FedCon: A Model Consistency-based Mechanism to Safeguard Federated Learning in Vulnerable and Heterogeneous IoT Environments

Xiaoyi Li, Xuewei Tao, Gengxiang Chen, Linlin You*, *Senior Member, IEEE*, Yongzheng Sun*, *Member, IEEE*

***Abstract*—Federated Learning (FL) offers a privacy-preserving and cost-efficient paradigm to collaboratively train a global model without exposing local data of clients in IoT. However, since client data remain decentralized, it makes the learning vulnerable to malicious behaviors, such as data poisoning attacks, through which, updates can be manipulated to corrupt the global model and damage the overall IoT system. To address such a security threat, gradient similarity-based methods have been proposed to remedy the impact of data poisoning in securing FL, mainly under independent and identically distributed (IID) scenarios. Since heterogeneous Non-IID data are widely distributed in IoT systems, and direct aggregation on local models is more native for FL to update the global model, it becomes challenging for current solutions to distinguish whether models are learned from clients with poisoned or imbalanced data samples. Therefore, this paper proposes FedCon, a model consistency-based mechanism to secure the FL process. It leverages the consistency of local models to identify benign models for global model aggregation. Experimental results demonstrate that FedCon can outperform state-of-the-art methods in defending against data poisoning attacks under both IID and Non-IID scenarios to not only boost model accuracy but also reduce attack success rate.**

***Index Terms*—Federated Learning, Model Consistency, Data Poisoning Attacks, Label Flipping Attacks, Learning Protection**

I. Introduction

Recent years have witnessed a rapid development of IoT through the proliferation of various smart and connectable devices [1]. To harness the massive amounts of data sensed and preserved within IoT, traditional methods rely on a central server for data collection, process, and storage. Under the increasing concerns of data security and privacy, centralized methods becomes less practical to support IoT systems, especially these with user sensitive data [2]. Thus, as a novel decentralized learning paradigm, Federated Learning (FL) [3] emerges to engage private-preserving collaboration among heterogeneous IoT devices to support applications in various domains, e.g., smart city [4]–[6], autonomous vehicles [7]–[11], and personal healthcare [12]–[15]. In general, FL allows distributed clients conduct local training based on their local datasets, and employs a server to aggregate local models uploaded by the clients for a globally shareable model [16].

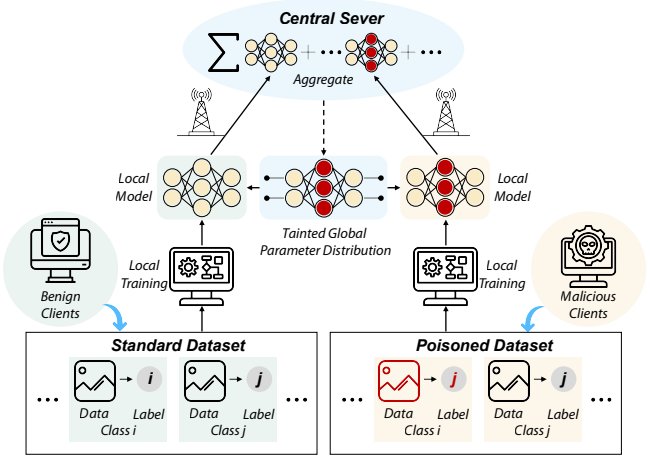


Fig. 1. Security threats introduced by data poisoning attacks in FL.

Since FL ensures user privacy by keeping local datasets opaque to the central server [17], [18], it loses the full control of training data, making the learning process vulnerable to data poisoning attacks [19]–[21]. As illustrated in Fig. 1, malicious clients can disrupt the global learning process by simply launching label flipping attacks (LFAs) [22], [23], which alter labels of selected samples from one class (source class) to another (target class). Consequentially, poisoned local models can be trained, and used to update the global model. If an appropriate defense mechanism is missing, these contaminated models can not only disrupt the overall model learning process making the global model hard to converge but also compromise the entire IoT systems making them vulnerable to encoded security threats.

To safeguard the learning, various defense methods have been proposed to counter these straightforward yet highly destructive data poisoning attacks. Broadly, they can be classified into two categories:

- **Robust Model Aggregation**: It intends to remedy the harm of adversarial updates by designing weighted model aggregation algorithms [24]–[27].
- **Malicious Update Detection**: It attempts to classify local updates into benign and malicious, and aggregate benign

*Corresponding author: Linlin You, Yongzheng Sun.

This work was supported by the National Key Research and Development Program of China (2023YFB4301900), the Guangdong Basic and Applied Basic Research Foundation (2023A1515012895), the Department of Science and Technology of Guangdong Province (2021QN02S161) and the Shenzhen Science and Technology Program (JCYJ20240813101300001). *(Xiaoyi Li, Xuewei Tao and Gengxiang Chen contributed equally to this work.)*

Xiaoyi Li, Gengxiang Chen and Linlin You are with the School of Intelligent Systems Engineering, Sun Yat-Sen University, Shenzhen 518107, CN.

Xuewei Tao and Yongzheng Sun are with the School of Mathematics, China University of Mining and Technology, Xuzhou 221116, China.

Manuscript received XX XX, 2025; revised XX XX, 2025.

updates for the global model [28]–[33].

Although many defense methods have been developed, they still suffer from several limitations while applying to IoT systems. First, most methods assume that local datasets used for model training are Independent and Identically Distributed (IID), meaning that the data of each client shares the same distribution. However, in real-world IoT environments, data are typically Non-IID, entailing that different clients have distinct data distributions [32]. For example, in an IoT-based healthcare system, an IID setting implies that all hospitals contribute patient data with similar disease distributions, whereas in a Non-IID setting, different hospitals may have patient populations with varying demographics and disease prevalence. Second, instead of directly processing local models uploaded by IoT clients at the server that are generally supported by FL frameworks to harness edge resources for a performance boost, local gradients are required by these methods to make the classification between malicious and benign updates [28]–[30]. As a result, it may not only lag the overall model training speed but also increase the learning overhead. Finally, as the information injected by manipulated data and introduced by imbalanced data may become marginal, similarity-based methods [32], [33] may become less efficient while the level of data heterogeneity increases.

To address these limitations, this paper proposes FedCon, which leverages model consistency to safeguard the collaborative and privacy-preserving learning process in FL. In general, FedCon identifies malicious models by comparing the response of local models toward the same stimulus. To be specific, first, based on the responses of models towards same pairs of stimuli [34]–[37], FedCon calculates model consistency score (MCS) and forms a Model Voting Matrix (MVM) that stores the MCS value of every model pair. Secondly, the central server scans the MVM row by row to distinguish benign and malicious models. Finally, the global model is updated by aggregating benign models that pass the voting process. In general, the main contributions of this paper can be summarized as follows:

- To defend against data poisoning attacks in FL, FedCon designs and implements a malicious model identification mechanism that can efficiently distinguish malicious and benign models by analyzing the consistency of models in response to a set of common stimuli rather than directly calculating their similarity based on uploaded gradients. Therefore, it can support the application of FL in common IoT systems consisting of Non-IID data.
- To safeguard FL in scenarios with high data heterogeneity level and malicious client rate, FedCon can overcome the shortcomings in similarity-based methods that may become inefficient when knowledge encoded in local data tends to be sparse and discrete in related feature space. Hence, it can stable the overall learning process of FL to correctly generate required model with high performance.
- To ensure compatibility with general FL frameworks, FedCon facilitates direct model exchange instead of model gradients. This design allows FedCon to integrate seamlessly with FL frameworks, enabling them to defend against data poisoning attacks without disrupting the learning process.

Moreover, based on standard datasets, FedCon is compared with state-of-the-art baselines (i.e., FedAVG [3], Median [24], PEFL [27], LFR-PPFL [32], FoolsGold [33]). It shows that FedCon can effectively resist label flipping attacks, even in scenarios with high data heterogeneity.

The rest of this paper is organized as follows. First, Section II summarizes the related work. Second, Section III presents preliminary concepts. Third, Sections IV and V introduce and evaluate FedCon, respectively. Finally, section VI concludes this paper and discusses future works.

II. Related Work

This section introduces data poisoning attacks LFAs under FL and summarizes the corresponding state-of-the-art defense methods.

A. Data poisoning attacks LFAs under FL

LFAs are initially designed to disrupt centralized machine learning algorithms by modifying portions of training datasets [38], [39]. In the context of FL, where private data is processed locally and modifications on local datasets are invisible to the global server, LFAs pose a significant security threat to FL [40], [41], either making the overall learning process less efficient or exposing the entire systems under certain security treads. To execute LFAs in FL, malicious clients need only alter the labels of local samples from the source class to the target class. While participating in the global learning process, these clients use their modified datasets to generate local updates and upload adversarial gradients or models to the global server. Since the sample features remain unchanged in LFAs, these attacks are more covert than other methods, such as Byzantine attacks. Additionally, malicious clients can alter fewer samples [23] to further enhance the stealth of the attack, making data poisoning effects similar to the effects introduced by Non-IID data.

B. Defense methods

Given the urgent need to support security-enhanced FL, many defense mechanisms have been proposed, which can generally be categorized into two types:

- **Robust Model Aggregation**: It is to minimize the hazard of LFAs by adaptively adjusting weights in model aggregation to update the global model. In [24], authors propose several strategies to exclude local updates from malicious clients, e.g., by calculating the median of each uploaded parameter as the global parameter (i.e., Median) or conditionally averaging uploaded gradients to form global gradients (i.e., Trimmed-mean). To design a more reasonable aggregation strategy, the Krum algorithm and Multi-Krum algorithm [25], which calculate the mark of every uploaded gradient based on the Euclidean distance between each gradient pair, are used to update the global model by gradients with the lowest score. In addition, a Multi-Krum-based framework, named as BREA [26],

is proposed to enhance the privacy-preserving learning process under a quantized stochastic gradient setting. Instead of eliminating malicious updates, the authors in [27] leverage coordinate-wise median as a benchmark, calculate the similarity between the benchmark and each uploaded gradient, and correspondingly, adjust the aggregation weight to stabilize the learning.

- **Malicious Update Detection**: It attempts to distinguish benign and malicious updates based on information uploaded from each client. Since malicious clients use modified datasets to train local models, their corresponding updates should hold diacritical feature spaces and can be separated by clustering algorithms (i.e., KMeams [28] or KPCA [29]). In [30], authors argue that when LFAs are launched, layers apart from the Fully Connected Layer (FC) are not influenced as they only extract data features. Therefore, the distributed weights inside the FC layer can be used to distinguish benign and malicious updates. Accordingly, a Hidden Markov Model (HMM) is introduced in [31] to estimate the probability of clients uploading benign information. Assuming that malicious clients behave likewise, [33] proposes a FoolsGold algorithm that mitigates impacts of LFAs by penalizing clients with similar updates. Finally, based on the observation that the cosine similarity between malicious and benign gradients gradually declines, [32] proposes a label-flipping-robust and privacy-preserving FL (LFR-PPFL) algorithm to distinguish malicious and benign clients based on temporal analysis of cosine similarity.

Despite that significant progress has been made to prevent the impact of data poisoning attacks, such as LFAs, current defense mechanisms still face several limitations. First, many methods exhibit low adaptability to support Non-IID scenarios, limiting their applicability in IoT systems and services. Second, to better distinguish malicious and benign updates, most approaches require clients to upload gradients instead of local models for the similarity analysis. It will increase the computational burden of the server, and slows down the overall learning process. Finally, the gradient-based similarity analysis often shows limited distinction, making it incapable to accuratelyClients engage in synergistic cooperation identify malicious clients, especially when data heterogeneity increases among IoT clients.

III. PRELIMINARY

A. Label-Flipping Attacks in FL

FL preserves the confidentiality of private data by ensuring that local data remains inaccessible to the central server [17], [18]. However, its decentralized nature makes it vulnerable to data poisoning attacks, in which, malicious clients can generate updates that diverge from the true gradient direction to interrupt the model learning process or contaminate the model to be learned. In general, as a typical attack, LFAs allow malicious clients manipulate only labels of training samples to generate local updates in FL that can alter the learning in an undesirable direction. It becomes even more severe in highly heterogeneous (Non-IID) settings, as variations in client data distributions can obscure these malicious behaviors. As a result, the global model may suffer from slower convergence and reduced performance across all clients.

B. Problem Formulation

In this work, we adopt a synchronous FL framework, where a central server is configured to ensure that K clients (participated in the learning) will work at the same pace. Specifically, in a global learning round, each local client k performs local training based on its private dataset D_k. Then, it uploads its local model w_k^t to the server and wait for further instructions, either to continue or terminate the training. Once the central server receives local models from all the clients, it starts to identify malicious models and aggregate benign models to update the global model w_{global}^t.

Under the synchronous setting, we assume that less than 50% of the participating clients are malicious. Moreover, given that it is challenging to distinguish contaminated updates from honest ones in Non-IID scenarios, our goal is to design a model consistency-based mechanism that can safeguard the privacy-preserving learning under an untrustworthy environment to ensure the convergence of the global model by aggregating local models trained based on original or manipulated data as defined in Formula 1,

$$\min_{w_{global}^t} F(w_{global}^t) \triangleq \frac{1}{K}\sum_{k=1}^{K} \eta F(w_k^t; D_k) \tag{1}$$

where $F(\cdot)$ is the cost function, η is the learning rate, D_k can be original or manipulated.

IV. METHODOLOGY

In this section, FedCon is introduced. Based on a standard FL system, which consists of a central server to coordinate multiple clients synchronously, the overall architecture of FedCon is designed as illustrated in Fig. 2, which includes three iterative phases in a global learning round:

- **P1: Local Training Phase**: Both benign and malicious clients train local models using their private datasets, and then, upload corresponding local models to the global server;
- **P2: Malicious Model Identification**: The server calculates pair-wise model consistency score (MCS) for uploaded local models, and distinguishes malicious and benign models based on model voting matrix (MVM);
- **P3: Global Aggregation Phase**: The server updates the global model by aggregating local models passing the voting process, and then dispatches the global model to clients to start a new global round.

Within a learning round to update the global model, the three phases will run consecutively until a predefined stop condition is met (i.e., maximum learning round or target model accuracy is reached). In the following subsections, their implementation details are described, respectively.

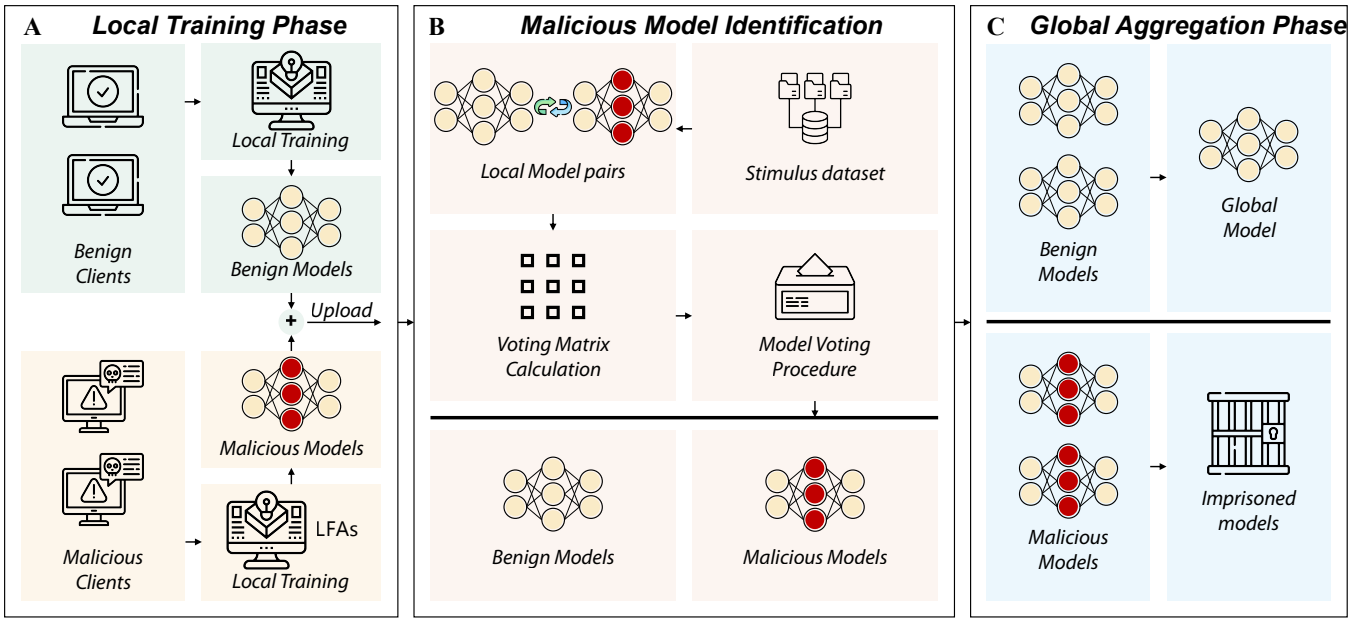


Fig. 2. The overview of FedCon, consisting of three phases: A) **Local Training Phase**, where malicious clients perform Label Flipping Attacks (LFAs); B) **Malicious Model Identification**, where model behaviors are evaluated using a shared stimulus dataset; and C) **Global Model Aggregation**, where only models identified as benign are aggregated to update the global model.

A. Local Training Phase

Assume that there are K local clients working collaboratively to train a globally shareable model under an IoT systems. However, during the learning, there are 1) benign clients that follow the instructions from the server to make their contributions by sharing their local models, and also 2) malicious clients that infiltrate the system intending to disturb the global learning process by launching LFAs (i.e., flipping source class labels into target class labels).

To be specific, in the system, local clients hold private datasets $D = \{D_1, D_2, ..., D_K\}$, and each dataset is divided into training set D_k^{train} and test set D_k^{test}. For malicious clients, during the preparation of local training, they alter their training set as defined in Formula 2,

$$D_k^{train} \xrightarrow{S \to T} D_k^{m-train} \tag{2}$$

where S, T, and $D_k^{m-train}$ are the source class, the target class, and the modified training set, respectively.

When local training starts, client k, first, receives the latest global model w^t generated in the recent t^{th} global round from the server, then sets it as the current local model w_k^t, and finally, initializes the training process according to Formula 3,

$$\begin{aligned} Benign: w_k^t &= w_k^t - \eta \nabla F_k(w_k^t, D_k^{train}) \\ Malicious: w_{k_m}^t &= w_{k_m}^t - \eta \nabla F_k(w_{k_m}^t, D_k^{m-train}) \end{aligned} \tag{3}$$

where η refers to the learning rate; $F_k(\cdot)$ is the loss calculated based on the local training set of client k; w_k^t and $w_{k_m}^t$ represent benign and malicious local models, respectively. Note that during the local training, different from benign clients, who use untouched training data D_k^{train} to update their local models, malicious clients intentionally train their local models based on manipulated training data $D_k^{m-train}$.

After the local training is completed, regardless of benign or malicious clients, they will upload their local model to the server and wait for further instructions from the server to start another round of local training or terminate the learning.

Since both benign and malicious models can be uploaded in a vulnerable environment, the central server must carefully distinguish them to avoid significant disturbance in model aggregation. To tackle this issue, FedCon introduces a malicious model identification phase as introduced below.

B. Malicious Model Identification

In deep neural networks, shallow layers are primarily responsible for learning the fundamental features of the dataset. In contrast, deep layers are designed to extract target-specific characteristics [42]. During LFAs, attackers only flip labels, leaving intrinsic data features unchanged, which will significantly impact the deep layers, especially the output layer. Hence, theoretically, malevolent targets ω_m of malicious clients presented in their corresponding local models shall be distinct from benign targets ω_b. Consequently, when the same stimulus is fed into ω_m and ω_b, respectively, it is expected to observe significant differences between the output layers. Based on this assumption, we calculate the Pearson coefficient of the output vectors to measure the concordance between each pair of local models uploaded by IoT clients, and generate a model consistency-based Matrix. Moreover, considering the heterogeneity among local knowledge extracted from IoT clients, a voting mechanism is designed to identify malicious clients more rationally.

At the beginning of each global learning round, the server distributes the latest global model w^t to each client and waits

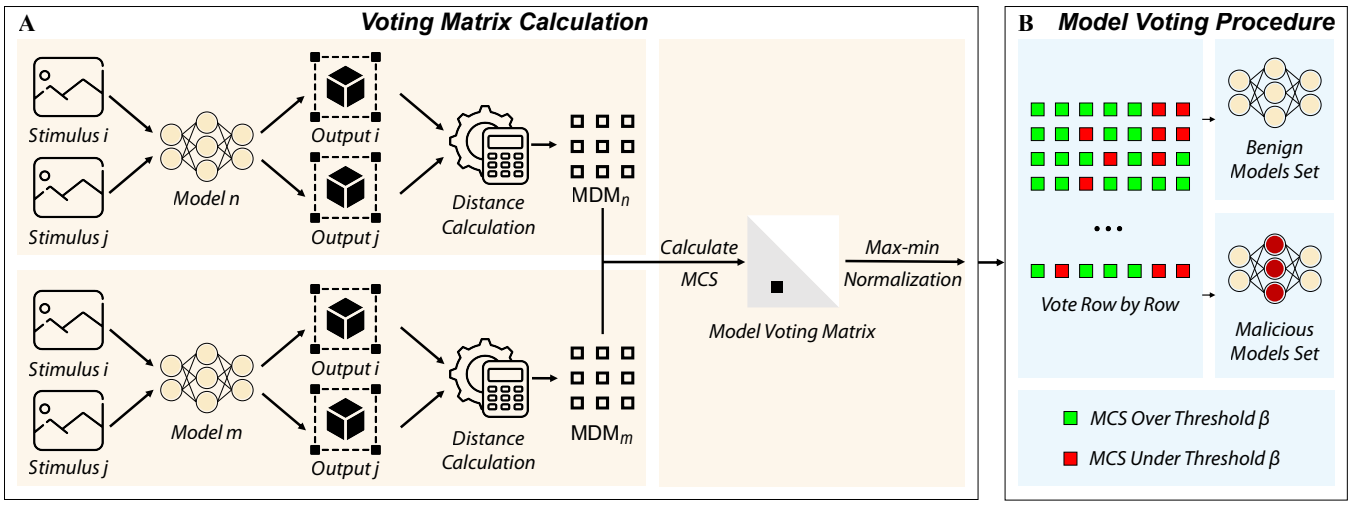


Fig. 3. The detailed workflow of Malicious Clients Identification in FedCon, including: A) **Voting Matrix Calculation**, where Model Consistency Scores (MCS) are calculated based on the Model Dissimilarity Matrices (MDMs) of local models; and B) **Model Voting Procedure**, where models that fail to align with the majority are classified as malicious and excluded from global model aggregation.

to receive updated local models. Once all local models are collected, the server will identify malicious models and aggregate benign models to update the global model. To precisely detect malicious models, FedCon implements a workflow as shown in Fig. 3, where a Model Voting Matrix (MVM) storing model consistency scores (MCSs) of two local models is first created. Afterward, by scanning each row of the matrix, malicious models can be voted and excluded from the global model aggregation process to ensure the poisoned knowledge will not be injected into the global model.

1) Model Voting Matrix (MVM) Calculation: Once the server receives all the local models uploaded by the clients, it evaluates the consistency of models. First, a stimulus dataset $S = \{S_1, S_2, \ldots, S_H\}$ is prepared by the server based on open datasets. Each local model M_n produces an output vector $o_n(S_i)$ in response to a given stimulus S_i. To remedy the intrinsic different in model output and better measure model behavior in the latent space through a unified value, for each local model, a Model Dissimilarity Matrix (MDM) is constructed according to Formula 4:

$$MDM_n[i,j] = Dis(o_n^i, o_n^j), \qquad i,j \leq H \tag{4}$$

where $o_n^i = o_n(S_i)$ stands for the output vector of M_n in response to a given stimulus S_i; H refers to the length of the stimulus dataset; and $Dis(\cdot)$ is the distance measurement function. Note that different $Dis(\cdot)$ can be used freely in practice, and in this study, by default, cosine similarity, as defined in Formula 5, is adopted to measure the alignment level of output vectors and enable a more nuanced fine-grained analysis.

$$cos(o_n^i, o_n^j) = \frac{o_n^i \cdot o_n^j}{|o_n^i||o_n^j|} \tag{5}$$

Accordingly, the model consistency score of two MDMs is computed to evaluate the output alignment of a local model pair n and m toward the same stimulus dataset according to Formula 6,

$$MCS_{n,m} = [\rho(MDM_n, MDM_m)]^2 \tag{6}$$

where $MCS_{n,m}$ refers to the MCS between local models M_n and M_m, and it measures the Pearson Correlation Coefficient ρ of their MDMs, namely MDM_n and MDM_m. Note that $MCS_{n,m}$ is equivalent to $MCS_{m,n}$.

After the computation of MCSs for every pair of received local models, a Model Voting Matrix (MVM) can be created with a form as defined in Formula 7,

$$MVM[n,m] = MCS_{n,m}, \qquad n,m \leq K \tag{7}$$

Aiming at standardizing the difference in model consistency to better distinguish benign and malicious updates, MVM can be rescaled and normalized according to Formula 8,

$$\overline{MVM}[n,m] = \frac{MVM[n,m] - MVM_{min}}{MVM_{max} - MVM_{min}}, n,m \leq K \tag{8}$$

The normalized $\overline{MVM}$ is then utilized in the model voting procedure to identify malicious clients.

2) Model Voting Procedure: Under the assumption that there are no more than 50% malicious clients, a voting procedure is designed to identify updates from them based on MVM. During the voting, a hyperparameter β is configured through methods such as grid search or Bayesian optimization and used as a threshold to distinguish benign and malicious models. Accordingly, the voting procedure of FedCon is implemented according to Formulas 9 and 10,

$$C_n = \sum_{k=1}^{K} \begin{cases} 1, & (\overline{MVM}[n,k]) \geq \beta \\ 0, & (\overline{MVM}[n,k]) < \beta \end{cases} \tag{9}$$

$$w_n^t = \begin{cases} 1, & Benign,\ if\ C_n \geq \frac{K}{2} \\ 0, & Malicious,\ if\ C_n < \frac{K}{2} \end{cases} \tag{10}$$

where C_n refers to the number of elements over threshold β corresponding to row n in $\overline{MVM}$ and w_n^t is the identification result of model n in the t^{th} learning round. Note that since the usage of different optimization methods in configuring β may affect the final voting result, their performance is evaluated in Section V-B.

Algorithm 1 The pseudocode of FedCon

1: **Client Side:**
2: **for** $k = 1$ to K **in parallel do**
3: $w_{local}^k \leftarrow w_{globel}^t$
4: **if** Malicious **then**
5: $D_k^{m-train} \xleftarrow{S \leftarrow T} D_k^{train}$
6: $w_{local}^k \leftarrow$ Train$(w_{local}^k, D_k^{m-train})$
7: **else**
8: $w_{local}^k \leftarrow$ Train$(w_{local}^k, D_k^{train})$
9: **end if**
10: Upload(w_{local}^k)
11: **end for**
12: **Server Side:**
13: Receive$(\{w_{local}^k\}_{k=1}^K)$
14: **while** All models received **do**
15: **for** $n = 1$ to K **do**
16: **for** $m = 1$ to K **do**
17: Calculating MDM_n, MDM_m using (4)
18: Calculating $MCS_{n,m}$ using (6)
19: **end for**
20: **end for**
21: $MVM \leftarrow$ Form using (6,7)
22: Malicious Models $\leftarrow$ Identify$(MVM,$(8,9))
23: Updating w_{globel}^t using (11)
24: Distribute(w^t)
25: $t \leftarrow t + 1$
26: **end while**

C. Global Aggregation Phase

After the identification of malicious models, the central server aggregates benign models on average according to Formula 11,

$$w^t = \sum_{n=1}^{N} \frac{1}{N_{Benign}} w_n^t \tag{11}$$

where N_{Benign} is the number of benign models. Once the server updates the global model, it will distribute the global model to each client and stand by to receive local models or stop the learning if the predefined termination condition is met (i.e., maximum learning rounds). It is worth noting that various weighted aggregation strategies [43]–[45] can be applied in FedCon to enhance the aggregation performance, and for simplicity, the average function is used by default.

D. Algorithm of FedCon

As shown in Algorithm 1, FedCon executes at both client and server sides.

- **Client Side** (Lines 1-9): For FedCon clients, they first receive global model to be trained, and then, set it as their initial local models to be updated. Afterward, benign clients start the training to update local models, while malicious clients launch data poisoning attacks, i.e., LFAs. Finally, both benign and malicious clients upload their updated local models w_k^t to the server.
- **Server Side** (Lines 10-22): The FedCon server dispatches the latest global model and wait for local models of FedCon clients. Once all clients have their local models uploaded, it stimulates every model pair by using the stimulus dataset and computes MVM. Then, it conducts the model voting procedure and identify malicious models. After that, the server updates the global model w^t according to the voting results and distributes it to each client. Finally, the global training round is increased from t to $t + 1$, and the whole system steps into the next round. Note that the global learning process ends when a stop condition is met (e.g., reaching target accuracy or maximum learning iterations).

E. Scalability of FedCon

The time complexity (TC) of FedCon can be calculated based on two main steps. First, to form MDMs (Model Dissimilarity Matrices), $\frac{H^2}{2}$ stimulus pairs are processed by each of K clients, and then, related responses with a feature length L are passed to a cosine function, whose TC is $O(L)$, to measure the alignment level between each of two responses. Accordingly, the TC of this step is $O(K \times \frac{H^2}{2} \times L)$ to generate K MDMs. Second, $\frac{K^2}{2}$ pairs of MDMs, whose dimension is H^2, is used to calculate Model Consistency Scores (MCSs) that form a symmetric matrix, called Model Voting Matrix (MVM). Therefore, the TC of this step is $O(\frac{K^2}{2} \times H^2)$. Since the two steps execute sequentially, the overall TC is the sum of the two TCs, and after simplification, it is $O(\frac{K \times (K+L) \times H^2}{2})$.

Moreover, the Computational Delay (CD) introduced by FedCon can be measured according to Formula 12,

$$CD = \frac{K \times (K + L) \times H^2}{2 \times FLOPS} \tag{12}$$

where $FLOPS$ represents the number of floating-point operations per second that is directly related to the configuration of the central server.

Accordingly, the additional Energy Consumption (EC) can be calculated as the product of the computational delay and the power consumption P of the central server, as depicted in Formula 13.

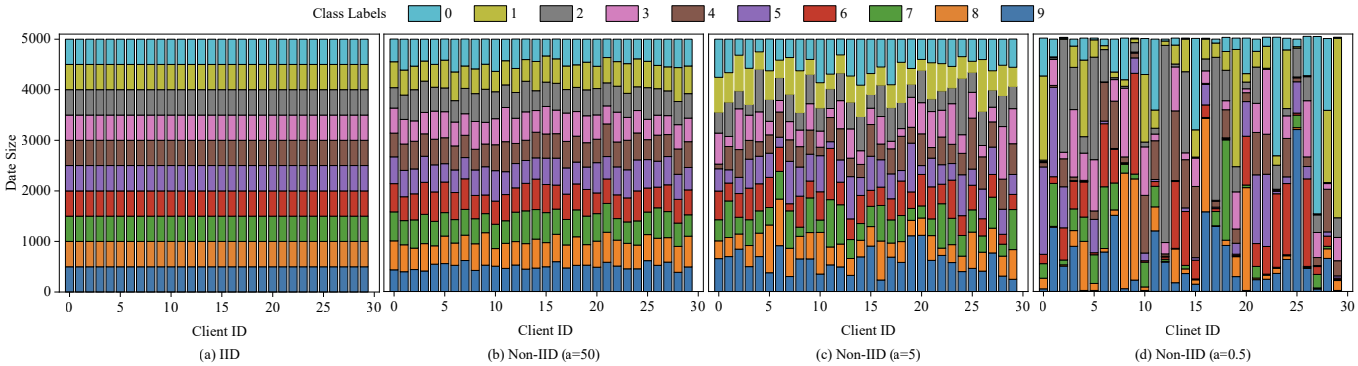


Fig. 4. Data Partition under IID and different Non-IID settings. Note that in Non-IID settings, heterogeneity is controlled by the value of α. Specifically, lower α value refers to higher heterogeneity. This research adopts 3 values (i.e., 50, 5, and 0.5) to synthesize client datasets.

$$EC = CD \times P \tag{13}$$

Since in our method, H is a truly small value, the additional TC, CD, and EC are comparable to those of state-of-the-art methods (e.g., FoolsGold [33] and LFR-PPFL [32]), both of which grow quadratically with the number of clients.

To improve the scalability of FedCon in supporting large-scale federated learning systems, a client selection strategy [44], [46] can be adopted to exclude clients with insufficient or less important information. By selecting a proportion R (where $R < 1$) of clients, the overall computational workload of FedCon can be reduced by a factor of R^2. Specifically, the TC is reduced from $O(\frac{K\times(K+L)\times H^2}{2})$ to $O(\frac{K\times(K+L)\times H^2\times R^2}{2})$, with similar reductions of R^2 applied to the CD and EC. Accordingly, the overall workload of FedCon can be properly controlled to efficiently and effectively support both cross-device and cross-silo federated learning. In cross-device scenarios, by applying client selection strategy, in each learning round, the number of clients can be controlled to extract sufficient local knowledge for the update of global model. Moreover, in cross-silo scenarios, where fewer clients handle large heterogeneous datasets, FedCon can still enhance model aggregation through the analysis of model consistency to mitigate the impact introduced by data heterogeneity. Therefore, FedCon can remain scalable and efficient across diverse federated learning environments.

In summary, unlike gradient-based methods, FedCon can enhance and support model-based workflow to prevent attacks such as LFAs by measuring the output alignment of uploaded models towards the same set of stimuli. Thus, under an open and vulnerable environment, it can run cost-efficiently to support FL in not only IID but also Non-IID scenarios as evaluated in the following section.

V. Evaluation

In this section, FedCon will be evaluated together with other state-of-the-art methods under the same setting.

A. Experiment Settings

To better illustrate the merit of the proposed method, without loss of generality for a fair comparison, five state-of-the-art baselines and three standard datasets are used in the experiment to learn a simple neural network, and related performance is analyzed by the same metrics.

TABLE I
Summary of Experiment Settings[1]

Term	MNIST	GTSRB	SFDDD
Target Accuracy	0.95	0.90	0.95
Learning Rate	0.001		
Batch Size	128		
Local Dataset Size	5000		
Malicious Clients Rate	40%		
Local Training Epoch	3		
Iteration	300		

[1]Except for Target Accuracy, all other settings are identical across the three datasets.

1) Learning Model: A variant of the classic convolutional neural network LeNet-5 is utilized [47], which consists of 3 convolution layers and 1 fully connected layer.

2) Datasets: Three standard datasets are utilized, namely:

- **MNIST [48]**: A classic dataset consisting of 10 classes of handwritten digits (0 to 9), commonly used in classification tasks. It includes 50,000 training images and 10,000 test images, each with a resolution of 28×28×1.
- **German Traffic Sign Recognition Benchmarks (GTSRB) [49]**: This dataset contains various types of German traffic signs, comprising 34,799 training images and 12,630 test images across 43 classes. Each image is a 32×32×3 color picture. For simplicity, the top 10 classes — Yield, No passing, Road work, No passing (3.5t), Priority road, Keep right, and Speed limit 30/50/70/80 — are used in this study to prepare local datasets.
- **State Farm Distracted Driver Detection(SFDDD) [50]** The dataset contains 22,424 color images, each with a resolution of 32×32×3. These images are categorized into 10 classes, including activities such as safe driving, drinking, and reaching behind.

3) Data Preparation: In addition to the common IID setting, Non-IID data is also prepared to reflect the high data heterogeneity in IoT. Specifically, under the IID setting, each client receives a local dataset with 5,000 training samples and 1,000 test samples. For the Non-IID setting, local datasets are

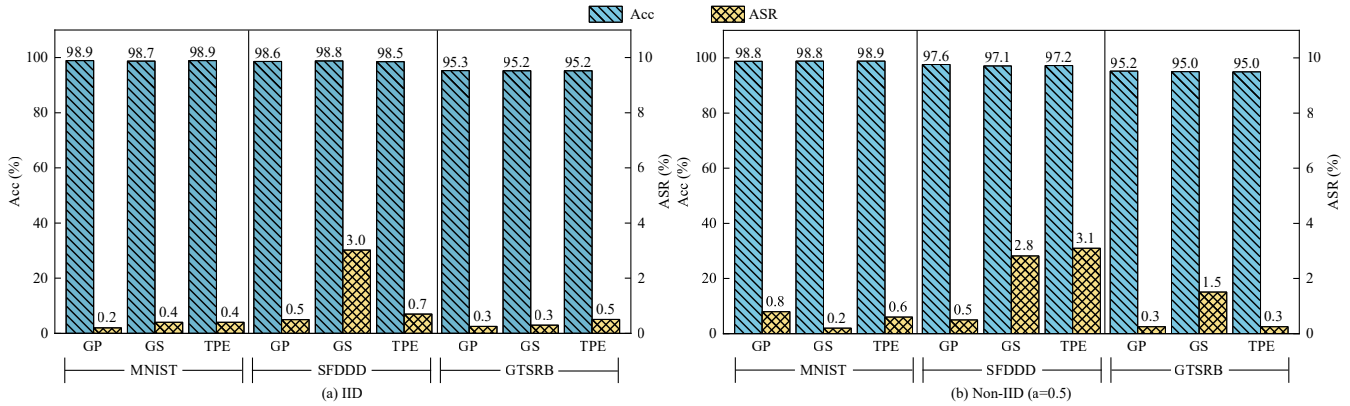


Fig. 5. Top-1 test accuracy and ASR of FedCon under different hyper-parameter optimization methods.

generated based on Dirichlet distribution [51], which controls the degree of heterogeneity by a hyperparameter α. Note that a lower α value indicates greater heterogeneity. Accordingly, we use three α values to create Non-IID data: $\alpha = 0.5$ [52] and $\alpha = 5$ [53], following prior studies, and also $\alpha = 50$ to incorporate a scenario with minimal heterogeneity. An overview of the IID and Non-IID data partitions is shown in Fig. 4. Note that in Non-IID cases, it may happen that malicious clients lack samples from the source class due to the uneven data distribution and to resolve that, related clients can add source class samples to $\frac{1}{10}$ of the total dataset to prepare poisoned training data.

4) Baselines: Five baselines are used, namely:

- **FedAVG [3]**: It is a standard FL method, in which clients upload their local models, and the server average them to produce the global model. Due to its vulnerability to data poisoning attacks, it serves as a baseline representing the worst-case scenario.
- **Median [24]**: It is a method, where clients upload their local models, and the server computes the median of each parameter to form the global model with the impact of data poisoning attacks mitigated.
- **PEFL [27]**: It is an algorithm against attacks by computing the correlation coefficient between each local gradient with coordinate-wise medians, and accordingly, assigning relevant aggregation weight to update the global model.
- **LFR-PPFL [32]**: It is a method that detects malicious clients by analyzing both the historical trends of uploaded gradients and also, the cosine similarity among current uploaded gradients.
- **FoolsGold [33]**: It is a mechanism that calculates the cosine similarity of historical gradients among clients. Based on the assumption that malicious clients tend to exhibit similar behavior, clients with higher similarity are assigned lower weights to update the global model.

5) Evaluation Metrics: Three evaluation metrics are used to measure the performance of different methods, namely:

- **Top-1 test accuracy (Acc)**: A commonly used metric in classification tasks calculated according to Formula 14, where TP, TN, FP, and FN refer to true positives, true negatives, false positives, and false negatives, respectively;

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

- **Attack Success Rate (ASR)**: A common index to measure the severity of LFAs as defined in Formula 15, where N_S refers to the number of testing samples with the source label S and N_T denotes the number of testing samples misclassified into the target label T:

$$ASR = \frac{N_T}{N_S} \tag{15}$$

- **Round required to reach the target accuracy (R_{tacc})**: The global learning round required to reach the target accuracy. It reflects the convergence speed of different FL protection mechanisms. Note that the target accuracy is 0.95 for MNIST, 0.9 for GTSRB and 0.95 for SFDDD.

6) System setup: Finally, an FL system with one central server and 30 clients is established. Each client trains its local model with a batch size of 128 and a learning rate of 0.001. Moreover, by default, 40% of the clients are set as malicious to launch LFAs (e.g., flipping label 9 to label 7 for the MNIST dataset). Since the calculation of model consistency requires common stimulus in FedCon, one image from each category is selected and used, which will be used for model training. Note that detailed experiment settings are listed in Table I. Moreover, related data and code prepared according to the above setting are publicly accessible via the link[1].

B. Evaluation on voting process

Within FedCon, the voting process to distinguish benign and malicious clients is controlled by a hyperparameter β, which can be auto-configured by optimization methods, such as conventional Grid Search (GS), Bayesian optimization with Gaussian Processes (GP), and Tree-structured Parzen Estimator (TPE). To evaluate the performance of these methods in selecting β, an ablation study is performed. As illustrated in Fig. 5, GS, GP and TPE all can achieve remarkable performance in selecting a proper β to not only ensure model

[1]https://github.com/IntelligentSystemsLab/FedCon

TABLE II
THE GENERAL PERFORMANCE [1,2] OF FEDCON AND BASELINES. IT INCLUDES 1) TOP-1 TEST ACCURACY Acc, 2) ATTACK SUCCESS RATE ASR, AND 3) COMMUNICATION ROUND REQUIRED TO REACH TARGET ACCURACY R_{tacc}.

Dataset	*Distribution*	*IID*			*Non-IID (α = 50)*			*Non-IID (α = 5)*			*Non-IID (α = 0.5)*		
	Metric	Acc↑	ASR↓	$R_{tacc}\downarrow$	Acc↑	ASR↓	$R_{tacc}\downarrow$	Acc↑	ASR↓	$R_{tacc}\downarrow$	Acc↑	ASR↓	$R_{tacc}\downarrow$
MNIST	FedAVG [3]	0.936	0.119	-	0.929	0.356	-	0.926	0.261	-	0.931	0.279	-
	Median [24]	0.976	0.094	**3**	0.971	0.168	4	0.968	0.172	**4**	0.962	0.204	**8**
	PEFL [27]	0.958	0.018	164	0.946	0.026	-	0.935	0.036	-	-	-	-
	LFR-PPFL [32]	0.956	0.038	197	0.958	0.042	184	0.951	0.073	199	0.949	0.108	-
	FoolsGold [33]	0.960	0.006	161	0.966	0.004	165	0.958	0.012	185	0.945	0.336	-
	FedCon (Ours)	**0.989**	**0.002**	**3**	**0.988**	**0.002**	**3**	**0.986**	**0.004**	**4**	**0.988**	**0.008**	17
GTSRB	FedAVG [3]	0.884	0.286	-	0.874	0.454	-	0.873	0.386	-	0.841	0.677	-
	Median [24]	0.936	0.030	14	0.924	0.053	20	0.913	0.176	32	0.904	0.374	34
	PEFL [27]	0.910	0.076	384	0.902	0.025	440	0.893	0.147	-	-	-	-
	LFR-PPFL [32]	0.916	0.045	298	0.910	0.086	496	0.888	0.157	-	0.884	0.338	-
	FoolsGold [33]	0.912	0.058	145	0.924	0.040	225	0.897	0.198	-	0.861	0.460	-
	FedCon (Ours)	**0.953**	**0.003**	**7**	**0.947**	**0.016**	**14**	**0.951**	**0.007**	**17**	**0.952**	**0.003**	**17**
SFDDD	FedAVG [3]	0.923	0.282	-	0.900	0.315	-	0.903	0.287	-	0.770	0.520	-
	Median [24]	0.954	0.182	93	0.949	0.227	-	0.936	0.291	-	0.941	0.323	-
	PEFL [27]	0.911	0.023	-	0.917	0.086	-	-	-	-	-	-	-
	LFR-PPFL [32]	0.941	0.012	-	0.949	0.030	-	0.941	0.034	-	0.935	0.150	-
	FoolsGold [33]	0.947	0.016	-	0.935	0.065	-	0.914	0.168	-	0.908	0.273	-
	FedCon (Ours)	**0.986**	**0.003**	**29**	**0.989**	**0.002**	**8**	**0.981**	**0.004**	**13**	**0.976**	**0.005**	**20**

[1]Results in **bold** refer to the best performance while underlined results refer to the second best performance.

[2]The sign **-** means related methods can not make the model converge or exceed the target accuracy.

accuracy but also prevent data poisoning attacks. It shows that in both common IID and high heterogeneity Non-IID settings ($\alpha = 0.5$), regardless of the usage of different parameter optimization methods, FedCon can consistently give a nearly-identical performance. It reveal the capability of FedCon in capturing the differences between benign and malicious models through the calculation of model consistency score. Since GP, in general, can achieve a more balanced performance, it is used in the following experiments to determine optimal β in model voting process.

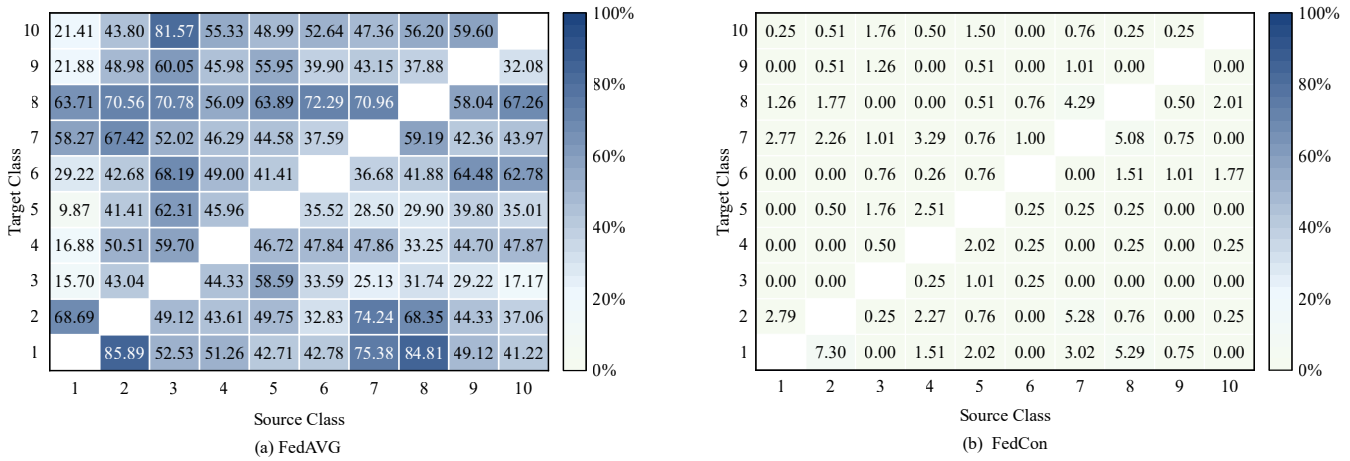
C. Evaluation on general performance

The performance of the proposed method FedCon is compared with the baselines based on three standard datasets under different levels of data heterogeneity, whose results are listed in Table II. Specifically, regardless of datasets, by considering general scenarios under trustworthy running environments (i.e., no data poisoning attacks), while the data heterogeneity increases, the model accuracy of the compared methods decreases, and in worst cases, can not reach the training target under Non-IID ($\alpha = 0.5$). In contrast, FedCon can consistently remain a high and stable model accuracy with an average improvement of 1.91%, 3.44%, and 3.88% in MNIST, GTSRB and SFDDD, respectively, compared to the second best method. Moreover, in Non-IID cases of SFDDD, where all the compared methods can not reach the predefined training target (i.e., 0.95), FedCon can still ensure the convergence of the model within few rounds (less than 20 rounds). It shows that the attack protection mechanisms introduced by FedCon do not cause potential drawbacks to support FL, instead, can bring additional benefits to train more accurate models.

While considering the efficacy in defending data poisoning attacks, in widely discussed IID cases, compared to the baseline FedAvg without any protection, all the methods with attack defense mechanisms can ensure that the trained model will not be contaminated to behave correctly with a low attack success rate (ASR). However, the success rate of attacks of compared methods (i.e., FoolsGold, Median, PEFL, and LFR-PPFL) increases when data heterogeneity among FL clients enlarges, as malicious and benign information introduced by the poisoned and imbalanced data samples, respectively, become marginal for the compared methods to distinguish. However, since FedCon measures the consistency of local models based their responses to given stimulus, it can better make the differences, and thus, significantly improve the attack protection performance. E.g., compared to the second best method, an average improvement in ASR is achieved by 92.59%, 99.11%, and 96.67% in Non-IID ($\alpha = 0.5$) cases of MNIST, GTSRB and SFDDD, respectively.

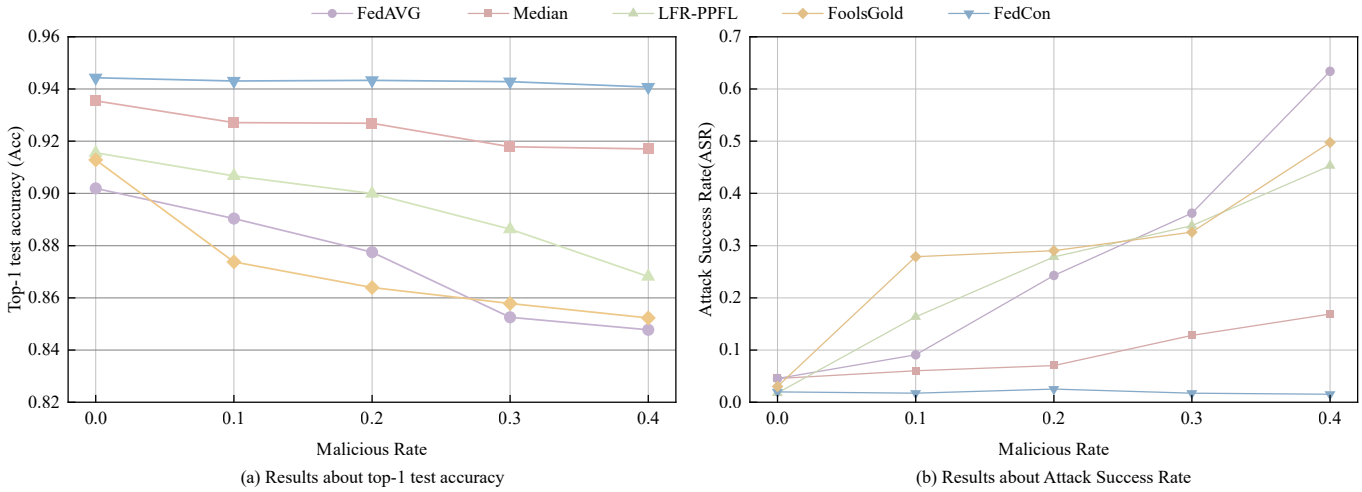
D. Evaluation on method robustness

In the above section, the ASR is measured by using a single pair of flipping from the source to target classes, and to make a more comprehensive evaluation revealing the robustness of the proposed method, each flipping pair of source-target classes in GTSRB is prepared under the heterogeneous setting, i.e., Non-IID ($\alpha = 0.5$), which represent a relatively complex scenario to make the experiment more robust and reliable. As shown in



(a) FedAVG (b) FedCon

Fig. 6. The performance of the baseline FedAVG and the proposed method FedCon against different label flipping pairs under Non-IID ($\alpha = 0.5$) setting of GTSRB. Note that each cell of the heatmap represents the value of ASR (%).



(a) Results about top-1 test accuracy (b) Results about Attack Success Rate

Fig. 7. The performance of FedCon and baselines with variant proportions of malicious clients in GTSRB under the setting of Non-IID ($\alpha = 0.5$).

Fig. 6, the baseline method FedAvg suffers severely due to the damage introduced by each flipping pair with a high ASR. In the worst cases, its ASR reaches up to about 86% indicating that the malicious knowledge can be easily injected, making the training less reliable with either fatal failure in model training or security threats in model usage. However, under the same attack setting, through the measurement of model consistency, FedCon can remain ASR under 0.95% on average for all kinds of label flipping attacks. Such a result indicates that FedCon is resistible against data poisoning attacks in avoiding related hazard to secure the learning in a vulnerable and heterogeneous IoT environments.

Moreover, another factor effecting the success of attacks to test the robustness of the method is the rate of malicious clients. Under the assumption that no more than 40% of clients are malicious, additional experiments are executed to test the model accuracy together with ASR under malicious rate from 0 to 40% by using Non-IID ($\alpha = 0.5$) setting of GTSRB. Specifically, as shown in Fig. 7 (a), the model accuracy of compared methods (i.e., FedAvg, FoolsGold, Median, and LFR-PPFL) declines dramatically with the increase of malicious client rate. However, FedCon can stabilize the performance of trained model by keeping a straight and highest accuracy curve. Moreover, besides the merit in ensuring the model accuracy under different malicious client rates, FedCon can also keep the lowest ASR as illustrated in Fig. 7 (b), in which, the ASR curves of compared methods grows significantly along with the increase of malicious client rate. Such results not only vividly illustrate the damage introduced by the data poisoning attacks in disturbing the model training process but also straightforwardly reveal the efficiency and effectiveness of FedCon in preventing the damage of these attacks launched by different proportions of malicious clients.

E. Discussion

1) Convergence Analysis of FedCon: Compared to traditional methods, the main component of FedCon affecting the model convergence is the model aggregation process implemented based on MVM. As recent research [54] has proven the convergence of FedAVG even on Non-IID data, to facilitate the convergence analysis of FedCon, we make following assumptions:

- The local loss function $F_k(w)$ for each client k is bounded below, and its gradient is also bounded by B_k.

$$\min_w F_k(w) > -\infty \quad \|\nabla F_k(w)\| \leq B_k \tag{16}$$

- The loss function $F_k(w)$ meets the L_k-smoothness and μ_k-Lipschitz continuity conditions.

$$\begin{aligned} \|\nabla F_k(w_2) - \nabla F_k(w_1)\| &\leq L_k \|w_2 - w_1\| \\ \|\nabla^2 F_k(w_2) - \nabla^2 F_k(w_1)\| &\leq \mu_k \|w_2 - w_1\| \end{aligned} \tag{17}$$

- The stochastic gradient computed with a sample from D_k is assumed to have bounded variance:

$$\begin{aligned} \|\nabla F_k(w, D_k) - \nabla F_k(w)\| &\leq \sigma_g \\ \|\nabla^2 F_k(w, D_k) - \nabla^2 F_k(w)\| &\leq \sigma_h \end{aligned} \tag{18}$$

  where σ_g and σ_h are the upper bound of the variance of the first-order and second-order stochastic gradients, respectively.

Based on these assumptions, we can analyze the convergence of FedCon. Since local models trend to converge to their respective local minima, by aggregating benign updates w_k^t, the global model w^t will gradually converge to a global optimum as well, and its convergence rate can be bounded as:

$$\begin{aligned} &\frac{1}{R}\sum_{t=0}^{R-1} \mathbb{E}\left[\|\nabla F(w^t)\|^2\right] \leq \\ &\frac{4(F(w_0) - F^*)}{\beta R} + \left(\frac{\sigma_g^2}{D_{avg}} + \frac{\sigma_h^2}{D_{avg}}\right) \end{aligned} \tag{19}$$

where R denotes the total training rounds, $\mathbb{E}$ represents the expected squared gradient norm at round t, F^* is the optimal value of the global loss function, β is a constant related to the step size and learning rate in the optimization process, D_{avg} is the average number of data samples across all clients.

Therefore, in theory, the model learned through FedCon is expected to converge to the global minimum F^* within R rounds. The convergence of FedCon is further supported by the experimental results in Section V-C.

2) Statistical Justification of FedCon: FedCon utilizes Pearson Correlation Coefficient (PCC) to calculate Model Consistency Score (MCS), which accesses the similarity between two models by measuring their linear relationship in response to a same set of stimuli as defined in Formula 6. Therefore, inheriting from PCC, a high MCS value (close to 1) means that the two models exhibit consistent behaviors in processing same inputs. In contrast, a low MCS value (close to 0) suggests a behavior divergence. Moreover, for a given local model, in each learning round, FedCon will compare it with rest of the local models received at the server, and accordingly, by measuring the inconsistency within these MCSs, FedCon can intuitively and easily identify models with controversial scores, and exclude them from the model aggregation process to stabilize the overall learning process.

In general, by using PCC to calculate MCS, FedCon can gain following benefits. First, unlike gradient-based methods, e.g., Krum [25] and Trimmed-Mean [24], FedCon can efficiently distinguish the variations introduced by heterogeneous data and manipulated samples, as the model trained by using malicious data will present different responses to a same set of stimuli compared to benign models. Moreover, instead of measuring the gradient similarity (in which, few information can be used to calculate malicious indicators), FedCon can configure the size of stimuli to obtain sufficient information to reveal the behavioral difference between local models, and accordingly, improve the robustness of the learning process. Finally, as an additional but important feature, through the measurement of model consistency at the server, FedCon is compatible with existing FL methods that can ease the configuration at client side to support a relatively simple task-loading processing and improve the scalability of FedCon.

In summary, compared with state-of-the-art methods, FedCon can improve the performance of FL in not only trustworthy but also malicious running environments without introducing drawbacks on model performance and training speed. Moreover, under more complex and severe attack situations, i.e., high data heterogeneity level and malicious client rate, differing from compared methods, FedCon can consistently maintain a stable and robust model performance as well as a low and rare ASR, which illustrates its supremacy in identifying malicious models based on their response consistency in processing given stimulus to secure the collaborative and privacy-preserving model learning process.

VI. Conclusion

This paper introduces FedCon that leverages model consistency to remedy the hazard caused by data poisoning attacks launched by malicious clients in FL. Specifically, first, based on same sets of stimulus, FedCon calculates the model consistency between each two local models uploaded by IoT clients, and use it to form the model voting matrix (MVM). Second, by scanning the MVM row by row, FedCon implements a voting processing to identify malicious models, and accordingly, update the global model based on benign models that pass the voting. In general, FedCon presents a novel alternative of current attack defendse methods by easing the requirement on sharing local gradients, and also relieving the inability in handling heterogeneous data, to make the protection compatible with general FL process that exchanges local models between the clients and server, and supports IoT clients with heterogeneous data.

To reveal the supremacy of the proposed method, FedCon is evaluated against five state-of-the-art baselines (i.e., FedAVG, Median, PEFL, LFR-PPFL, and FoolsGold) based on three standard datasets (i.e., MNIST, GTSRB, and SFDDD). The results show that without introducing additional drawbacks to support the general FL process, FedCon can achieve the highest model accuracy compared to the second best method with average improvements of 1.91%, 3.44%, and 3.88% in MNIST, GTSRB and SFDDD, respectively. In the meanwhile, under the most complicated Non-IID settings, FedCon can also remain the lowest attack success rate, which, compared to the second best method, reduces about 92.59%, 99.11%, and 96.6% for MNIST, GTSRB and SFDDD, respectively. Finally, regardless the increase of malicious client rate (assuming that

it does not exceed 40%), FedCon can successfully eliminate the impact introduced the attacks to protect the collaborative and privacy-preserving model learning process in vulnerable and heterogeneous IoT environments.

Aiming at improving the adaptivity of FedCon for ubiquitous IoT, it is worthwhile to expand the current study in three aspects. First, besides label flipping attacks, the capability of FedCon in defending other types of attacks (e.g., backdoor attacks) can be further examined and enhanced to better support IoT systems running widely in vulnerable and untrustworthy Internet. Second, since current model consistency only considers the difference of local updates within one learning round, factors measuring temporal changes among different learning rounds can be studied and used to better identify malicious ones that launch attacks with different frequency (e.g., causally or continuously), and stabilize the overall model learning by mitigating the fluctuations introduced by local models. Moreover, investigating the integration of adaptive thresholding techniques in adjusting the MCS decision boundary will be an important direction to further refine the ability of FedCon to accommodate varying attack intensities and dynamic learning conditions. Finally, to support the application of large-scale AI models, related security issues can be identified and addressed through FedCon to secure model training and deployment processes in IoT systems.

REFERENCES

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, 2014.

[2] T. Zhang, L. Gao, C. He, M. Zhang, B. Krishnamachari, and A. S. Avestimehr, "Federated learning for the internet of things: Applications, challenges, and opportunities," *IEEE Internet of Things Magazine*, vol. 5, no. 1, pp. 24–29, 2022.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282.

[4] S. Pandya, G. Srivastava, R. Jhaveri, M. R. Babu, S. Bhattacharya, P. K. R. Maddikunta, S. Mastorakis, M. J. Piran, and T. R. Gadekallu, "Federated learning for smart cities: A comprehensive survey," *Sustainable Energy Technologies and Assessments*, vol. 55, p. 102987, 2023.

[5] S. P. Ramu, P. Boopalan, Q.-V. Pham, P. K. R. Maddikunta, T. Huynh-The, M. Alazab, T. T. Nguyen, and T. R. Gadekallu, "Federated learning enabled digital twins for smart cities: Concepts, recent advances, and future directions," *Sustainable Cities and Society*, vol. 79, p. 103663, 2022.

[6] J. C. Jiang, B. Kantarci, S. Oktug, and T. Soyata, "Federated learning in smart city sensing: Challenges and opportunities," *Sensors*, vol. 20, no. 21, p. 6230, 2020.

[7] L. You, M. Danaf, F. Zhao, J. Guan, C. L. Azevedo, B. Atasoy, and M. Ben-Akiva, "A federated platform enabling a systematic collaboration among devices, data and functions for smart mobility," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4060–4074, 2023.

[8] Z. Guo, L. You, S. Liu, J. He, and B. Zuo, "Icmfed: An incremental and cost-efficient mechanism of federated meta-learning for driver distraction detection," *Mathematics*, vol. 11, no. 8, p. 1867, 2023.

[9] Z. Du, C. Wu, T. Yoshinaga, K.-L. A. Yau, Y. Ji, and J. Li, "Federated learning for vehicular internet of things: Recent advances and open issues," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 45–61, 2020.

[10] L. You, M. Hao, J. Sun, Y. Wang, C. Rong, C. Yuen, P. Santi, and C. Ratti, "Toward a personalized autonomous transportation system: Vision, challenges, and solutions," *The Innovation*, vol. 5, no. 6, 2024.

[11] S. Fukumoto, R. Litextsuperscript, K. Zeng, H. Nan, and Z. Su, "Investigations and time estimation on federated learning for future internet of vehicles," *IEEE Internet of Things Journal*, pp. 1–1, 2025.

[12] H. Elayan, M. Aloqaily, and M. Guizani, "Sustainability of healthcare data analysis iot-based systems using deep federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7338–7346, 2022.

[13] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–37, 2022.

[14] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, pp. 1–19, 2021.

[15] A. Rauniyar, D. H. Hagos, D. Jha, J. E. Håkegård, U. Bagci, D. B. Rawat, and V. Vlassov, "Federated learning for medical applications: A taxonomy, current trends, challenges, and future research directions," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 7374–7398, 2024.

[16] L. You, Z. Guo, C. Yuen, C. Y.-C. Chen, Y. Zhang, and H. V. Poor, "A framework reforming personalized internet of things by federated meta-learning," *Nature Communications*, vol. 16, no. 1, p. 3739, 2025.

[17] X. Pei, X. Deng, S. Tian, L. Zhang, and K. Xue, "A knowledge transfer-based semi-supervised federated learning for iot malware detection," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 3, pp. 2127–2143, 2023.

[18] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. H. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1364–1381, 2022.

[19] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, and J. Liu, "Data poisoning attacks on federated machine learning," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 11 365–11 375, 2022.

[20] T. Krauß, J. König, A. Dmitrienko, and C. Kanzow, "Automatic adversarial adaption for stealthy poisoning attacks in federated learning," in *Network and Distributed System Security Symposium (NDSS)*, 2024.

[21] G. Chen, X. Li, L. You, A. M. Abdelmoniem, Y. Zhang, and C. Yuen, "A data poisoning resistible and privacy protection federated learning mechanism for ubiquitous iot," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[22] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*. Springer, 2020, pp. 480–501.

[23] D. Cao, S. Chang, Z. Lin, G. Liu, and D. Sun, "Understanding distributed poisoning attack in federated learning," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2019, pp. 233–239.

[24] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[25] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[26] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[27] X. Liu, H. Li, G. Xu, Z. Chen, X. Huang, and R. Lu, "Privacy-enhanced federated learning against poisoning adversaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4574–4588, 2021.

[28] S. Shen, S. Tople, and P. Saxena, "Auror: Defending against poisoning attacks in collaborative deep learning systems," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, 2016, pp. 508–519.

[29] D. Li, W. E. Wong, W. Wang, Y. Yao, and M. Chau, "Detection and mitigation of label-flipping attacks in federated learning systems with kpca and k-means," in *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2021, pp. 551–559.

[30] Z. Lv, H. Cao, F. Zhang, Y. Ren, B. Wang, C. Chen, N. Li, H. Chang, and W. Wang, "AWFC: Preventing Label Flipping Attacks Towards Federated Learning for Intelligent IoT," *The Computer Journal*, vol. 65, no. 11, pp. 2849–2859, 10 2022. [Online]. Available: https://doi.org/10.1093/comjnl/bxac124

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

[32] X. Shen, Y. Liu, F. Li, and C. Li, "Privacy-preserving federated learning against label-flipping attacks on non-iid data," *IEEE Internet of Things Journal*, 2023.
[33] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings," in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 301–316.
[34] N. Kriegeskorte, M. Mur, and P. A. Bandettini, "Representational similarity analysis-connecting the branches of systems neuroscience," *Frontiers in systems neuroscience*, p. 4, 2008.
[35] J. Mehrer, C. J. Spoerer, N. Kriegeskorte, and T. C. Kietzmann, "Individual differences among deep neural network models," *Nature communications*, vol. 11, no. 1, p. 5725, 2020.
[36] S. Liu, Q. Chen, and L. You, "Fed2a: Federated learning mechanism in asynchronous and adaptive modes," *Electronics*, vol. 11, no. 9, p. 1393, 2022.
[37] G. Chen, K. Li, A. M. Abdelmoniem, and L. You, "Exploring representational similarity analysis to protect federated learning from data poisoning," in *Companion Proceedings of the ACM Web Conference 2024*, ser. WWW '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 525–528. [Online]. Available: https://doi.org/10.1145/3589335.3651503
[38] F. A. Yerlikaya and Ş. Bahtiyar, "Data poisoning attacks against machine learning algorithms," *Expert Systems with Applications*, vol. 208, p. 118101, 2022.
[39] H. Xiao, H. Xiao, and C. Eckert, "Adversarial label flips attack on support vector machines," in *ECAI 2012*. IOS Press, 2012, pp. 870–875.
[40] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.
[41] P. M. Sánchez Sánchez, A. H. Celdrán, T. Schenk, A. L. B. Iten, G. Bovet, G. M. Pérez, and B. Stiller, "Studying the robustness of anti-adversarial federated learning models detecting cyberattacks in iot spectrum sensors," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–12, 2022.
[42] M. Zeiler, "Visualizing and understanding convolutional networks," in *European conference on computer vision/arXiv*, vol. 1311, 2014.
[43] J. Jiang, S. Ji, and G. Long, "Decentralized knowledge acquisition for mobile internet applications," *World Wide Web*, vol. 23, no. 5, pp. 2653–2669, 2020.
[44] L. You, S. Liu, Y. Chang, and C. Yuen, "A triple-step asynchronous federated learning mechanism for client activation, interaction optimization, and aggregation enhancement," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 199–24 211, 2022.
[45] S. Chen, C. Shen, L. Zhang, and Y. Tang, "Dynamic aggregation for heterogeneous quantization in federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6804–6819, 2021.
[46] L. You, Z. Guo, B. Zuo, Y. Chang, and C. Yuen, "Slmfed: A stage-based and layerwise mechanism for incremental federated learning to assist dynamic and ubiquitous iot," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16 364–16 381, 2024.
[47] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.
[48] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[49] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.
[50] A. Montoya, D. Holman, SF_data_science, T. Smith, and W. Kan, "State farm distracted driver detection," https://www.kaggle.com/c/state-farm-distracted-driver-detection, Feb. 2016, google.
[51] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 7252–7261. [Online]. Available: https://proceedings.mlr.press/v97/yurochkin19a.html
[52] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 965–978.
[53] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 10 713–10 722.
[54] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

Xiaoyi Li receives his B.Eng in the School of Intelligent Systems Engineering, Sun Yat-Sen University, China, in 2024, where he is now pursuing the master's degree. His research interests include federated learning, privacy security, ethical considerations in large models, machine learning, and their applications in the Internet of Things (IoT).

Xuewei Tao received his B.S. and M.S. degrees in China University of Mining and Technology, Xuzhou, China, in 2009 and 2014, respectively. He is currently working toward the Ph.D. degree with China University of Mining and Technology, Xuzhou, China. His current research interests include machine Learning, and their applications in Internet of Things and Intelligent Systems.

Gengxiang Chen receives his B.Eng in the School of Intelligent Systems Engineering, Sun Yat-Sen University, China, in 2022, where he is now pursuing the master's degree. His research interests include Federated Learning, Data and Privacy Security, Machine Learning, and their applications in Internet of Things and Intelligent Transportation Systems.

Linlin You is an associate professor at the School of Intelligent Systems Engineering, Sun Yat-sen University, and also a research affiliate at the Intelligent Transportation System Lab, Massachusetts Institute of Technology. He was a senior postdoc at the Singapore-MIT Alliance for Research and Technology and a research fellow at the Architecture and Sustainable Design Pillar of Singapore University of Technology and Design. He received his Ph.D. in Computer Science from the University of Pavia in 2015. He published more than 100 journal and conference papers in the research fields of Smart Cities, Multi-source Data Fusion, Machine Learning, and Federated Learning. He is an Associate Editor of Springer Nature Computer Science, Editorial Board Member of Scientific Reports, and Youth Editor of The Innovation (Cell Press).

Yongzheng Sun is a Professor at the School of Mathematics, China University of Mining and Technology. He received his B.S. and M.S. degrees in applied mathematics from Jiangsu Normal University, Xuzhou, China, in 2001 and 2004, respectively, and the Ph.D. degree in applied mathematics from Fudan University, Shanghai, China, in 2010. He Published more than 70 journal papers in the research fields of complex networks, machine Learning, and coordinated control of multi-agent systems. He is an Editorial Board Members of PLOS Complex Systems.