Information Fusion 126 (2026) 103529

Contents lists available at ScienceDirect

Information Fusion

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

Full length article

Mitigating malicious model fusion in federated learning via confidence-aware defense

Qilei Li [a,b,*], Pantelis Papageorgiou [b], Gaoyang Liu [c], Mingliang Gao [d], Linlin You [e], Chen Wang [c], Ahmed M. Abdelmoniem [b]

[a] Faculty of Artificial Intelligence in Education, Central China Normal University, Wuhan, 430079, China
[b] School of Electronic Engineering and Computer Science, Queen Mary University of London, London, E1 4NS, United Kingdom
[c] Huazhong University of Science and Technology, Wuhan, China
[d] Shandong University of Technology, Zibo, China
[e] Sun-Yat Sen University, Shenzhen, China

ARTICLE INFO

Keywords:
Federated learning
Security and privacy
Malicious attacks
Internet-of-Things
Model robustness

ABSTRACT

Federated Learning (FL) has emerged as a promising approach for decentralized machine learning in Internet of Things (IoT) applications, where privacy-sensitive data remains distributed across devices. However, FL systems are vulnerable to attacks that are happening in malicious clients via data poisoning and model poisoning. Once such malicious models are fused in the global server, it will deteriorate the global model's performance. Existing defense methods typically mitigate specific types of poisoning but are often ineffective against others. To overcome this issue, we propose a simple yet effective framework called Confidence-Aware Defense (CAD). It aims to achieve accurate, robust, and versatile detection of malicious attacks. CAD evaluates the reliability of client updates by leveraging the confidence scores produced by each FL client model. Our key insight is that poisoning attacks, regardless of attack type, will cause the model to deviate from its previous state, thus leading to increased uncertainty when making predictions. Therefore, CAD is comprehensively effective for various types of poisoning attacks, including model poisoning and data poisoning. The proposed CAD method accurately identifies and mitigates malicious updates, even under varying attack intensities and data heterogeneity. CAD is evaluated on standard FL benchmarks (CIFAR-10, MNIST, Fashion-MNIST) under non-IID settings and both model and data poisoning attacks. It achieves up to 97.8% accuracy on MNIST and sustains over 64% accuracy under 50% poisoning on CIFAR-10. CAD surpasses all prior defense methods in robustness and performance. These results demonstrate the practicality of CAD in securing FL systems against various threat scenarios.

1. Introduction

Federated Learning (FL) [1] is a distributed machine learning paradigm that enables multiple clients, such as edge devices, to collaboratively train a model without exchanging their local private data. This paradigm aligns seamlessly with the Internet of Things (IoT) ecosystem, where a vast number of interconnected devices generate massive volumes of decentralized, privacy-sensitive data. A typical FL system for IoT applications is shown in Fig. 1. IoT devices, ranging from wearable sensors to smart home appliances, often collect user-specific information that cannot be transmitted to centralized servers due to privacy concerns, bandwidth limitations, or regulatory compliance requirements (*e.g.*, GDPR). FL addresses these challenges by enabling local data processing on IoT devices while only sharing model updates, making it a suitable solution for large-scale, privacy-preserving IoT applications [2,3]. Federated learning can be categorized based on data partitioning characteristics into horizontal federated learning (HFL), vertical federated learning (VFL), and federated transfer learning (FTL) [4,5]. HFL deals with scenarios where clients share the same feature space but differ in sample space, such as image classification tasks across multiple institutions. VFL assumes that clients hold data with overlapping samples but different feature spaces, often seen in collaborative financial or e-commerce applications. FTL handles cases

* Correspondence to: Completed the work while at Queen Mary University of London, UK.
E-mail addresses: qilei.li@ccnu.edu.cn (Q. Li), ahmed.sayed@qmul.ac.uk (A.M. Abdelmoniem).

https://doi.org/10.1016/j.inffus.2025.103529
Received 15 March 2025; Received in revised form 2 July 2025; Accepted 11 July 2025
Available online 23 July 2025

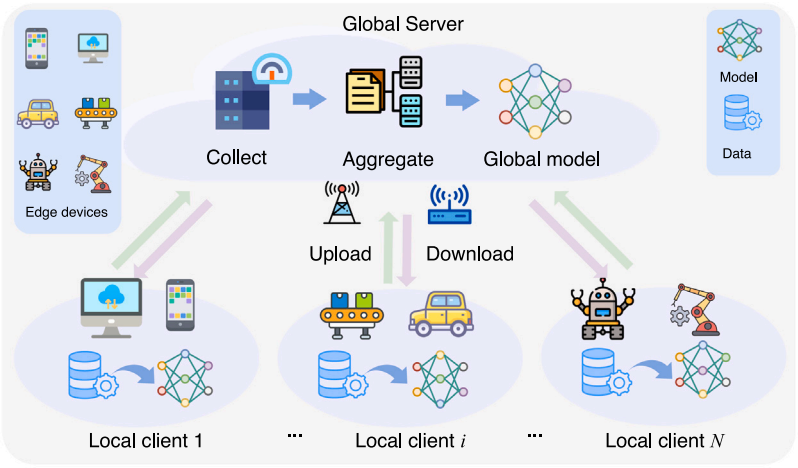

Fig. 1. Illustration of a federated learning framework for IoT with edge devices. The system involves multiple local clients, such as IoT devices and edge devices, who are collaboratively training a global model. Local data is processed, and models are trained locally on devices, with *local model updates* uploaded to a central global server. The server aggregates the updates into a unified global model, which is then distributed to local clients for further training.

where both sample and feature spaces vary significantly across clients, using transfer learning techniques to enable collaboration.

However, the decentralized nature of FL introduces significant security challenges, particularly in the context of IoT systems. Malicious attacks originating from distributed local clients can severely compromise the globally trained model's performance through poisoning strategies such as data poisoning and model poisoning [6–8]. In data poisoning attacks, adversaries manipulate clients' local training data by injecting biases or false patterns to mislead the global model during training; *e.g.*, label shuffling, where data-label pairs are randomly reassigned, can cause substantial confusion in the updated global model [9,10]. On the other hand, model poisoning attacks involve directly tampering with the model updates sent to the central server, with Byzantine attacks being a representative example where malicious clients submit arbitrary or random updates to disrupt the learning process and degrade model performance [11,12]. Both attack types not only degrade predictions and hinder the convergence of deep neural networks but can also cause a complete model collapse in extreme cases. The consequences are particularly severe in IoT applications, where FL is often employed in safety-critical systems such as healthcare monitoring, autonomous vehicles, and industrial control. An undetected malicious attack in these scenarios could impair the system's functionality and pose risks to human safety. Therefore, developing robust and efficient methods to detect and defend against such attacks is of great importance for ensuring the reliability of FL systems in IoT environments.

Existing defense mechanisms primarily target specific types of poisoning attacks, under a moderate attack. Byzantine-robust methods such as Krum [13] and Trimmed Mean [14] aim to mitigate the impact of outliers among client updates but often fall short when dealing with data poisoning attacks. However, these methods generally assume that the majority of clients are honest, which may not hold true in practical scenarios where the proportion of malicious clients is significant [11]. Recent advanced methods are primarily driven by the analysis of fine-grained model consistency. In that context, FLDetector [15] improves upon existing defenses by detecting malicious clients through the consistency of their model updates across historical iterations, and flags clients whose updates deviate significantly. DeFL [16] focuses on critical learning periods within the training process to determine the gradient changes and remove malicious clients. FedRoLA [17] enhances robustness by analyzing the per-layer consistency of model updates and conducts aggregation based on the alignment of updates at each layer. However, these methods are simply designed to address specific types of attacks under a certain degree of attack Intensities, and exhibit limited robustness and versatility against a wide range of attack types and intensities

Given the inherent uncertainty of FL systems and the attacks' complexity, being capable of addressing only certain types of attack is far from sufficient. A robust and comprehensive defense approach is needed to ensure the integrity and reliability of the global model across all scenarios, including various types of attacks with differing levels of severity and degrees of data heterogeneity. To meet this demand, we propose a simple yet effective defense model, termed Confidence-Aware Defense (CAD), based on the confidence scores of clients' local models, which measures how certain a model is about its predictions, with higher scores indicating greater certainty and lower scores reflect higher uncertainty. Our key insight is that malicious attacks, regardless of attack type, will cause the model to deviate from its previous state, thus leading to lower confidence in its predictions. Inspired by this observation, we propose to use the confidence scores as a criterion to evaluate the reliability of client updates. Based on this, we develop CAD as an effective defense mechanism for both model poisoning and data poisoning attacks. CAD's design helps to identify and mitigate malicious updates regardless of the attack type and attack intensity. Specifically, the proposed approach includes the following steps: 1. Collecting confidence scores of each client update: During each training round, we collect the confidence scores of model updates from each client. 2. Establishing confidence boundaries: Based on the collected confidence scores, we assess the confidence of each client model update and set the boundaries as a reference. 3. Detecting and handling malicious updates: Based on the boundaries, we identify updates with lower confidence scores and take appropriate actions, such as discarding or dampening these updates.

Our key contributions are summarized as follows:

- We identify the intrinsic correlation between model confidence and poisoning attacks, which serves as the foundation for a straightforward yet effective confidence-based method to detect malicious clients and determine whether an attack has occurred on a local client. Our approach is versatile and applicable to both model poisoning and data poisoning attacks, under various data heterogeneities.
- By detecting malicious attacks on client model updates through confidence scores, the proposed method demonstrates effectiveness against certain poisoning attacks of varying intensities. Whether facing moderate (25% malicious clients), severe (50% malicious clients), or extreme (75% malicious clients) activities, our approach ensures higher model accuracy and stability.

- We validate the proposed method through extensive experiments on multiple datasets and demonstrate its effectiveness in enhancing the security and performance of FL models against a wide range of attacks. The results show that our confidence-based defense mechanism outperforms existing methods in terms of robustness and accuracy in detecting and mitigating malicious updates.

The rest of the paper is organized as follows: Section 2 reviews related work, including FL in IoT, malicious client attacks, and defense mechanisms. Section 3 introduces the proposed Confidence-Aware Defense (CAD) and its core components. Section 4 presents experimental results demonstrating CAD's robustness and effectiveness. Finally, Section 5 concludes the paper with a summary and future directions.

2. Background and related work

2.1. Federated learning (FL) in IoT

The objective of Federated Learning (FL) [18] is to facilitate collaborative training of a shared global model across multiple learning entities while keeping locally generated data private and secure [19–21]. It is particularly well-suited for IoT applications, which devices often collect vast amounts of sensitive data, such as health metrics, user behavior, or environmental information. These data cannot be shared directly due to privacy concerns or communication constraints [22,23]. FL addresses these challenges by dividing the learning process into two key stages: local training and global aggregation. Each IoT device trains a local model using its private dataset during the local training phase. This phase involves updating the model parameters based solely on the local data [24]. After completing the local training, each device sends the updated model parameters to a central server or edge aggregator for global aggregation. The global aggregation stage combines the received model updates from participating devices to form a new global model. The most widely used aggregation approach is Federated Averaging (FedAvg) [24], which computes a weighted average of model parameters from participating devices. Despite its advantages in preserving data privacy and enabling decentralized learning, FL in IoT systems faces significant security challenges [18]. The decentralized nature of FL makes it vulnerable to attacks, including data poisoning, where adversaries manipulate local datasets [6], and model poisoning, where malicious devices send crafted model updates to compromise the global model [21]. These challenges are exacerbated in IoT scenarios, where devices are resource-constrained and operate in highly heterogeneous environments [25]. Given the critical role of FL in privacy-sensitive IoT scenarios, developing efficient and robust defense mechanisms to detect and mitigate the impact of malicious activities is imperative [26,27].

2.2. Malicious poisoning attacks

FL models are susceptible to malicious attacks, which can significantly degrade the performance of the global model, as illustrate in Fig. 2. The malicious attacks can be broadly categorized into data poisoning and model poisoning.

Data Poisoning Attacks: In data poisoning attacks, malicious clients intentionally manipulate their local training data to degrade the performance of the global model. Common strategies include label flipping, where the labels of training samples are deliberately swapped, and label shuffling, where the labels of training samples are randomly reassigned. These attacks aim to bias the model's training process, and further lead to inaccurate or harmful predictions. Rahman et al. [28] highlights how local model poisoning through data manipulation can significantly impact the robustness of federated learning systems. Shen et al. [29] propose a data poisoning attack that specifically address the label flipping poisoning attack. Similarly, Smith et al. [30] examine clean-label poisoning attacks, where the manipulated data appears legitimate but is designed to subvert the performance of the model.

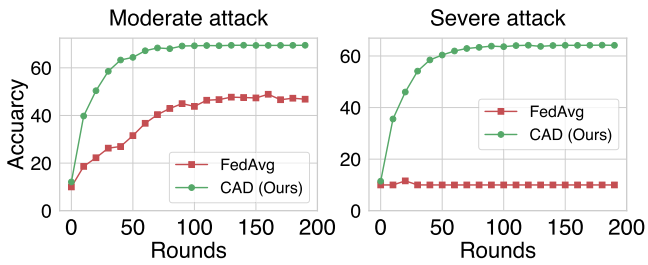

Fig. 2. Impact of malicious attacks on model convergence under the "Little Is Enough" attack at various intensities. FedAvg aggregation degrades significantly, while our CAD consistently maintains high accuracy and robustness.

Model Poisoning Attacks: In contrast to data poisoning, model poisoning refers to malicious clients directly sending manipulated model updates to the central server during the global aggregation phase. These attacks are conducted by altering the model parameters in a way that maximizes the negative impact on the global model. Zhang et al. [31] have demonstrated how backdoor attacks could be performed by injecting specific patterns into the model updates, enabling attackers to trigger incorrect behaviors under certain conditions. Liu et al. [32] propose FLOW framework to detect and mitigate model poisoning attacks in FL systems. Lee et al. [33] analyze the adversarial impact of model attacks on federated learning models. Additionally, Byzantine attacks, which involve arbitrary or adversarial behavior by clients, can be classified under model poisoning when clients send random or adversarial updates to the central server. Recent Byzantine-robust methods like Krum++ [34] and Enhanced Trimmed Mean [35] aim to filter out these abnormal updates but can struggle when the attacks are sophisticated.

2.3. Adversarial robustness in FL

Adversarial robustness has become a critical area of research in machine learning, focusing on developing models and algorithms that can withstand adversarial manipulations and malicious attacks. In the context of federated learning, adversarial robustness is particularly important due to the decentralized nature of FL and the increased risk of adversarial clients or poisoned updates. Techniques such as adversarial training [36], robust aggregation methods [13,14], and certified defenses [37] have been proposed to enhance the resilience of machine learning models against a wide range of attacks. Recent works have also explored the adaptation of these techniques to federated settings, aiming to defend against both data and model poisoning attacks [12,27,38]. Our proposed method is inspired by these advances and seeks to further improve the robustness of FL systems by leveraging confidence-aware detection and aggregation strategies.

2.4. Common defenses against malicious attacks

Recently, numerous works have been proposed to detect and mitigate the aforementioned attacks [6,15–17]. DeFL [16] is proposed as a defense mechanism that secures critical learning periods during the training process to reduce the impact of model poisoning attacks. Zhang et al. [15] introduced FLDetector to integrate statistical analysis, anomaly detection, and machine learning methods to enhance the accuracy of detecting poisoned updates. FedRoLA [17] strengthens robustness by analyzing per-layer consistency in model updates and conducting aggregation based on the alignment of updates at each layer. Despite the improved accuracy in detecting malicious clients, these existing methods are often tailored to specific types of attacks.

Table 1
Comparison of state-of-the-art FL defenses and our proposed CAD.

Method	Detection criteria	Assume majority honest	Aggregation	Robust under high attack rate
FLDetector [15]	Historical gradient consistency	Yes	Standard/FedAvg	No
DeFL [16]	Gradient dynamics during critical learning periods	Yes	Standard/FedAvg	Limited
FedRoLA [17]	Per-layer alignment of model updates	Yes	Layer-wise Aggregation	Limited
CAD (Ours)	Confidence score clustering	No	Confidence-weighted	Yes

Additionally, they typically rely on the fragile assumption that the attack intensity is moderate, meaning less than 25% of the participating clients are malicious, which may not hold in a practical FL system where higher proportions of malicious clients can be present. To resolve these limitations, in this paper, we propose the CAD model to address both types of attacks and varying intensities. The comparison of state-of-the-art FL defenses and our proposed CAD is shown in Table 1.

2.5. Model confidence

Model confidence [39] refers to a model's certainty regarding its predictions. Higher confidence scores indicate greater confidence, while lower scores suggest higher uncertainty. It provides insights into the model's decision-making process by identifying the predictions that the model is less sure about [40,41]. Recent studies have also explored uncertainty estimation in broader federated and ensemble-based decision systems. For instance, confidence and uncertainty modeling have been applied to federated medical decision-making for COVID-19 treatment allocation [42], intelligent evaluation in IoT-based agriculture systems [43], and ensemble-based multi-level meta-decision frameworks [44]. While these works focus on different application domains, they highlight the importance of uncertainty handling in distributed and federated environments.

Super Loss is a confidence-aware loss designed for curriculum learning, which incorporates confidence scores to determine the reliability of each training instance [45]. It introduces a regularization to adjust the loss dynamically based on the confidence score (σ^j) for sample j in the training set D. The Super Loss is defined as:

$$\mathcal{L}_{\text{super}} = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{1}{\sigma^{j}{}^{2}}\mathcal{L}_j + \log\sigma^j\right), \tag{1}$$

where $\mathcal{L}_i$ is the task loss for sample i, and σ^j is the confidence score, and N is the number of samples. The confidence score σ^j is obtained by the following closed-form solution:

$$\sigma^j = \exp\left(-\mathrm{W}\left(\frac{1}{2}\max\left(-\frac{2}{e}, \frac{\mathcal{L}_i - \log(\tau)}{\lambda}\right)\right)\right), \tag{2}$$

where $\log(\tau)$ is the constant, and τ is set to the number of classes, and λ is the regularization parameter. In this work, motivated by the observation that malicious attack will disrupt the local model's convergence against the original global initialization, and making the learned model less confident in the output, we propose to use per-client confidence accumulation to determine the quality of the learned model. Note that we use Eq. (2) solely for estimating the confidence score, in contrast to the original design of Super Loss, which uses it as a regularization term. The rationale is to make the CAD model versatile and applicable to various tasks, and to explicitly demonstrate that the effectiveness stems from the detection algorithm rather than modifications to the loss function. We adopt the Super Loss for uncertainty estimation as it derives confidence directly from per-sample loss, which serves as a model-agnostic and calibration-insensitive measure without relying on logit-based outputs.

Table 2
Mathematical notations.

Notation	Description
t	Number of training round
n	Number of clients
$\mathcal{D}_i$	Local dataset of client i
θ	Global model parameters
θ_i	Local model parameters of client i
$\mathcal{L}_i^j$	Individual loss for sample j in $\mathcal{D}_i$
σ_i	Confidence score for client i
S	Set of confidence scores from all clients
$\mu_{\text{lower}}, \mu_{\text{upper}}$	Centroids from k-means clustering
$\mathcal{H}$	Set of honest clients
$\mathcal{M}$	Set of malicious clients
$\mathcal{L}_{\mathcal{H}}$	Data lengths of honest clients
W_{orig}	Original weights based on data lengths
R	Re-weight regularization factors
$r_{\text{norm},i}$	Normalized regularization factor
$w_{\text{final},i}$	Final weight for client i
W	Lambert W function

3. Methodology

3.1. Problem formulation

We consider an FL system with N clients, each with its local dataset $\mathcal{D}_i$. The goal is to train a global model θ by aggregating the local updates θ_i from client $\{C_i\}_{i=1}^N$ while mitigating the impact caused by potential malicious clients, which can perform two types of attacks, including ***Data poisoning attacks***: These include label manipulation where the local dataset is intentionally corrupted to mislead the global model. ***Model poisoning attacks***: Here, the model updates sent by the clients are manipulated to degrade the performance of the global model. To achieve this goal, we propose a Confidence-Aware Defense (CAD) mechanism based on the client-level confidence scores ($\{\sigma_i\}_{i=1}^N$), which reflects the uncertainty of the local model's predictions and helps in identifying malicious updates. *The detection operation is performed on the global server to select the honest local updates in each local training round $t \in \{1, 2, \ldots, T\}$, where T is the total number of training rounds.* As shown in Fig. 3, the framework consists of five steps: (1) initializing and distributing the global model; (2) training local models and calculating confidence score; (3) uploading models and confidence scores, (4) detecting malicious clients via clustering, and (5) aggregating honest updates using re-weighted aggregation. The notations used in formulating CAD are summarized in Table 2.

3.2. Confidence-driven detection

3.2.1. Client confidence estimation

For each client, the confidence score is estimated via their local dataset $\mathcal{D}_i$. Referring to Eq. (2), for each local client, we calculate the accumulated confidence score for all samples in the local dataset and then compute an average to obtain the per-client confidence score σ_i^t. For each client C_i, in each local training round $t \in \{1, 2, \ldots, T\}$, the accumulated confidence score σ_i^t is defined as the average of its confidence scores computed on its dataset's samples $\sigma_i^t = \frac{1}{N_i}\sum_{j=1}^{N_i}\sigma_i^{t,j}$, where N_i is the number of samples in $\mathcal{D}_i$. This σ_i^t reflects the overall

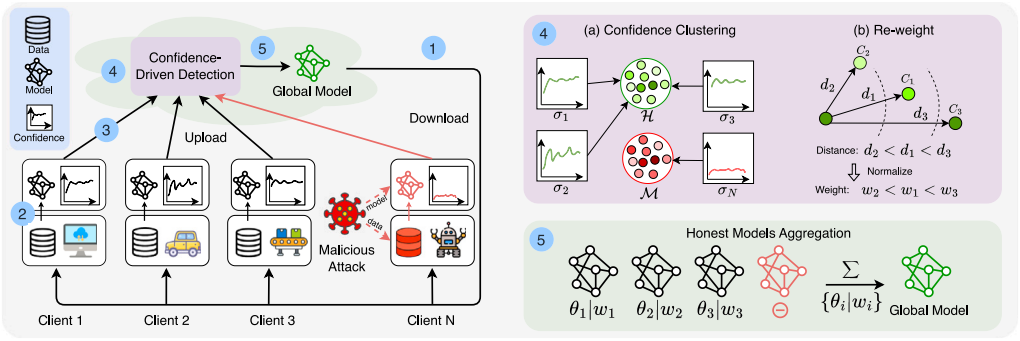

Fig. 3. The proposed Confidence-Aware Defense (CAD) framework. It encompasses five key steps: 1. Initialize the global model and distribute it to clients. 2. Clients train their personalized, confidence-aware local models. 3. Clients upload their local models and associated confidence scores to the server. 4. The server identifies potentially malicious clients through confidence clustering. 5. The server aggregates the models of honest clients by re-weighted aggregation.

uncertainty of the local model's predictions for client i in round t. Once the confidence scores for all clients have been computed, they will be normalized by min–max scaling, which transforms the confidence scores into a uniform range [0, 1]. Let $S^t = \{\sigma_i^t\}_{i=1}^n$ represent the set of confidence scores from n clients. The objective of normalization is to ensure that confidence scores from different clients are directly comparable by aligning them to a common scale. This prevents inconsistencies in score magnitude from affecting the clustering process and improves detection accuracy. The normalized confidence scores S^t_{norm} are formulated as:

$$S^t_{\text{norm}} = \frac{S^t - \min(S^t)}{\max(S^t) - \min(S^t)}. \tag{3}$$

3.2.2. Clustering based detection

After normalizing the per-client confidence scores, the next step is to separate the clients into two groups, namely honest and malicious clients. To achieve this, we employ a clustering strategy that groups clients based on their confidence scores. Specifically we apply two-way k-means clustering upon the normalized confidence scores S^t_{norm}, so as to obtain two centroids that establish the lower bound and upper bound:

$$\{\mu^t_{\text{lower}}, \mu^t_{\text{upper}}\} = k\text{-means}(S^t_{\text{norm}}, 2). \tag{4}$$

Once the centroids are obtained, the following step is to classify the clients as honest or malicious based on the distance in terms of the normalized confidence score to the cluster centroids. Specifically, a client C_i is deemed honest if its normalized confidence score σ_i^t is closer to the centroid μ^t_{upper} (representing higher confidence) than to the centroid μ^t_{lower} (representing lower confidence). Conversely, a client is classified as malicious if the opposite holds true:

$$\begin{aligned} \mathcal{H}^t &= \{i \mid |\sigma_i^t - \mu^t_{\text{upper}}| < |\sigma_i^t - \mu^t_{\text{lower}}|\}, \\ \mathcal{M}^t &= \{i \mid |\sigma_i^t - \mu^t_{\text{lower}}| \leq |\sigma_i^t - \mu^t_{\text{upper}}|\}. \end{aligned} \tag{5}$$

By classifying clients, we can focus on aggregating updates from only honest clients and get rid of malicious ones, thereby enhancing the quality and robustness of FL's trained models.

3.3. Re-weighted aggregation

Based on the identified set of malicious clients, the next step is to perform weight aggregation. Let $\Theta^t_{\mathcal{H}}$ denote the set of honest local model updates, and $\mathcal{L}^t_{\mathcal{H}}$ represent the corresponding set of number of samples for these honest clients. These are defined as:

$$\Theta^t_{\mathcal{H}} = \{\theta_i^t \mid i \in \mathcal{H}^t\}, \quad \mathcal{L}^t_{\mathcal{H}} = \{\ell_i \mid i \in \mathcal{H}^t\}, \tag{6}$$

by which we isolate the local updates of clients that are considered trustworthy, so as to ensure that only reliable information will be used in the aggregation process. Subsequently, we define the original weights $w^t_{\text{orig},i}$ with respect to the number of available samples for these honest clients. This weight is calculated as the ratio of the number of client's samples over the amount of total training of all honest clients:

$$\{w^t_{\text{orig},i}\}_{i\in\mathcal{H}^t} = \{\frac{\ell_i}{\sum_{j\in\mathcal{H}^t}\ell_j}\}_{i\in\mathcal{H}^t}. \tag{7}$$

In addition, to *adaptively* optimize the weight based on the trustworthiness (confidence) of each honest client, we apply a re-weighting strategy, for which a re-weighting regularization factor r_i^t is then calculated for each honest client based on their normalized confidence scores. The set of re-weighting regularization factors for honest clients is denoted as $R^t = \{\sigma_i^t \mid i \in \mathcal{H}^t\}$, which are then normalized $r^t_{\text{norm},i} = \frac{\sigma_i^t}{\sum_{j\in\mathcal{H}^t}\sigma_j^t}$, so that one can ensure that the aggregation process give appropriate importance to individual client models. Once the original weights and the re-weighting regularization factors are obtained, we combine both to obtain the final weights as $w^t_{\text{final},i} = r^t_{\text{norm},i} \cdot w^t_{\text{orig},i}$. As a final step, the global model parameters are then aggregated by the re-weighted average, which ensures the contributions from different clients are proportional to their combined confidence scores and original weights. Let $\Theta^t_{\mathcal{H}}$ be the set of models from the honest clients, then the updated global model θ^t_{global} is obtained as:

$$\theta^t_{\text{global}} = \frac{\sum_{i\in\mathcal{H}^t} w^t_{\text{final},i} \cdot \theta_i^t \cdot \ell_i}{\sum_{i\in\mathcal{H}^t} w^t_{\text{final},i} \cdot \ell_i}. \tag{8}$$

We summarize the process of CAD in Algorithm 1. By interactively conducting this detection and selected aggregation, our proposed CAD model effectively identifies and aggregates updates from honest clients, while ensuring that the aggregation process gives more weight to clients with higher confidence scores, thus improving the overall performance and security of the FL model.

3.4. Algorithm complexity analysis

3.4.1. Computational complexity

The computational complexity of the proposed method consists of three main steps: confidence estimation, clustering-based detection, and re-weighted aggregation. For confidence estimation, the total complexity for all clients is:

$$O(N_{\text{total}} \cdot F), \quad N_{\text{total}} = \sum_{i=1}^{n} N_i. \tag{9}$$

Clustering-based detection uses two-way k-means, with a complexity of $O(n \cdot t)$, where n is the number of clients and t is the number of iterations. Re-weighted aggregation for n_h honest clients, each with d model parameters, costs $O(n_h \cdot d)$. Thus, the total computational complexity is:

$$O(N_{\text{total}} \cdot F) + O(n \cdot t) + O(n_h \cdot d). \tag{10}$$

Algorithm 1: Proposed CAD Framework

Input: Initialized global model $\theta^{t=0}_{\text{global}}$, local datasets $\{\mathcal{D}_i\}_{i=1}^{N}$, number of clients N, number of rounds T.
Output: Updated global model $\theta^{T}_{\text{global}}$.

for *each round* $t = 1, \ldots, T$ **do**
  Step 1: Local Training - Client
  for *each client* C_i, $i = 1, \ldots, N$ **do**
    Train model θ_i^t on dataset $\mathcal{D}_i$ based on $\theta^{(t-1)}_{\text{global}}$.
    Compute confidence score $\sigma_i^t = \frac{1}{N_i} \sum_{j=1}^{N_i} \sigma_i^{t,j}$.
  end
  Step 2: Confidence Normalization - Server
  Normalize scores $S^t = \{\sigma_i^t\}$ by min-max scaling:

$$S^t_{\text{norm}} = \frac{S^t - \min(S^t)}{\max(S^t) - \min(S^t)}.$$

  Step 3: Malicious Client Detection - Server
  Cluster by two-way k-means:

$$\{\mu^t_{\text{lower}}, \mu^t_{\text{upper}}\} = k\text{-means}(S^t_{\text{norm}}, 2).$$

  Classify clients into honest $\mathcal{H}$ or malicious $\mathcal{M}$:

$$\mathcal{H}^t = \{i \mid |\sigma_i^t - \mu^t_{\text{upper}}| < |\sigma_i^t - \mu^t_{\text{lower}}|\}.$$

  Step 4: Re-weighted Aggregation - Server
  Compute final weights:

$$w^t_{\text{final},i} = r^t_{\text{norm},i} \cdot w^t_{\text{orig},i}.$$

  Aggregate global model:

$$\theta^t_{\text{global}} = \frac{\sum_{i \in \mathcal{H}^t} w^t_{\text{final},i} \cdot \theta_i^t \cdot \ell_i}{\sum_{i \in \mathcal{H}} w^t_{\text{final},i} \cdot \ell_i}.$$

  Step 5: Model Distribution - Server
  Distribute the global model θ^t_{global} to clients.
end
return θ^T_{global}.

Overall, the proposed method introduces minimal additional computational overhead compared to traditional approaches, ensuring efficiency and scalability for large-scale systems.

3.4.2. Communication overhead

The proposed method significantly reduces communication costs by excluding malicious client updates compared with conventional federated learning methods. In conventional federated learning, all n clients upload their model updates to the server, resulting in a communication cost of $O(n \cdot d)$, where d is the number of model parameters. In contrast, the proposed framework identifies and excludes malicious clients, reducing the cost of model uploads to:

$$O((1-\eta) \cdot n \cdot d), \tag{11}$$

where η is the proportion of malicious clients. Each client also transmits a scalar confidence score, which incurs a negligible additional cost of $O(n)$. After aggregating the updates, the server broadcasts the global model to all clients, costing $O(n \cdot d)$. The total communication overhead is $O((1-\eta) \cdot n \cdot d) + O(n) + O(n \cdot d)$, which simplifies to $O((2-\eta) \cdot n \cdot d)$. This reduction is particularly beneficial in IoT systems, where bandwidth is often limited, and many clients may be malicious. For example, with $\eta = 0.5$ (50% malicious clients), the communication cost is halved compared to traditional methods.

4. Experiments and evaluations

Our evaluation aims to address the following questions:

1. Can prioritizing updates based on confidence scores enhance model robustness against various attack intensities?
2. Does the re-weighted aggregation mechanism improve model performance compared to baseline defenses, especially under severe attacks?
3. How does CAD compare to state-of-the-art defenses regarding malicious client detection, model accuracy, stability in diverse datasets, models, and data heterogeneity?

4.1. Experimental setup

In the following, we detail the experimental setup, including datasets, DNN models, attacks, and defenses.

4.1.1. Datasets and evaluation metric

We evaluate CAD on a wide range of FL benchmarks, including: CIFAR-10 [46], MNIST [47], and Fashion-MNIST [48], commonly used in prior studies [16,17]. We employed a heterogeneous partitioning strategy to create non-IID local data where samples and class distributions are unbalanced among clients. The factor is sampled from a Dirichlet distribution $\texttt{Dir}_N(\alpha)$ [49], with α determining the level of heterogeneity, We illustrate the influence of α on data heterogeneity in Fig. 4. We observe that a smaller α value refers to high heterogeneity. We set $\alpha = 0.5$ to reflect a severe attack environment following the conventional setting [15]. We evaluated the performance of the model via classification accuracy. Although our experiments focus on image classification datasets, the CAD framework is model- and task-agnostic. It can be applied to any federated learning setting where models produce confidence scores or predictive probabilities

4.1.2. DNN models

We tested CAD model on three representative DNN models, including: AlexNet [50], VGG-11 [51], and a Multi-Layer Perceptron (MLP) [52] with layers consisting of 784, 512, and 10 neurons respectively. Specifically, AlexNet and VGG-11 were employed for the CIFAR-10 dataset, the MLP for MNIST, and AlexNet for Fashion-MNIST.

4.1.3. Baseline defenses

We compared CAD with state-of-the-art defense models including: TrimMean [14], AFA [53], FLDetector [15], DeFL [16], FedRoLA [17]. Additionally, we included FedAvg [24] as a baseline method for comparison. We consider two settings regarding the adversary's knowledge [16]: (a) ***Full***, where the adversary knows the gradients of honest clients, and (b) ***Partial***, where the adversary is unaware of the gradient updates shared by honest clients. We evaluated the malicious client detection performance by measuring the True Positive Rate (TPR) and the False Positive Rate (FPR).

4.1.4. Malicious client attacks

We evaluate the CAD model under two common malicious client attacks: model poisoning and data poisoning. Below, we detail the mathematical formulations of attack strategies.

- **Label Shuffling (LS)** [9] is a type of *data poisoning attack* where malicious clients randomly reassign labels in their local dataset. This introduces noise into the training process and misleads the global model. For a malicious client C_m, the poisoned dataset is generated as:

$$\mathcal{D}_m = \{(x_i, \hat{y}_i) \mid (x_i, y_i) \in \mathcal{D}_m, \hat{y}_i = \text{shuffle}(y_i)\},$$

  where $\text{shuffle}(y_i)$ represents a random reassignment of the true label y_i. This results in incorrect patterns the global model learns, degrading its performance.

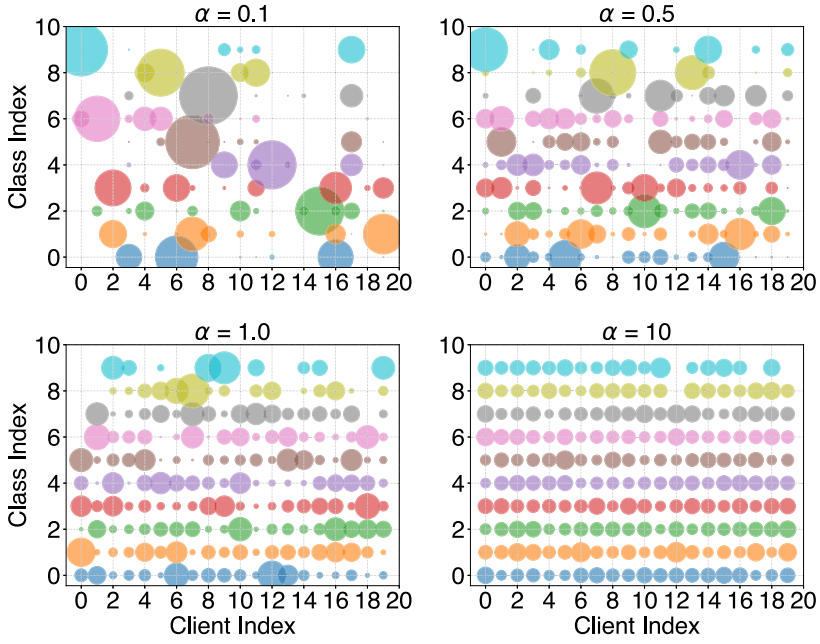

Fig. 4. Client-class distributions for different α values in a federated learning setting for different data heterogenesis. Larger points indicate higher sample sizes, with the x-axis showing clients and the y-axis showing classes.

- **Little is Enough (LIE)** [54] is a *model poisoning attack* where malicious clients add noise to the aggregate gradients of honest clients. Let $\{\nabla\theta_h^i\}_{i=1}^k$ represent the gradients from k honest clients. The adversary computes the mean and standard deviation of these gradients:

$$\mu = \frac{1}{k}\sum_{i=1}^{k}\nabla\theta_h^i, \quad \sigma = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(\nabla\theta_h^i - \mu)^2}.$$

  The malicious gradient is then crafted as $\nabla\theta_m = \mu + z\cdot\sigma$, where z is a noise coefficient chosen to evade detection.
- **Min–Max (MM)** [55] is a *model poisoning attack* where malicious gradients are equidistant from the cluster of honest gradients, maximizing disruption while remaining undetected. Let $d(\cdot,\cdot)$ denote the Euclidean distance. The malicious gradient $\nabla\theta_m$ satisfies:

$$\max_{h\in\{1,\dots,k\}} d(\nabla\theta_m, \nabla\theta_h^h) \le \max_{h_1,h_2\in\{1,\dots,k\}} d(\nabla\theta_h^{h_1}, \nabla\theta_h^{h_2}).$$

  This ensures the malicious gradient stays within the range of honest gradients but disrupts model convergence.
- **Min-Sum (MS)** [55] is another *model poisoning attack*, where malicious gradients minimize the sum of squared distances to all honest gradients. The adversary computes the malicious gradient $\nabla\theta_m$ by solving:

$$\sum_{h=1}^{k} d(\nabla\theta_m, \nabla\theta_h^h)^2 \le \sum_{h_1,h_2\in\{1,\dots,k\}} d(\nabla\theta_h^{h_1}, \nabla\theta_h^{h_2})^2.$$

  To maximize the attack, malicious clients use identical $\nabla\theta_m$.

4.1.5. Implementation details

We simulated $N = 50$ clients in the experiments. Each client was trained to update its local model for 20 local epochs using the stochastic gradient descent (SGD) optimizer. The number of FL training rounds was set to 200 to ensure convergence for all DNN models. The learning rate was set to 0.01. The batch size was set to 16, with a weight decay of 1×10^{-3}. We evaluated the robustness of our CAD model under three degrees of attack severity defined by the percentage of malicious clients: *moderate* (25%), *severe* (50%), and *extreme* (75%). For data poisoning attacks, confidence scores were directly obtained during model training. For model poisoning attacks, confidence scores were obtained by further running inference on local training data before sending it to the parameter server. The implementation is done in PyTorch [56] and the experiments are ran on NVIDIA A100 GPU.

4.2. Experimental results

4.2.1. Robustness under various attack intensities

We evaluated the robustness of the proposed CAD model and other state-of-the-art defenses under different levels of attack intensity with the CIFAR-10 dataset with the VGG model. The results, as shown in Table 3, indicate that under moderate attacks (25% malicious clients), most defense methods perform reasonably well, with our CAD model achieving the highest accuracy across all attack types. However, as the intensity of the attacks increases to severe (50%) and extreme (75%), the performance of most methods significantly deteriorates. In contrast, the CAD model consistently maintains high accuracy and robustness by outperforming other methods under all attack intensities. This demonstrates the effectiveness of our confidence-based detection and re-weighted aggregation approach in mitigating the impact of various attack types, even under severe attack conditions.

4.2.2. Applicable to various models and datasets

Table 4 compares the performance of CAD with other state-of-the-art defense methods across different models and datasets. The results demonstrate that CAD consistently outperforms other defenses with high accuracy under various attacks. For MNIST dataset with MLP model, most defense methods perform well under label shuffling attacks, but CAD achieves the highest accuracy across all attack types. Similarly, for Fashion MNIST dataset with AlexNet model, CAD maintains superior performance, especially against LIE and MM attacks, where other methods significantly collapsed in accuracy. In the CIFAR-10 dataset, both with VGG-11 and AlexNet models, the robustness of CAD stands out. While other defenses struggle to converge, particularly under LIE and MM attacks, CAD consistently achieves the best results. This highlights the effectiveness of our method in providing robust defense against various attacks, even under different model and dataset combinations. These results demonstrated that CAD is not only effective

Table 3
Robustness to various modes of attack on CIFAR-10 with VGG-11 model. Best results are shown in **bold**.

Intensity	Defence	Min–Max		Min-Sum		LIE		Label Shuffle	
		Full	Partial	Full	Partial	Full	Partial	Full	Partial
Moderate (25%)	FedAvg	55.40	64.37	49.44	59.20	49.84	52.89	68.32	68.60
	TrimMean	53.69	61.35	51.58	60.33	31.63	34.22	65.20	65.20
	AFA	68.89	68.71	68.99	69.17	69.09	68.80	68.88	69.09
	FLDetector	32.66	32.37	25.93	35.61	61.94	56.92	39.74	46.32
	DeFL	65.67	69.04	62.97	68.74	48.02	51.51	67.95	68.57
	FedRoLA	69.23	69.03	68.59	68.76	68.67	68.95	68.48	68.59
	CAD (Ours)	**69.85**	**69.43**	**69.94**	**69.43**	**69.67**	**69.18**	**69.36**	**69.47**
Severe (50%)	FedAvg	19.02	18.37	21.40	22.25	12.94	15.04	62.91	62.70
	TrimMean	16.92	17.27	17.49	17.20	11.55	12.35	37.26	36.88
	AFA	16.65	17.89	15.42	17.42	10.42	10.56	60.32	60.85
	FLDetector	23.61	20.02	20.47	20.73	14.41	11.96	34.84	38.81
	DeFL	21.60	27.50	48.26	48.26	23.17	23.95	63.02	62.31
	FedRoLA	19.52	19.60	21.40	22.10	13.42	14.17	62.52	61.98
	CAD (Ours)	**64.00**	**65.07**	**64.75**	**65.13**	**64.44**	**64.96**	**64.27**	**64.19**
Extreme (75%)	FedAvg	16.71	17.72	17.79	17.60	10.56	12.08	50.51	53.46
	TrimMean	14.57	15.03	17.52	16.60	10.01	10.42	21.17	20.23
	AFA	11.54	11.62	13.84	10.03	10.06	10.51	20.23	22.17
	FLDetector	16.71	17.72	17.79	16.31	11.04	12.08	18.33	18.04
	DeFL	20.11	19.44	19.42	20.02	16.06	16.76	52.18	52.72
	FedRoLA	16.71	16.86	17.79	17.59	10.00	10.00	54.39	52.46
	CAD (Ours)	**56.96**	**57.04**	**56.88**	**56.48**	**56.75**	**56.54**	**57.68**	**57.79**

Table 4
Defense robustness across various models and datasets. Test accuracy of different defense methods with various attacks across different models and datasets under severe attack settings. The best results are shown in **bold**.

Model	Defence	MM	MS	LIE	LS
Cifar10 (VGG-11)	FedAvg	19.02	21.40	12.94	62.91
	AFA	16.65	15.42	10.42	60.32
	FLDetector	23.61	20.47	14.41	34.84
	DeFL	21.60	48.26	23.17	63.02
	FedRoLA	19.52	21.40	13.42	62.52
	CAD (Ours)	**64.00**	**64.75**	**64.44**	**64.27**
Cifar10 (AlexNet)	FedAvg	19.86	19.75	10.16	59.63
	AFA	18.68	19.47	10.01	57.02
	FLDetector	17.24	17.17	10.00	32.47
	DeFL	30.69	49.05	18.98	59.42
	FedRoLA	19.52	19.75	10.16	59.33
	CAD (Ours)	**62.92**	**62.60**	**62.90**	**62.76**
Fashion MNIST (AlexNet)	FedAvg	78.71	74.58	15.66	88.06
	AFA	23.70	23.70	10.00	88.56
	FLDetector	78.37	74.98	10.00	76.51
	DeFL	87.73	86.83	10.11	88.25
	FedRoLA	78.09	74.58	15.66	88.05
	CAD (Ours)	**88.43**	**88.32**	**87.86**	**88.79**
MNIST (MLP)	FedAvg	95.81	94.47	52.45	97.36
	AFA	94.33	92.70	11.35	96.73
	FLDetector	93.24	92.36	45.60	96.92
	DeFL	94.29	93.29	88.90	97.47
	FedRoLA	95.81	94.47	52.45	97.40
	CAD (Ours)	**97.87**	**97.73**	**97.08**	**97.83**

in mitigating attacks for a specific benchmark but also adaptable to various models and datasets, thereby making it a reliable defense method.

4.2.3. Robustness to data heterogeneity

We further investigated the robustness of CAD across different levels of data heterogeneity by varying the key parameter α in the Dirichlet distribution Dir(α). A smaller α indicates a higher degree of non-IID data. We simulated a moderate LIE attack and a severe Label Shuffle attack on the Cifar10 dataset with VGG-11 model. As shown in Fig. 5, as the non-IID degree decreases (*i.e.*, , as α increases), the global model accuracy improves at a much higher rate compared to FedAvg, as indicated by the blue shaded area, consistently under both Full and Partial scenarios. These results confirm the robustness of our CAD model to varying data heterogeneity.

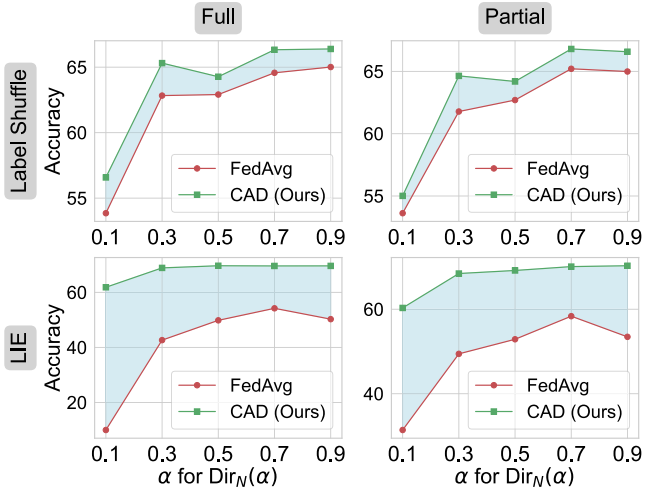

Fig. 5. Robustness to data heterogeneity by varying degrees of Non-IID data, as controlled by the parameter α in the Dirichlet distribution Dir(α). The improvement over the FedAvg baseline is shown in the blue shaded area.

4.2.4. Malicious clients detection accuracy

We evaluated the detection performance of our proposed CAD model under LIE and label shuffle attack scenarios considering both moderate and severe intensities. The evaluation was conducted on the Cifar10 dataset with the VGG-11 model at full attack setting. As shown in Fig. 6, our CAD model can accurately detect malicious clients, and converge to high TPR and low FPR after only a few training rounds. Therefore, it can exclude malicious clients and lead to a significant improvement in overall performance, as shown in the blue line over the orange line. Its advantage is more obvious when more clients are being attacked, with detection accuracy remaining high and stable even under severe attack conditions while the FedAvg was collapsed. This demonstrates that CAD is effective under varying intensities and thereby significantly enhances model performance compared to the FedAvg baseline.

4.2.5. Impact of re-weighting

We ablated the impact of the re-weighting operation and the results are shown in Fig. 7. It can be observed that the re-weighting operation can consistently enhance performance, especially at higher attack

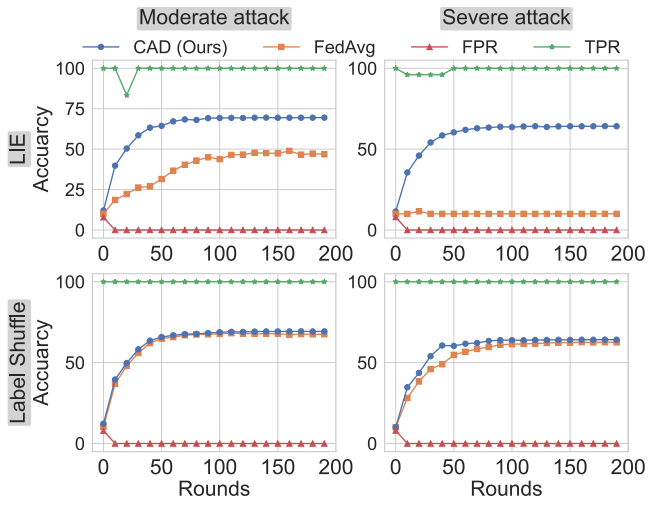

Fig. 6. Malicious client detection accuracy evaluated on the Cifar10 dataset with the VGG-11 model.

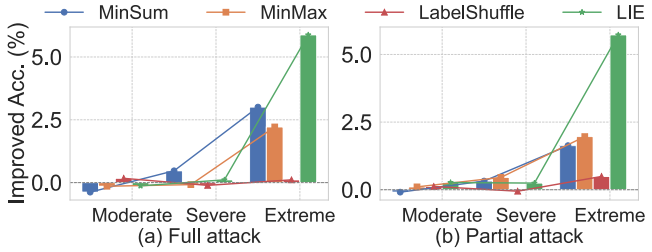

Fig. 7. Effect of applying re-weighting model accuracy across different attack intensities. For each level of attack severity, there are four bars representing different attackers. As attack intensities increase, we can observe consistent improvements with particularly noticeable under "Severe" and "Extreme" conditions.

rates. The improvement is particularly clear in "Severe" and "Extreme" conditions, with detecting attacks like LIE and MinSum benefiting the most. This effectiveness is likely due to the fact that only a few honest clients remain under high attack rates, therefore, it is crucial to identify and prioritize them. This further demonstrates that the confidence score is a reliable metric for assessing the quality of client models, thereby supporting the underlying assumption of our confidence-aware CAD framework.

4.3. Limitations and discussion

One challenge in the proposed CAD framework is handling dynamic attack strategies where malicious clients adapt their behavior over time to evade detection. Such adversaries may alter updates or confidence scores to resemble honest clients, which reduces the effectiveness of static clustering in distinguishing between the two groups. For example, overlapping confidence scores between malicious and honest clients can lower detection accuracy, especially against sophisticated and evolving threats. To address this limitation, future research could focus on adaptive or online learning techniques that update the detection mechanism based on real-time and historical data, and also to consider the anonymity between identity and model updates [57] to ensures accuracy and cost-effectiveness client selection against advanced adversaries. By doing so, CAD can adjust clustering boundaries dynamically to account for changes in attack patterns, and further improve robustness against advanced strategies. While the CAD framework improves the robustness of federated learning systems by leveraging confidence scores from client models, this design introduces potential privacy risks. Confidence scores, although less sensitive than raw data, may still leak information about a client's local data distribution or training data characteristics. For instance, consistently high or low confidence in specific classes could reflect underlying data imbalances or user-specific preferences. An adversary with access to these scores may be able to infer sensitive properties of a client's dataset, such as data scarcity, class distribution, or even the presence of specific categories. To address this issue, future work may explore incorporating techniques such as differential privacy or confidence score obfuscation to further enhance privacy protection.

5. Conclusion

In this paper, we proposed the CAD model, an attack defense mechanism for FL that achieves accurate malicious client detection to address both model poisoning and data poisoning attacks. The proposed CAD model estimates per-client confidence scores to reflect the model's uncertainty, which are then referred to identify clients into honest and malicious categories. By re-weighting the contributions of honest clients based on their confidence scores, Extensive experiments conducted on various benchmarks with non-IID data demonstrate that CAD significantly improves the accuracy, stability, and robustness of FL systems against specific poisoning attacks. However, further evaluation is needed to assess its effectiveness against adaptive adversaries that may evolve to evade confidence-based detection. Besides, our current approach assumes a fixed binary partition (k=2) between honest and malicious clients. While this assumption works well when a client is either fully honest or fully malicious, it may overlook more nuanced behaviors, such as partially poisoned clients, or clients switching between honest and malicious behaviors. Future work could explore adaptive or soft clustering techniques to better capture these behaviors, along with experimental evaluations against such sophisticated attackers.

CRediT authorship contribution statement

Qilei Li: Writing – original draft, Conceptualization, Writing – review & editing, Formal analysis. **Pantelis Papageorgiou:** Conceptualization, Methodology. **Gaoyang Liu:** Conceptualization, Methodology. **Mingliang Gao:** Funding acquisition, Conceptualization, Formal analysis. **Linlin You:** Supervision, Resources. **Chen Wang:** Supervision, Resources. **Ahmed M. Abdelmoniem:** Project administration, Resources, Methodology.

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 in part by UKRI-EPSRC GRANT Reference EP/X035085/1, and National Natural Science Foundation of China GRANT Reference 62272183. The research presented in this paper was conducted while Qilei Li was with Queen Mary University of London, UK.

Data availability

Data will be made available on request.

References

[1] F. Herrera, D. Jiménez-López, A. Argente-Garrido, N. Rodríguez-Barroso, C. Zuheros, I. Aguilera-Martos, B. Bello, M. García-Márquez, M.V. Luzón, FLEX: Flexible federated learning framework, Inf. Fusion 117 (2025) 102792.

[2] A.K. Abed, Utilizing artificial intelligence in cybersecurity: A study of neural networks and support vector machines, Babylon. J. Netw. 2025 (2025) 14–24.

[3] A. Denis, A. Thomas, W. Robert, A. Samuel, S.P. Kabiito, Z. Morish, M. Sallam, G. Ali, M.M. Mijwil, A survey on artificial intelligence and blockchain applications in cybersecurity for smart cities, SHIFRA 2025 (2025) 1–45.

[4] E.T.M. Beltrán, M.Q. Pérez, P.M.S. Sánchez, S.L. Bernal, G. Bovet, M.G. Pérez, G.M. Pérez, A.H. Celdrán, Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges, IEEE Commun. Surv. Tutor. 25 (4) (2023) 2983–3013.

[5] D. Shenaj, G. Rizzoli, P. Zanuttigh, Federated learning in computer vision, IEEE Access 11 (2023) 94863–94884.

[6] P. Fang, J. Chen, On the vulnerability of backdoor defenses for federated learning, in: Proc. of AAAI, 2023.

[7] K.N. Kumar, C.K. Mohan, L.R. Cenkeramaddi, The impact of adversarial attacks on federated learning: A survey, IEEE Trans. Pattern Anal. Mach. Intell. (2023).

[8] J. Zhou, J. Hu, J. Xue, S. Zeng, Secure fair aggregation based on category grouping in federated learning, Inf. Fusion 117 (2025) 102838.

[9] V. Tolpegin, S. Truex, M.E. Gursoy, L. Liu, Data poisoning attacks against federated learning systems, in: Proc. of ESORICA, 2020.

[10] C. Xie, K. Huang, P.-Y. Chen, B. Li, DBA: Distributed backdoor attacks against federated learning, in: Proc. of ICLR, 2020.

[11] M. Fang, X. Cao, J. Jia, N.Z. Gong, Local model poisoning attacks to Byzantine-robust federated learning, in: Proc. of USENIX, 2020.

[12] A.N. Bhagoji, S. Chakraborty, P. Mittal, S. Calo, Analyzing federated learning through an adversarial lens, in: Proc. of ICML, 2019.

[13] P. Blanchard, E.M. El Mhamdi, R. Guerraoui, J. Stainer, Machine learning with adversaries: Byzantine tolerant gradient descent, in: Proc. of NeurIPS, 2017.

[14] D. Yin, Y. Chen, R. Kannan, P. Bartlett, Byzantine-robust distributed learning: Towards optimal statistical rates, in: ICML, 2018.

[15] Y. Zhang, et al., FLDetector: Defending federated learning against model poisoning attacks via detecting malicious clients, in: Proc. of ACM SIGKDD, 2022.

[16] G. Yan, H. Wang, X. Yuan, J. Li, DEFL: Defending against model poisoning attacks in federated learning via critical learning periods awareness, in: Proc. of AAAI, 2023.

[17] X.Y. Gang Yan, J. Li, FedRoLA: Robust federated learning against model poisoning via layer-based aggregation, in: Proc. of ACM SIGKDD, 2024.

[18] N. Rodríguez-Barroso, D. Jiménez-López, M.V. Luzón, F. Herrera, E. Martínez-Cámara, Survey on federated learning threats: Concepts, taxonomy on attacks and defences, experimental study and challenges, Inf. Fusion 90 (2023) 148–173.

[19] D. Chiaro, E. Prezioso, M. Ianni, F. Giampaolo, FL-enhance: A federated learning framework for balancing non-IID data with augmented and shared compressed samples, Inf. Fusion 98 (2023) 101836.

[20] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. (TIST) 10 (2) (2019) 1–19.

[21] P. Kairouz, H.B. McMahan, et al., Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (1–2) (2021) 1–210.

[22] J. Wang, H. Deng, Y. Wang, J. Xie, H. Zhang, Y. Li, S. Guo, Multi-sensor fusion federated learning method of human posture recognition for dual-arm nursing robots, Inf. Fusion 107 (2024) 102320.

[23] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, Knowl.-Based Syst. 216 (2021).

[24] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proc. of AISTATS, 2017.

[25] W. Li, J. Li, et al., Federated learning in mobile edge computing: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 2031–2063.

[26] L. Lyu, H. Yu, Q. Yang, Threats to federated learning: A survey, IEEE Internet Things J. 7 (6) (2020) 5390–5409.

[27] E. Bagdasaryan, et al., Backdoor attacks on federated learning, in: Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics, AISTATS, 2020, pp. 2938–2948.

[28] X. Li, N. Wang, S. Yuan, Z. Guan, FedIMP: Parameter importance-based model poisoning attack against federated learning system, Comput. Secur. (2024) 103936.

[29] X. Shen, Y. Liu, F. Li, C. Li, Privacy-preserving federated learning against label-flipping attacks on non-iid data, IEEE Internet Things J. 11 (1) (2023) 1241–1255.

[30] E. Smith, R. Brown, Defense mechanisms for clean-label poisoning attacks in federated learning, IEEE Trans. Neural Netw. Learn. Syst. 34 (3) (2023) 1234–1245.

[31] W. Zhang, L. Chen, Backdoor attacks in federated learning: Methods and defense mechanisms, J. Mach. Learn. Res. 25 (2) (2023) 567–580.

[32] S. Liu, Z. Li, Q. Sun, L. Chen, X. Zhang, L. Duan, FLOW: A robust federated learning framework to defend against model poisoning attacks in IoTs, IEEE Internet Things J. (2023).

[33] J. Lee, M. Park, Adversarial model poisoning in federated learning: Analysis and mitigation, IEEE Trans. Neural Netw. Learn. Syst. 35 (4) (2023) 987–998.

[34] H. Wang, K. Zhou, Krum++: Enhanced Byzantine-robust federated learning, Proc. ICML 40 (1) (2023) 123–135.

[35] F. Liu, Y. Zhao, Enhanced trimmed mean: A robust aggregation method for federated learning, in: Proc. of NeurIPS, 2023.

[36] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations, ICLR, 2018.

[37] J.M. Cohen, E. Rosenfeld, J.Z. Kolter, Certified adversarial robustness via randomized smoothing, in: International Conference on Machine Learning, ICML, 2019.

[38] C. Fung, C.J. Yoon, I. Beschastnikh, The limitations of federated learning in sybil settings, in: 23rd International Symposium on Research in Attacks, Intrusions and Defenses, RAID, 2020.

[39] S.I. Gass, L.S. Joel, Concepts of model confidence, Comput. Oper. Res. 8 (1981) 341–346.

[40] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: ICML, 2016.

[41] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: Proc. of NeurIPS, 2017.

[42] A. AlSereidi, S.Q.M. Salih, R. Mohammed, A.A. Zaidan, H. Albayati, D. Pamucar, A.S. Albahri, B. Zaidan, K. Shaalan, J. Al-Obaidi, et al., Novel federated decision making for distribution of anti-SARS-CoV-2 monoclonal antibody to eligible high-risk patients, Int. J. Inf. Technol. Decis. Mak. 23 (01) (2024) 197–268.

[43] N. Mourad, S. Qahtan, B. Zaidan, H.A. Ibrahim, A. Zaidan, Bi-level hierarchical ensemble intelligent approach for evaluating spatio-temporal semantic data management systems in IoT-based agriculture 5.0, Expert Syst. Appl. 276 (2025) 127083.

[44] B. Zaidan, W. Ding, H. Alsattar, N. Mourad, A. Zaidan, S. Qahtan, T.F. Ng, Y.-R. Zeng, I. Alshakhatreh, Exploration and development of a structured multi-level fusion in an ensemble-based large-scale meta-decision model, Inf. Fusion 117 (2025) 102911.

[45] T. Castells, P. Weinzaepfel, J. Revaud, SuperLoss: A generic loss for robust curriculum learning, in: Proc. of NeurIPS, 2020.

[46] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Tech. Rep., University of Toronto, 2009.

[47] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE (1998).

[48] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.

[49] J. Darroch, D. Ratcliff, A characterization of the Dirichlet distribution, J. Amer. Statist. Assoc. 66 (335) (1971) 641–643.

[50] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Proc. NeurIPS (2012).

[51] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proc. of ICLR, 2015.

[52] M.-C. Popescu, V.E. Balas, L. Perescu-Popescu, N. Mastorakis, Multilayer perceptron and neural networks, WSEAS Trans. Circuits Syst. 8 (7) (2009) 579–588.

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

[54] G. Baruch, M. Baruch, Y. Goldberg, A little is enough: Circumventing defenses for distributed learning, Adv. Neural Inf. Process. Syst. 32 (2019).

[55] V. Shejwalkar, A. Houmansadr, Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning, in: NDSS, 2021.

[56] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, in: Proc. of NeurIPS, 2019.

[57] X. Chen, Y. Gao, H. Deng, AIFL: Ensuring unlinkable anonymity and robust incentive in cross-device federated learning, IEEE Internet Things J. (2024).