respectively. Therefore, in the non-adversarial setting, FLAD exhibits higher global accuracy.

**Robustness.** The global accuracy of FLAD under different attacks is at most 2.1% lower than FedAvg without attack in Table III. However, other FL defences show reduced global accuracy in some attack scenarios. For example, the accuracy of Bulyan, FLAME and FLD can be up to 8.4%, 59.1% and 2.5% lower than FLAD respectively in Table III(b). Moreover, FLAD safeguards against targeted attacks while maintaining global accuracy, where FLAD's backdoor accuracy is maintained at the same 10% as when there are no attacks in Table IV.

TABLE III
GLOBAL MODEL ACCURACY FROM DIFFERENT DEFENSES UNDER DIFFERENT ATTACKS. PERCENTAGES INDICATE THE PERCENTAGE OF MALICIOUS CLIENTS AND % ARE OMITTED.

(a) MNIST (0.8, 20%)

| ATTACKS | FEDAVG | KRUM | BULYAN | FLTRUST | FLAME | FLD | FLAD |
|---|---|---|---|---|---|---|---|
| NONE | 94.9 | 90.2 | 94.3 | 94.7 | 70.5 | 95.1 | 95.2 |
| GAUSSIAN | 9.1 | 91.1 | 94.6 | 95.0 | 43.4 | 94.3 | 95.2 |
| SIGN-FLIPPING | 31.4 | 90.6 | 94.7 | 94.8 | 57.3 | 94.0 | 95.0 |
| ZERO-GRADIENT | 10.1 | 91.6 | 94.1 | 94.7 | 64.7 | 93.8 | 95.1 |
| AGR-AGNOSTIC | 27.4 | 87.5 | 91.6 | 92.6 | 33.4 | 93.3 | 94.5 |
| MPAF | 19.8 | 88.2 | 92.7 | 92.9 | 45.3 | 93.4 | 94.7 |
| BACKDOOR-DATA | 94.7 | 91.2 | 94.2 | 93.6 | 72.4 | 94.1 | 95.0 |
| MODEL-REPLACE | 94.3 | 89.4 | 93.8 | 93.7 | 70.8 | 94.1 | 94.9 |
| DBA | 94.3 | 87.1 | 94.1 | 93.5 | 53.8 | 94 | 94.7 |
| AVERAGE | 52.9 | 89.7 | 93.8 | 93.9 | 56.8 | 94.0 | **94.9** |

(b) CIFAR-10 (0.1, 20%)

| ATTACKS | FEDAVG | KRUM | BULYAN | FLTRUST | FLAME | FLD | FLAD |
|---|---|---|---|---|---|---|---|
| NONE | 68.8 | 53.5 | 64.1 | 54.4 | 47.4 | 68.3 | 68.6 |
| GAUSSIAN | 14.3 | 62.3 | 63.1 | 53.3 | 9.9 | 67.5 | 69.0 |
| SIGN-FLIPPING | 10.0 | 54.6 | 63.9 | 54.6 | 11.8 | 67.3 | 68.2 |
| ZERO-GRADIENT | 10.0 | 50.7 | 62.0 | 53.8 | 10.9 | 66.8 | 69.3 |
| AGR-AGNOSTIC | 11.2 | 46.9 | 59.6 | 50.2 | 19.8 | 65.2 | 66.7 |
| MPAF | 10.1 | 49.1 | 60.8 | 52.7 | 23.5 | 65.9 | 67.2 |
| BACKDOOR-DATA | 67.2 | 51.7 | 62.3 | 53.6 | 45.7 | 67.6 | 69.1 |
| MODEL-REPLACE | 68.2 | 49.9 | 61.0 | 51.6 | 46.7 | 67.9 | 69.4 |
| DBA | 68.8 | 51.8 | 63.7 | 55.4 | 46.1 | 68.1 | 69.0 |
| AVERAGE | 36.5 | 52.3 | 62.3 | 53.3 | 29.1 | 67.2 | **68.5** |

(c) CIFAR-10 (0.5, 20%)

| ATTACKS | FEDAVG | KRUM | BULYAN | FLTRUST | FLAME | FLD | FLAD |
|---|---|---|---|---|---|---|---|
| NONE | 64.8 | 29.0 | 60.7 | 58.0 | 41.5 | 64.2 | 64.6 |
| GAUSSIAN | 14.1 | 46.7 | 62.6 | 55.2 | 9.1 | 63.4 | 64.1 |
| SIGN-FLIPPING | 10.0 | 31.0 | 62.3 | 56.8 | 11.2 | 63.0 | 63.9 |
| ZERO-GRADIENT | 12.1 | 29.5 | 62.0 | 55.8 | 9.7 | 63.3 | 64.1 |
| AGR-AGNOSTIC | 9.6 | 42.1 | 62.1 | 54.2 | 8.6 | 63.2 | 64.0 |
| MPAF | 11.3 | 42.2 | 63.5 | 56.7 | 10.0 | 63.4 | 64.1 |
| BACKDOOR-DATA | 64.0 | 33.9 | 61.3 | 50.8 | 40.8 | 64.1 | 64.2 |
| MODEL-REPLACE | 64.4 | 24.3 | 60.7 | 51.0 | 39.9 | 63.6 | 63.9 |
| DBA | 65.0 | 40.2 | 61.7 | 51.4 | 41.1 | 63.1 | 64.1 |
| AVERAGE | 35.0 | 35.4 | 61.9 | 54.4 | 23.5 | 63.5 | **64.1** |

(d) FEMNIST (20%)

| ATTACKS | FEDAVG | KRUM | BULYAN | FLTRUST | FLAME | FLD | FLAD |
|---|---|---|---|---|---|---|---|
| NONE | 77.4 | 74.5 | 76.1 | 76.7 | 58.5 | 77.1 | 77.4 |
| GAUSSIAN | 11.2 | 75.6 | 77.0 | 77.1 | 31.9 | 76.5 | 77.2 |
| SIGN-FLIPPING | 15.7 | 74.9 | 76.8 | 76.5 | 42.8 | 76.3 | 76.9 |
| ZERO-GRADIENT | 9.8 | 73.6 | 75.4 | 76.9 | 50.3 | 75.9 | 77.0 |
| AGR-AGNOSTIC | 12.8 | 70.7 | 73.5 | 73.8 | 28.1 | 74.9 | 75.8 |
| MPAF | 17.4 | 72.0 | 72.8 | 74.5 | 40.3 | 75 | 76.6 |
| BACKDOOR-DATA | 75.4 | 72.8 | 76.1 | 76.3 | 62.3 | 76.1 | 77.0 |
| MODEL-REPLACE | 75.3 | 73.3 | 76.7 | 76.0 | 60.7 | 76.1 | 77.1 |
| DBA | 76.2 | 70.1 | 76.9 | 75.4 | 49.8 | 76.4 | 77.0 |
| AVERAGE | 41.2 | 73.1 | 75.7 | 75.9 | 47.2 | 76.0 | **76.9** |

(e) ASSIST2009 (20%)

| ATTACKS | FEDAVG | KRUM | BULYAN | FLTRUST | FLAME | FLD | FLAD |
|---|---|---|---|---|---|---|---|
| NONE | 65.4 | 63.7 | 63.0 | 63.6 | 64.5 | 64.4 | 65.1 |
| GAUSSIAN | 50.9 | 63.3 | 63.0 | 57.3 | 63.5 | 64.1 | 65.0 |
| SIGN-FLIPPING | 60.1 | 63.1 | 64.1 | 60.3 | 63.5 | 64.6 | 65.4 |
| ZERO-GRADIENT | 60.1 | 63.1 | 64.1 | 60.3 | 63.8 | 64.7 | 65.4 |
| AVERAGE | 59.2 | 63.3 | 63.5 | 60.4 | 63.8 | 64.5 | **65.3** |

**Efficiency.** FLAD only requires client to train its own local model in each iteration, does not introduce additional computational overhead for clients and does not add additional communication overhead between client and server compared to FedAvg. As shown in Figure 5, the convergence speed of FLAD is comparable to FedAvg without attacks under different attacks. However, other defences converge significantly slower than the no-attack FedAvg as shown in Figure 6, which shows that FLAD global model converges faster than other FL defences, more closely matching no-attack FedAvg.

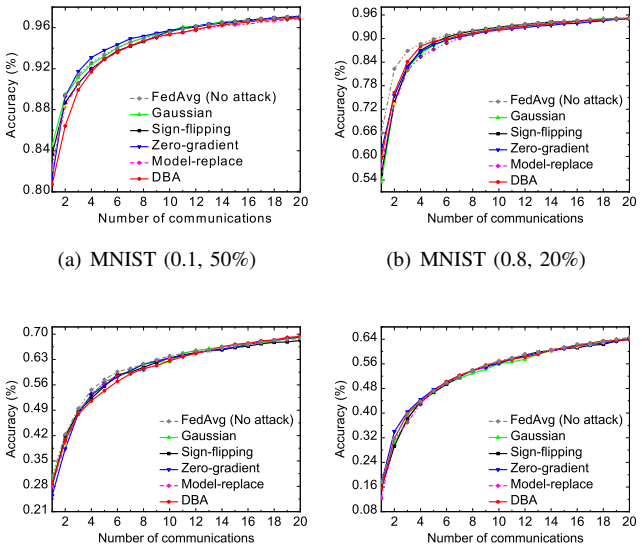
TABLE IV
BACKDOOR MODEL ACCURACY FROM DIFFERENT DEFENSES UNDER DIFFERENT TARGETED ATTACKS.

(a) MNIST (0.1, 50%)

| ATTACKS | FEDAVG | MEDIAN | KRUM | FLTRUST | FLAME | FLD | FLAD |
|---|---|---|---|---|---|---|---|
| NONE | 10.0 | 10.1 | 10.1 | 10.0 | 10.0 | 10.0 | 10.0 |
| BACKDOOR-DATA | 99.9 | 99.1 | 27.6 | 10.3 | 10.1 | 10.1 | 9.9 |
| MODEL-REPLACE | 83.7 | 54.2 | 0.0 | 10.1 | 9.9 | 10.0 | 9.8 |
| DBA | 100.0 | 63.0 | 100.0 | 10.2 | 10.5 | 10.1 | 10.0 |
| AVERAGE | 73.4 | 56.6 | 34.4 | 10.2 | 10.1 | 10.1 | **9.9** |

(b) CIFAR-10 (0.1, 20%)

| ATTACKS | FEDAVG | MEDIAN | KRUM | FLTRUST | FLAME | FLD | FLAD |
|---|---|---|---|---|---|---|---|
| NONE | 10.0 | 9.9 | 10.2 | 10.0 | 10.1 | 10.0 | 10.0 |
| BACKDOOR-DATA | 57.2 | 10.2 | 10.0 | 11.0 | 10.2 | 10.4 | 11.0 |
| MODEL-REPLACE | 58.0 | 9.6 | 10.5 | 10.8 | 9.8 | 9.9 | 10.3 |
| DBA | 59.2 | 10.1 | 9.9 | 12.0 | 10.0 | 10.1 | 9.6 |
| AVERAGE | 46.1 | 10.0 | 10.2 | 11.0 | 10.0 | 10.1 | **10.0** |




(a) MNIST (0.1, 50%) (b) MNIST (0.8, 20%)

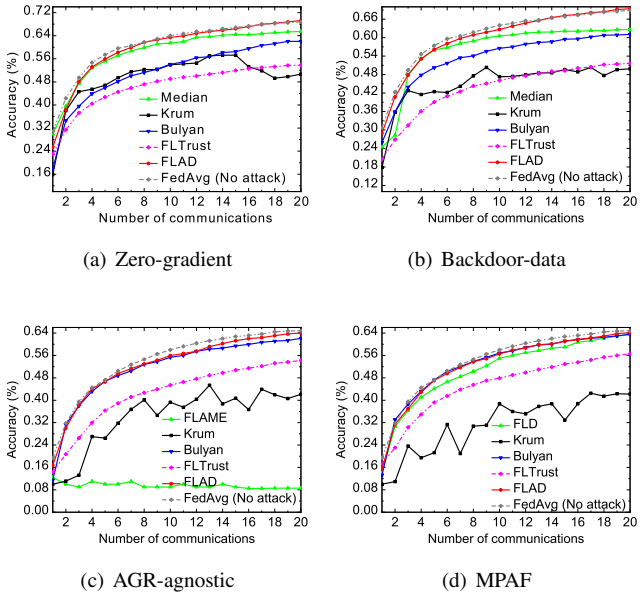
(c) CIFAR-10 (0.1, 20%) (d) CIFAR-10 (0.5, 20%)

Fig. 5. Global accuracy of FLAD under different attacks.

**Generalizability.** FLAD has been experimentally demonstrated to be applicable across various scenarios, including different numbers of clients and varying local training epochs. As shown in Figure 7(a), when the number of clients increases from 50 to 1000, only a marginal decrease in accuracy is observed, primarily due to the increasing dispersion of data. The overall accuracy improves with an increase in the number of local training epochs as shown in Figure 7(b). Furthermore,

TABLE V
DETECTION ACCURACY IN RWD, FLAME, FLD AND FLAD, WHERE *FPR* IS THE RATE OF HONEST NODES MISCLASSIFIED AS MALICIOUS, *FNR* IS THE RATE OF MALICIOUS NODES MISCLASSIFIED AS HONEST, *Recall* IS THE RATE OF MALICIOUS NODES CORRECTLY DETECTED, AND *Precision (Pre)* IS THE RATE OF PREDICTED MALICIOUS NODES THAT ARE ACTUALLY MALICIOUS.

| ATTACKS | RWD | | | | FLAME | | | | FLD | | | | FLAD | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FPR | FNR | RECALL | PRE | FPR | FNR | RECALL | PRE | FPR | FNR | RECALL | PRE | FPR | FNR | RECALL | PRE |
| GAUSSIAN | 0.0 | 0.0 | 100.0 | 100.0 | 63.1 | 27.5 | 72.5 | 27.7 | 6.5 | 0.0 | 100.0 | 92.9 | 1.3 | 0.0 | 100.0 | 96.2 |
| SIGN-FLIPPING | 100.0 | 100.0 | 0.0 | 0.0 | 59.2 | 18.0 | 82.0 | 31.6 | 4.0 | 0.0 | 100.0 | 95.6 | 0.5 | 0.0 | 100.0 | 98.5 |
| ZERO-GRADIENT | 100.0 | 100.0 | 0.0 | 0.0 | 67.9 | 13.5 | 86.5 | 29.8 | 4.4 | 0.0 | 100.0 | 94.8 | 0.3 | 0.0 | 100.0 | 99.0 |
| BACKDOOR-DATA | 100.0 | 100.0 | 0.0 | 0.0 | 83.6 | 30.5 | 69.5 | 21.7 | 2.1 | 0.0 | 100.0 | 97.1 | 0.5 | 0.0 | 100.0 | 98.5 |
| MODEL-REPLACE | 0.0 | 0.0 | 100.0 | 100.0 | 65.3 | 46.0 | 54.0 | 21.6 | 5.3 | 0.0 | 100.0 | 93.7 | 0.3 | 0.0 | 100.0 | 99.0 |
| DBA | 100.0 | 100.0 | 0.0 | 0.0 | 75.7 | 19.0 | 81.0 | 26.3 | 7.8 | 0.0 | 100.0 | 90.3 | 1.7 | 0.0 | 100.0 | 95.2 |
| AVERAGE | 66.7 | 66.7 | 33.3 | 33.3 | 69.1 | 25.8 | 74.3 | 26.5 | 5.0 | 0.0 | 100.0 | 94.1 | **0.8** | **0.0** | **100.0** | **97.7** |

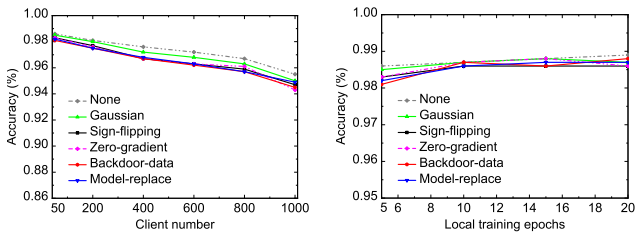



(a) Zero-gradient (b) Backdoor-data

(c) AGR-agnostic (d) MPAF

Fig. 6. Global accuracy versus interaction number for different defences against different attacks in CIFAR-10(0.1, 20%) (a) and (b), CIFAR-10(0.5, 20%) (c) and (d).

FLAD is capable of maintaining a high global accuracy even when the proportion of malicious clients reaches 50%, as discussed in Section V-B4.




(a) Clients (b) Local training epochs

Fig. 7. Global model accuracy with different numbers of clients/local training epochs under different attacks on MNIST (0.1, 20%).

*2) Malicious Clients Detection Accuracy:* The defense idea of FLAD is to detect and reject malicious gradients and to aggregate only honest local gradients. RWD [34], FLAME [8] and FLD [32] also aim at detecting and rejecting malicious gradients. Table V shows average malicious detection results during client-server interactions in MNIST (0.1, 20%). As shown in Table V, RWD works only in Gaussian and model-replacement attack and FLAME is not capable of identifying malicious nodes completely. Although both FLD and FLAD achieve 100% recall, FLAD demonstrates higher precision and lower FPR. Although FLAD may have cases of false positives, it can maintain a relatively high precision in our experiments. FLAD can successfully resist attacks, outperforming RWD, FLAME and FLD significantly.

*3) Comparing Different Variants of FLAD:* As shown in Figure 1, the FLAD has three key components: server dataset, Feature Extraction Model, and DBSCAN clustering. According to the usage characteristics of each component, four variants of FLAD are considered for ablation experiments.

- **FLAD-onlyServer**: Only the server dataset is utilized for training to evaluate the performance of the server model, i.e., direct aggregation $W_s^t$ in Algorithm 1.
- **FLAD-withServer**: The server model gradients $W_s$ are aggregated along with the honest gradients, i.e. add the model of server dataset for aggregation.
- **FLAD-NoDBSCAN**: $\boldsymbol{\mu}$, $\boldsymbol{eps}$ is the output of Algorithm 1. If the feature extraction value of local gradient is in $[\boldsymbol{\mu} - \boldsymbol{eps}, \boldsymbol{\mu} + \boldsymbol{eps}]$, it is honest gradient, otherwise it is malicious gradient.
- **FLAD-NoFCNN**: The server does not use Feature Extraction Model and directly adopts DBSCAN to cluster. The parameter $eps$ is the maximum value of $L_2$ norm between any two server model gradients.

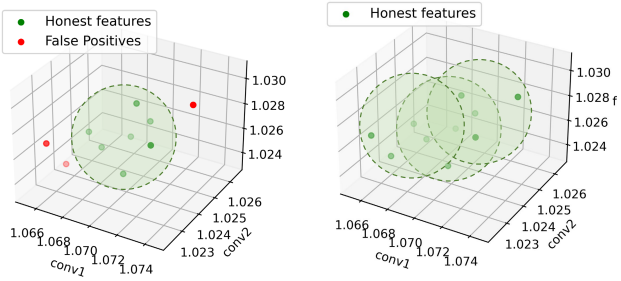
The results in Table VI demonstrate that FLAD outperforms the four variants. Due to the limited size of the server dataset and the fewer training epochs of the server model compared to the local models on clients, both the model obtained by aggregating the server gradients with honest gradients in FLAD-withServer and the global model trained solely on the server dataset in FLAD-onlyServer experience a decline in accuracy. FLAD-NoDBSCAN direct use of range partitioning may ignore a part of honest local gradients due to data heterogeneity, as shown in Figure 8. Therefore, malicious detection effect of FLAD is better than FLAD-NoDBSCAN. In FLAD-NoFCNN, since local gradients belong to high-

TABLE VI
EXPERIMENTAL RESULTS OF DIFFERENT FLAD VARIANTS UNDER DIFFERENT ATTACKS IN MNIST (0.8, 20%). THE RESULTS ARE PRESENTED AS GLOBAL ACCURACY (ACC) AND CORRECT DIVISION RATE (DIV) WHICH INDICATES THE RATIO OF NODES CORRECTLY DIVIDED AMONG ALL NODES.

| VARIANTS | NO ATTACK | | GAUSSIAN | | SIGN-FLIPPING | | ZERO-GRADIENT | | AGR-AGNOSTIC | | MPAF | | AVERAGE | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ACC | DIV | ACC | DIV | ACC | DIV | ACC | DIV | ACC | DIV | ACC | DIV | ACC | DIV |
| FLAD-ONLYSERVER | 72 | - | - | - | - | - | - | - | - | - | - | - | 72 | - |
| FLAD-WITHSERVER | 94 | 100 | 94 | 100 | 95 | 100 | 95 | 100 | 93 | 98 | 95 | 100 | 94 | 100 |
| FLAD-NODBSCAN | 94 | 90 | 94 | 87 | 32 | 9 | 93 | 94 | 95 | 78 | 94 | 58 | 84 | 69 |
| FLAD-NOFCNN | 46 | 9 | 38 | 30 | 56 | 29 | 65 | 30 | 92 | 49 | 94 | 50 | 65 | 33 |
| FLAD | 95 | 100 | 95 | 100 | 95 | 100 | 95 | 100 | 95 | 100 | 95 | 100 | 95 | 100 |

TABLE VII
THE TIME (IN SECONDS) REQUIRED FOR THE SERVER TO AGGREGATE THE GLOBAL GRADIENT PER ROUND UNDER MPAF ATTACK IN DIFFERENT DEFENSES. FOR FLTRUST AND FLAD, TIMES ARE GIVEN AS "SERVER TRAINING/AGGREGATION". THE LAST COLUMN INDICATES CLIENT LOCAL TRAINING TIME PER ROUND, WHICH CAN RUN PARALLEL TO SERVER TRAINING.

| DATASET | FEDAVG | BULYAN | KRUM | FLAME | FLD | FLTRUST | FLAD | LOCAL TRAINING/ROUND |
|---|---|---|---|---|---|---|---|---|
| MNIST (0.8, 20%) | 0.01 | 0.56 | 0.57 | 0.66 | 0.94 | 0.40/0.08 | 0.69/0.04 | 0.25 |
| FEMNIST (20%) | 0.02 | 0.77 | 0.81 | 0.87 | 1.06 | 0.58/0.12 | 0.74/0.04 | 0.36 |
| CIFAR-10 (0.5, 20%) | 0.06 | 12.54 | 11.53 | 1980.77 | 10.9 | 1.09/79.02 | 1.65/0.07 | 1.90 |



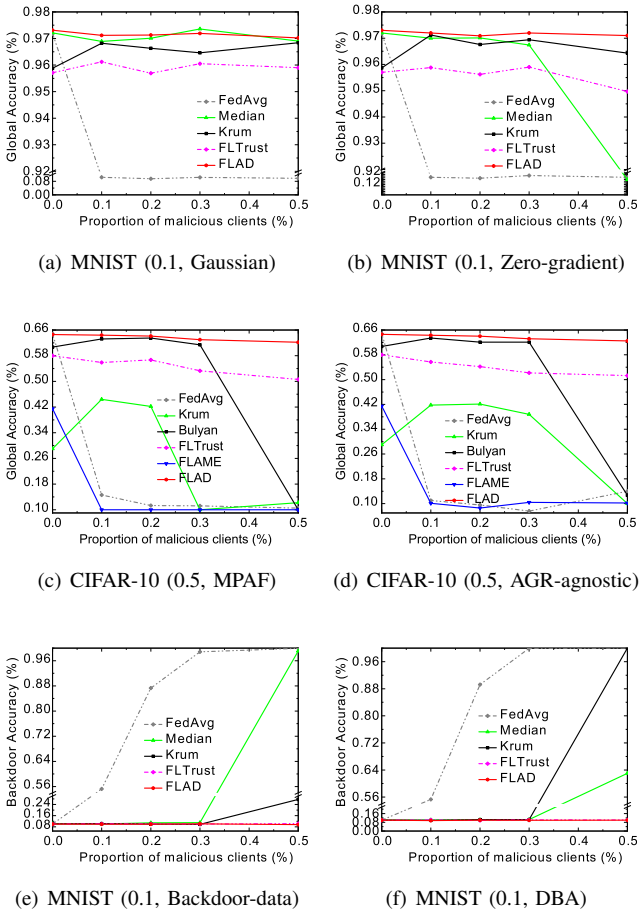

(a) FLAD-NoDBSCAN (b) FLAD

Fig. 8. Comparison of feature vector classified by FLAD-NoDBSCAN and FLAD (with DBSCAN).

dimensional vectors, the direct use of DBSCAN to classify high-dimensional local gradients without using FEM dimensionality reduction will bring huge errors. In general, FLAD outperforms four ablation variants. Both DBSCAN and neural network Feature Extraction Models play an integral role in the performance improvement of FLAD.

*4) Proportion of Malicious Client:* The global and backdoor accuracy of FLAD hardly changes as the proportion of attackers increases, similar to FedAvg without attacks in Figure 9. Although the global accuracy of FLTrust remains almost stable as the proportion of attackers increases, it is significantly lower than that of FLAD. The global accuracy of Median and Bulyan drops sharply when the proportion reaches 50%. Therefore, FLAD can tolerate up to 50% of malicious clients and has a large advantage over other defenses.

*5) Comparison of Aggregation Time:* As shown in Table VII, FLAD demonstrates significantly lower aggregation time compared to Bulyan, Krum, FLAME and FLD, showcasing a clear advantage in computational efficiency. While FLAD incurs slightly higher server model training time than FLTrust (e.g., 0.69s vs. 0.40s for MNIST and 1.65s vs. 1.09s for CIFAR-10), it exhibits a substantial advantage in aggregation time. Notably, on the complex CIFAR-10, FLAD maintains a low detection and aggregation time 0.07s, which is only a

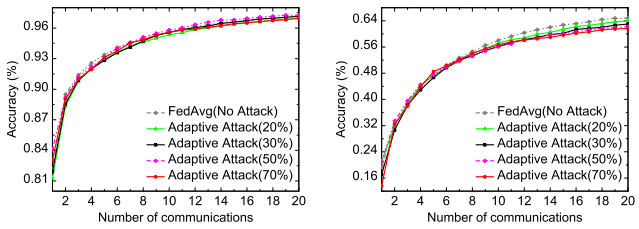



(a) MNIST (0.1, Gaussian) (b) MNIST (0.1, Zero-gradient)

(c) CIFAR-10 (0.5, MPAF) (d) CIFAR-10 (0.5, AGR-agnostic)

(e) MNIST (0.1, Backdoor-data) (f) MNIST (0.1, DBA)

Fig. 9. Global and backdoor accuracy of different defenses against different attacks across varying proportions of malicious clients.

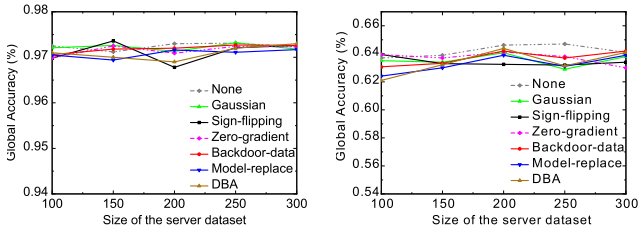



(a) MNIST (0.1) (b) CIFAR-10 (0.5)

Fig. 10. Global accuracy under adaptive attacks with different proportion of malicious clients.

fraction of FLTrust's 79.02s. Moreover, server model training is independent of the clients and can be executed in parallel with local client training. For more complex datasets, where client local training time exceeds server model training time, the actual time consumption of FLAD and FLTrust depends primarily on their aggregation time. FLAD's aggregation time is significantly lower than FLTrust's. Overall, FLAD achieves efficient aggregation while greatly reducing the computational overhead for detection and aggregation on the server.

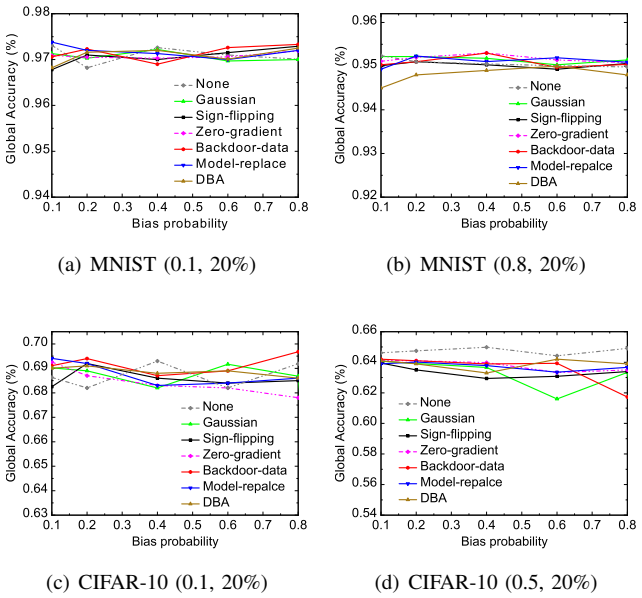
*6) Evaluation of Local Model Adaptive Attack:* Adaptive attack on FLAD is described in Algorithm 5. In Figure 10, even with malicious client proportions reaching 70%, FLAD's convergence rate and accuracy under adaptive attack resemble those of FedAvg without attacks. Therefore, the current state-of-the-art local model adaptive attack [14] proves ineffective against FLAD. When the attack strength is weak enough for FLAD to misjudge the malicious gradients, the malicious gradients are close enough to honest gradients to have almost no effect on the accuracy of aggregated global model.




(a) MNIST (0.1, 20%) (b) CIFAR-10 (0.5, 20%)

Fig. 11. The impact of server dataset size on global accuracy under different attacks.

*7) Impact of the Server Dataset:* We evaluate the impact of size and bias probability $P$ of $D_s$ on FLAD's global accuracy in Figure 11 and Figure 12. Figure 11 shows the global accuracy of FLAD under different attacks as the $D_s$ increases from 100 to 300, where $D_s$ is sampled uniformly at random using Case I in Section V-A1. We observe that the global accuracy barely changes as the size of $D_s$ changes, fluctuating slightly around 97.1% and 63.4% similar to FedAvg without attacks. This may be due to the fact that FLAD can successfully divide the malicious and honest clients using a small number of server dataset. Figure 12 shows that for common attacks, changes of




(a) MNIST (0.1, 20%) (b) MNIST (0.8, 20%)

(c) CIFAR-10 (0.1, 20%) (d) CIFAR-10 (0.5, 20%)

Fig. 12. The impact of server dataset bias probability on global accuracy under different attacks.

bias probability $P$ of $D_s$ do not affect the global accuracy, which is similar to FedAvg without attacks.

According to the experimental part of FLTrust, observing the global accuracy of FLTrust with different bias probabilities $P$, we find that FLTrust is more sensitive to changes in bias probabilities compared to FLAD. For MNIST (0.1), when $P$ reaches 0.8, the test error rate of FLTrust under targeted attack is as high as 78% (global accuracy 22%) while FLAD still maintains 97%. This may be because FLAD learns honest features obtained from clean data, whereas FLTrust directly uses clean data to bootstrap trust. Therefore, compared with FLTrust, which also uses server dataset to bootstrap trust, our FLAD is more robust to the distribution transformations of server dataset.

## VI. PFLAD: Privacy-Preserving FLAD

### A. Privacy-Preserving FL

Although FL prevents clients from sharing their private data directly, the process of exchanging parameters can also leak the client's private information to attackers, resulting in *inference attack* [3], such as DLG [42], iDLG (improved deep leakage from gradients) [43] and GRNN (generative regression neural network) [44]. These inference attacks highlight the need for protection of parameters such as transmission gradients in FL.

Most existing privacy-preserving FL is based on well-known privacy-preserving techniques, including Homomorphic Encryption (HE), Secure Multi-Party Computation (SMPC), and Differential Privacy (DP) [3], [45]. PEFL [5] is a privacy-enhancing FL that uses Paillier encryption, but it requires to encrypt and decrypt up to four times between the server and the cloud platform in each iteration. BREA [46] uses the Feldman verifiable secret sharing, but it introduces significant

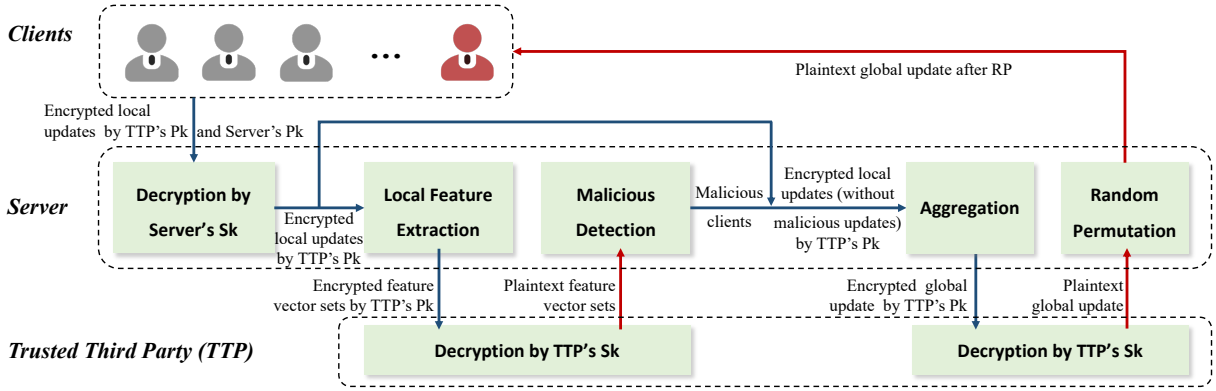



Fig. 13. The framework of PFLAD.

multi-party communication and computational overhead. DP-FedAvg [47] uses DP with adaptive clipping, which protects privacy but introduces the accuracy penalty, while not being resistant to poisoning attacks. Moreover, there are privacy-preserving FL such as FedCG [48] and PPCL [49] based on the FL training itself. However, few schemes can achieve both Byzantine-robust against model poisoning attacks and privacy-preserving against inference attacks for FL.

### B. The Framework of PFLAD

We use CKKS homomorphic encryption [9] which can handle floating-point numbers for transmission gradient privacy protection, thus achieving privacy-preserving FL. Moreover, in order to maintain local gradients secret to semi-honest clients and server, a Trusted Third Party (TTP) needs to be introduced for homomorphic encryption and decryption.

Figure 13 shows the framework of PFLAD on the basis of FLAD, where red arrows represent plaintext and blue arrows represent ciphertext. Specifically, Client $i$ encrypts local gradients $P_{ks}(P_{kt}(w_i^t))$ using the public key of TTP $P_{kt}$ and server $P_{ks}$. After decryption by $S_{ks}$, the server uses homomorphic technology to input $P_{kt}(w_i^t)$ into FEM to obtain $P_{kt}(\text{Pro}_i^t)$, which is then sent to TTP for decryption $S_{kt}(P_{kt}(\text{Pro}_i^t)) = \text{Pro}_i^t$. TTP returns $\text{Pro}_i^t$ to server and Algorithm 3 is executed to get malicious clients. After rejecting malicious updates, the server obtains aggregation global gradients $P_{kt}(W^t) = \text{FedAvg}(P_{kt}(w_i^t))$, $i \in [1, n-c]$. $P_{kt}(W^t)$ is sent to TTP for decryption, and then TTP uses $P_{ks}$ to encrypt $P_{ks}(W^t)$ preventing direct leakage of the original global gradients $W^t$ to clients. If semi-honest clients collude with each other, they may infer honest gradients from $W^t$, thus causing indirect information leakage for honest clients. Therefore, the server uses Random Permutation (RP) [49] to scramble global model $W^t$ to mitigate privacy leakage caused by semi-honest clients. Finally, the server returns the global model after RP $W_{RP}^t$ to clients.

**Random Permutation (RP).** Drawing on the PPCL [49], we use the RP of model parameters, which takes advantage of the computational characteristics of neural networks to achieve global model parameter transformation without affecting accuracy. RP is mainly applicable to scenarios with global models consisting of convolutional neural networks (CNN) and fully connected neural networks (FCNN). Specifically, let $f_i \in \mathbb{R}^{n_i \times n_{i+1}}$ denote the convolution kernel with input channel $n_i$ and output channel $n_{i+1}$, and consider $f_i$ as a two-dimensional matrix with $n_i \times n_{i+1}$. After randomizing the column index $n_{i+1}$ of $f_i$, the row index $n_{i+1}$ of $f_{i+1} \in \mathbb{R}^{n_{i+1} \times n_{i+2}}$ needs to be permuted in the same order. $f_{i+1}$ can continue to randomize the column index $n_{i+2}$, then $f_{i+2}$ row indexes will be permuted accordingly. Moreover, the RP can also be used for FCNN, where the weights and bias of the network are disrupted in the same order. Since the order of the global model with RP at each interaction is random, the semi-honest or malicious clients cannot obtain more sensitive information from global parameters $W^t$.

**Remark.** FEM and feature extraction values are in plaintext in PFLAD. However, recovering original $d$-dimensional gradients from one-dimensional feature extraction value is relative to solving a $d$-element primary equation, which is impossible in theory.

### C. Security Analysis of PFLAD

The privacy and semantic security of the CKKS homomorphic encryption against chosen-plaintext attacks (IND-CPA security) guarantee the strong security of PFLAD [9]. In this section, we provide a simulator-based hybrid proof to further demonstrate that during the execution of PFLAD, the joint view of the server and TTP will not reveal any user's private data, except for the privacy information inferred from the computed results. Consistent with the threat model and related assumptions in previous literature [5], [49], we make the following assumptions.

**Assumption 3.** *According to the threat model and the definition of the TTP, we assume that both the server and TTP are honest and curious, but the TTP will not engage in collusion attacks with the server or any client.*

**Assumption 4.** *We assume the presence of malicious attackers among the clients, while all clients are assumed to be honest and curious.*

**Proposition 2(Honest but Curious Server and TTP).** *Given a client set $U = \{U_1, \cdots, U_n\}$ and a set $C = \{S, TTP\}$, where $S$ represents the server. Define $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U}}$ as a random*

*variable representing the joint view of the server and TTP in $C$ during the actual execution of PFLAD. There exists a Probabilistic Polynomial Time (PPT) simulator* $\mathbf{SIM}$ *such that the output of* $\mathbf{SIM}$ *is computationally indistinguishable from* $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U}}$*:*

$$\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U}} \equiv \mathbf{SIM}, \tag{18}$$

*where $\equiv$ denotes computational indistinguishable.*

**Proof.** $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U}}$ encompasses all internal states and messages received by $C$ during the execution of PFLAD. We employ the standard hybrid proof method similar to [50] to prove this proposition. We perform a series of subsequent modifications and operations on the random variables within $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U}}$ and define a PPT simulator $\mathbf{SIM}$ such that the output of $\mathbf{SIM}$ is computationally indistinguishable from $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U},\kappa}$. The detailed proof is as follows.

**Hybrid 1.** Initialize a random variable to ensure that it is statistically indistinguishable from $\mathbf{REAL}_{\mathbf{C}}^{\mathbf{U}}$ in distribution.

**Hybrid 2.** In this hybrid, we simulate the change in the behavior of client $U_i \in U$ such that each client $U_i$ encrypts a randomly selected vector $\boldsymbol{\alpha_i}$ using the TTP's public key $P_{kt}$ instead of the original $\boldsymbol{w_i}$ (for honest clients) or $\boldsymbol{X_i}$ (for malicious clients). Since only the content of the ciphertext is altered, the IND-CPA security property of HE and the non-collusion setting between the server and TTP guarantee that this hybrid is indistinguishable from the previous one.

**Hybrid 3.** Building upon Hybrid 2, in this hybrid, we encrypt the encrypted random vectors $P_{kt}(\boldsymbol{\alpha_i})$ using the server's public key $P_{ks}$, rather than the original $P_{kt}(\boldsymbol{w_i})$ or $P_{kt}(\boldsymbol{X_i})$. Since only the content of the ciphertext is altered, this hybrid is indistinguishable from the previous one.

**Hybrid 4.** In this hybrid, we simulate the server's execution of local feature extraction algorithm. We modify the input to the algorithm, using $P_{kt}(\boldsymbol{\alpha})$ in place of $P_{kt}(\boldsymbol{w})$, where $\boldsymbol{\alpha} = \{\boldsymbol{\alpha_1}, \cdots, \boldsymbol{\alpha_n}\}$ and $\boldsymbol{w} = \{\boldsymbol{w_1}, \cdots, \boldsymbol{w_n}\}$. Since only the content of the ciphertext is altered, when performing homomorphic computation $\boldsymbol{F_{fc}}(P_{kt}(\boldsymbol{\alpha}))$, the IND-CPA security property of HE and the non-collusion setting between the server and TTP guarantee that this hybrid is indistinguishable from the previous one.

**Hybrid 5.** In this hybrid, we simulate the server's local gradient aggregation. For honest clients $U_i \in U$ where $i \in [1, n-c]$, we perform homomorphic addition and multiplication using randomly encrypted vectors $P_{kt}(\boldsymbol{\alpha_i})$ to obtain the aggregated value, instead of using $P_{kt}(\boldsymbol{w_i})$ for computation. Since only the content of the ciphertext is altered, when performing homomorphic computation $\frac{1}{n-c}\sum_{i=1}^{n-c}[P_{kt}(\boldsymbol{\alpha_i})]$, the IND-CPA security property of HE and the non-collusion setting between the server and TTP guarantee that this hybrid is indistinguishable from the previous one.

The proof proves that there exists a simulator $\mathbf{SIM}$ as described above so that its output is computationally indistinguishable from the output of $\mathbf{REAL}$. Therefore, PFLAD maintains privacy security for the server and TTP, meaning that honest but curious server and TTP cannot gain access to the private gradient information of clients.

**Proposition 3 (Clients).** *PFLAD provides privacy security for clients, meaning that malicious or honest but curious clients cannot compromise the privacy of other clients.*

**Proof.** Malicious or honest but curious clients attempt to infer the presence of target samples in the training dataset by exploiting incorrect gradient vectors [51]. This attack heavily relies on aggregation rules that involve only averages. In PFLAD, the globally aggregated gradients are sent to each client after being randomly permuted, shuffling the specific data arrangement within the gradients. As a result, malicious or honest but curious clients cannot infer whether target samples are used in the training. Therefore, PFLAD ensures privacy security for clients.

In conclusion, PFLAD enables the protection of FL transmission gradients and provides privacy security for all clients, server and TTP.

### *D. Performance Analysis of PFLAD*

Experiments[4] on MNIST show that PFLAD achieves global model accuracy under different attacks similar to FLAD in Table VIII.

TABLE VIII
GLOBAL MODEL ACCURACY OF PFLAD UNDER DIFFERENT ATTACKS IN MNIST(0.1) AND MNIST(0.8).

| | NONE | GAUSSIAN | SIGN-FLIPPING | ZERO-GRADIENT | MODEL-REPLACE |
|---|---|---|---|---|---|
| MNIST (0.1, 20%) | 97.3 | 97.1 | 97.0 | 97.1 | 97.2 |
| MNIST (0.1, 50%) | 97.3 | 97.0 | 97.0 | 97.1 | 96.9 |
| MNIST (0.8, 20%) | 95.1 | 95.1 | 95.0 | 95.1 | 95.0 |

## VII. FURTHER DISCUSSIONS AND LIMITATIONS

**Feature Extraction Model Training.** The feature extraction model has to be trained in every iteration. The global model converges differently in each iteration, leading to varying honest gradient features. Therefore, it is necessary to train the feature extraction model for honest gradients in every iteration, which can be achieved for servers with abundant computational resources. However, we acknowledge that it is less friendly to computing-constrained servers.

**The Quality of Server Dataset $D_s$.** We acknowledge that the performance of FLAD suffers when $D_s$ is poisoned. However, our FLAD only needs a small amount of clean $D_S$, easily obtainable by the server through methods like manual data generation and labeling [6].

**Privacy in Feature Extraction Values.** In our PFLAD, recovering the original $d$-dimensional local gradient vector from the one-dimensional feature extraction value is relative to solving a $d$-element primary equation. It is impossible to solve this equation in theory. However, it is an interesting future work that clients potentially leverage external knowledge to solve the equation, thus leading to a privacy breach of the original gradients.

[4]**CKKS Settings.** Polynomial mod degree 8192. Modal size of the coefficients of the polynomial [40, 21, 21, 21, 21, 21, 21, 21, 40]. Global scaling factor $2^{21}$. Galois Keys are introduced for batch rotation.

## VIII. Conclusion and Future Work

We proposed FLAD, a Byzantine-robust FL defence based on gradient feature anomaly detection. Specifically, the server collects a small clean server dataset to bootstrap trust and trains feature extraction models (FEMs) to extract client local gradients features. DBSCAN is used to evaluate the similarity between the feature extraction values of clients. Through cosine similarity and relative length difference with standard features, malicious clients can be identified. Extensive experiments on non-IID and IID datasets showed that FLAD achieved higher accuracy, robustness, and efficiency compared to state-of-the-art defenses. Moreover, we implemented privacy-preserving FLAD (PFLAD) using CKKS and random permutation to ensure transmitted gradient privacy. In our future work, we will focus on improving the extraction accuracy and efficiency of FEMs, while minimizing the requirements on the server dataset, and will further explore ways to optimizing the efficiency of FLAD.

## Acknowledgments

This work was supported by the Science and Technology Innovation Plan of the Shanghai Science and Technology Commission in China under Grant number 24BC3200600.

## References

[1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," arXiv:1912.04977 [cs.LG], 2021. [Online]. Available: https://doi.org/10.48550/arXiv.1912.04977

[2] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 12:1–12:19, 2019. [Online]. Available: https://doi.org/10.1145/3298981

[3] L. Lyu, H. Yu, X. Ma, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, "Privacy and robustness in federated learning: Attacks and defenses," arXiv:2012.06337 [cs.CR], 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2012.06337

[4] W. Jiang, H. Li, S. Liu, X. Luo, and R. Lu, "Poisoning and evasion attacks against deep learning algorithms in autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4439–4449, 2020. [Online]. Available: https://doi.org/10.1109/TVT.2020.2977378

[5] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574–4588, 2021. [Online]. Available: https://doi.org/10.1109/TIFS.2021.3108434

[6] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "FLTrust: Byzantine-robust federated learning via trust bootstrapping," in *Proceedings of the 2021 Network and Distributed System Security Symposium*. ISOC, 2021. [Online]. Available: https://doi.org/10.14722/ndss.2021.24434

[7] C. Xie, M. Chen, P.-Y. Chen, and B. Li, "CRFL: Certifiably robust federated learning against backdoor attacks," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021, pp. 11 372–11 382. [Online]. Available: https://proceedings.mlr.press/v139/xie21a.html

[8] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni, F. Koushanfar, A.-R. Sadeghi, and T. Schneider, "FLAME: Taming backdoors in federated learning," in *Proceedings of the 31st USENIX Security Symposium*. USENIX Association, 2022, pp. 1415–1432. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/nguyen

[9] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*. Springer, 2017, pp. 409–437. [Online]. Available: https://doi.org/10.1007/978-3-319-70694-8_15

[10] F. Lin, Q. Ling, and Z. Xiong, "Byzantine-resilient distributed large-scale matrix completion," in *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2019, pp. 8167–8171. [Online]. Available: https://doi.org/10.1109/ICASSP.2019.8683121

[11] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis, "Federated variance-reduced stochastic gradient descent with robustness to Byzantine attacks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4583–4596, 2020. [Online]. Available: https://doi.org/10.1109/TSP.2020.3012952

[12] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 1544–1551, 2019. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33011544

[13] X. Cao and N. Z. Gong, "MPAF: Model poisoning attacks to federated learning based on fake clients," in *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop*. IEEE, 2022, pp. 3396–3404. [Online]. Available: https://doi.org/10.1109/CVPRW56347.2022.00383

[14] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to Byzantine-robust federated learning," in *Proceedings of the 29th USENIX Security Symposium*. USENIX Assciation, 2020, pp. 1605–1622. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/fang

[15] V. Shejwalkar and A. Houmansadr, "Manipulating the Byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *Proceedings of the 2021 Network and Distributed System Security Symposium*. ISOC, 2021. [Online]. Available: https://doi.org/10.14722/ndss.2021.24498

[16] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning*, 2012. [Online]. Available: https://icml.cc/2012/papers/880.pdf

[17] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security – ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I*. Springer, 2020, pp. 480–501. [Online]. Available: https://doi.org/10.1007/978-3-030-58951-6_24

[18] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948. [Online]. Available: https://proceedings.mlr.press/v108/bagdasaryan20a.html

[19] Z. Sun, P. Kairouz, A. T. Suresh, and H. B. McMahan, "Can you really backdoor federated learning?" arXiv:1911.07963 [cs.LG], 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1911.07963

[20] C. Xie, K. Huang, P.-Y. Chen, and B. Li, "DBA: Distributed backdoor attacks against federated learning," in *Proceedings of the 2019 International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rkgyS0VFvr

[21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[22] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. NeurIPS, 2017, pp. 119–129. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html

[23] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659. [Online]. Available: https://proceedings.mlr.press/v80/yin18a.html

[24] R. Guerraoui and S. Rouault, "The hidden vulnerability of distributed learning in Byzantium," in *Proceedings of the 35th International*

*Conference on Machine Learning*. PMLR, 2018, pp. 3521–3530. [Online]. Available: https://proceedings.mlr.press/v80/mhamdi18a.html

[25] S. Rajput, H. Wang, Z. Charles, and D. Papailiopoulos, "DETOX: A redundancy-based framework for faster and more robust gradient aggregation," in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. NeurIPS, 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/415185ea244ea2b2bedeb0449b926802-Abstract.html

[26] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1142–1154, 2022. [Online]. Available: https://doi.org/10.1109/TSP.2022.3153135

[27] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, "Analyzing federated learning through an adversarial lens," in *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 2019, pp. 634–643. [Online]. Available: https://proceedings.mlr.press/v97/bhagoji19a.html

[28] J. Bernstein, J. Zhao, K. Azizzadenesheli, and A. Anandkumar, "signSGD with majority vote is communication efficient and fault tolerant," arXiv:1810.05291 [cs.DC], 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1810.05291

[29] C. P. Wan and Q. Chen, "Robust federated learning with attack-adaptive aggregation," arXiv:2102.05257 [cs.LG], 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2102.05257

[30] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010. [Online]. Available: https://doi.org/10.1007/s10994-010-5188-5

[31] M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li, "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy*. IEEE, 2018, pp. 19–35. [Online]. Available: https://doi.org/10.1109/SP.2018.00057

[32] Z. Zhang, X. Cao, J. Jia, and N. Z. Gong, "FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, 2022, pp. 2545–2555. [Online]. Available: https://doi.org/10.1145/3534678.3539231

[33] S. Shen, S. Tople, and P. Saxena, "AUROR: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 2016, pp. 508–519. [Online]. Available: https://doi.org/10.1145/2991079.299112

[34] R. Ganjoo, M. Ganjoo, and M. Patil, *Mitigating Poisoning Attacks in Federated Learning*. Springer, 2022, pp. 687–699. [Online]. Available: https://doi.org/10.1007/978-981-16-7167-8_50

[35] P. Rieger, T. D. Nguyen, M. Miettinen, and A.-R. Sadeghi, "DeepSight: Mitigating backdoor attacks in federated learning through deep model inspection," in *Proceedings of the 2022 Network and Distributed System Security Symposium*. ISOC, 2022. [Online]. Available: https://doi.org/10.14722/ndss.2022.23156

[36] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN," *ACM Transactions on Database Systems*, vol. 42, no. 3, pp. 19:1–19:21, 2017. [Online]. Available: https://doi.org/10.1145/3068335

[37] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," arXiv:1212.5701 [cs.LG], 2012. [Online]. Available: https://doi.org/10.48550/arXiv.1212.5701

[38] Y. LeCun, C. Cortes, and C. J. C. Burges. (1998) MNIST handwritten digit database. The original URL did not have the dataset any longer. An alternative source is https://www.kaggle.com/datasets/hojjatk/mnist-dataset. [Online]. Available: http://yann.lecun.com/exdb/mnist

[39] A. Krizhevsky, V. Nair, and G. Hinton. (2009) The CIFAR-10 dataset. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[40] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar. FEMNIST dataset. [Online]. Available: https://github.com/TalwalkarLab/leaf/tree/master/data/femnist

[41] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv:1412.6572 [stat.ML], 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1412.6572

[42] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. NeurISP, 2019, pp. 8389:1–8389:11. [Online]. Available: http://papers.neurips.cc/paper/by-source-2019-8389

[43] B. Zhao, K. R. Mopuri, and H. Bilen, "iDLG: Improved deep leakage from gradients," arXiv:2001.02610 [cs.LG], 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2001.02610

[44] H. Ren, J. Deng, and X. Xie, "GRNN: Generative regression neural network–a data leakage attack for federated learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 13, no. 4, pp. 65:1–65:24, 2022. [Online]. Available: https://doi.org/10.1145/3510032

[45] H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, H. Möllering, T. D. Nguyen, P. Rieger, A.-R. Sadeghi, T. Schneider, and H. Yalame, "SAFELearn: Secure Aggregation for private FEderated Learning," in *Proceedings of the 2021 IEEE Security and Privacy Workshops*. IEEE, 2021, pp. 56–62. [Online]. Available: https://doi.org/10.1109/SPW53761.2021.00017

[46] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020. [Online]. Available: https://doi.org/10.1109/JSAC.2020.3041404

[47] G. Andrew, O. Thakkar, B. McMahan, and S. Ramaswamy, "Differentially private learning with adaptive clipping," in *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*. NeurISP, 2021, pp. 17 455–17 466. [Online]. Available: https://papers.neurips.cc/paper_files/paper/2021/hash/91cff01af640a24e7f9f7a5ab407889f-Abstract.html

[48] Y. Wu, Y. Kang, J. Luo, Y. He, L. Fan, R. Pan, and Q. Yang, "FedCG: Leverage conditional GAN for protecting privacy and maintaining competitive performance in federated learning," in *Proceedings of the 31st International Joint Conference on Artificial Intelligence*. IJCAI, 2022, pp. 2334–2340. [Online]. Available: https://www.ijcai.org/proceedings/2022/0324.pdf

[49] H. Yan, L. Hu, X. Xiang, Z. Liu, and X. Yuan, "PPCL: Privacy-preserving collaborative learning for mitigating indirect information leakage," *Information Sciences*, vol. 548, pp. 423–437, 2021. [Online]. Available: https://doi.org/10.1016/j.ins.2020.09.064

[50] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. H. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1364–1381, 2020. [Online]. Available: https://doi.org/10.1109/TDSC.2020.3005909

[51] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy*. IEEE, 2019, pp. 739–753. [Online]. Available: https://doi.org/10.1109/SP.2019.00065

**Peng Tang** is a postdoctoral in the School of Cyber Science and Engineering, at Shanghai Jiao Tong University. He received his M.S. degree in Computer Science from Beijing University of Posts and Telecommunications in 2017 and his Ph.D. degree in Cyber Security from Shanghai Jiao Tong University in 2022. His research focuses on privacy protection, data science, and AI security.

**Xiaoyu Zhu** received the B.S. degree in information security from Xidian University, Xi'an, China, in 2022. She is currently pursuing the master's degree in the School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. Her research focuses on privacy protection, data science, and AI security.

**Weidong Qiu** received the Ph.D. degree in Computer Software Theory from Shanghai Jiao Tong University, Shanghai, China, in 2001 and received the M.S. degree in Cryptography from Xidian University, Xi'an, China, in 1998. He is currently a professor and doctoral supervisor in the School of Cyber Science and Engineering, Shanghai Jiao Tong University. His main research areas include data science, privacy computing, cryptography and computer forensics.

**Zheng Huang** received the Ph.D. degree in Computer Science from Shanghai Jiao Tong University, Shanghai, China, in 2003. He is currently an Associate Professor in the School of Cyber security, Shanghai Jiao Tong University. His main research areas include Machine Learning and Security.

**Zhenyu Mu** received the B.S. degree in information security from Shanghai Jiao Tong University, Shanghai, China, in 2022. He is currently pursuing the master's degree in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. His research focuses on graph neural networks and data mining.

**Shujun Li** (M'2008, SrM'2012) is Professor of Cyber Security at the School of Computing and Director of the Institute of Cyber Security for Society (iCSS), University of Kent, U.K. He has published over 100 scientific papers, including five Best Papers. His research interests are mostly about inter-disciplinary topics related to cyber security and privacy, human factors, digital forensics and cybercrime, multimedia computing, AI and data science, and more recently education and learning. His work covers multiple application domains, including but not limited to digital health, smart cities, smart homes, and e-tourism. Professor Li is a Fellow of the BCS, and a co-founder and a Vice President of the Association of British Chinese Professors (ABCP). He received multiple awards and honours including the 2022 IEEE Transactions on Circuits and Systems Guillemin-Cauer Best Paper Award and a 2022 Research and Innovation Award from the IEEE SMC Society TCHS.