

Secure Federated Learning Based on the OPTICS Clustering Algorithm and Client Selection

Chunmei Ma, Zhengyang Zhang, Baogui Huang, Chuanwen Luo*, Guangshun Li, and Junhua Wu

Abstract: In Federated Learning (FL), the aggregation servers inevitably face attacks from malicious clients because they cannot directly constrain the behaviour of clients. Effectively detecting and mitigating the impact of these malicious clients on the global model is a critical challenge. This paper proposes a secure aggregation algorithm, named Ordering Points To Identify the Clustering Structure (OPTICS) Clustering Algorithm and Client Selection Federated Learning (OCACSFL), designed to identify and exclude malicious clients. By employing an improved OPTICS clustering algorithm, OCACSFL enables the aggregation server to effectively distinguish between benign and malicious clients, ensuring that malicious updates do not contribute to the global model aggregation. If a client is detected to upload malicious updates multiple times, OCACSFL permanently rejects that client's updates. Experimental results demonstrate that OCACSFL can efficiently detect and exclude malicious clients across various datasets and machine learning models, showcasing exceptional performance and robustness.

Key words: Federated Learning (FL); model poisoning attacks; malicious client detection; Ordering Points To Identify the Clustering Structure (OPTICS) clustering algorithm; client selection

1 Introduction

Federated Learning (FL)^[1] has emerged as a new paradigm in machine learning^[2–6] and has shown significant potential for advancement across various fields, including natural language processing,

autonomous driving, financial technology, image classification, and medical imaging diagnostics^[7, 8]. Traditional centralized machine learning algorithms require collecting substantial data from clients^[9]. However, these centralized approaches may lead to privacy concerns, as data collectors can directly access users' sensitive information^[10, 11]. In the FL framework, multiple clients collaboratively train a global model^[12, 13], while keeping their training data locally stored. Clients only need to upload model updates, such as gradients or weights, to a central server, ensuring that original data remain stored locally and are not directly accessible to external entities, thereby effectively safeguarding user privacy^[14, 15].

FL systems still face numerous security challenges^[16]. Since the central server cannot directly access clients' private data or control their actions, malicious clients can deceive the server by sending tampered, harmful model updates, thus launching poisoning attacks on the global model. Based on the

• Chunmei Ma, Zhengyang Zhang, Baogui Huang, Guangshun Li, and Junhua Wu are with School of Computer Science, Qufu Normal University, Rizhao 276800, China. E-mail: chunmeima@qfnu.edu.cn; Zhangzhengyang@qfnu.edu.cn; hjbaogui@qfnu.edu.cn; guangshunli@qfnu.edu.cn; shdwjh@163.com.

• Chuanwen Luo is with School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China, and also with Engineering Research Center for Forestry-Oriented Intelligent Information Processing, National Forestry and Grassland Administration, Beijing 100083, China. E-mail: chuanwenluo@bjfu.edu.cn.

* To whom correspondence should be addressed.

Manuscript received: 2025-04-01; revised: 2025-06-06; accepted: 2025-12-31

attack objectives, model poisoning attacks are typically categorized as untargeted and targeted attacks. In untargeted attacks^[17, 18], the compromised global model makes incorrect predictions on a wide range of test inputs without focusing on a specific target. Fang et al.^[19] proposed an untargeted attack framework that transforms untargeted attacks into an optimization problem, designing model updates from malicious clients to maximize the difference in the global model's updates before and after the attack. In targeted attacks^[17, 18, 20], the compromised global model only makes incorrect predictions on attacker-selected test inputs, while maintaining accuracy on other test inputs. For example, in image classification tasks, targeted attacks can manipulate the model to misclassify an "apple" as a "watermelon". Targeted attacks can be divided into two categories: naive approach and model replacement^[21]. The naive approach significantly undermines model performance by introducing mislabeled data into the training set^[22]. However, in FL, the aggregation process inherently mitigates the impact of such malicious data on model accuracy. As a result, the naive approach has only a limited effect on the model's performance, even when a large number of attack rounds are conducted. In contrast, model replacement allows malicious clients to upload carefully crafted model updates that induce misclassification in the global model, producing a more pronounced effect^[21].

Existing defenses against model poisoning attacks primarily rely on Byzantine-robust FL methods^[23–25]. These methods aim to enable the central server to learn an accurate global model even if some clients are malicious and submit harmful model updates. In theory, Byzantine-robust FL methods can limit the impact of malicious clients on global model parameters. However, these approaches have notable limitations: some demonstrate robustness only when the number of malicious clients is small^[23, 24], while others depend on the aggregation server having access to a clean and representative validation dataset^[25]. Additionally, we find that current methods^[23, 24] either exclude certain clients or substitute a subset of clients for the full client set, failing to ensure that all selected benign clients participate in model aggregation. These limitations result in some benign data being underutilized, thereby leading to wasted data resources.

In this paper, we propose a novel Byzantine-robust aggregation method named Ordering Points To Identify

the Clustering Structure (OPTICS) Clustering Algorithm and Client Selection Federated Learning (OCACSFL). Inspired by Le et al.^[26], we first apply the density-based clustering algorithm OPTICS to partition clients into multiple clusters. Under the assumption that the majority of clients are benign, the largest cluster is selected as the benign client set for model aggregation in each round. Second, the server dynamically evaluates each client's reputation. If a client's credit score is smaller than a predefined threshold, the server ceases to accept the client's model updates. To prevent wastage of benign data, we introduce the concept of "benign distance difference" to determine if all participating clients in the current round are benign. Finally, acknowledging the risk that, in certain rounds, a majority of malicious clients could be selected and compromise OCACSFL's decision-making process, we integrate a model rollback mechanism inspired by residual neural networks^[27]. This allows the server to roll back to a previous, stable model in extreme cases, ensuring the robustness and integrity of the overall system. OCACSFL is compatible with various aggregation methods, such as FedAvg^[1] and FedProx^[28], without affecting the underlying aggregation strategy.

OCACSFL has been evaluated on three benchmark datasets (MNIST, FMNIST, and CIFAR-10) under three types of attacks: sign-flipping attacks^[29], additive noise attacks^[29], and backdoor attacks^[21]. The experimental results demonstrate that OCACSFL achieves strong performance across all three datasets, effectively mitigating the impact of each attack. The model accuracy remains close to the baseline results without attacks.

The contributions of this paper are summarized as follows.

(1) We propose the OCACSFL scheme, leveraging an enhanced OPTICS clustering algorithm and dynamic credit scoring to bolster Byzantine resilience. First, we apply the OPTICS clustering algorithm to cluster model updates, with the average reachability distance serving as the clustering criterion to accurately identify benign clients for aggregation. Additionally, a dynamic credit scoring mechanism adjusts the participation priorities of clients based on their historical performance, ensuring that clients with high reputations are given precedence in model training, thereby optimizing the overall robustness.

(2) A model update evaluation criterion and a model

rollback mechanism are introduced to improve the robustness of OCACSFL. By observing the reachability distance differences in model updates, we can detect malicious clients in each round, reducing misclassification and minimizing benign client wastage. Furthermore, when malicious clients are in the majority, the rollback mechanism effectively restores model performance, mitigating the negative impact of these clients on model training.

(3) We evaluate OCACSFL's effectiveness across multiple datasets and attack scenarios. For instance, on the MNIST and FMNIST datasets, OCACSFL successfully detects 100% of malicious clients under sign-flipping attacks, while maintaining model accuracy within 4% of the unperturbed federated averaging baseline.

The remainder of this paper is organized as follows. Section 2 presents related work, summarizing existing defense methods against model poisoning attacks. Section 3 provides preliminary knowledge, introducing the fundamental concepts of federated learning and the OPTICS clustering algorithm. Section 4 states the problem, describing the real-world application scenarios and design objectives targeted by our proposed method. Section 5 elaborates on the principles and implementation details of the proposed OCACSFL approach. Section 6 presents experimental results, demonstrating the effectiveness and advantages of OCACSFL. Finally, Section 7 concludes with a summary of our main contributions and an outlook on future research directions.

2 Related Work

Byzantine-robust FL algorithms can generally be categorized into two types. The first type of defense modifies the aggregation rule by employing calculations, such as the median or geometric center, to produce the final global model, rather than relying on the simple weighted average used in FedAvg^[30]. For example, Xie et al.^[31] proposed the median aggregation method, which aggregates updates by taking the median of each dimension, thereby reducing the impact of malicious updates. Yin et al.^[32] introduced the trimmed mean aggregation method, which excludes extreme values in updates and then computes the mean of the remaining values to enhance model robustness. Pillutla et al.^[33] proposed a robust aggregation method called RFA, which further strengthens resilience to malicious updates using a geometric median-based

aggregation strategy. Han et al.^[30] presented an attack-tolerant aggregation algorithm based on parameter importance, designed to mitigate or eliminate the impact of malicious clients. However, these defense strategies remain vulnerable to large-scale attacks by malicious clients, especially in rounds where half or more of the selected clients are compromised, significantly increasing the risk of system failure. Li et al.^[34] introduced Ditto, which uses regularization to balance global and local models, thereby reducing the impact of malicious clients on the global model.

The second defense approaches aim to enhance model robustness by detecting and removing malicious updates^[30]. Theoretically, Krum^[23] can safeguard the global model from the influence of malicious clients during model aggregation by selecting local model updates with the smallest discrepancy to their $n - m - 2$ nearest neighbors, where n is the total number of clients and m is the number of malicious clients. However, it is important to note that Krum requires the number of malicious clients, which is often difficult to ascertain in real-world scenarios, posing a challenge to the method's effectiveness. Although FLTrust^[25] exhibits robustness against a significant proportion of malicious clients, it depends on the assumption that the server has access to a clean validation dataset whose statistical distribution closely matches that of the overall training data. In typical FL scenarios, where the server lacks such a validation dataset, the global model may still be compromised by a large number of malicious clients. Li et al.^[16] proposed an algorithm named ClippedClustering, based on hierarchical clustering methods^[35, 36], to identify malicious clients. This algorithm enhances the robustness of clustering-based approaches by incorporating automatic clipping of model updates. Considering practical constraints such as transmission efficiency, computational resources, and channel conditions, clients are not selected to participate in aggregation every round. Instead, the server selects a subset of clients each round for model aggregation. This may result in a round where malicious clients constitute the majority, causing potentially irreversible damage to the aggregated model. Le et al.^[26] leveraged the DBSCAN clustering algorithm^[37] to partition all clients into distinct clusters. Assuming benign clients are in the majority, the cluster with the largest number of clients is selected as the benign client group. The client identified as benign is allowed to participate in model aggregation.

Additionally, Le et al.^[26] incorporated Differential Privacy (DP) techniques^[38–41] by adding noise to the uploaded model updates of clients that meet specific criteria, thus preserving client privacy. However, the performance of the algorithm proposed in Ref. [26] is sensitive to key hyperparameters, such as the neighborhood radius (denoted as ϵ) and the density threshold (denoted as MinPts). If ϵ is too large or MinPts is too small, there is a risk of misclassifying malicious clients as benign, leading to the inclusion of malicious models in the global aggregation and negatively impacting the overall performance. On the contrary, if ϵ is too small or MinPts is too large, benign clients may be excluded from the cluster of benign clients, reducing the training samples and causing a loss of accuracy.

3 Preliminary

3.1 FL

Suppose there is a parameter aggregation server and N clients. D_i denotes the local dataset of client c_i , $i \in \{1, 2, \dots, N\}$ and $n_i = |D_i|$ represents the number of samples of D_i . The objective of FL is to find a set of model parameters θ that minimizes the global loss function $L(\theta)$ in Eq. (1),

$$\arg \min_{\theta} L(\theta) = \sum_{i=1}^N \frac{n_i}{n} L_i(\theta) \quad (1)$$

where $n = \sum_{i=1}^N n_i$, and $L_i(\theta)$ denotes the local loss of θ on client c_i .

In each round of federated learning, the server sends the global model to the clients. Each client then updates the model using its local data and obtains a new local model. Subsequently, the client uploads its local model update to the server for model aggregation. The server gets a new global model by aggregating all local model updates. And then, the server initiates the next round of training. This process is repeated until the global model reaches optimal performance or a predefined number of rounds is completed.

Without loss of generality, we consider the t -th training round. Let $g_i^t = \theta_i^t - \theta^{t-1}$ represent the model updates uploaded by client c_i in round t , where θ_i^t is the local model obtained by client c_i in round t , and θ^{t-1} is the global model of the previous round (i.e., round $t-1$). After receiving model updates from all clients, the server aggregates the model updates and obtains the t -th round global model in Eq. (2),

$$\theta^t = \theta^{t-1} + \sum_{i=1}^N \frac{n_i}{n} g_i^t \quad (2)$$

3.2 Related definitions of OPTICS clustering algorithm

OPTICS^[42] is a density-based clustering algorithm used to uncover clustering structures with varying densities. Unlike traditional distance-based clustering algorithms (such as K-means^[43]), OPTICS identifies cluster structures by considering the density of data points, making it more effective for handling noisy data and clusters with irregular shapes. Compared to the DBSCAN^[37] algorithm, OPTICS is less sensitive to input parameters, enhancing the stability of clustering. The OPTICS method does not require prior specification of the number of clusters and can effectively eliminate the interference of outliers. In this paper, we employ an enhanced OPTICS clustering algorithm to partition model updates into benign and malicious categories. We first introduce several critical algorithmic parameters fundamental to the OPTICS framework.

Given a set of model updates G , the OPTICS clustering detection algorithm involves the following definitions^[26, 42].

eps-neighborhood radius: A model update g_j is considered to reside within the ϵ -neighborhood of another model update g_i if the Euclidean distance $d(g_i, g_j) \leq \epsilon$.

Core model update: A model update g_i is considered a core model update if the number of model updates within its ϵ -neighborhood radius is at least MinPts . Here, MinPts represents the minimum number of model updates required within the ϵ -neighborhood radius for g_i to qualify as a core model update.

Core distance: For a given MinPts and ϵ that make g_i a core model update, the smallest neighborhood radius required to qualify g_i as a core model update is referred to as its core distance. It is mathematically defined as Eq. (3),

$$\begin{aligned} \text{core-distance}(g_i) &= \begin{cases} \text{undefined,} & \text{if } |N_{\epsilon}(g_i)| < \text{MinPts} \\ d(g_i, N_{\epsilon}^{\text{MinPts}}(g_i)), & \text{otherwise} \end{cases} \quad (3) \end{aligned}$$

where $N_{\epsilon}(g_i)$ denotes the set of model updates within the ϵ -neighborhood of model update g_i , and $N_{\epsilon}^{\text{MinPts}}(g_i)$ represents the MinPts -th nearest neighbor of g_i within $N_{\epsilon}(g_i)$. Here, $d(g_i, N_{\epsilon}^{\text{MinPts}}(g_i))$ is the

Euclidean distance between model update g_i and its MinPts-nearest neighbor in $N_{\text{eps}}(g_i)$.

Reachability distance: For a given MinPts and eps, the reachability distance of model update g_j relative to model update g_i is defined in Eq. (4),

$$\text{reachability-distance}(g_i, g_j) = \begin{cases} \text{undefined}, & \text{if } |N_{\text{eps}}(g_i)| < \text{MinPts} \\ \max(\text{core-distance}(g_i), d(g_i, g_j)), & \text{otherwise} \end{cases} \quad (4)$$

4 Problem Formulation

4.1 Threat model

The attacker can control a number of malicious clients, which may be either fabricated and injected into the federated learning system by the attacker or genuine clients compromised through malicious means. In this paper, we focus on the latter scenario, where the attacker gains access to and manipulates a client's local training model, local data, loss function, and learning rate. During each global training round, benign client c_i computes and sends the authentic model update $g_i^t = \theta_i^t - \theta^{(t-1)}$ to the aggregation server. In contrast, malicious clients send carefully altered model updates $g_i^t \neq \theta_i^t - \theta^{(t-1)}$ to the aggregation server in an attempt to disrupt global model training.

4.2 Defense background

In practical federated learning applications, the number of clients is typically very large, making it highly difficult for an attacker to compromise a significant portion—let alone more than half of them. Such an endeavor would demand exceptional capabilities. Furthermore, as the attacker gains control over more clients, the likelihood of detection increases—a scenario attackers generally aim to avoid. As a result, much of the research assumes that only a small fraction of clients are compromised, with the majority remaining benign^[35, 44]. This paper similarly operates under the assumption that benign clients dominate in the federated learning system.

4.3 Design objectives

(1) **Secure aggregation:** The aggregation server can detect and mitigate the influence of malicious clients during global model aggregation.

(2) **Malicious client filtering:** If a client is repeatedly identified as malicious over multiple

training rounds, it will no longer be allowed to upload its model updates.

5 Methodology

Byzantine attacks mainly take two forms: tampering with local data or manipulating the local model. Regardless of the method used by malicious clients to disrupt the global model, their behavior is ultimately reflected in the uploaded model updates, which exhibit significant deviations from those of benign clients. Based on this characteristic, we propose a Byzantine-robust aggregation method, called OCACSFL, which is based on an improved OPTICS clustering algorithm. The main motivation of OCACSFL is as follows. The malicious and benign updates exhibit large different distributions in space. If benign updates constitute the majority, the density-based clustering algorithm can effectively distinguish them. The OPTICS clustering algorithm, as a density-based approach, is very suitable for this scenario. Without loss of generality, the t -th round workflow of OCACSFL is illustrated in Fig. 1, and the specific steps are as follows.

Step 1: Client selection: The aggregation server selects $|H|$ clients based on their credit scores, where H denotes the set of clients (see Section 5.4 for details). We use $c_1, c_2, \dots, c_{|H|}$ to denote the selected $|H|$ clients.

Step 2: Local model training: Each selected client trains a local model using its dataset.

Step 3: Uploading model updates: $|H|$ clients upload their model updates to the aggregation server.

Step 4: Secure aggregation: Upon receiving the $|H|$ model updates from the $|H|$ selected clients, the aggregation server performs secure aggregation (Step 4).

Steps 2 and 3 follow the standard federated learning process and will not be discussed in detail here. The client selection in Step 1 is based on each client's credit score, which, in turn, is determined by Step 4. Therefore, we will now focus on explaining Step 4 in detail.

The general process of secure aggregation is as follows: First, the aggregation server inputs the model updates into the OPTICS clustering algorithm to generate two queues, as shown in Step 4.1 of Fig. 1. The server then analyzes these two queues to determine whether there are any malicious model updates. If no malicious updates are detected, all model updates are

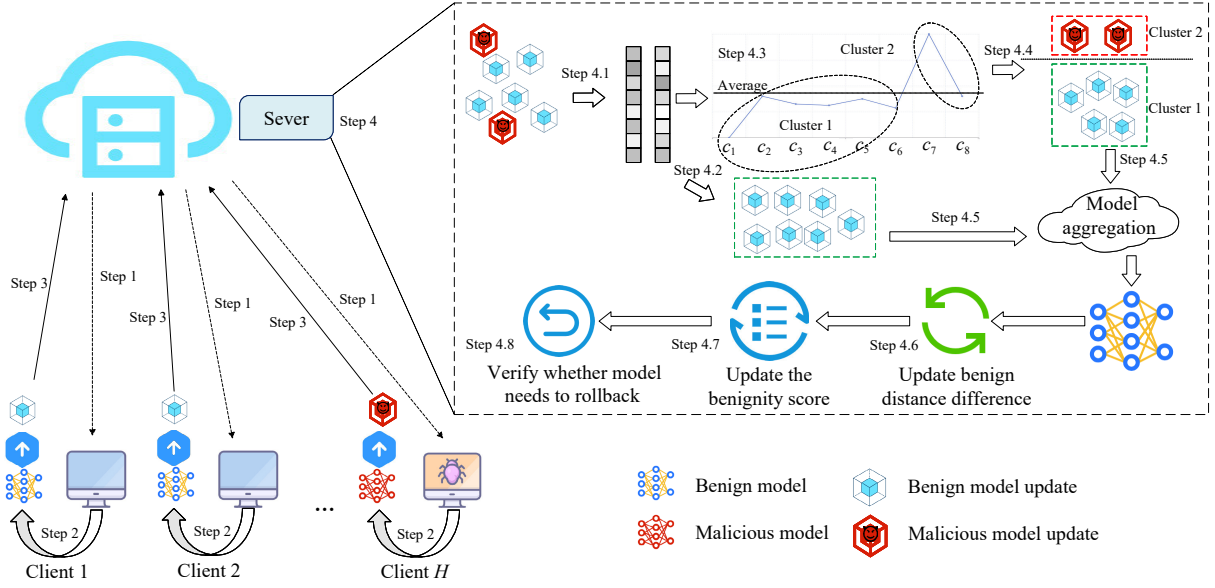


Fig. 1 Workflow of OCACSFL. OCACSFL consists of two main modules. Steps 4.1–4.5 and 4.8 employ the improved OPTICS clustering algorithm—or, when triggered, the model rollback mechanism—to obtain the benign model; Steps 4.6 and 4.7 adaptively updates the OCACSFL parameters.

aggregated, as detailed in Steps 4.2 and 4.5. If malicious model updates are identified, the server further analyzes and excludes these updates from the aggregation, as described in Steps 4.3, 4.4, and 4.5. After the global model for the current round is obtained, the aggregation server updates the relevant parameters, as outlined in Steps 4.6 and 4.7. If the accuracy of the global model does not meet expectations, the server reverts the global model to the state from the previous round, as illustrated in Step 4.8.

5.1 Byzantine-robust aggregation based on OPTICS clustering

Next, we will describe in detail the execution process of Step 4.1. Let ϵ be the maximum distance (or $+\infty$) among all model updates, and MinPts be 2. Under this setting, each model update is considered a core model update. The $|H|$ model updates are input into the OPTICS algorithm, which produces two queues: orders_clis and order_dists . The queue orders_clis represents the sequence of processed model updates. Without loss of generality, assume that $\text{orders_clis} = [g_1, g_2, \dots, g_H]$, then the value of $\text{order_dists}[h]$ represents the minimum reachability distance between g_h and $g_1, g_2, \dots, g_{h-1}$ ($h \in \{2, 3, \dots, |H|\}$). Initially, the elements of orders_clis and order_dists are undefined. First, the core distance of each model update is calculated according to Eq. (3). Let $\text{UG} = G$ represent the set of unprocessed model updates. A model update,

denoted as g_i , is randomly selected from UG . g_i is marked as processed, inserted into orders_clis , and assigned a corresponding value of 0 in order_dists . Then, the reachability distances between g_i and all other unprocessed model updates are calculated using Eq. (4) and stored in the reach_dists vector. When there are still unprocessed model updates, the following steps are repeated: select the smallest value from reach_dists , denoted as minDist , and assume that the corresponding model update is g_j . g_j is marked as processed, g_j and minDist are inserted into orders_clis and order_dists , respectively. Subsequently, reach_dists is updated. The reachability distances between g_j and all other unprocessed model updates using Eq. (4). If the distance between g_j and any model update is smaller than the corresponding value in reach_dists , the corresponding value is replaced by the smaller distance. The detailed procedure of the OPTICS algorithm is presented in Algorithm 1.

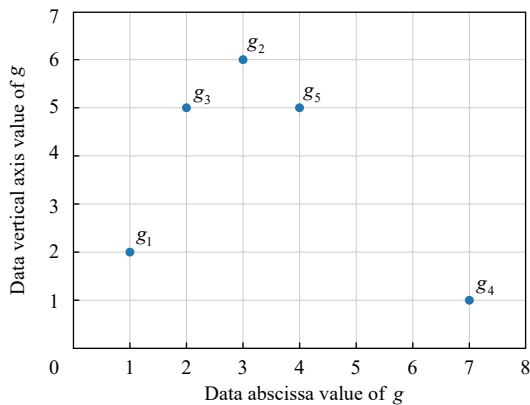
The following example illustrates the operational procedure of Algorithm 1. We suppose $|H| = 5$ and $G = \{g_1, g_2, g_3, g_4, g_5\}$. We suppose $g_1 = (1, 2)$, $g_2 = (3, 6)$, $g_3 = (2, 5)$, $g_4 = (7, 1)$, and $g_5 = (4, 5)$ as shown in Fig. 2, then their corresponding core distances are 3.16, 1.41, 1.41, 5.00, and 1.41, respectively. Initially, let $\text{UG} = G$. Server randomly select a model update from UG . Without loss of generality, g_1 is assumed to be selected. Then, g_1 is marked as processed (i.e., $\text{UG} = \text{UG} \setminus \{g_1\}$),

Algorithm 1 OPTICS cluster**Require:** Set of local updates G **Ensure:** orders_clis and order_dists

```

1: Initialize: orders_clis = [ ], order_dists = [ ], and reach_dists = [ ]
2:  $UG = G$ ;
3: Calculate core distance based on Eq. (3);
4: Randomly select a model update, denoted as  $g_i$ ;
5:  $UG = UG \setminus \{g_i\}$ ;
6: orders_clis [1] =  $g_i$ ;
7: order_dists [1] = 0, idx = 2;
8: reach_dists [ $i$ ] =  $+\infty$ ;
9: for  $\forall g_j \in UG$  do
10: Calculate reachability distance ( $g_i, g_j$ ) based on Eq. (4);
11: reach_dists [ $j$ ] = reachability-distance ( $g_i, g_j$ );
12: end for
13: while  $UG$  is not empty do
14: Select the model update corresponding to the minimum value in reach_dists, denoted as  $g_j$  (that is, reach_dists [ $j$ ] is the smallest);
15:  $UG = UG \setminus \{g_j\}$ ;
16: orders [idx] =  $g_j$ ;
17: order_dists [idx] = reach_dists [ $j$ ], idx = idx + 1;
18: reach_dists [ $j$ ] =  $+\infty$ ;
19: for  $\forall g_k \in UG$  do
20: Calculate reachability distance ( $g_j, g_k$ ) based on Eq. (4);
21: reach_dists [ $k$ ] = reachability-distance ( $g_j, g_k$ );
22: end for
23: end while
24: Return orders and order_dists

```

**Fig. 2** Data distribution of the example.

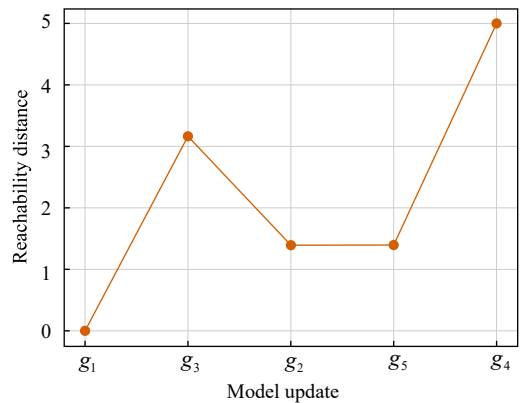
orders_clis [1] is set to g_1 , and order_dists [1] is set to 0. In addition, reach_dists [1] is set to $+\infty$. Then, reach_dists are updated as $\{+\infty, 4.47, 3.16, 6.08, 4.24\}$. Since

reach_dists [3] is smallest, server next selects g_3 . g_3 is marked as processed (i.e., $UG = UG \setminus \{g_3\}$), orders_clis [2] is set to g_3 , and order_dists [2] is set to 3.16. Subsequently, reach_dists is updated, and reach_dists [3] is set to $+\infty$. The reachability distances between g_3 and g_2, g_4, g_5 are 1.41, 6.40, 2.00, respectively. Due to $1.41 < 4.47$, $6.08 < 6.40$, and $2.00 < 4.24$, reach_dists [2] and reach_dists [5] are replaced with 1.41 and 2.00, respectively. Server repeats the process until UG is empty. Finally, orders_clis = $\{g_1, g_3, g_2, g_5, g_4\}$ and order_dists = $\{0, 3.16, 1.41, 1.41, 5.00\}$. A reachability distance plot (Fig. 3) is generated by plotting orders_clis on the horizontal axis and order_dists on the vertical axis.

Next, the clustering process is described. First, server computes the average value of order_dists as defined in Eq. (5),

$$\text{Avg_dist} = \frac{\sum_{h=2}^{|H|} \text{order_dists}[h]}{|H| - 1} \quad (5)$$

The Avg_dist serves as the threshold for cluster partitioning. The first element of order_dists is excluded from this calculation. Clearly, the first model update in the orders_clis sequence is assigned to the initial cluster. Subsequently, server compares each order_dists [h] ($h = 2, 3, \dots, |H|$) with Avg_dist. If order_dists [h] is smaller than Avg_dist, the orders_clis [h] is grouped into the same cluster as orders_clis [$h-1$]. Otherwise, orders_clis [h] becomes the first member of a new cluster (see Steps 4.3 and 4.4 in Fig. 1). For the above example, the value of Avg_dist of order_dists is 2.745. The clustering process is shown in Fig. 4, it is clear that g_1, g_2 , and g_5 are below Avg_dist, and g_3 and g_4 are above Avg_dist. The

**Fig. 3** Reachability distance plot of the example.

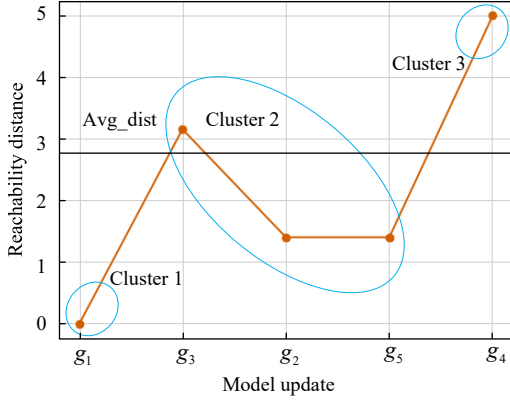


Fig. 4 OPTICS clustering process.

clustering result is shown in Fig. 5. Clearly, g_1 is in a cluster, g_2 , g_3 , and g_5 are in a cluster, and g_4 is in a cluster. Based on the assumption that benign model updates constitute the majority, the largest cluster is identified as containing benign model updates. Finally, the server aggregates these benign model updates to generate the aggregated global model for the current round, as illustrated in Step 4.5 of Fig. 1.

The adoption of the mean value for both cluster partitioning and benign update evaluation stems from two fundamental rationales. First, the mean value is dynamic and only determined after all clients have uploaded their model updates, making it difficult for attackers to predict it in advance. This unpredictability prevents attackers from reducing the difference between malicious and benign model updates. Second, using the mean value effectively leverages the spatial distribution characteristics of benign and malicious model updates. Benign model updates are generally clustered closely together, whereas they are spatially distant from malicious model updates. Employing the mean value helps to clearly distinguish between these

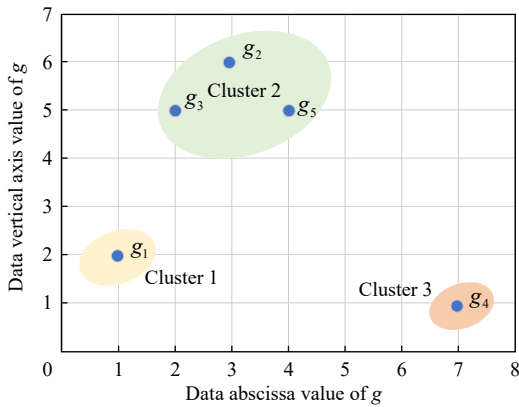


Fig. 5 Result of OPTICS clustering.

distributions, reducing the likelihood of misclassification.

The Byzantine-robust aggregation method based on OPTICS clustering is similar to the Krum algorithm mentioned above in calculating the model updates differences, but it differs in how it distinguishes between benign and malicious model updates. Krum identifies benign model updates by selecting local model updates with the smallest model updates differences from $n-m-2$ neighboring model updates. However, in practical scenarios, the number of benign and malicious model updates in each round is usually unknown and varies, making it challenging to optimally set the parameter m . If m is too small, malicious model updates may be mistakenly classified as benign. Conversely, if m is too large, some benign model updates may be misclassified as malicious, causing a loss of benign data from the aggregation. In contrast, the OPTICS-based Byzantine-robust aggregation method does not require predefining the numbers of benign and malicious model updates, relying instead on the assumption that benign model updates are in the majority.

5.2 Benign distance difference

The accuracy of OPTICS clustering relies on a clear distinction between benign and malicious model updates. In practice, however, due to the heterogeneity of training data across clients, local models trained by different clients inherently exhibit variations. This implies that when the differences between model updates are relatively small, forcibly categorizing them into benign or malicious may lead to misclassification, where some benign model updates are incorrectly labeled as malicious, resulting in wasted training data. To address this issue, this paper introduces the concept of “benign distance difference”. Before clustering the model updates, the “maximum reachability distance difference” between the updates is computed. If the “maximum reachability distance difference” is smaller than the predefined “benign distance difference”, all model updates are considered benign.

First, we define the “maximum reachability distance difference” (denoted as $\max\text{DisDiff}$) as the difference between the maximum and minimum values of order_dists in each round, as shown in Eq. (6),

$$\max\text{DisDiff} = \max \{ \text{order_dists} [i] \} - \min \{ \text{order_dists} [i] \} \quad (6)$$

where $i \in \{2, 3, \dots, H\}$. Next, we define the benign

distance difference (denoted as BenignDisDiff) as the difference between the maximum and minimum reachability distances among all benign model updates in each round, formalized in Eq. (7),

$$\text{BenignDisDiff} = \max \{\text{order_dists}[i]\} - \min \{\text{order_dists}[i]\} \quad (7)$$

where $g_i \in \text{benign_C}$, and benign_C is the cluster of benign model updates.

Extensive experiments show that even all model updates are benign, the “maximum reachability distance difference” inevitably exhibits minor fluctuations across different rounds. To prevent such fluctuations from compromising the accuracy of the aggregation server in identifying benign clients, we introduce a relaxation adjustment to the “benign distance difference”. Specifically, in each round, we scale BenignDisDiff by a relaxation factor α in Eq. (8),

$$\text{BenignDisDiff} = \text{BenignDisDiff} \times \alpha \quad (8)$$

This modification strategically broadens the tolerance threshold of BenignDisDiff, explicitly accounting for inherent fluctuations in reachability distances induced by benign data heterogeneity, thereby enhancing the robustness of benign cluster classification criteria through systematic inclusion of natural variance patterns.

When setting the hyperparameter α , a basic principle must be followed. The value of α must not be taken such that the BenignDisDiff extends to the same reachability distance difference as that between malicious model updates. Otherwise, the aggregation server may incorrectly identify a malicious model updates as benign. We conducted a lot of experiments on MNIST, FMNIST, and CIFAR-10 datasets to check the range of reachability distance between benign models (shown in Table 1) and between benign and malicious models (shown in Table 2) when Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP) models are attacked. The experimental results show that the reachability distance between benign and malicious models is at least twice as large—and can be up to 10^4 times greater—than the distances measured among benign models. Therefore, in this paper, α is set to 2.

We use distance difference rather than distance as a judgement of whether all model updates in this round are benign, mainly based on the following empirical findings. As shown in Figs. 6 and 7, with the

Table 1 Range of reachability distance among benign model updates.

Dataset–Model	Attack		
	Sign	Noise	Backdoor
MNIST–MLP	0–5	0–5	0–5
FMNIST–MLP	0–5	0–5	0–5
MNIST–CNN	0–20	0–20	0–20
FMNIST–CNN	0–200	0–200	0–200
CIFAR10–CNN	0–300	0–300	0–300

Table 2 Reachability distance between benign model updates and malicious models updates.

Dataset–Model	Attack		
	Sign	Noise	Backdoor
MNIST–MLP	10–120	5.0×10^4 – 5.3×10^4	3.0×10^3 – 2.0×10^4
FMNIST–MLP	10–120	5.0×10^4 – 5.2×10^4	5.0×10^3 – 2.0×10^4
MNIST–CNN	45–150	2.0×10^4 – 2.5×10^4	5.0×10^4 – 5.0×10^6
FMNIST–CNN	9.0×10^4 – 8.0×10^6	2.0×10^4 – 3.0×10^4	2.0×10^6 – 5.0×10^7
CIFAR10–CNN	8.0×10^4 – 1.0×10^7	5.8×10^6 – 6.0×10^6	2.0×10^4 – 3.0×10^7

progression of federated learning training rounds, both the reachability distance between benign model updates and the reachability distance separating benign from malicious updates exhibit a decreasing trend. In contrast, BenignDisDiff maintains relative stability throughout training phases, while consistently remaining smaller than maxDisDiff observed in scenarios involving malicious updates.

In each round, after receiving the model updates uploaded by clients, the aggregation server computes maxDisDiff and compares it with BenignDisDiff. If $\text{maxDisDiff} < \text{BenignDisDiff}$, all model updates are considered benign and directly aggregated (as shown in Steps 4.2 and 4.5 of Fig. 1). Otherwise, the

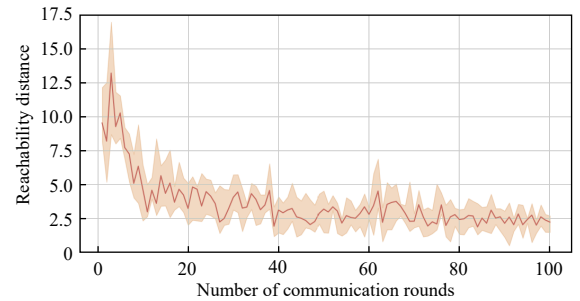


Fig. 6 Reachability distances between CNN benign model updates on the MNIST dataset (where the shading is the range of reachability distances among order_dists obtained in each round, and the solid line is the median number of order_dists. Obviously the difference between the shaded high and low points in each round is maxDisDiff.)

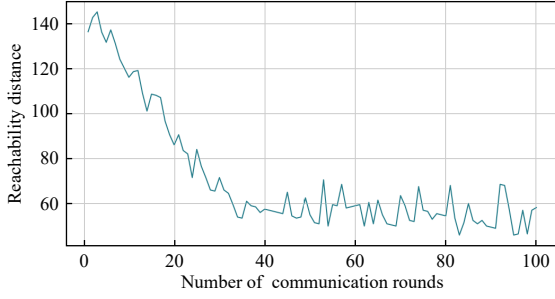


Fig. 7 Reachability distance between CNN benign and malicious model updates on the MNIST dataset.

OPTICS algorithm is executed to cluster the updates and filter out malicious ones (Steps 4.3 and 4.4 in Fig. 1). And then, the parameter server aggregates benign updates (Step 4.5 in Fig. 1).

5.3 Model rollback mechanism

Although in practical federated learning scenarios, the number of malicious clients usually constitutes only a small portion of all clients, during each round of aggregation when the server selects participating clients, situations may still arise as shown in Fig. 8, where the number of selected malicious clients exceeds half of the total selected clients for that round. Although the probability of such an occurrence is extremely low, if it does happen, the method proposed in this paper for distinguishing between benign and malicious model updates will fail. To address this situation, we propose a model rollback mechanism to ensure the robustness of the global model.

Without loss of generality, we illustrate the operation process for round t ($t > 1$). The aggregation server first sends the global model w_t to clients. The clients test the accuracy of w_t with their own test data and upload the test results to the aggregation server. The

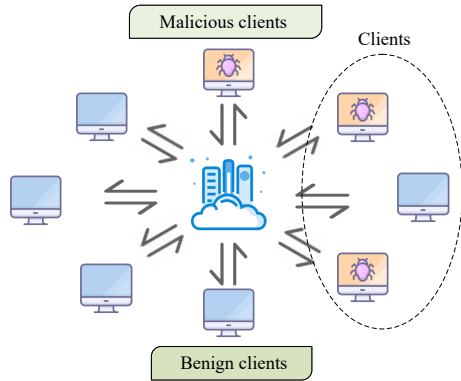


Fig. 8 Majority of selected malicious clients.

aggregation server calculates the average of the test results, denoted by AvgACC_t . If $\text{AvgACC}_t < \beta \times \text{AvgACC}_{t-1}$, the aggregation model w_t rolls back to the aggregation model w_{t-1} in round $t-1$, i.e., $w_t = w_{t-1}$, where β ($0 < \beta < 1$) is called the rollback threshold. The model rollback process is shown in Step 4.8 in Fig. 1.

We conduct extensive experiments to check how the test accuracy of different models varies on different datasets, assuming that there is no malicious model attack. The experimental results are shown in Figs. 9–11. As can be seen from Fig. 9, on the Independent and Identically Distributed (IID) dataset, the accuracy of the models rises smoothly without large oscillations. Whereas, when the data presents non-IID, there is a large oscillation in the test accuracy of the model. On the simple datasets MNIST and FMNIST, the test accuracy of the global model w_t decreases by no more than 10% compared to the test accuracy of w_{t-1} , as shown in Fig. 10. On the complex dataset CIFAR-10, the test accuracy of the global model w_t decreases by no more than 15% compared to the test accuracy of the global model w_{t-1} , as shown in Fig. 11. That is, the test accuracy of the global model w_t is not less than 85% of the test accuracy of w_{t-1} . Therefore, in the experiments of this paper, the value of

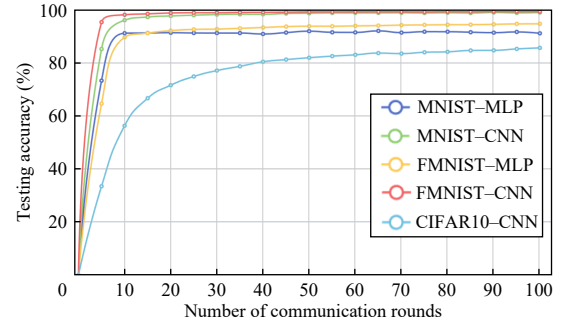


Fig. 9 Test accuracy of models on IID-databases.

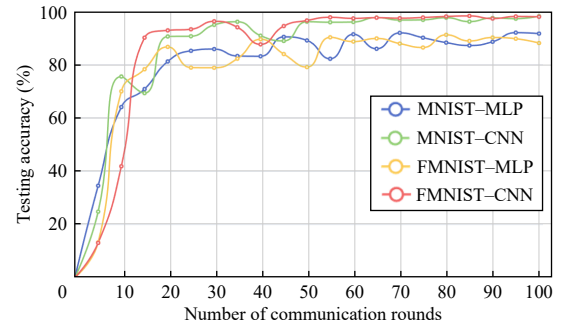


Fig. 10 Test accuracy of models on non-IID databases MNIST and FMNIST.

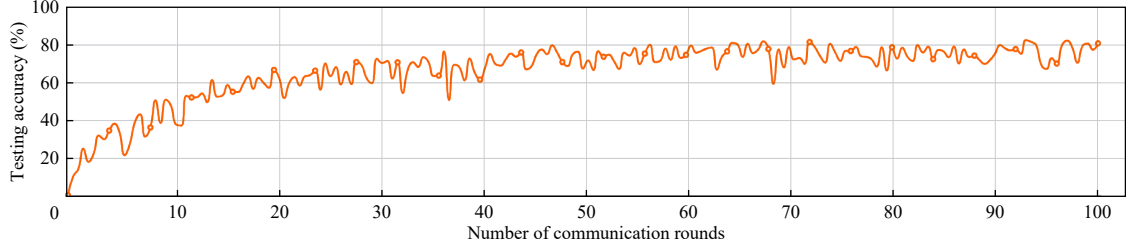


Fig. 11 Test accuracy of models on non-IID database CIFAR-10.

β is set to 0.85. In practical applications, users can set the value of β according to the complexity of the dataset and the requirement of robustness to Byzantine attacks.

Due to the heterogeneity of training data from different clients, the server randomly selects different clients to update the global model, and it is normal for the test accuracy of the global model to oscillate during the updating process. Under the attack of malicious models, the test accuracy of global models is more likely to oscillate. Regardless of the reason for the significant drop in global model test accuracy, the model rollback mechanism effectively prevents the degradation of model performance and ensures the convergence of the global model.

5.4 Client selection

Before the start of each training round, the aggregation server first selects $|H|$ clients to participate in that round of training. In this paper, we propose a client selection mechanism based on credit score, as shown in Step 4.7 of Fig. 1. The aggregation server records the credit score of each client. During a certain training round, if a client is recognised as malicious, the client's credit score will be deducted; on the contrary, if a client is recognised as benign, its credit score will be increased. Let score_i denote the credit score of client c_i , and the credit scores of all clients are initialised to score_upper . The selected probability of client c_i is defined as Eq. (9),

$$p_i = \frac{\text{score}_i}{\sum_{j=1}^N \text{score}_j} \quad (9)$$

At the end of a training round, the server updates the credit score of client c_i in Eq. (10),

$$\text{score}_i = \begin{cases} \min(\text{score}_i + s, \text{score_upper}), & \text{if } c_i \text{ is benign} \\ \text{score}_i - s \times \gamma^{r_i-1}, & \text{if } c_i \text{ is malicious} \end{cases} \quad (10)$$

where $s > 0$ denotes the credit score awarded to a benign client. That is, in a training round, if c_i is identified as a benign client, its credit score is increased by s until it reaches the upper bound score_upper . $\gamma > 1$ denotes the multiplier by which the credit score is deducted when a client is identified as a malicious client, and r_i denotes the number of times that a client c_i has been "consecutively" identified as a malicious client, with the initial value of r_i set to 0. For example, suppose that c_i is selected in 4 consecutive training rounds ($t, t+1, t+2, t+3$). Among them, c_i is identified as a malicious client in rounds $t, t+1$, and $t+3$, while it is identified as a benign client in round $t+2$. Then, $r_i = 1$ at the end of round t , $r_i = 2$ at the end of round $t+1$, r_i is reset to 0 at the end of round $t+2$, and $r_i = 1$ at the end of round $t+3$. In order to quickly identify and eliminate malicious clients, the credit score of a malicious client is deducted exponentially, while that of a client recognised as a benign client is increased linearly. If $\text{score}_i \leq 0$, then c_i is identified as a malicious client and the probability of being selected by the aggregation server is 0. This means that c_i will never have a chance to attack the aggregation model.

If a malicious client continuously performs attacks on the aggregation model, it will be quickly recognised by the aggregation server and excluded from the FL system. Whereas the probability of a benign client being misidentified as a malicious client and excluded from the FL system is very small. We verify the above conclusion in Section 6.4.3.

The pseudocode for the main algorithm of this paper is shown in Algorithms 2 and 3.

6 Experiment

In this section, we evaluate the robust federated learning performance of our OCACSFL method for image classification tasks on three public datasets. The effectiveness of OCACSFL is demonstrated by comparing it with two baseline defense mechanisms

Algorithm 2 Secure aggregation

Require: Set of local model updates G , set of selected clients H , score at global round t , and upper limit of credit score score_upper

Ensure: Global model w_t

```

1: for client  $c_i$  in  $H$  do
2:   if  $\text{score}_i \leq 0$  then
3:     Delete  $g_i$ ;
4:   end if
5: end for
6: Calculate the distance between each pair of  $g_i$  and  $g_j$ ;
7:  $\text{max\_distance} = \max(\text{distance})$ ;
8: OPTICS clustering algorithm is performed to obtain
   orders and order_dists; // Algorithm 1
9: Calculate  $\text{maxDisDiff}$  through Eq. (6);
10: if  $\text{maxDisDiff} < \text{BenignDisDiff}$  then
11:   for client  $c_i$  in  $H$  at global round  $t$  do
12:     if  $\text{score}_i < \text{score\_upper}$  then
13:        $\text{score}_i = \text{score}_i + s$ ;
14:     end if
15:   end for
16:   Return  $\text{fedavg}(G)$ ; //  $\text{fedavg}(\cdot)$  is the federated averaging
   function
17: else
18:    $\text{Average} = \text{avg}(\text{order\_dists})$ ;
19:    $\text{New cluster}_{j=0} = []$ ;
20:   for client  $c_i$  in  $H$  at global round  $t$  do
21:     if  $\text{order\_dists}[i] < \text{Average}$  then
22:       Add  $c_i$  to  $\text{cluster}_j$ ;
23:     else
24:        $\text{New cluster}_{j=j+1} = []$ ;
25:       Add  $c_i$  to  $\text{cluster}_j$ ;
26:     end if
27:   end for
28:   Define the cluster with the largest number of clients as
    $\text{benign\_cluster}$ ;
29:   for client  $c_i$  in  $H$  at global round  $t$  do
30:     if client  $c_i$  in  $\text{benign\_cluster}$  then
31:       if  $\text{score}_i < \text{score\_upper}$  then
32:          $\text{score}_i = \text{score}_i + s$ ;
33:       else
34:          $\text{score}_i = \text{score}_i - s \times \text{multiple}^{t-1}$ ;
35:       end if
36:     end if
37:   end for
38:   Update  $\text{BenignDisDiff}$  through Eq. (7);
39:   Return  $\text{fedavg}(g_i \text{ in } \text{benign\_cluster})$ ;
40: end if

```

Algorithm 3 Main process of OCACSFL

Require: Initial global model w_0 , global round T , and data of clients D

Ensure: Final global model w_T

```

1: Initialize: Score of clients, upper limit of credit score
    $\text{score\_upper}$ , drop threshold  $\beta$ , and relaxed factor  $\alpha$ 
2: for  $t < T$  do
3:   Select  $|H|$  clients from clients pool on Eq. (9);
4:   for client  $c_i$  in  $H$  do
5:     Receive  $w_{t-1}$  from the server;
6:     Train  $w_{t-1}$  on  $D_i$ ;
7:     Send the local model updates  $g_i$  to server;
8:   end for
9:   Server receives all  $g_i$  from  $|H|$  clients;
10:  Obtain  $w_t$  by Algorithm 2;
11:  if  $\text{AvgACC}_t < \beta \times \text{AvgACC}_{t-1}$  then
12:     $w_t = w_{t-1}$ ;
13:  end if
14: end for
15: Return global model  $w_T$ 

```

and an ideal baseline without attacks. The experiments are implemented using PyTorch framework.

6.1 Experimental setup

In our experiments, we assume a typical federated learning scenario consisting of an aggregation server and 100 clients, with a portion of them being malicious clients. We validate the effectiveness of the proposed method under both targeted and untargeted attack types. Specifically, we conduct experiments in scenarios with 10 and 30 malicious clients for both targeted and untargeted attacks, and in each round of global training, 10 clients are randomly selected to participate in secure aggregation. Based on the Ref. [21], we consider the following attack types for testing.

Sign-flipping attack: The sign-flipping attack is an untargeted attack in which malicious clients manipulate the uploaded data by flipping the sign of the model updates, thus adjusting the model updates direction to induce incorrect model predictions. These perturbations are almost imperceptible to human observers. In the experiment, malicious clients executing a sign-flipping attack invert the model updates before uploading. Specifically, the local update of malicious client k is defined as $w^k = \sigma \hat{w}^k$, where $\hat{w}^k$ is the true model update, and σ is a flip constant. In this experiment, σ is set to -1 .

Additive noise attack: The additive noise attack is

another untargeted attack, where malicious clients add specific noise (typically Gaussian white noise) to the uploaded data, effectively misleading the model's output. It is worth noting that adding controlled noise can effectively protect sensitive information in model updates, but excessive noise can negatively affect model's accuracy. In the experiment, malicious clients conduct a noise-based adversarial attack via injecting Gaussian noise with a mean of 0 and standard deviation of 1 to the model updates prior to model transmission. Specifically, the local update of malicious client k is given by $w^k = \hat{w}^k + \varepsilon$, where $\hat{w}^k$ is the true model update and ε represents Gaussian noise with mean 0 and standard deviation 1.

Backdoor attack: The backdoor attack is a targeted attack in which malicious clients introduce a “backdoor” during model training to produce a specified incorrect output under certain trigger conditions, while maintaining the main model's performance under normal conditions. In this experiment, malicious clients manipulate local models to misclassify certain images within each dataset. In the MNIST dataset, images labeled as “6” are reassigned to the “0” category, and images labeled as “7” are reassigned to the “1” category. In the FMNIST dataset, images labeled as “Shirt” are reassigned to the “T-shirt” category, and images labeled as “Sneaker” are reassigned to the “Sandal” category. In the CIFAR-10 dataset, images labeled as “Dog” are reassigned to the “Cat” category, and images labeled as “Horse” are reassigned to the “Deer” category.

The experimental parameters are configured as follows: each client is initialized with a credit score of 5 points, s is set to 1, score_upper is set to 5, γ is set to 2, α is set to 2, and β is set to 0.85. In all experiments, the global aggregation rounds are set to 100, except for the CIFAR-10 dataset under non-IID conditions, where the global aggregation rounds are set to 1000. Each round consists of 10 training epochs, with a batch size of 50. The Adam optimizer is used with a learning rate of 0.01, a momentum parameter of 0.9, and a weight decay coefficient of 1×10^{-4} .

6.2 Datasets and models

To validate the effectiveness of the OCACSFL method, we conduct experiments on three public datasets—MNIST, FMNIST, and CIFAR-10—using two common machine learning models, MLP and CNN.

MNIST dataset: The MNIST dataset contains 70 000

images of handwritten digits (0–9), each image sized at 28 pixels $\times$ 28 pixels, with 60 000 images in the training set and 10 000 in the test set.

FMNIST dataset: The FMNIST dataset is an alternative to the MNIST dataset, specifically designed for research in federated learning. It contains over 60 000 images of merchandise across 10 categories (T-shirt, Trousers, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot), with each image sized to 28 pixels $\times$ 28 pixels.

CIFAR-10 dataset: The CIFAR-10 dataset consists of 60 000 color images of 10 classes (Airplane, Car, Bird, Cat, Deer, Dog, Frog, Horse, Ship, and Truck), each with a resolution of 3 channels $\times$ 32 pixels $\times$ 32 pixels.

For the MNIST and FMNIST datasets, we train an MLP model with a dense layer of 64 units. A CNN model is trained on the MNIST, FMNIST, and CIFAR-10 datasets. The CNN model for the MNIST dataset includes two convolutional layers with kernel sizes of 5×5 and 32/16 output channels, i.e., $5 \times 5 \times 32$ and $5 \times 5 \times 16$, followed by a fully connected layer with 50 units. For the FMNIST dataset, the CNN model consists of two convolutional layers ($5 \times 5 \times 16$ and $5 \times 5 \times 32$) along with two batch normalization layers (corresponding to the 16 and 32 channels). For the CIFAR-10 dataset, the CNN model contains four convolutional layers ($3 \times 3 \times 32$, $3 \times 3 \times 64$, $3 \times 3 \times 128$, and $3 \times 3 \times 128$) and two batch normalization layers (32 and 128 channels).

6.3 Baselines

FedAvg^[1]: The aggregation server receives model updates from clients and then performs a weighted aggregation based on the proportion of each client's dataset relative to the total dataset size. In this study, we evaluate the performance of the FedAvg algorithm under both attack and no-attack conditions.

Krum^[23]: The Krum algorithm selects a single local model update to represent the entire global model update. The selected local update is the one that minimizes the sum of Euclidean distances to its nearest $n - m - 2$ local updates.

GeoMed^[24]: GeoMed generates a global model update using the geometric median of all local model updates, which may not correspond to any of the local model updates themselves.

ClippedClustering^[16]: ClippedClustering first prunes the oversized model updates, then clusters the model updates by hierarchical clustering, selects the

cluster containing the highest number of model updates as a benign cluster, and uses the model updates in it to participate in model aggregation.

6.4 Experimental results

6.4.1 Robustness to untargeted attacks

In the experiments, we consider two types of untargeted attacks: sign-flipping attack and additive noise attack. The experimental results of the sign-flipping attack are shown in Tables 3 and 4. Under sign-flipping attack conditions, the proposed method demonstrates strong defense performance across all

three public datasets. When the proportion of malicious clients is 10%, the number of malicious clients participating in each round is relatively small, and both Krum and GeoMed achieve satisfactory results. In this case, the advantages of OCACSFL are not yet significant. However, when the proportion of malicious clients increases to 30%, the number of malicious clients participating in each round significantly increases, and the defense performance of Krum and GeoMed decreases to varying degrees across different datasets, especially on the CIFAR-10 dataset, where the test accuracy approaches that of FedAvg without

Table 3 Robustness on IID datasets. The experiment results are presented as model accuracy.

(Unit: %)

Dataset	Model	Method	No-attack	Sign-flipping		Additive noise		Backdoor	
				10%	30%	10%	30%	10%	30%
MNIST	MLP	Krum	92.6	91.1	85.3	91.4	90.8	91.7	90.6
		GeoMed	92.6	91.3	60.9	91.6	90.9	92.0	73.6
		ClippedClustering	92.6	91.9	90.2	82.3	59.6	9.8	43.4
		ClippedClustering (all)	92.6	92.3	91.8	83.1	56.6	91.4	90.8
		OCACSFL	92.6	92.3	91.8	92.4	91.9	92.4	91.5
		FedAvg	92.6	62.1	15.0	84.5	48.2	42.1	20.2
	CNN	Krum	99.2	96.1	12.1	98.9	98.7	99.1	10.3
		GeoMed	99.2	98.9	11.9	99.1	98.2	98.7	10.2
		ClippedClustering	99.2	98.7	11.4	11.5	16.7	98.7	11.3
		ClippedClustering (all)	99.2	99.0	97.8	99.0	56.8	98.8	97.5
		OCACSFL	99.2	99.0	97.8	99.0	98.8	99.2	97.5
		FedAvg	99.2	90.2	10.0	12.4	11.1	10.0	9.7
FMNIST	MLP	Krum	94.2	92.4	89.2	90.2	90.1	93.3	90.6
		GeoMed	94.2	93.6	53.5	94.1	88.7	93.7	89.2
		ClippedClustering	94.2	94.1	91.1	82.4	60.2	29.2	58.6
		ClippedClustering (all)	94.2	94.2	94.1	94.1	87.8	86.2	83.6
		OCACSFL	94.2	94.2	92.1	94.1	94.1	93.9	92.3
		FedAvg	94.2	75.6	10.0	66.7	46.7	63.6	32.8
	CNN	Krum	99.3	98.7	76.4	98.8	98.8	99.1	10.6
		GeoMed	99.3	99.1	68.9	99.3	98.5	99.2	39.6
		ClippedClustering	99.3	97.5	93.4	95.6	42.2	99.1	9.8
		ClippedClustering (all)	99.3	99.1	97.2	99.3	78.5	99.2	97.2
		OCACSFL	99.3	99.1	98.2	99.3	98.8	99.2	97.2
		FedAvg	99.3	95.7	10.0	89.8	79.2	10.0	10.0
CIFAR10	CNN	Krum	82.8	66.3	10.0	65.1	64.4	68.6	10.0
		GeoMed	82.8	82.7	10.0	82.4	10.0	82.3	10.0
		ClippedClustering	82.8	82.6	10.0	10.0	10.0	10.0	10.0
		ClippedClustering (all)	82.8	82.7	78.5	40.7	10.0	78.6	67.8
		OCACSFL	82.8	82.5	80.9	82.6	79.2	82.5	78.4
		FedAvg	82.8	10.0	10.0	10.0	10.0	10.0	10.0

Note: The “No-attack” column is the result of our baseline experiment, i.e., the experimental result of FedAvg under no-attack. The model accuracy is represented by the average accuracy over the final 10% of training rounds. ClippedClustering selects 10% of clients per round. ClippedClustering (all), as specified in Ref. [16], selects all clients in each round. The best results are bolded.

Table 4 Robustness on non-IID datasets. The experiment results are presented as model accuracy.

(Unit: %)

Dataset	Model	Method	No-attack	Sign-flipping		Additive noise		Backdoor	
				10%	30%	10%	30%	10%	30%
MNIST	MLP	Krum	88.4	42.3	38.7	42.2	41.8	56.3	51.4
		GeoMed	88.4	64.8	20.9	85.7	79.7	82.3	28.6
		ClippedClustering	88.4	86.2	85.6	72.1	50.4	30.8	14.2
		ClippedClustering (all)	88.4	88.1	87.9	88.4	50.7	10.0	11.9
		OCACSFL	88.4	86.8	85.2	86.1	83.2	87.2	81.9
		FedAvg	88.4	28.1	19.2	51.3	41.2	10.0	10.0
	CNN	Krum	96.9	14.7	10.0	10.0	10.0	10.0	10.0
		GeoMed	96.9	82.4	16.3	94.9	80.4	94.6	10.0
		ClippedClustering	96.9	94.5	10.2	10.3	10.0	11.3	10.2
		ClippedClustering (all)	96.9	94.7	88.6	86.3	10.3	82.6	10.3
		OCACSFL	96.9	94.7	90.4	95.6	89.0	93.7	87.5
		FedAvg	96.9	40.8	10.8	10.0	10.0	10.0	10.0
FMNIST	MLP	Krum	88.9	72.7	50.1	53.6	61.1	57.4	61.2
		GeoMed	88.9	61.5	18.3	88.7	77.1	88.5	10.0
		ClippedClustering	88.9	88.3	87.3	73.8	47.3	10.3	11.3
		ClippedClustering (all)	88.9	88.6	88.5	71.8	52.4	54.4	78.2
		OCACSFL	88.9	86.6	85.8	87.1	82.6	88.2	81.2
		FedAvg	88.9	37.2	17.8	62.1	40.3	10.0	10.0
	CNN	Krum	97.5	10.6	10.4	10.0	10.0	10.0	10.0
		GeoMed	97.5	97.2	42.8	94.1	81.8	93.2	10.0
		ClippedClustering	97.5	97.3	20.9	52.3	21.8	11.3	9.8
		ClippedClustering (all)	97.5	97.2	82.7	81.2	15.2	52.7	57.1
		OCACSFL	97.5	93.6	83.2	81.8	81.2	91.4	90.8
		FedAvg	97.5	35.8	10.0	45.4	10.0	10.0	10.0
CIFAR10	CNN	Krum	77.6	10.0	10.0	10.0	10.0	10.0	10.0
		GeoMed	77.6	10.0	10.0	10.0	10.0	70.2	10.0
		ClippedClustering	77.6	61.4	10.0	10.0	10.0	10.0	10.0
		ClippedClustering (all)	77.6	77.3	66.7	74.8	57.4	10.0	10.0
		OCACSFL	77.6	76.4	68.9	74.8	71.2	73.2	72.0
		FedAvg	77.6	10.0	10.0	10.0	10.0	10.0	10.0

Note: The “No-attack” column is the result of our baseline experiment, i.e., the experimental result of FedAvg under no-attack. The model accuracy is represented by the average accuracy over the final 10% of training rounds. ClippedClustering selects 10% of clients per round. ClippedClustering (all), as specified in Ref. [16], selects all clients in each round. The best results are bolded.

any defense mechanism. In contrast, the proposed method maintains stable defense performance, with results close to those of FedAvg under no attack conditions. The ClippedClustering method performs well when all clients participate in aggregation each round, but when only a subset of clients is selected for aggregation, the model accuracy declines, particularly on the MNIST dataset, CNN models, and in the scenario with 30% malicious clients, where the accuracy drop is particularly significant.

The experimental results of the additive noise attack are shown in Columns 7 and 8 of Tables 3 and 4. Under additive noise attacks, the proposed method still demonstrates good defense performance across all three public datasets. Due to the relatively weak nature of additive noise attacks, Krum and GeoMed also perform well in most cases, especially on the MNIST and FMNIST datasets, where they achieve about 90% accuracy, and even close to 99% accuracy in CNN models. When the proportion of malicious clients

reaches 30%, the performance of Krum, GeoMed, and OCACSFL declines less significantly compared to other attacks, and their defense abilities remain stable. However, the ClippedClustering method does not perform well against additive noise attacks in many cases, regardless of the proportion of clients selected for aggregation each round. We hypothesize that the reason for this is shown in Fig. 12, where the model updates are shifted in different directions due to the random nature of the additive noise attack. The ClippedClustering method eventually divides the clients into two categories, but when the noise causes model updates to shift in multiple directions, ClippedClustering fails to completely separate malicious and benign clients, and both categories may include malicious clients, leading to a significant decrease in model accuracy.

6.4.2 Robustness to backdoor attacks

The experimental results of the backdoor attack are shown in Tables 3 and 4. In backdoor attacks, when the proportion of malicious clients is 10%, only a few malicious clients are selected for training in each round, OCACSFL, Krum, and GeoMed all demonstrate good defense performance. However, when the proportion of malicious clients increases to 30%, the performance of Krum and GeoMed declines significantly more than that of the proposed method, especially in the CNN model, where the accuracy of Krum and GeoMed drops to around 10%. ClippedClustering also performs poorly in this scenario, and we hypothesize that its performance degradation mechanism is similar to that observed under additive noise attacks. In backdoor attacks, model updates shift due to the label modification introduced by malicious clients. When different labels are substituted, the model updates move in different directions, causing both categories of clients in the final clustering to contain malicious clients. This substantially affects model accuracy.

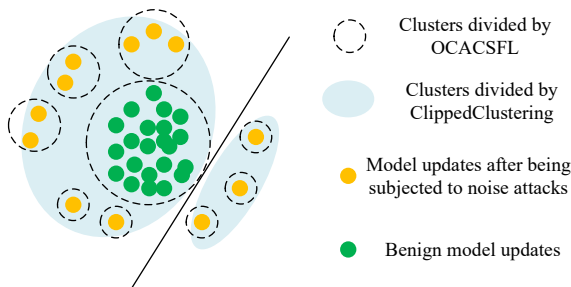


Fig. 12 Model updates shift after adding noise.

In examining the experimental results, we observe the following three phenomena. First, under non-IID conditions, the Krum method fails to achieve effective defense performance with the CNN model. We hypothesize that this is due to the significant data heterogeneity among clients in non-IID scenarios. Krum selects a single model from all uploaded models as the final aggregated model, and this chosen model may represent only a single class or a few classes of data. As a result, the model struggles to accurately classify other categories, which adversely impacts overall classification performance. Second, when the proportion of malicious clients increases from 10% to 30%, the GeoMed method experiences the most significant performance drop. We believe this is because GeoMed calculates the geometric median of the uploaded models to serve as the final aggregated model. When the number of malicious clients rises, these malicious models exert a greater influence on the calculation of the geometric median, resulting in substantial median shifts that considerably reduce the effectiveness of the final model. Lastly, when malicious clients make up the majority of selected clients in a given round, all defense methods compared in this study fail, and the loss in model accuracy becomes irrecoverable. Figure 13 shows the variation in model accuracy of different defense methods over training rounds under sign-flipping attacks on the CIFAR-10 dataset. The FedAvg (no-attack) curve represents the ideal baseline performance in the absence of attacks. When five out of ten selected clients are malicious in round 15, the performance of Krum begins to decline, and the accuracy of GeoMed drops to approximately 10%, with no chance of recovery. However, because OCACSFL identifies and excludes one malicious client within the first 14 rounds, it is less affected in round 15. Nonetheless, in

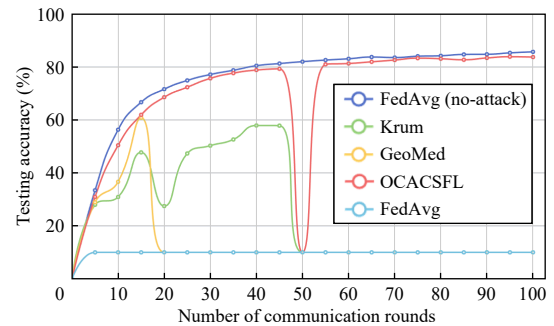


Fig. 13 Model accuracy on the IID CIFAR-10 dataset under a sign-flipping attack with 30% malicious clients.

round 45, when the number of malicious clients among the selected clients rises to eight (though this scenario is rare), the performance of both Krum and OCACSFL significantly decreases. At this point, due to the model rollback mechanism, OCACSFL can recover to a better state in the next round, whereas Krum is unable to compensate for the loss. The experiments also reveal that without the rollback mechanism, OCACSFL likewise cannot recover from substantial performance degradation. This further demonstrates the necessity of the rollback mechanism in OCACSFL.

We speculate that the reasons for the irrecoverable losses are as follows: When malicious clients constitute the majority, the loss of the aggregated model increases considerably, resulting in abnormally large model updates and causing substantial shifts in the global model. In the subsequent normal aggregation rounds, because the loss function values are relatively small, the model updates are also small, and the “pull-back” effect on the shifted model is quite limited. Consequently, the global model remains far from its ideal state in subsequent rounds and struggles to recover to optimal performance.

6.4.3 Effectiveness in detecting malicious clients

We evaluate the recall rate of OCACSFL for detecting malicious clients under sign-flipping attacks in two scenarios, where the proportions of malicious clients are 10% and 30%, respectively (as shown in Table 5). The reason for the failure to achieve a 100% recall rate is due to the high number of malicious clients and the insufficient number of training rounds, resulting in some of the malicious clients not participating in the aggregation during the 100 training rounds. As illustrated in Figs. 14 and 15, the recall rate for malicious clients gradually increases as the number of training rounds increases. In our experiments, when the number of training rounds is increased to 200, the recall rate for malicious clients reaches 100%. We

Table 5 Recall rates of malicious clients in OCACSFL.

(Unit: %)

Model	Attack	Proportion of malicious clients	
		10%	30%
MLP	Sign	100.0	100.0
	Noise	90.0	96.7
	Poison	100.0	100.0
CNN	Sign	90.0	96.7
	Noise	90.0	96.7
	Poison	100.0	96.7

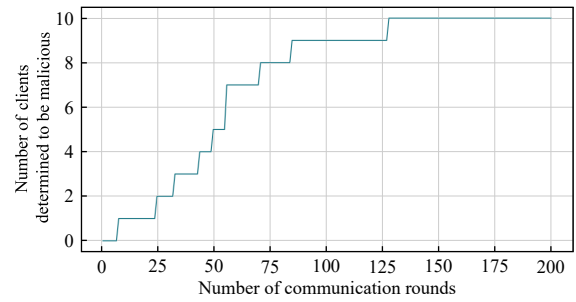


Fig. 14 Change in recall rate for 10% of malicious clients.

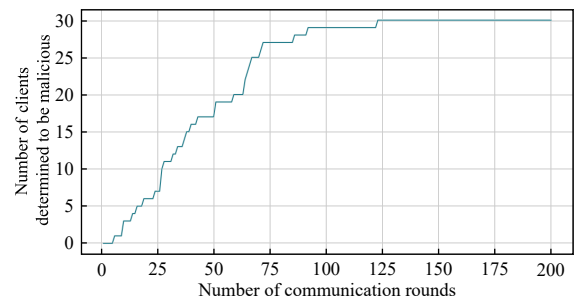


Fig. 15 Change in recall rate for 30% of malicious clients.

further evaluate the Adjusted Rand Index (ARI) of OCACSFL for clustering benign and malicious clients under sign-flipping attacks during the training process. The results indicate that the ARI consistently remains at 1 throughout training, demonstrating the effectiveness of OCACSFL in accurately distinguishing benign clients from malicious ones.

7 Conclusion and Future Work

This paper proposes OCACSFL, a method based on OPTICS density clustering for detecting malicious model updates. OCACSFL identifies and excludes malicious model updates from clients in federated learning, while permanently rejecting updates from clients identified as malicious across multiple rounds. By leveraging the benign distance differential, OCACSFL reduces unnecessary data wastage, and allows each round’s model to revert to a previous model with higher accuracy when the current round experiences performance degradation. Extensive experiments demonstrate that OCACSFL achieves high accuracy in model precision, validating its robust capability in countering model poisoning attacks.

Future work will focus on extending the application of OCACSFL to a broader range of machine learning domains, such as natural language processing and image segmentation. Additionally, we plan to integrate differential privacy techniques to enhance client

privacy protection and prevent dishonest servers from compromising sensitive client information.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 62372266, 62472251, 62202054, U24A20244, and 61771289) and the Young Elite Scientists Sponsorship Program by the China Association for Science and Technology (No. 2023QNRC001).

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in *Proc. 20th Int. Conf. Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.
- [2] H. Liu, N. Li, H. Kou, S. Meng, and Q. Li, FSRPCL: Privacy-preserve federated social relationship prediction with contrastive learning, *Tsinghua Science and Technology*, vol. 30, no. 4, pp. 1762–1781, 2025.
- [3] J. Chen, Z. Lin, W. Lin, W. Shi, X. Yin, and D. Wang, FedMUA: Exploring the vulnerabilities of federated learning to malicious unlearning attacks, *IEEE Trans. Inf. Foren. Sec.*, vol. 20, pp. 1665–1678, 2025.
- [4] Z. Xiong, W. Li, and Z. Cai. Federated generative model on multi-source heterogeneous data in IoT, in *Proc. 37th AAAI Conf. Artificial Intelligence*, Washington, DC, USA, 2023, pp. 10537–10545.
- [5] Y. Jiang, J. Shen, Z. Liu, C. W. Tan and K. Y. Lam, Toward efficient and certified recovery from poisoning attacks in federated learning, *IEEE Trans. Inf. Foren. Sec.*, vol. 20, pp. 2632–2647, 2025.
- [6] Z. He, L. Wang, and Z. Cai, Clustered federated learning with adaptive local differential privacy on heterogeneous IoT data, *IEEE Internet Things J.*, vol. 11, no. 1, pp. 137–146, 2024.
- [7] R. Li, H. Wang, Q. Lu, J. Yan, S. Ji, and Y. Ma, Research on medical image classification based on improved FedAvg algorithm, *Tsinghua Science and Technology*, vol. 30, no. 5, pp. 2243–2258, 2025.
- [8] M. Xhemrishi, J. Östman, A. Wachter-Zeh, and A. G. I. Amat, FedGT: Identification of malicious clients in federated learning with secure aggregation, *IEEE Trans. Inf. Foren. Sec.*, vol. 20, pp. 2577–2592, 2025.
- [9] W. Zhuang, C. Chen, J. Li, C. Chen, Y. Jin, and L. Lyu, When foundation model meets federated learning: Motivations, challenges, and future directions, arXiv preprint arXiv: 2306.15546, 2025.
- [10] X. Nie, L. T. Yang, F. Fan, and Z. Yang, Tensor-based efficient federated reinforcement learning for cyber-physical-social intelligence, *Big Data Mining and Analytics*, vol. 8, no. 4, pp. 867–879, 2025.
- [11] X. Chen, J. Pang, and T. Sun, Deep reinforcement learning based resource provisioning for federated edge learning, *High Confid. Comput.*, vol. 5, no. 2, p. 100264, 2025.
- [12] C. Li, B. Gu, Z. Zhao, Y. Qu, G. Xin, J. Huo, and L. Gao, Federated transfer learning for on-device LLMs efficient fine tuning optimization, *Big Data Mining and Analytics*, vol. 8, no. 2, pp. 430–446, 2025.
- [13] S. Zhu, R. Li, Z. Cai, D. Kim, D. Seo, and W. Li, Secure verifiable aggregation for blockchain-based federated averaging, *High Confid. Comput.*, vol. 2, no. 1, p. 100046, 2022.
- [14] F. Tavassolian, M. Abbasi, A. Ramezani, A. Taherkordi, and M. R. Khosravi, ResFed: An accurate and light federated multi-shot pre-trained model on edge devices, *Tsinghua Science and Technology*, vol. 30, no. 4, pp. 1539–1551, 2025.
- [15] Z. Xiong, Z. Cai, D. Takabi, and W. Li, Privacy threat and defense for federated learning with non-i.i.d. data in AIoT, *IEEE Trans. Industr. Inform.*, vol. 18, no. 2, pp. 1310–1321, 2022.
- [16] S. Li, E. C. H. Ngai, and T. Voigt, An experimental study of byzantine-robust aggregation schemes in federated learning, *IEEE Trans. Big Data*, vol. 10, no. 6, pp. 975–988, 2024.
- [17] X. Cao and N. Z. Gong, MPAF: Model poisoning attacks to federated learning based on fake clients, in *Proc. 2022 IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops*, New Orleans, LA, USA, 2022, pp. 3395–3403.
- [18] V. Shejwalkar and A. Houmansadr, Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning, in *Proc. 28th Annu. Network and Distributed System Security Symp.*, <https://dblp.org/db/conf/ndss/ndss2021.html#conf/ndss/SheljwalkarH21>, 2021.
- [19] M. Fang, X. Cao, J. Jia, and N. Z. Gong, Local model poisoning attacks to Byzantine-Robust federated learning, in *Proc. 29th USENIX Security Symp.*, Virtual Event, pp. 1605–1622, 2020.
- [20] Y. Huang, S. Gupta, Z. Song, K. Li, and S. Arora, Evaluating gradient inversion attacks and defenses in federated learning, in *Proc. 35th Conf. Neural Information Processing Systems*, Virtual Event, 2021, pp. 7232–7241.
- [21] Z. Gu and Y. Yang, Detecting malicious model updates from federated learning on conditional variational autoencoder, in *Proc. 2021 IEEE Int. Parallel and Distributed Processing Symp.*, Portland, OR, USA, 2021, pp. 671–680.
- [22] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, How to backdoor federated learning, in *Proc. 33th Int. Conf. Artificial Intelligence and Statistics*, Palermo, Italy, 2020, pp. 2938–2948.

- [23] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, in *Proc. 31st Int. Conf. Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 118–128.
- [24] Y. Chen, L. Su, and J. Xu, Distributed statistical machine learning in adversarial settings: Byzantine gradient descent, in *Proc. 2018 ACM Int. Conf. Measurement and Modeling of Computer Systems*, Irvine, CA, USA, 2018, p. 96.
- [25] X. Cao, M. Fang, J. Liu, and N. Z. Gong, FLTrust: Byzantine-robust federated learning via trust bootstrapping, in *Proc. 28th Annu. Network and Distributed System Security Symp.*, <https://dblp.org/db/conf/ndss/ndss2021.html#conf/ndss/CaoF0G21>, 2021.
- [26] J. Le, D. Zhang, X. Lei, L. Jiao, K. Zeng, and X. Liao, Privacy-preserving federated learning with malicious clients and honest-but-curious servers, *IEEE Trans. Inf. Foren. Sec.*, vol. 18, pp. 4329–4344, 2023.
- [27] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778.
- [28] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, Federated optimization in heterogeneous networks, in *Proc. 3rd Conf. on Machine Learning and Systems*, Austin, TX, USA, 2020, pp. 429–450.
- [29] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, Learning to detect malicious clients for robust federated learning, arXiv preprint arXiv: 2002.00211, 2020.
- [30] S. Han, S. Park, F. Wu, S. Kim, B. Zhu, X. Xie, and M. Cha, Towards attack-tolerant federated learning via critical parameter analysis, in *Proc. 2023 IEEE/CVF Int. Conf. Computer Vision*, Paris, France, 2023, pp. 4976–4985.
- [31] C. Xie, O. Koyejo, and I. Gupta, Generalized byzantine-tolerant SGD, arXiv preprint arXiv: 1802.10116, 2018.
- [32] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018, pp. 5650–5659.
- [33] K. Pillutla, S. M. Kakade, and Z. Harchaoui, Robust aggregation for federated learning, *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022.
- [34] T. Li, S. Hu, A. Beirami, and V. Smith, Ditto: Fair and robust federated learning through personalization, in *Proc. 38th Int. Conf. Machine Learning*, Virtual Event, 2021, pp. 6357–6368.
- [35] Z. Li, L. Liu, J. Zhang, and J. Liu, Byzantine-robust federated learning through spatial-temporal analysis of local model updates, in *Proc. 2021 IEEE 27th Int. Conf. Parallel and Distributed Systems*, Beijing, China, 2021, pp. 372–379.
- [36] F. Sattler, K. R. Müller, T. Wiegand, and W. Samek, On the Byzantine robustness of clustered federated learning, in *Proc. 2020 IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Barcelona, Spain, 2020, pp. 8861–8865.
- [37] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, Portland, OR, USA, 1996, pp. 226–231.
- [38] T. D. Nguyen, P. Rieger, H. Chen, H. Yalame, H. Möllering, H. Fereidooni, S. Marchal, M. Miettinen, A. Mirhoseini, S. Zeitouni, et al., FLAME: Taming backdoors in federated learning, in *Proc. 31st USENIX Security Symp.*, Boston, MA, USA, 2022, pp. 1415–1432.
- [39] D. Wang and S. Guan, FedFR-ADP: Adaptive differential privacy with feedback regulation for robust model performance in federated learning, *Inf. Fusion*, vol. 116, p. 102796, 2025.
- [40] M. Adnan, M. Haider Syed, A. Anjum, and S. Rehman, A framework for privacy-preserving in IoV using federated learning with differential privacy, *IEEE Access*, vol. 13, pp. 13507–13521, 2025.
- [41] X. Liu, J. Wang, X. Xiong, and H. Sun, Federated learning data protection scheme based on personalized differential privacy in psychological evaluation, *Neurocomputing*, vol. 611, p. 128653, 2025.
- [42] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, OPTICS: Ordering points to identify the clustering structure, *ACM SIGMOD Rec.*, vol. 28, no. 2, pp. 49–60, 1999.
- [43] J. Macqueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, L. M. Lecam and J. Neyman, eds. Berkeley, CA, USA: University of California Press, 1967, pp. 281–297.
- [44] V. Shejwalkar, A. Houmansadr, P. Kairouz, and D. Ramage, Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning, in *Proc. 2022 IEEE Symp. Security and Privacy*, San Francisco, CA, USA, 2022, pp. 1354–1371.



Chunmei Ma received the BEng degree in computer science and technology from Qufu Normal University, China, in 2001, and the MS degree from Nanjing University of Posts and Telecommunications, China, 2007. She is currently an associate professor at School of Computer Science, Qufu Normal University, China. Her current research interests include big data, Internet of Things (IoTs), and distributed algorithms.



Zhengyang Zhang received the BEng degree in computer science and technology from Qufu Normal University, China, in 2023, where he is currently a master student. His research interests include federated learning and privacy protection.



Baogui Huang received the PhD degree from Qufu Normal University, China, in 2020, where he is currently an associate professor at School of Computer Science. He was a visiting scholar at The George Washington University, Washington, DC, USA. His current research interests include wireless networks and distributed computing, artificial intelligence, and edge computing. He is a member of the China Computer Federation (CCF).



Chuanwen Luo received the PhD degree from Renmin University of China, China, in 2020. He is currently working as an associate professor at School of Information Science and Technology, Beijing Forestry University, China. He was a visiting scholar at Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA, in 2019. His research interests include various topics in the application of wireless networks, ad hoc and sensor networks, and algorithm design and analysis.



Guangshun Li received the BEng degree in computer science and technology from Qufu Normal University, China, in 2001, and the MEng and PhD degrees from Harbin Engineering University, China, in 2004 and 2008, China. He is currently a full professor at School of Computer Science, Qufu Normal University, China. He was a visiting scholar at The Hong Kong Polytechnic University, in the second half year of 2019. He has already published more than 50 articles. His research interests include wireless networks, IoTs, and big data.



Junhua Wu received the MEng and PhD degrees from Harbin Engineering University, China, in 2005 and 2009, respectively. She is currently a professor at School of Computer Science, Qufu Normal University, China. She has authored/co-authored more than 40 journal and conference papers. Her research interests include network security, blockchain, and artificial intelligence.