Journal of Information Security and Applications 94 (2025) 104210

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Robust clustering federated learning with trusted anchor clients

Maozhen Zhang [a], Yi Li [a], Fei Wei [b], Bo Wang [a,*], Yushu Zhang [c]

[a] Dalian University of Technology, Dalian, China
[b] Alibaba Group, Hangzhou, China
[c] Jiangxi University of Finance and Economics, Jiangxi, China

ARTICLE INFO

Keywords:
Federated learning
Model Poisoning attack
Malicious clients detection
Deep learning

ABSTRACT

Federated Learning (FL) is a distributed machine learning framework that has attracted widespread attention. However, its decentralized architecture makes it vulnerable to attack with malicious data or model injection. While existing methods are can defend against a limited number of malicious clients, the challenge of defending against model poisoning attacks from a large number of malicious clients remains an unresolved issue. To address these challenges. We propose the Robust Clustering Federated Learning with Trusted Anchor Clients, which aims to provide clean global models for specified trusted client's enterprise (trusted client as anchor client), even in the presence of a substantial number of malicious clients. Specifically, it performs classification by extracting clustering factors from the differences between anchor clients and other clients. It then identifies trustworthy clusters as aggregation clusters to identify the most likely benign clients. Extensive experiments on two datasets demonstrate that our method maintains robust defense efficacy, even in scenarios involving numerous malicious clients (more than 50%) or highly non-independent, non-identically distributed data.

1. Introduction

Federated Learning (FL) is a decentralized machine learning paradigm in which multiple devices or servers collaboratively train a global model while retaining their raw data locally [1–8]. By design, FL enhances data privacy and mitigates the need for centralized data collection. Specifically, in each training round, clients receive the current global model, perform local updates using their private data, and send the resulting model updates to the server. The server then aggregates these updates to obtain an improved global model. This updated global model is then broadcast back to the clients for the next training round [9,10]. Among the three canonical types of federated learning (FL)—horizontal, vertical, and federated transfer learning—this work focuses exclusively on horizontal FL. In horizontal FL, all participating clients share the same feature space (i.e., data have the same attributes) but possess different data samples. This setting has been widely adopted in real-world applications such as next-word prediction [1,11], credit scoring [12,13], and IoT device collaboration [14,15], and remains the dominant scenario in contemporary FL research and practice.

In FL, each client trains a model on its own local dataset and participates in collaborative training by uploading model updates to the server. However, due to its distributed nature, FL is vulnerable to model poisoning attacks [16–22], where malicious clients submit crafted updates to compromise the global model. Based on the attacker's goal, poisoning attacks are typically categorized as targeted or untargeted. In targeted attacks, the adversary attempts to manipulate the model's behavior on specific inputs or labels—for example, by forcing the model to misclassify particular instances (e.g., backdoor attacks). In contrast, untargeted poisoning attacks aim to degrade the overall performance of the global model without focusing on specific targets, often by injecting noise or inconsistent gradients to induce convergence failure or accuracy drop. Such untargeted attacks are especially damaging, as they can silently and broadly destabilize the training process. In this work, we focus on detecting and mitigating untargeted poisoning attacks, which remain a significant and underexplored threat in federated settings.

Untargeted poisoning attacks aim to hinder model convergence or drive training toward a sub-optimal solution [23–29]. These attacks can be realized through either data poisoning (e.g., manipulating local datasets) or model poisoning (i.e., submitting crafted malicious updates to the central server). Unlike targeted attacks that aim to manipulate predictions on specific inputs, untargeted attacks broadly degrade the overall performance of the global model, often in subtle ways that make detection more challenging. To mitigate poisoning threats, a variety of server-side robust aggregation strategies have been proposed, particularly for targeted attacks. These methods seek to approximate the true global update by applying statistical techniques such as the

* Corresponding author.
E-mail addresses: maozhenzhang@mail.dlut.edu.cn (M. Zhang), liyi@dlut.edu.cn (Y. Li), feiwei@alibaba-inc.com (F. Wei), bowang@dlut.edu.cn (B. Wang), zhangyushu@jxufe.edu.cn (Y. Zhang).

https://doi.org/10.1016/j.jisa.2025.104210

Available online 23 August 2025

coordinate-wise median or trimmed mean [30–33]. However, existing approaches predominantly rely on server-side filtering and do not incorporate client-side information or collaboration. Consequently, they suffer from two major limitations: (1) their robustness degrades significantly as the proportion of malicious clients increases, and (2) they exhibit poor performance under non-IID data distributions, where natural variations among benign client updates can resemble adversarial behavior. Designing effective defenses in federated learning thus remains a critical challenge, particularly under realistic assumptions of data heterogeneity and partial trust.

To address these challenges, we propose `TACRC-FL` (Trusted Anchor Client-guided Robust Clustering for Federated Learning), a novel cluster-based defense against model poisoning attacks. TACRC-FL leverages gradient-space representations of client updates to partition participants into distinct clusters, guided by trusted anchor clients that help distinguish between benign and malicious updates. Specifically, TACRC-FL introduces a novel gradient norm vector computation method, guided by the parameters of one or more trusted anchor clients. These anchor clients upload authenticated model updates through a tamper-proof verification mechanism, preventing hijacking or tampering. By using the anchor-provided reference, client updates are projected into a feature space, enabling classification into clusters. Each cluster is then assigned a trust score, and global aggregation is conducted with respect to these trust scores. A closely related method is FLTrust [34], which requires the server to hold an auxiliary dataset for maintaining a clean reference model. However, this assumption violates the standard FL paradigm, which prohibits direct access to raw data on the server side. In contrast, TACRC-FL requires no server-side dataset and remains effective under non-IID data distributions and high ratios of malicious participants. We consider a realistic deployment in which several state-owned (official) banks act as the initiating organizers of a federated-learning consortium together with smaller regional or privately-owned banks. The state-owned banks tightly audit and control the devices under their administration, so the clients belonging to them can serve as trusted anchor clients. In contrast, clients operated by smaller private banks undergo less stringent vetting and may include malicious participants. By letting the anchor clients from the state-owned banks guide the verification and aggregation processes, the consortium can effectively mitigate poisoning attempts originating from untrusted organizations while still benefiting from their data contributions, ultimately yielding a more robust global model.

In summary, we make the following contributions:

- We propose a `Tamper-proof Verification` mechanism that applies cryptographic key agreement to protect client model parameters in FL.
- We introduce a federated gradient norm vector computation method, guided by `Anchor client` model parameters. Based on this, we construct client clusters by measuring gradient feature discrepancies between anchor clients and other participants.
- We leverage trusted clients to identify and aggregate benign clusters, yielding a coarser yet robust global model while reducing communication overhead.
- We conduct extensive experiments on two benchmark datasets to evaluate the performance and robustness of `TACRC-FL` against several state-of-the-art model poisoning attacks. Our results show that `TACRC-FL` achieves up to a 10× improvement in defense effectiveness compared to existing approaches.

2. Background

2.1. Federated learning

In this work, we consider a typical FL setting as used in [1], which involves a central server s and N clients. Each client $i \in [\mathcal{N}]$ holds a local dataset D_i. The size of the dataset is denoted as $|D_i| = n_i$. Each local dataset may observe a different distribution, i.e., the local datasets are non-IID. The objective of FL is to coordinate the clients (with their local data) to train a global model $\boldsymbol{w}$. At each training iteration t, we use w_t to represent the global model and w_t^i to represent the local model updated by participant i. The server updates the global model by aggregating the local model updated with learning rate η.

Specifically, at the t-iteration, the FL system repeats the follow three steps to obtain the global model w_t from the current w_{t-1}.

Step I. The server sends the global model w_{t-1} from the previous round to the server.

Step II. Each client performs its local learning process with its local training data and the received global model w_{t-1}. During the local learning process, each client uses its own private data for random gradient descent:

$$w_t^i = w_{t-1}^i + \eta_i \cdot \nabla L_i(w_{t-1}; D_i)$$

where η_i is the learning rate of local model training and $\nabla L_i(w_{t-1}; D_i)$ denotes the gradient of local optimization loss. When the local training is complete, the client sends the latest local model w_t^i back to the server

$$w_t = w_{t-1} + \eta \cdot \sum_{i=1}^{\mathcal{N}} (w_t^i - w_{t-1}), \text{ for } t = 1, \ldots, T.$$

Where η is the learning rate of global model, it is usually calculated based on the number of local clients $|\mathcal{N}|$ and the number of samples $|D_i|$.

Step III. The server use aggregator $\mathcal{A}$ to aggregates local model updates obtain the updated global w_t.

2.2. HDBSCAN

The density-based spatial clustering of noise applications (DBSCAN) [35] is a clustering algorithm that uses a predefined maximum distance to determine whether two points belong to the same group. Extending DBSCAN, the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [36] determines a different maximum distance for each clustering group according to the density of different points. HDBSCAN avoids setting the maximum distance and the number of clusters in advance and can cluster adaptively in the iteration.

2.3. Byzantine-attack

The vulnerability of FL gives malicious clients ample opportunity to disrupt training. Attackers can poison local updates either by injecting new adversarial clients or by compromising originally benign ones. Most robust aggregation defenses—such as Krum, Trimmed Mean, and Median—assume that fewer than 50% of participants are malicious [30,31], yet scenarios in which adversaries form a majority remain largely unexplored. Recent adaptive attacks further exploit this gap: by tuning the magnitude and direction of poisoned gradients to mimic the benign majority, they can evade existing filters and still impair convergence [26]. Empirical studies confirm that seemingly simple strategies—such as adding bounded random noise [23] or inverting the signs of selected client updates [24]—can significantly degrade the performance of aggregation algorithms including FedAvg, Krum, Trimmed Mean, and Median. Consequently, even with current defenses in place, untargeted poisoning continues to pose a serious threat to model integrity and accuracy in federated learning.

2.4. Byzantine-robust FL methods

The model updates in Byzantine-robust FL are typically high-dimensional vectors, and identifying benign updates involves analyzing the relationships among these vectors. However, the excessive dimensionality of model updates incurs significant computational overhead. As previously mentioned, `Krum` [30] requires computing the Euclidean

Algorithm 1 TACRC-FL

Input: cluster C_j, client model w_i, anchor model w_j
Parameter: number of iterations T
Output: final global model w^*
for $t = 1$ to T **do**
  Verify if anchor model w_j is tampered using Algorithm 2 ▷ Tamper-proof check
  if verification passed **then**
    $C_1, \ldots, C_l \leftarrow \text{CLUSTER}(w_i, w_j)$ ▷ Use Algorithm 3
    $C_{\text{agg}} \leftarrow \text{DETECTCLUSTERS}(\{C_1, ..., C_l\})$ ▷ Use Algorithm 4
    $w^* \leftarrow \text{FEDAVG}(\{w_i \mid i \in C_{\text{agg}}\})$ ▷ Aggregate selected client updates
return w^*

Algorithm 2 Tamper-Proof Verification

Input: client u, masked model $[w_u^{local}]$, location index I_{index}, mask function $H(\cdot)$ ▷ Input parameters
Output: Boolean value (True / False) ▷ Verification result
$mask_u \leftarrow H(S^{sk})$ ▷ Generate mask using secret key
$w_u^{local} \leftarrow [w_u^{local}] - mask_u$ ▷ Recover the original local model
$(S^{pk}, S^{sk}) \leftarrow \text{KA.gen}(w_u^{local}, I_{index})$ ▷ Generate key pair
$\tilde{S}_{u,s} \leftarrow \text{KA.agg}(S^{pk}, S^{sk})$ ▷ Compute the agreement shared key
if $\tilde{S}_{u,s} = S_{u,s}$ **then** ▷ Check authentication match
  return True ▷ Verification passed
else
  return False ▷ Verification failed

distances between each client and all others, selecting the one with the smallest sum of distances to the $N-f-2$ nearest neighbors, where f is the number of malicious clients. `Trimmed Mean` [31] requires sorting values at each coordinate and removing extreme values, while `Median` [31] takes the coordinate-wise median across clients.

Detecting abnormal (i.e., potentially attacked) model updates can be formulated as a classification task. A natural approach is to adopt clustering-based classification [32,33,37,38]. Sattler et al. [33] explored the use of clustered FL under Byzantine threats by partitioning client updates based on cosine similarity. However, this method is only effective when the number of malicious clients is small. Tao et al. [32] proposed a robust aggregation algorithm based on IFCA (Iterative Federated Clustering Algorithm) [39], which incorporates coordinate-wise `Median` and `Trimmed Mean` into IFCA to enhance robustness. Nevertheless, the high dimensionality d still introduces significant learning challenges. Our proposed aggregation factor addresses this issue by reducing the dimensional burden encountered during clustering. However, both of the above methods still require full pairwise computation over all client updates. Moreover, they remain effective only when the proportion of malicious clients is relatively small; their performance degrades significantly when this proportion increases.

Some defenses have been proposed to handle large-scale malicious participation. For instance, `EVA Detection` [40] trains an autoencoder using an auxiliary dataset and detects potential attacks by comparing discrepancies between generated and real data. `FLTrust` [34] relies on a server-side trusted dataset for validating updates. However, these approaches typically depend on external datasets obtained outside the federation, which violates the core principle of FL—namely, that the server should not have access to raw client data.

3. Framework of TACRC-FL

3.1. Threat model

We follow a FL setup where a central server coordinates multiple distributed clients. Among the clients, an arbitrary number may be compromised and controlled by the adversary. The adversary's capabilities include full control over its own local data and model updates. However, the adversary has no access to the data or model parameters of benign clients. The adversary aims to influence the aggregation of the global model by manipulating its locally trained models. Specifically, the attacker's objective is to degrade the overall accuracy of the global model or steer its convergence toward a sub-optimal point.

3.2. Overview

As depicted in Algorithm 1, the workflow of `TACRC-FL` consists of the following steps: (1) Perform tamper-proof verification to determine whether the anchor model has been modified. (2) Extract clustering factors and perform clustering based on the differences between trusted anchor model updates and other client updates. (3) After clustering, aggregate the client updates within each cluster. Anchor clients are then used to evaluate the aggregated cluster models and assign trust scores. (4) In each iteration, select the cluster with the highest trust score, and use predefined thresholds δ_1 and δ_2 to determine additional trustworthy clusters. Clusters whose trust scores fall outside these threshold ranges are considered potentially malicious and are excluded from the aggregation list. An overview of `TACRC-FL` is shown in Fig. 1, and the full procedure is summarized as pseudo-code in Algorithm 1.

3.3. Tamper-proof verification

We employ the key agreement method [41] to implement a tamper-proof verification as depicted in Algorithm 2. In Fig. 2, we present the flowchart illustrating the process of key agreement and model masking in the proposed algorithm. Here, $u \in N$ is the client, s is the server. S represents a secret key, h and g are parameters in key exchange protocol, h is a randomly selected large prime number, and g is an integer selected as a generator in a finite field.

Key Agreement. The trusted anchor client u engages in key agreement with the server s to obtain the shared key $S_{u,s}$. Firstly, the trusted client and the server generate their respective private keys, denoted as S_u^{sk} and S_s^{sk}, respectively. Unlike the server, the trusted client randomly selects the parameter location of the model. Clinet generate h using the values of these model parameters. Then, it calculates the private key S_u^{sk} using the function **KA.gen** and records the index of the parameter location used as I_{index}. Subsequently, the trusted client and the server compute their respective public keys S_u^{pk} and S_s^{pk} from their private keys and then exchange them. Upon receiving the other party's public key, each party calculates the agreement shared key $S_{u,s}$.

Mask Model and Model Tampering Verification. The trusted client applies a random mask to the model using the shared key $S_{u,s}$, as illustrated in (1) and (2). The mask is generated from the shared key $S_{u,s}$ and added to the model parameters to obtain the masked model.

$$mask_u = H(w_u^{local}, S_{u,s}), \tag{1}$$

$$[w_u^{local}] = w_u^{local} + mask_u, \tag{2}$$

here, $H(\cdot)$ is the mask generation function. After applying the mask, the trusted client sends the masked model $[w_u^{local}]$, mask generation function $H(\cdot)$, private key calculation function **KA.gen**, and parameter

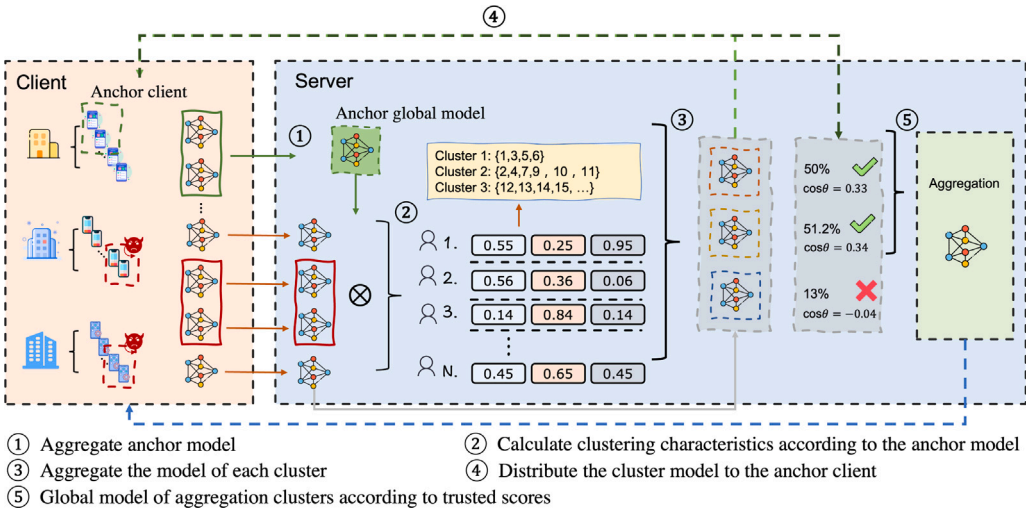


Fig. 1. Overview of TACRL-FL.

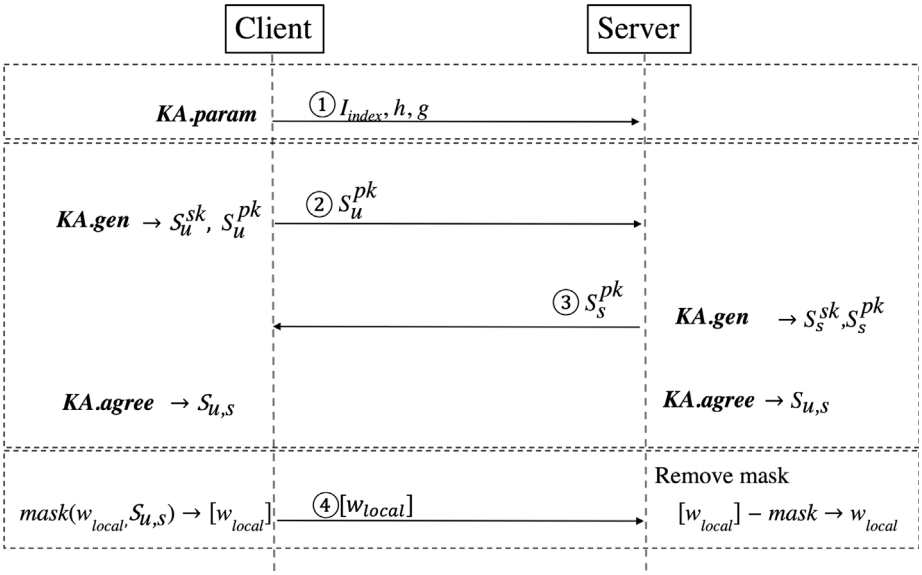


Fig. 2. Key agreement and model masking.

index I_{index} to the server. The server calculates the mask by using the shared key $S_{u,s}$ and removes the mask from the model to obtain w_u^{local}. It then employs the key calculation function **KA.gen** and the parameter index I_{index} to compute the agreement shared key $\tilde{S}_{u,s}$ between the trusted client and the server. To verify that the trusted client has not been tampered with, the consistency between $\tilde{S}_{u,s}$ and $S_{u,s}$ is checked. In addition, the mask of the model prevents malicious clients from accessing the trusted model so that malicious clients do not know the real update direction of the clean model to design more complex attacks. The server checks whether $\tilde{S}_{u,s}$ and $S_{u,s}$ are the same. The same calculated key means the model parameters have not been tampered with.

3.4. Clustering factor extraction and clustering

In FL, it is widely acknowledged that the distribution of model parameters reflects the variation in model training. The distinction between malicious and benign models lies in malicious participants injecting harmful information into the model, disrupting the model's convergence, or injecting backdoor tasks. Malicious attacks aim to make the global model deviate from the optimal model w^*, forcing the global model to converge toward a secondary solution or diverge. These attacks result in model updates that deviate more from the normal model or exhibit larger update gradients, as depicted in Fig. 3(a).

Based on the considerations above, we use a trusted cluster model as an anchor to indicate the correct direction. The specific process of clustering factor extraction is shown in Algorithm 3. As presented in the algorithm, we calculate the angle between the trusted model direction and other unknown attribute models to avoid deviating from the correct direction during model updates and optimization:

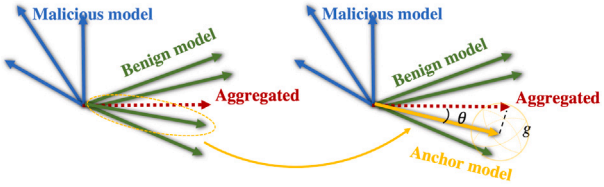
$$q_i = \frac{\langle \tilde{g}, g_i \rangle}{\|\tilde{g}\| \|g_i\|}, \tag{3}$$

here, $\tilde{g}$ denotes the aggregated gradient of the anchor cluster, and g_i is the update from client i. We further include a locality measure $\texttt{Neighbor}(g_i, k)$, defined as the average ℓ_2-distance to the k nearest updates, to capture neighborhood structure. Because benign clients share the same optimization goal as the anchor, their updates are mutually similar, whereas malicious updates deviate significantly. By combining q_i and the k-NN distance, our clustering step groups updates with similar orientations and magnitudes, as depicted in Fig. 3(b). Clusters

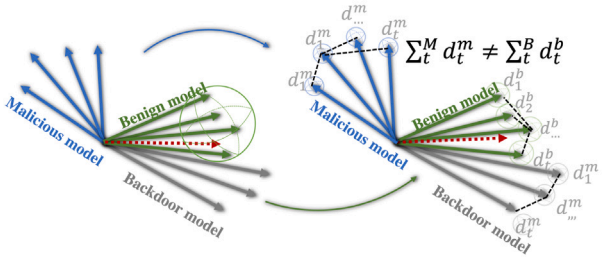
Algorithm 3 Clustering Factor Extraction and Clustering

Input: local models $\{w_t^i\}_{i=1}^{N_u}$, anchor models $\{w_t^j\}_{j=1}^{N_a}$, previous global model w_{t-1} ▷ Models at iteration t
Parameter: k (nearest neighbors) ▷ Used for k-NN distance
Output: clusters $C = \{C_1, \ldots, C_l\}$

$\tilde{w}_t \leftarrow w_{t-1} + \frac{1}{N_a}\sum_{j=1}^{N_a} w_{t-1}^j$ ▷ Anchor-guided pseudo model
$\tilde{g} \leftarrow \| \tilde{w}_t - w_{t-1} \|_2$ ▷ Anchor gradient norm
for $i \leftarrow 1$ to N_u **do** ▷ Process every non-anchor client
  $g_i \leftarrow \| w_t^i - w_{t-1} \|_2$ ▷ Client i gradient norm
  $\Delta g_i \leftarrow \| \tilde{g} - g_i \|_2$ ▷ Distance to anchor gradient
  $q_i \leftarrow \frac{\langle \tilde{g}, g_i \rangle}{\|\tilde{g}\| \, \|g_i\|}$ ▷ Cosine similarity with anchor
  $d_i^k \leftarrow \text{NEIGHBOR}(g_i, k)$ ▷ k-NN distance in gradient space
  $X_i \leftarrow \{\Delta g_i, q_i, d_i^k\}$ ▷ Feature vector for clustering
$C_1, \ldots, C_l \leftarrow \text{HDBSCAN}(\{X_i\}_{i=1}^{N_u})$ ▷ Density-based clustering
return $C_1, \ldots, C_l$ ▷ Return final clusters



(a) Guidelines for anchor client updates



(b) The gradient of the least k nearest neighbor

Fig. 3. Anchor client and neighbor distance.

that closely align with the anchor direction are deemed benign; those that do not are flagged as potentially malicious for further scrutiny.

$$d_{i,j} = \sqrt{(g_i - g_j)^2}, \qquad d_i^k = \min \sum_{j=1, j\neq i}^{k} d_{i,j}, \tag{4}$$

where $d_{i,j}$ is the distance between the model updates gradient g_i and other model updates gradient g_j, d_i^k is the k nearest distance of the model updates gradient g_i. Additionally, to prevent adversarial attacks that carefully manipulate the distances between models, we employ the update gradient difference between the anchor model and each unknown model as a clustering factor. This approach is intended to enhance the system's robustness against potential attacks. The model update gradient is calculated by (5).

$$g_i = \|w_t^i - w_{t-1}\|_2, \qquad \Delta g_i = \|\tilde{g} - g_i\|_2, \tag{5}$$

here, $\tilde{g}$ is the anchor model update gradient. To avoid the need for adjusting the neighborhood radius of DBSCAN in each training round, we adopt HDBSCAN to form clusters, eliminating the requirement for setting the density threshold ϵ in DBSCAN.

3.5. Benign model updates detection

We use clustering factors to partition models with unknown trustworthiness into multiple clusters. The model updates within each cluster are aggregated to form a cluster model, which is then sent to the anchor client for evaluation. The trust score s is assessed using the anchor client's local training set:

$$s_a^j \leftarrow \texttt{Anchor\_Client\_Evaluate}(w_t^j), \tag{6}$$

where Anchor_Client_Evaluate$(\cdot)$ denotes the evaluation of the cluster model w_t^j on the anchor client's private dataset. The resulting trust scores from all anchor clients are then uploaded to the server and averaged. Simultaneously, we compute the cosine similarity between each cluster model's update and the anchor model's update.

Based on the averaged trust scores and cosine similarities, we apply threshold values δ_1, δ_2, and δ_3 to filter out suspicious clusters. Clusters that meet the filtering criteria are added to the aggregation list. The objective is to identify and exclude potentially malicious updates from being aggregated, ensuring that the final global model is primarily influenced by benign participants.

The full detection process of benign cluster model updates is described in Algorithm 4. Specifically, TACRC-FL leverages anchor clients to assess and select clusters with high confidence in their integrity. This process serves as a robust pre-selection step for FedAvg, ensuring that aggregation in each round only includes the most reliable clusters.

Algorithm 4 Benign Model Updates Detection

Input: Clusters $\{C_1, \ldots, C_l\}$ of client updates
Parameter: anchor clients $A = \{1, \ldots, N_a\}$; thresholds $\delta_1, \delta_2, \delta_3$
Output: Aggregation candidate list C_{agg}
for each cluster C_j, $j = 1$ to l **do**
  $w_t^j \leftarrow w_{t-1} + \eta \sum_{i \in C_j} (w_t^i - w_{t-1})$ ▷ Compute pseudo aggregated model for cluster
Store all cluster models: $w_t^j \leftarrow \{w_t^1, \ldots, w_t^l\}$
for each cluster C_j, $j = 1$ to l **do**
  for each anchor client $a = 1$ to N_a **do**
    $s_a^j \leftarrow \text{AnchorClientEvaluate}(w_t^j)$ ▷ Evaluate model by anchor client
  $\overline{s}^j \leftarrow \frac{1}{N_a} \sum_{a \in A} s_a^j$ ▷ Mean anchor score for cluster
  $\cos \overline{\theta}^j \leftarrow \frac{1}{|C_j|} \sum_{i \in C_j} \cos \theta_i^j$ ▷ Mean cosine similarity
$\tilde{s} \leftarrow \max\{\overline{s}^1, \ldots, \overline{s}^l\}$ ▷ Max anchor score across clusters
$\cos \tilde{\theta} \leftarrow \max\{\cos \overline{\theta}^j\}$ ▷ Max mean cosine similarity
for each cluster C_j, $j = 1$ to l **do**
  if $\cos \overline{\theta}^j = \cos \tilde{\theta}$ **then**
    Add C_j to C_{agg} ▷ Best-aligned cluster
  else if $\delta_1 < \cos \tilde{\theta} - \cos \overline{\theta}^j$ **and** $\tilde{s} - \overline{s}^j \leq \delta_2$ **and** $\overline{s}^j \geq \delta_3$ **then**
    Add C_j to C_{agg} ▷ Cluster passes adaptive threshold test
return C_{agg}

4. Performance evaluation

4.1. Experimental setup

Datasets. We used two image classification datasets from different mains in out experiments.

Table 1
Model architecture used on MNIST.

Layer type	Size
Conv + ReLU	$10 \times 5 \times 5$
Max pooling	2×2
Conv + ReLU	$20 \times 5 \times 5$
Max pooling	2×2
FC + ReLU	50
FC (Logits)	10

Table 2
Model architecture used on CIFAR-10.

Layer type	Size
Conv + ReLU	$32 \times 3 \times 3$
Max pooling	2×2
Conv + ReLU	$64 \times 3 \times 7$
Max pooling	2×2
FC + ReLU	512
FC (Logits)	10

Table 3
Accuracy (%) under various model poisoning attacks on non-IID datasets ($\alpha = 0.5$). Our method TACRC-FL consistently outperforms other defenses across different attack types.

Dataset	Method	FedAvg	MKrum	Trim	Median	RFA	RLR	Ours
MNIST ($\alpha = 0.5$)	No Att.	**98.57**	98.19	98.21	97.97	97.91	98.11	98.28
	Fang	97.22	87.72	93.84	93.72	09.76	93.26	**98.41**
	LIE	93.09	10.35	87.96	92.28	97.42	93.44	**98.45**
	Min-Max	97.77	95.34	96.47	95.68	09.80	96.92	**98.50**
	Min-Sum	97.76	96.80	95.90	93.76	79.87	97.17	**98.48**
	Sign.	97.24	97.73	96.91	97.12	97.03	96.94	**97.96**
	Add.	94.97	98.41	98.12	97.98	97.98	97.41	**98.34**
	Avg.	96.66	83.50	95.34	95.50	69.96	96.17	**98.34**
CIFAR10 ($\alpha = 0.5$)	No Att.	**60.26**	55.38	56.90	57.73	45.13	58.01	57.63
	Fang	52.52	36.98	44.30	36.20	38.28	40.37	**58.34**
	LIE	42.92	23.08	30.58	29.13	44.98	31.80	**57.69**
	Min-Max	53.25	44.84	51.17	54.53	43.27	55.81	**57.07**
	Min-Sum	51.99	51.07	53.23	49.48	47.97	57.77	**57.96**
	Sign.	44.72	55.76	49.26	52.24	41.99	50.40	**57.20**
	Add.	26.42	58.22	**58.78**	55.86	50.57	32.66	57.28
	Avg.	47.44	46.47	49.17	47.88	44.59	46.68	**57.59**

- MNIST: 10-class handwritten digit image classification dataset consisting of 60,000 training samples and 10,000 testing samples.
- CIFAR10: The CIFAR10 dataset consists of 60,000 32×32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10 000 test images.

Non-IID. To simulate different degrees of non-IID data distribution among these clients, we used the Dirichlet distribution to sample the data. The parameter α of the Dirichlet distribution controls the level of heterogeneity in the labels. Smaller values of α resulted in greater label heterogeneity. Unless otherwise specified, we set $\alpha = 0.5$ in our experiments.

Model. We trained on the above two datasets to verify the versatility of TACRC-FL. Specifically, for the MNIST dataset and CIFAR10 dataset, we employed a convolution neural network (CNN) for training. The architecture of the CNN model was detailed in Tables 1 and 2.

FL Setting. In each round, 128 clients participated in the training. Each round, we randomly sampled 32 clients to evaluate the impact of attack and defense on model training. Each client trains 5 epoch locally, batch size of $B = 16$ for MNIST and $B = 64$ for CIFAR10. In addition, we constructed a rigorous adversary scenario by configuring 2 parameters, i.e., malicious client number and non-IID degree (this is determined by the parameter α sampled by dirichlet). Unless otherwise specified, our experiments were performed under the settings of 8 malicious client clients and $\alpha = 0.5$. In addition, We set thresholds $\delta_1 = 0.05$, $\delta_2 = 0.05$ and $\delta_3 = 0.2$ in TACRC to control aggregation of clusters. It is worth noting that in order to reduce the amount of computation, our method only uses the neuronal values randomly sampled by the fully connected layer when calculating the nearest neighbors. The features are normalized before clustering.

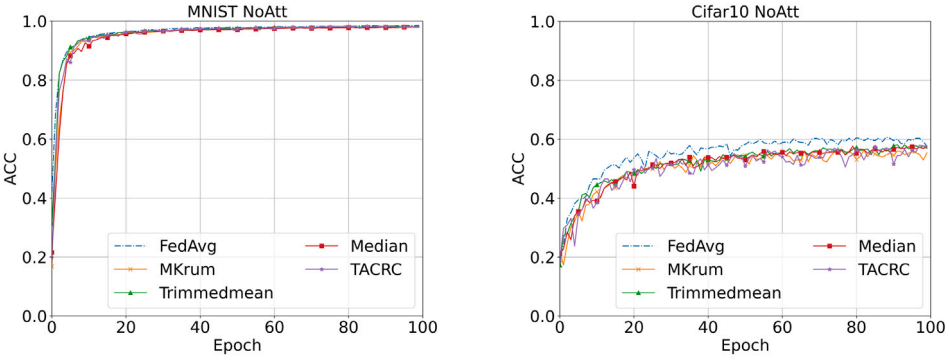
Table 4
Defense performance against adversarially crafted Byzantine attacks under non-IID data partitioning ($\alpha = 0.5$). Our proposed TACRC-FL achieves consistently strong results compared to state-of-the-art baselines.

Dataset	Attack	FedAvg	MKrum	Trim	Median	RFA	RLR	Ours
MNIST ($\alpha = 0.5$)	No Att.	**98.57**	98.19	98.21	97.97	97.91	98.11	98.28
	Adv.	53.88	98.32	96.37	96.49	97.45	79.36	**98.41**
CIFAR10 ($\alpha = 0.5$)	No Att.	**60.26**	55.38	56.90	57.73	45.13	57.01	57.63
	Adv.	10.00	58.29	41.49	51.52	33.45	10.00	**57.80**

Attack Settings. We consider seven types of untargeted model poisoning attacks. In each case, the attacker is assumed to have access to the gradient of the benign client's local model update. The attacks include: (1) **Fang** [24], (2) **LIE** [42], (3) **Min-Max**, (4) **Min-Sum**, (5) **Sign Flipping** (abbreviated as **Sign.**), (6) **Additive Noise** (**Add.**), and (7) the average performance across all attack types (**Avg.**). These abbreviated names are used consistently in our result tables for clarity. In the following, we briefly describe the implementation details of each attack.

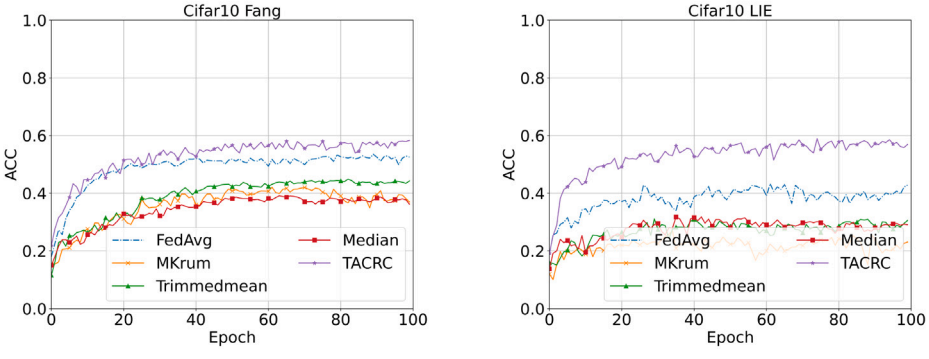
- **Fang**. Is an optimization based model poisoning attack. Specially, the adversary uses the update gradient of the benign model to calculate the average μ, and then solves for the scaling coefficient λ and direction $-sign(\cdot)$. λ is used to narrow the update gradient in the direction of malicious updates [24].
- **LIE**. The adversary computes the average μ and standard deviation σ of benign model gradients. The resulting malicious update is $\mu + 2\sigma$. The calculated coefficients are used to add noise to each dimension of the model update to influence the final global model. And ensure that the malicious model gradients is within the monitoring range of defense [42].
- **Min-Max**. Use the maximum sum of squares of any two benign models to update the gradients as an upper bound. Compute the malicious gradient, that the sum of the square of the malicious gradient update and the gradient update of any client does not exceed the upper bound [26].
- **Min-Sum**. This method Compute the sum of squares of all benign gradients as an upper bounded. Ensure that the maximum sum of the squared distances of the distance between the malicious gradient and all the benign gradients does not exceed the previously calculated upper bound [26].
- **SignFlipping**. SignFlipping Attack is an untargeted attack, where the malicious clients flip the signs of their local model updates [43,44]. Since there is no change in the magnitude of the local model updates, the SignFlipping attack can make hard-thresholding based defense fail [45].
- **AdditiveNoise**. Add random noise to each dimension [43,44]. This approach does not consider benign update gradients and server aggregation. It is easier to detect, but more impact on the model.
- **Adversarial lens**. This type of method adopts an adversarial perspective, where an alternating minimization strategy is used to iteratively optimize both stealthiness and attack efficacy. By balancing the trade-off between invisibility and adversarial objectives, the attack becomes more effective and harder to detect [46].

Robust aggregation. We compared our approach, TACRC-FL, with four aggregation rules [1,30,31], and conducted experiments on multiple attacks using two datasets. The aggregation rules considered were FedAvg [1], MKrum [30], Trimmedmean [31], Median [31], RFA [47] and RLR [48]

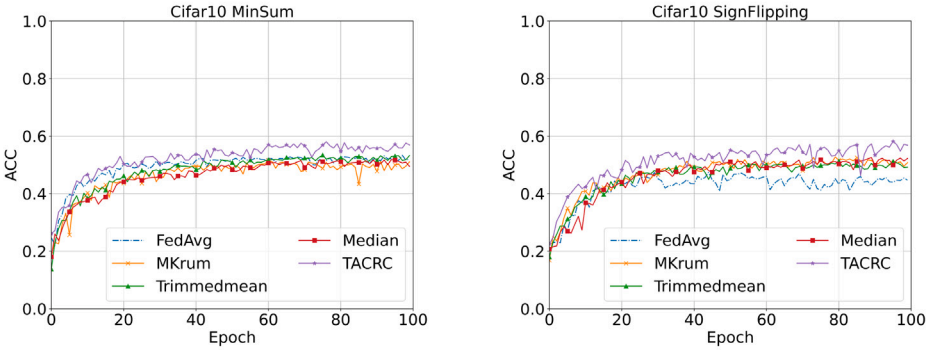


(a) MNIST (b) CIFAR10

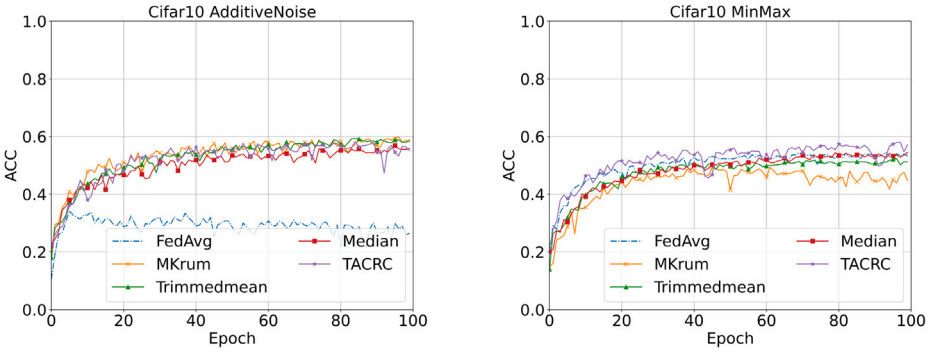
Fig. 4. w/o Attack under MNIST and CIFAR10 with $\alpha = 0.5$.



(a) Fang (b) Lie

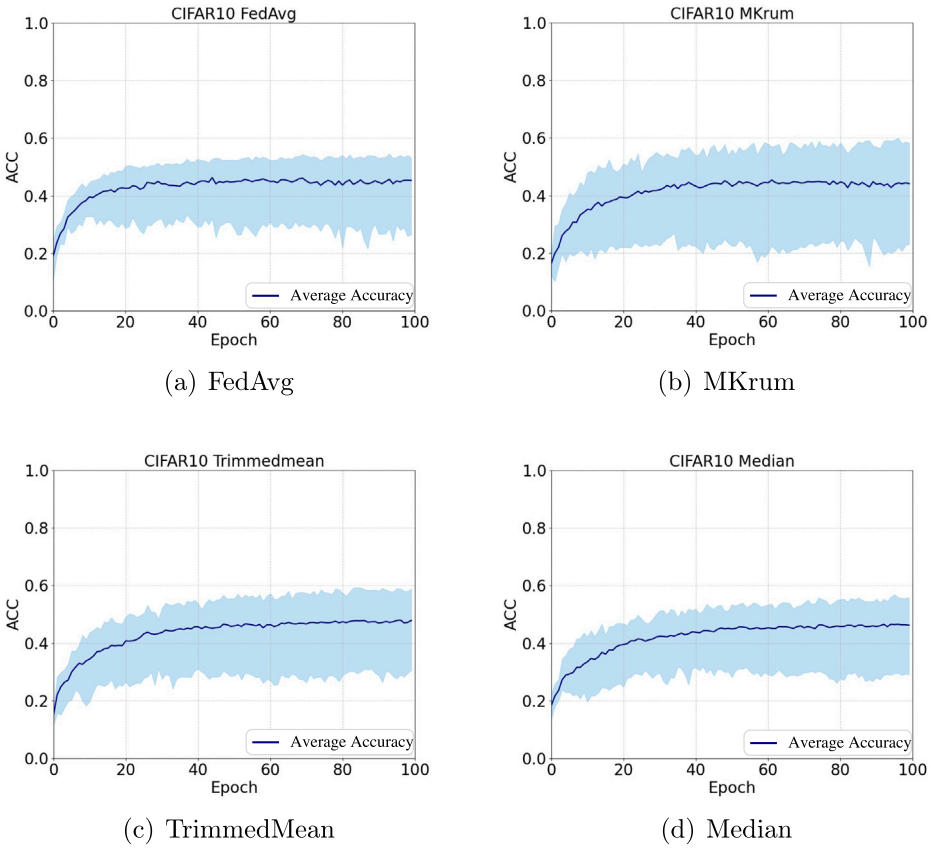


(c) Min-Sum (d) SignFlipping



(e) AdditiveNoise (f) Min-Max

Fig. 5. Attack and defense under CIFAR10, using non-IID partitioned datasets with $\alpha = 0.5$.



(a) FedAvg (b) MKrum

(c) TrimmedMean (d) Median

Fig. 6. Six kinds of attack impact on four defense method.

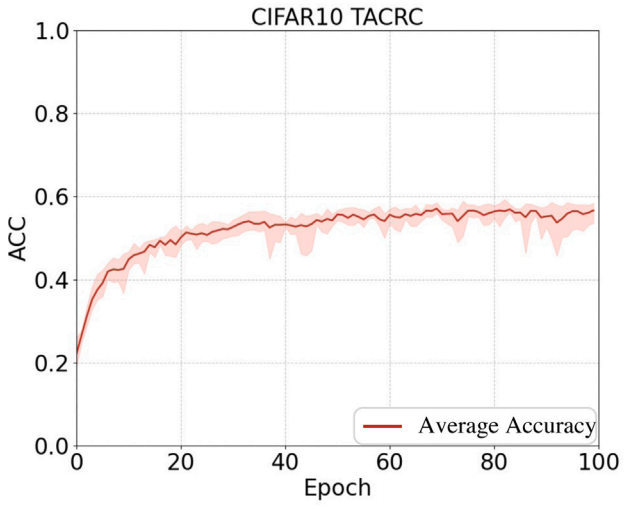


Fig. 7. Six kinds of attack impact on TACRC-FL.

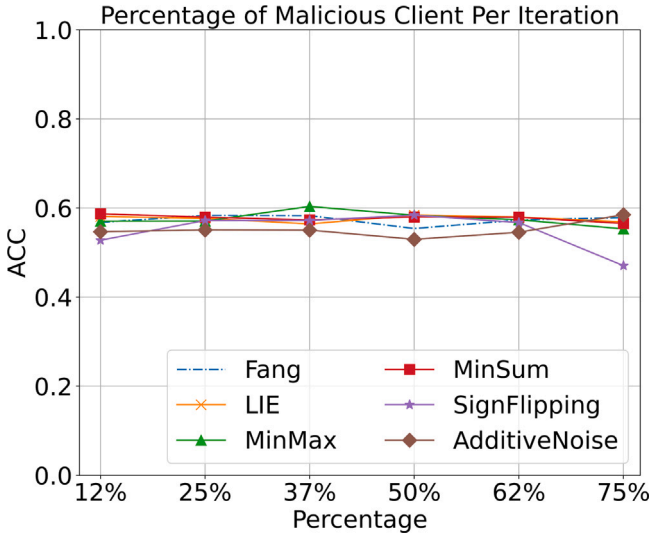


Fig. 8. The impact of different proportion of malicious client on our method under six attacks.

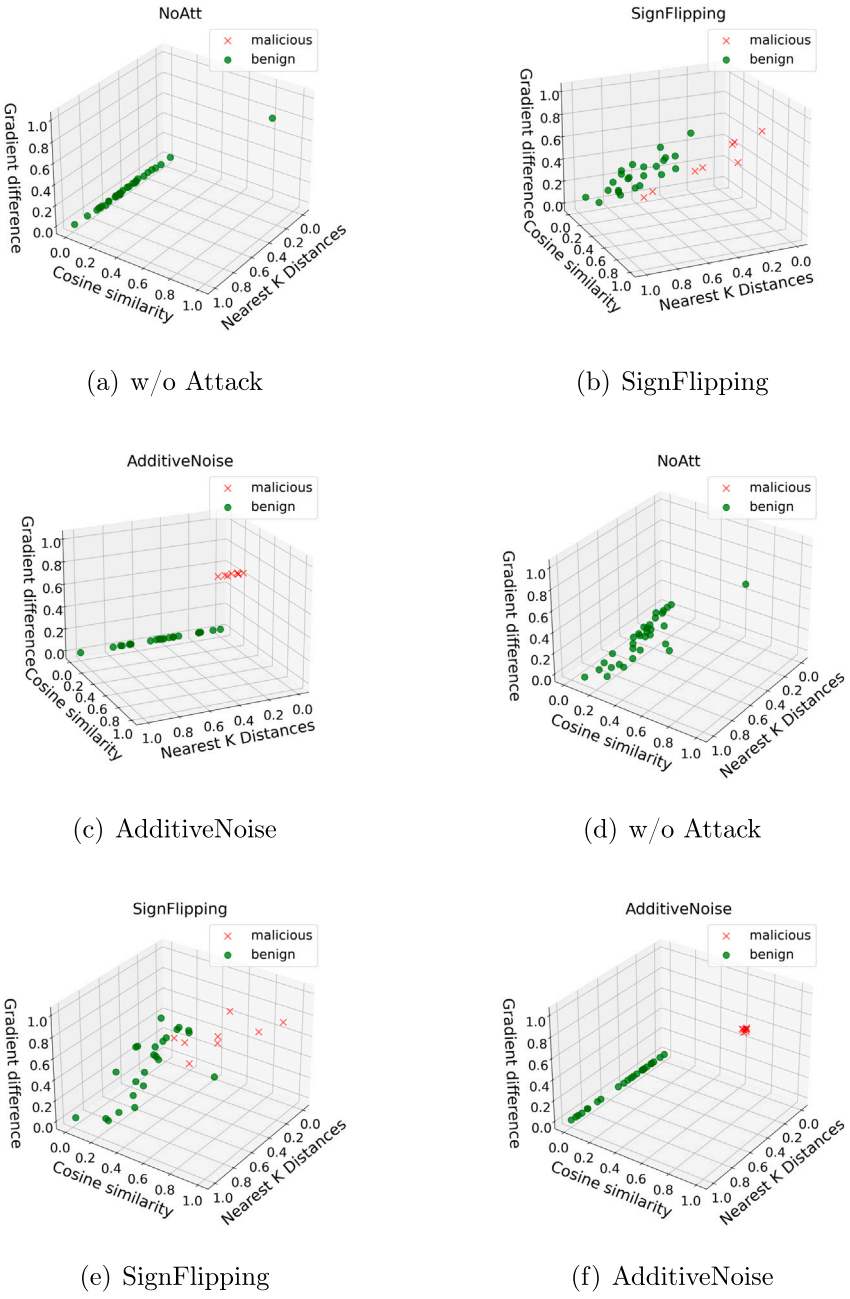


Fig. 9. Cluster feature on MNIST and CIFAR10, (a), (b), (c) show the results on MNIST, (d), (e), (f) show the result on CIFAR10. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Experiment

All experiments were performed on a server running Ubuntu 22.04LTS, with 1 NVIDIA GeForce RTX 3090 Ti (with 24 GB memory) and 64 GB RAM.

4.3. Experimental result

Performance Evaluation. We evaluate the performance of our proposed TACRC−FL against six state-of-the-art model poisoning attacks and compare it with other state-of-the-art defenses. Table 3 presents the test accuracy using LeNet on non-IID partitioned MNIST and CIFAR-10 datasets with a Dirichlet coefficient of $\alpha = 0.5$.

Fig. 4 illustrates the model performance under no attack with $\alpha = 0.5$. The accuracy of our proposed method TACRC−FL is close to that of FedAvg without attack, indicating minimal performance degradation in benign settings. Fig. 5 depicts the performance under six attack types on CIFAR-10. It is evident that TACRC−FL significantly mitigates the impact of strong model poisoning attacks. Among these, the Fang and LIE attacks are the most detrimental, substantially degrading FedAvg and bypassing other defenses. Although SignFlipping and AdditiveNoise are relatively easier to detect, they still cause considerable damage to FedAvg. Min-Max and Mix-Sum result in only slight reductions in accuracy, but these impacts are also effectively alleviated by our defense.

Table 3 summarizes the accuracy of seven aggregation methods after 100 rounds of training under all attack types and the no-attack setting. On MNIST, the model is easier to train, and attacks have a relatively smaller effect. FedAvg achieves an average accuracy of 96.46%, with a maximum of 98.57% and a minimum of 93.09%, resulting in an accuracy drop of 5.48%. In contrast, TACRC−FL achieves a

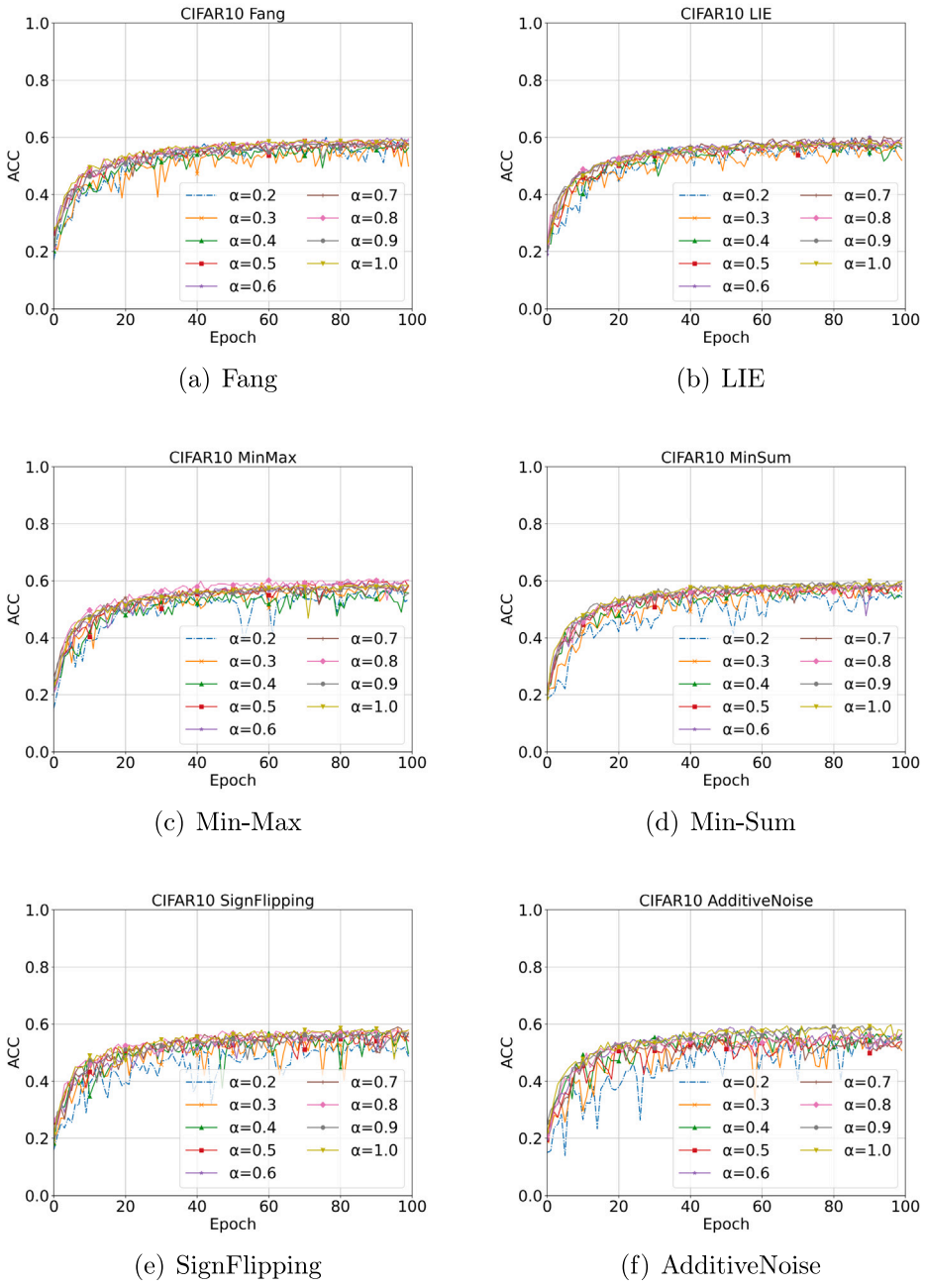


Fig. 10. Accuracy evaluate of different α setting under six kinds of attacks on CIFAR10.

maximum accuracy of 98.50%, a minimum of 97.96%, and an average drop of only 0.54%, showing that our method is approximately 10× more effective in resisting attacks. Table 4 further compares the global model accuracy under adversarially robust Byzantine attack scenarios across seven aggregation algorithms. Without defense, the global model converges to suboptimal accuracy. Compared to existing defenses, our method achieves consistently competitive results.

For CIFAR-10, the average accuracy of FedAvg under six attacks is 47.44%, with the worst being 44.59%. Our method achieves a significantly higher average accuracy of 57.59%. The average attack impact under TACRC−FL is 2.67% (60.26–57.59), while the best baseline defense (TrimmedMean) suffers an average impact of 11.09% (60.26–49.17). Therefore, TACRC−FL is approximately 4× more effective at mitigating poisoning attacks on this dataset.

TACRC-FL Stability Against Attacks. Fig. 6 illustrate the fluctuations in the impact of six attacks on four aggregation methods. Compared with Fig. 7, the global model obtained by the four aggregation methods has relatively large fluctuations in the six kinds of attack, indicating that it cannot better resist multiple attacks. On the contrary, the global model evaluated by our aggregation method TACRC−FL has less fluctuation, indicating that it has better defense effect against all six kinds of attacks.

The Influence of Adversary Number. In our experiment, 32 clients were randomly selected to participate in training in each round, and the proportion of existing clients was ϵ. Then the aggregate global model is obtained by the aggregation algorithm $\mathcal{A}$. In this experiment, we set epsilon to a different value 12.5%, 25%, 37.5%, 50%, 62.5% and 75% respectively. Fig. 8 showed the evaluation result for different attacks with different proportions of the number of malicious clients. Although

Table 5
Accuracy of our defense method under varying δ_1 while keeping $\delta_2 = 0.05$ and $\delta_3 = 0.2$ fixed. The results indicate the defense is not overly sensitive to δ_1, confirming its robustness.

δ_1	δ_2	δ_3	Adv.	Sign.	Add.
0.05	0.05	0.2	57.80	57.20	57.08
0.10	0.05	0.2	57.38	57.45	53.04
0.15	0.05	0.2	56.59	57.49	53.09
0.20	0.05	0.2	57.26	53.95	51.84
0.25	0.05	0.2	56.29	55.25	55.61

Table 6
Accuracy of our defense method under varying δ_2 while keeping $\delta_1 = 0.05$ and $\delta_3 = 0.2$ fixed. The defense maintains stable performance across a wide range of δ_2 settings against different attack types.

δ_1	δ_2	δ_3	Adv.	Sign.	Add.
0.05	0.05	0.2	57.20	57.08	57.80
0.05	0.10	0.2	55.98	57.27	57.44
0.05	0.20	0.2	53.81	53.81	58.68
0.05	0.30	0.2	56.37	56.39	56.20
0.05	0.40	0.2	57.69	57.54	55.35
0.05	0.50	0.2	57.13	57.19	51.71

the proportion of malicious clients had increased to 75%, our method can still effectively defend and get a convergent global model.

Validity of Clustering Factors. To validate the effectiveness of the proposed clustering factor in distinguishing between malicious and benign models, we visualized the clustering factor in three-dimensional space, as shown in Fig. 9. To validate the effectiveness of the proposed clustering factor in distinguishing between malicious and benign models, we visualized the clustering factor in three-dimensional space. Due to the different local dataset and heterogeneous settings of the client, the clustering factors of the model are scattered in the feature space of the cluster. Fig. 9(a)–(f) shows the cluster factors of MNIST and CIFAR10 under two kinds of attacks and without attack. Green represents benign model updates and red represents malicious model updates. In our clustering space, out clustering factor can obviously separate benign and malicious. We think that the clustering factor in this part is extensible. After taking advantage of other exception assessments, this section can be populated to enhance TACRC-FL's ability to defend against many types of attacks.

The Influence of Data Non-IID. We evaluate our method on different dirichlet coefficient α, α ranges from 0.2 to 1.0. Fig. 10(a)–(f) correspond to the accuracy changes of six attacks under different α Settings of CIFAR10. It shows that the reduction of α does not result in a performance loss for our method, which indicates that our method defends well against non-independent, identically distributed data.

Hyperparameter Analysis. We conduct a comprehensive analysis of the hyperparameters δ_1, δ_2, and δ_3, as shown in Tables 5, 6, and 7. In each experiment, we fix two of the parameters and vary the remaining one to evaluate the robustness of our defense method. The results demonstrate that our approach maintains strong robustness across a wide range of hyperparameter settings. Based on the overall performance, we select $\delta_1 = 0.05$, $\delta_2 = 0.05$, and $\delta_3 = 0.2$ as the default configuration for the evaluation of our defense.

5. Conclusion

This paper presents a robust aggregation algorithm, TACRC-FL, for federated learning (FL). It distinguishes benign model updates from malicious ones by leveraging a set of trusted anchor clients. Specifically, we design three clustering factors that effectively separate benign and malicious client updates based on the anchor models. Leveraging these anchors, the proposed method identifies and filters out malicious model clusters to mitigate their adverse effects on global model aggregation. Extensive experiments under various Byzantine threat scenarios demonstrate that TACRC-FL achieves strong robustness without requiring server-side clean data, and remains effective even in the presence of a high proportion of malicious clients and severe non-IID data distributions.

While this work focuses on untargeted Byzantine attacks, we acknowledge that targeted threats—such as backdoor attacks—pose significant and stealthy risks in FL. As part of future work, we plan to extend TACRC-FL to detect and defend against such targeted poisoning behaviors, further enhancing its applicability in real-world federated systems.

Table 7
Accuracy of our defense method under varying δ_3 while keeping $\delta_1 = 0.05$ and $\delta_2 = 0.05$ fixed. The results demonstrate consistent robustness across different δ_3 values against adversarial, sign-flipping, and additive noise attacks.

δ_1	δ_2	δ_3	Adv.	Sign.	Add.
0.05	0.05	0.1	58.07	52.32	57.78
0.05	0.05	0.2	57.20	57.08	57.80
0.05	0.05	0.3	55.65	57.18	56.04
0.05	0.05	0.4	56.19	55.72	57.69
0.05	0.05	0.5	56.64	57.02	56.89

CRediT authorship contribution statement

Maozhen Zhang: Project administration, Methodology, Investigation, Funding acquisition. **Yi Li:** Supervision, Software, Resources. **Fei Wei:** Investigation, Formal analysis, Data curation. **Bo Wang:** Visualization, Validation, Supervision, Funding acquisition. **Yushu Zhang:** Visualization, Resources, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62372452).

Data availability

Data will be made available on request.

References

[1] McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR; 2017, p. 1273–82.
[2] Kairouz P, McMahan HB, Avent B, Bellet A, Bennis M, Bhagoji AN, Bonawitz K, Charles Z, Cormode G, Cummings R, et al. Advances and open problems in federated learning. Found Trends® Mach Learn 2021;14(1–2):1–210.
[3] Gao D, Yao X, Yang Q. A survey on heterogeneous federated learning. 2022, arXiv preprint arXiv:2210.04505.
[4] Zhang C, Xie Y, Bai H, Yu B, Li W, Gao Y. A survey on federated learning. Knowl-Based Syst 2021;216:106775.
[5] Nguyen DC, Ding M, Pathirana PN, Seneviratne A, Li J, Poor HV. Federated learning for internet of things: A comprehensive survey. IEEE Commun Surv Tutor 2021;23(3):1622–58.

[6] Beltrán ETM, Pérez MQ, Sánchez PMS, Bernal SL, Bovet G, Pérez MG, Pérez GM, Celdrán AH. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. IEEE Commun Surv Tutor 2023.
[7] Zhu J, Cao J, Saxena D, Jiang S, Ferradi H. Blockchain-empowered federated learning: Challenges, solutions, and future directions. ACM Comput Surv 2023;55(11):1–31.
[8] Wu F, Guo S, Qu Z, He S, Liu Z, Gao J. Anchor sampling for federated learning with partial client participation. In: International conference on machine learning. PMLR; 2023, p. 37379–416.
[9] Ding J, Tramel E, Sahu AK, Wu S, Avestimehr S, Zhang T. Federated learning challenges and opportunities: An outlook. In: ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing. ICASSP, IEEE; 2022, p. 8752–6.
[10] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. ACM Trans Intell Syst Technol (TIST) 2019;10(2):1–19.
[11] Badr MM, Mahmoud M, Fang Y, Abdulaal M, Aljohani AJ, Alasmary W, Ibrahem MI. Privacy-preserving and communication-efficient energy prediction scheme based on federated learning for smart grids. IEEE Internet Things J 2023.
[12] Cheng K, Fan T, Jin Y, Liu Y, Chen T, Papadopoulos D, Yang Q. Secureboost: A lossless federated learning framework. IEEE Intell Syst 2021;36(6):87–98.
[13] Lee CM, Delgado Fernandez J, Potenciano Menci S, Rieger A, Fridgen G. Federated learning for credit risk assessment. In: Proceedings of the 56th hawaii international conference on system sciences. 2023, p. 10.
[14] Samarakoon S, Bennis M, Saad W, Debbah M. Federated learning for ultra-reliable low-latency V2V communications. In: 2018 IEEE global communications conference. GLOBECOM, IEEE; 2018, p. 1–7.
[15] Xu C, Qu Y, Xiang Y, Gao L. Asynchronous federated learning on heterogeneous devices: A survey. Comput Sci Rev 2023;50:100595.
[16] Imteaj A, Mamun Ahmed K, Thakker U, Wang S, Li J, Amini MH. Federated learning for resource-constrained IoT devices: panoramas and state of the art. Fed Transf Learn 2022;7–27.
[17] Kasyap H, Tripathy S. Beyond data poisoning in federated learning. Expert Syst Appl 2024;235:121192.
[18] Tan S, Hao F, Gu T, Li L, Liu M. Collusive model poisoning attack in decentralized federated learning. IEEE Trans Ind Inform 2023.
[19] Wang S, Li Q, Cui Z, Hou J, Huang C. Bandit-based data poisoning attack against federated learning for autonomous driving models. Expert Syst Appl 2023;227:120295.
[20] Cao X, Jia J, Zhang Z, Gong NZ. Fedrecover: Recovering from poisoning attacks in federated learning using historical information. In: 2023 IEEE symposium on security and privacy. SP, IEEE; 2023, p. 1366–83.
[21] Chen X, Yu H, Jia X, Yu X. Apfed: Anti-poisoning attacks in privacy-preserving heterogeneous federated learning. IEEE Trans Inf Forensics Secur 2023.
[22] Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V. How to backdoor federated learning. In: International conference on artificial intelligence and statistics. PMLR; 2020, p. 2938–48.
[23] Xie C, Koyejo O, Gupta I. Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In: Uncertainty in artificial intelligence. PMLR; 2020, p. 261–70.
[24] Fang M, Cao X, Jia J, Gong N. Local model poisoning attacks to {Byzantine-Robust} federated learning. In: 29th USeNIX security symposium. USeNIX security 20, 2020, p. 1605–22.
[25] So J, Güler B, Avestimehr AS. Byzantine-resilient secure federated learning. IEEE J Sel Areas Commun 2020;39(7):2168–81.
[26] Shejwalkar V, Houmansadr A. Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning. In: NDSS. 2021.
[27] Zhu B, Wang L, Pang Q, Wang S, Jiao J, Song D, Jordan MI. Byzantine-robust federated learning with optimal statistical rates. In: International conference on artificial intelligence and statistics. PMLR; 2023, p. 3151–78.
[28] Jebreel NM, Domingo-Ferrer J. FL-Defender: Combating targeted attacks in federated learning. Knowl-Based Syst 2023;260:110178.
[29] Han S, Park S, Wu F, Kim S, Zhu B, Xie X, Cha M. Towards attack-tolerant federated learning via critical parameter analysis. In: Proceedings of the IEEE/CVF international conference on computer vision. 2023, p. 4999–5008.
[30] Blanchard P, El Mhamdi EM, Guerraoui R, Stainer J. Machine learning with adversaries: Byzantine tolerant gradient descent. Adv Neural Inf Process Syst 2017;30.
[31] Yin D, Chen Y, Kannan R, Bartlett P. Byzantine-robust distributed learning: Towards optimal statistical rates. In: International conference on machine learning. PMLR; 2018, p. 5650–9.
[32] Tao Z, Yang K, Kulkarni SR. Byzantine-robust clustered federated learning. 2023, arXiv preprint arXiv:2306.00638.
[33] Sattler F, Müller K-R, Wiegand T, Samek W. On the byzantine robustness of clustered federated learning. In: ICASSP 2020-2020 IEEE international conference on acous tics, speech and signal processing. ICASSP, IEEE; 2020, p. 8861–5.
[34] Cao X, Fang M, Liu J, Gong NZ. Fltrust: Byzantine-robust federated learning via trust bootstrapping. 2020, arXiv preprint arXiv:2012.13995.
[35] Ester M, Kriegel H-P, Sander J, Xu X, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd. Vol. 96, 1996, p. 226–31.
[36] Campello RJ, Moulavi D, Sander J. Density-based clustering based on hierarchical density estimates. In: Pacific-Asia conference on knowledge discovery and data mining. Springer; 2013, p. 160–72.
[37] He Z, Wang L, Cai Z. Clustered federated learning with adaptive local differential privacy on heterogeneous iot data. IEEE Internet Things J 2023.
[38] Mehta M, Shao C. A greedy agglomerative framework for clustered federated learning. IEEE Trans Ind Inform 2023.
[39] Ghosh A, Chung J, Yin D, Ramchandran K. An efficient framework for clustered federated learning. Adv Neural Inf Process Syst 2020;33:19586–97.
[40] Li S, Cheng Y, Wang W, Liu Y, Chen T. Learning to detect malicious clients for robust federated learning. 2020, arXiv preprint arXiv:2002.00211.
[41] Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K. Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 2017, p. 1175–91.
[42] Baruch G, Baruch M, Goldberg Y. A little is enough: Circumventing defenses for distributed learning. Adv Neural Inf Process Syst 2019;32.
[43] Li L, Xu W, Chen T, Giannakis GB, Ling Q. RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 33, 2019, p. 1544–51.
[44] Wu Z, Ling Q, Chen T, Giannakis GB. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. IEEE Trans Signal Process 2020;68:4583–96.
[45] Sun Z, Kairouz P, Suresh AT, McMahan HB. Can you really backdoor federated learning?. 2019, arXiv preprint arXiv:1911.07963.
[46] Bhagoji AN, Chakraborty S, Mittal P, Calo S. Analyzing federated learning through an adversarial lens. In: International conference on machine learning. PMLR; 2019, p. 634–43.
[47] Pillutla K, Kakade SM, Harchaoui Z. Robust aggregation for federated learning. IEEE Trans Signal Process 2022;70:1142–54.
[48] Ozdayi MS, Kantarcioglu M, Gel YR. Defending against backdoors in federated learning with robust learning rate. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 35, 2021, p. 9268–76.