Data Quality Detection Mechanism Against Label Flipping Attacks in Federated Learning

Yifeng Jiang, Weiwen Zhang, *Member, IEEE*, and Yanxi Chen

数据集都比较简单:cifar-10,fashion-mnist,mnist

***Abstract*— Federated learning (FL) is an emerging framework that enables massive clients (e.g., mobile devices or enterprises) to collaboratively construct a global model without sharing their local data. However, due to the lack of direct access to clients' data, the global model is vulnerable to be attacked by malicious clients with their poisoned data. Many strategies have been proposed to mitigate the threat of label flipping attacks, but they either require considerable computational overhead, or lack robustness, and some even cause privacy concerns. In this paper, we propose Malicious Clients Detection Federated Learning (MCDFL) to defense against the label flipping attack. It can identify malicious clients by recovering a distribution over a latent feature space to detect the data quality of each client. We demonstrate the effectiveness of our proposed strategy on two benchmark datasets, i.e., CIFAR-10 and Fashion-MNIST, by considering different neural network models and different attack scenarios. The results show that, our solution is robust to detect malicious clients without excessive costs under various conditions, where the proportion of malicious clients is in the range of 5% and 40%.**

检测恶意客户端

***Index Terms*— Federated learning, deep learning, label flipping, data poisoning.**

I. Introduction

FEDERATED Learning (FL) has been touted as a promising strategy of machine learning with the concern of data privacy [1], [2], [3]. Machine learning is widely used in many applications, such as face recognition [4] and fingerprint liveness detection [5]. As the rise of data privacy concern, model training is conducted under the paradigm of FL. Specifically, without a centralized data manager that collects and validates datasets, FL allows the data to be kept locally on clients and introduces a central server to coordinate with the clients and learn a global model. Such a paradigm is optimized with locally-computed updates from each client [6], [7].

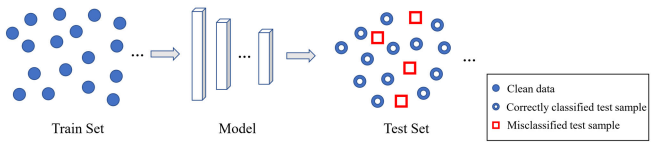
Manuscript received 13 April 2022; revised 19 January 2023; accepted 10 February 2023. Date of publication 27 February 2023; date of current version 2 March 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62002071, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515011577, in part by the Guangzhou Basic Research Project under Grant 202102020420, in part by the Top Youth Talent Project of Zhujiang Talent Program under Grant 2019QN01X516, and in part by the Guangdong Provincial Key Laboratory of Cyber-Physical System under Grant 2020B1212060069. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Valeria Loscri. *(Corresponding author: Weiwen Zhang.)*

The authors are with the School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China (e-mail: jiangyifeng@mail2.gdut.edu.cn; zhangww@gdut.edu.cn; chenyanxi@mail2.gdut.edu.cn).

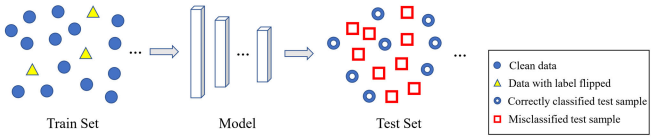
Digital Object Identifier 10.1109/TIFS.2023.3249568

Although the privacy concern is addressed, FL is vulnerable to poisoning attacks. The poisoning attacks can be divided into two categories: model poisoning and data poisoning [8]. Model poisoning attempts to influence learning through creating and sending adversarial gradients to the central coordinator. Data poisoning adds poisoned instances or exploit misclassification to introduce dirty data instead of affecting local training. In the above two attack strategies, the latter is easier to implement, which is powerful and secluded [9], [10]. Specifically, under the privacy protection mechanism of FL, the data of each participant is invisible such that the central server cannot thoroughly inspect the local training data. In other words, the federated model can only learn the corresponding knowledge from the data that the participants have, no matter whether that data is clean or poisoned. Implementers of data poisoning take advantages of such insight by injecting carefully crafted adversarial samples into the training data with the aim of misclassification during testing. The consequences of such misclassification may be severe, especially in some sensitive areas such as medical diagnosis, where a wrong medical diagnosis can lead to a loss of user trust and even life-threatening consequences [11].

Unfortunately, traditional defense strategies against data poisoning attacks in centralized ML settings are not feasible in FL due to the privacy protection mechanism. For example, anomaly detection [12] and data augmentations [13] are not feasible defense strategies in FL settings as they require the access to users' local data. As one of the data poisoning attacks, label flipping attack has attracted much attention because of its simple implementation and remarkable effect. Label flipping attack [14] is a typical application of data poisoning attacks in centralized ML settings, which could be also implemented in federated settings with significant effects [15]. Specifically, the attackers can undermine the training model via modifying the labels of an unnoticeable fraction of training data. This attack could be carried out by ordinary malicious participants without the prior knowledge of the FL system, e.g., FL process, model type and parameters. An example of label flipping attacks is shown in Fig. 1. It illustrates that even if a small fraction of sample labels are flipped, the number of misclassified test samples will also significantly increase. It is claimed that label flipping attacks could significantly increase the misclassification rate of the deep learning model with only around 50 poisoning samples, even up to 90% [16]. As a result, the global model



(a) Train on clean dataset



(b) Train on dataset with label flipping attack

Fig. 1. An example of label-flipping attack. (a) A toy example that illustrates the normal process of model training and its testing result. (b) A toy example that illustrates how label-flipping attack work: attacker secretly tampers with several labels, aiming for maximizing the misclassification of test set via misleading the model training.

is vulnerable to be attacked by malicious clients. To improve the robustness of real-world federated learning systems, it is imperative to investigate defense strategies for FL in the case of data poisoning.

Many efforts have been proposed to tackle data poisoning attacks in FL. They are mainly from three perspectives. The first perspective considers the impact of data on model accuracy [17], [18], which requires retraining or constructing additional dataset. The second perspective aims to detect malicious clients by the difference between benign and malicious model output layer's weights [8], [19]. However, these approaches require explicit knowledge of the potentially vulnerable source class or try to find it by a brute-force search. The third perspective aims to detect malicious clients with the assist of some prior knowledge [20], [21], [22]. But such knowledge is often inaccessible or difficult to estimate in privacy-concerned FL settings. Although most existing work have shown effectiveness in defending against data poisoning attacks, they are still not robust enough on handling label flipping attacks, with either considerable cost or less practicality. Therefore, it is still a challenge to tackle with the label flipping attacks in FL [23].

Due to the limitations of prior art, in this paper, we focus on robust and lightweight defence against label flipping attacks in FL, without prior knowledge and considerable overhead. Specifically, we propose a novel defense strategy called MCDFL via detecting training data quality of every client by means of a generative network. The generative network is derived only from the predictor module θ^p of global model, which allows it to be trained only on the server side. Given label sequence as input, it can generate feature representations, which are compatible with the predictions of the global model. Moreover, with a latent space that have much smaller dimensions compared with the input space, the generator would be lightweight. Thus, MCDFL is promising to achieve malicious detection with less overhead. The learned generator are subsequently used for privacy-ensured data quality detection of every active participant. Depending on the data quality, the server could distinguish between benign and malicious clients under restricted conditions, including inaccessible local data, limited computing resources, limited storage resources.

We perform extensive experiments on two benchmark datasets, i.e., CIFAR-10 and Fashion-MNIST. The experiments show that, when the proportion of malicious clients is in the range of 5% and 40%, the proposed MCDFL is robust to identify them. Moreover, the training accuracy of FedAvg with MCDFL applied is around 98% on Fashion-MNIST and even up to 100% on CIFAR-10, which is 20%-points or even 30%-points more than pure FedAvg. This improvement indicates that the proposed MCDFL can accurately extract the knowledge of local training data from each client. Therefore, the proposed MCDFL can robustly identify malicious clients and allow every benign participant to cooperate in the construction of the global model.

The contributions of this work are summarized as follows.

- We propose MCDFL against label flipping attacks in FL, which only requires to train a lightweight generator on the server for data quality detection. The detection requires neither retraining nor local update information, and thus can get rid of the corresponding heavy computational costs and privacy concerns.
- Contrary to certain prior works that need prior knowledge, the proposed MCDFL enables straightforward malicious detection.
- With data quality detection, the proposed MCDFL is robust to defend against label flipping attacks under various conditions, where the proportion of malicious clients is in the range of 5% and 40%.
- The generator could impose an inductive bias to regulate local training, which is able to accelerate model convergence and lead to better generalization performance.

The following is the organization of the rest of this paper. In Section II, we discuss the related work. In Section III, we give some notations and preliminaries before introducing our method. In Section IV, we formally introduce our defense strategy. In Section V, we show the limitation of existing detection strategy and conduct experiments to evaluate our method on two popular image classification tasks. The conclusion is presented in Section VI.

II. Related Work

Some strategies have been proposed for mitigating data poisoning attacks, but they require either significant computational costs, or some difficult-to-obtain prior knowledge. The strategy considering the impact of data on model accuracy can be divided into passive strategy and active strategy. The passive strategy requires retraining on client's local data. For example, Tramèr et al. [17] determined whether a sample was clean by measuring its incremental effect to the model performance when it participated in the training. Such a strategy is effective in the detection of some poisoning attacks, but it requires significant computational overhead for retraining and evaluating the performance of the model for each training sample, which also limits its application to large datasets. On the other hand, active strategies require additional datasets to train classification models. For example, Doku and Rawat [18] collected additional data to construct a prototype dataset homogeneous with clients, and trained a support vector

machine to detect malicious clients. However, it is not always easy to construct such dateset, even though there is a large amount of data on the internet while most of them are noisy.

Some strategies leverage clients gradient information for malicious detection. Tolpegin et al. [8] used clustering and PCA based strategy to detect the differences between benign and malicious model output layer's weights. However, these approaches also require considerable computational cost to find the potentially vulnerable source class by a brute-force search. Its detection results also lack robustness, especially when the proportion of malicious clients is higher than 20% (c.f. Fig. 3 in Section V). Li et al. [19] extended Tolpegin's study, using clustering and KPCA based strategy. However, it still suffers from the aforementioned computational overhead and robustness issues. Moreover, both of them need to transfer model updates, which may cause privacy leakage as the semi-honest server is still capable to recover some sensitive information of users through the received updates [24]. Liu et al. [25] also detected malicious clients via the similarity of local updates, and adopted homomorphic encryption as the underlying technology to avoid privacy leakage from updates. However, it still suffers from considerable computational costs. On the contrary, our proposed method is able to detect malicious clients without the update information, which requires no significant additional computational costs.

Prior knowledge is leveraged in some works for handling data poisoning attacks. For example, the authors in [20], and [21] required to assume the proportion of attackers to defense against poisoning attacks. Jebreel et al. [26] assumed knowledge on the clients' label distribution and adopted different strategies for each situation. However, such prior knowledge is not easy to obtain or estimate in practice. In contrast, our proposed method can perform effective recognition without any prior knowledge.

In addition, authors in [22], [27], and [28] adopted outlier detection. Specifically, they aimed to remove poisoned data points by examining clients' local dataset, or remove maliciously relevant neurons/weights from the infected models by taking full control of server's training process, which are inappropriate in federated learning settings. Qayyum et al. [29] proposed a hybrid learning-based strategy to detect malicious clients by parameter updates. However, it required the assumption that no malicious clients participated in the previous rounds of FL system. Ma et al. [30] proposed a privacy guaranteed FL framework for medical diagnosis based on the trimmed optimization with multi-key computation to defend against data poisoning attacks. However, that method is not robust enough as it requires manual selection of FL hyperparameters rather than automated tuning in adversarial settings, which is not required in our proposed method.

III. Notations and Preliminaries

In this section, we introduce notation definitions throughout the paper and the general purpose of federated learning, as well as assumptions about the capabilities of the attacker.

A. Notation Definitions

We present the notations as follows. Let $\mathcal{X} \subset R^p$ be an instance space, and $\mathcal{Z} \subset R^d$ ($\mathcal{Z} = \{\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_n\}$) be a latent feature space with $d<p$, where n is the total number of label types, and $\mathcal{Z}_j$ is the latent feature space of the j-th label, $\mathcal{Y} \subset R$ be an output space. The model θ we use is divided into two components, including a feature extractor θ^f and a predictor module θ^p, i.e., $\theta := [\theta^f; \theta^p]$. Specifically, the feature extractor θ^f yields feature representations $\mathcal{Z}$ based on input instance $\mathcal{X}$: $\mathcal{X} \to \mathcal{Z}$, and the predictor module θ^p yields y' based on feature representations $\mathcal{Z}$: $\mathcal{Z} \to y'$, where y' is the simplex over $\mathcal{Y}$. With $l : y' \times \mathcal{Y} \to R$ as a convex loss function, the risk of model θ can be defined as $L(\theta) := \mathbb{E}_{x \in D}[l(h(f(x; \theta^f); \theta^p), \mathcal{M}^*(x))]$, where $\mathbb{E}_{x \in D}$ indicates that x is sampled from data distribution D, h is a predictor, $\mathcal{M}^*$ is a ground-truth labeling function, i.e., $\mathcal{M}^*(x)$ is the label corresponding to x in user's local data.

The general purpose of federated learning is to train a global model parameterized by θ and minimize its risk on the training data of each user [7]:

$$\min_{\theta} \sum_{i=1}^{N} a_i L(\theta) \tag{1}$$

where N is the total number of clients, a_i is the aggregated weight of client i, $L(\theta) = \frac{1}{|D|}\mathbb{E}_{x \in D}(L_\theta(x))$ is the expected risk of a model parameterized by θ on data distribution D. Practically, the empirical risk of client i is $L(\theta_i) := \frac{1}{|D_i|}\mathbb{E}_{x \in D_i}[l(h(f(x; \theta_i^f); \theta_i^p), \mathcal{M}^*(x))]$.

B. Attacker's Capability

We assume that the central server is honest while a part of FL clients are malicious. We also assume that each malicious client can only manipulate its own training data, and it cannot access or manipulate its or other clients' learning process, such as detection process, loss function calculation, or the server's aggregation process.

An attacker can flip any label by recruiting more malicious clients to counteract the effect of benign clients. For an attack to be successful, the attacker's influence on the target label must be greater than the influence from benign clients. Since the global model is compromised between benign and malicious clients in early training rounds, the early selection of clients resembles a randomly selected state. If the number of malicious clients is too close to, or even exceeds the number of benign clients, malicious clients may outweigh the influence of benign clients, which will make the detection strategy mistake tampered labels as real ones. Thus, for the sake of testing the robustness within a reasonable range, we expect the strategy can defend against a significant fraction of malicious clients, from 5% to 40%.

IV. MCDFL: Malicious Clients Detection Federated Learning

In this section, we formally introduce our proposed detection strategy MCDFL. Figure 2 illustrates an overview of the detection process. We design a generator to learn the

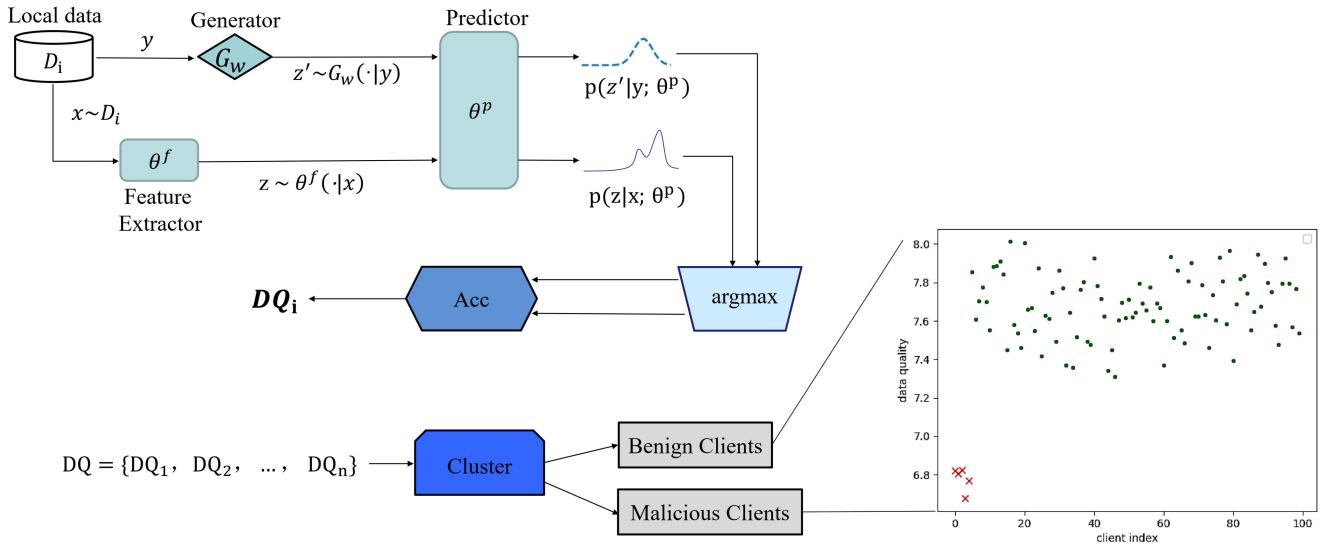


Fig. 2. Overview of MCDFL: a generator $G_w(\cdot|y)$ is learned by the server to extract latent features without observing each client's local data. The generator is then sent to active users for the detection of data quality and to benign clients for adjusting the local training.

knowledge of global model over the latent feature space, which is the assemble of multiple user models. The latent space is used as a criterion to detect whether the features of the samples correspond to the labels. Such a feature extraction allows us to obtain the quality of clients' local data, and clients with poor data quality will be considered as malicious clients. The latent feature can also be used as an inductive bias to regulate the training of the local model, which is able to accelerate model convergence and lead to better generalization performance.

A. Latent Feature Extraction

Our core idea is trying to detect data quality via recovering an aggregated distribution of benign clients' input space, which could be achieved via extracting knowledge from benign clients. However, such knowledge is non-observable in traditional FL. One approach is to characterize the knowledge that is consistent with the ground-truth data distributions of benign clients via learning a conditional distribution $Q^* : y \to x$ [31]:

$$Q^* = \underset{Q:y\to x}{argmax}\ \mathbb{E}_{y\sim p(y)}\mathbb{E}_{x\sim Q(x|y)}[\log p(y|x)] \tag{2}$$

in which there are two unknown distributions, $p(y)$ and $p(y|x)$, which are the prior and posterior distributions of users' local labels, respectively. In order to learn such distributions without requiring any prior knowledge, one approach is to utilize their empirical approximations rather than the ground-truth prior and posterior distributions. Specifically, $p(y)$ can be estimated as:

$$\hat{p}(y) \propto \sum_{b\in B} \mathbb{E}_{x\sim D_b}[I(\mathcal{M}^*(x) = y)] \tag{3}$$

where $I(\cdot)$ is an indicator function. B is the set of benign users. And $p(y|x)$ can be approximated by exploiting the ensemble knowledge from local models of benign clients:

$$log\,\hat{p}(y|x) \propto \frac{1}{|B|}\sum_{b\in B} \log p(y|x;\theta_b) \tag{4}$$

It seems that, with the above approximations, Eq. (2) can be directly optimized. However, the input space $\mathcal{X}$ may be of high dimension, which will bring computation overloads, and may also cause leakage of information about the user data profile. Therefore, such a strategy will get stuck in the same predicament as the previous work. A promising alternative solution is to recover a distribution, which is more compact than the raw instance space, and can thus protect local data privacy. Specifically, it can be a distribution $G^* : y \to z$ over a latent space:

$$G^* = \underset{G:\mathcal{Y}\to z}{argmax}\ \mathbb{E}_{y\sim\hat{p}(y)}\mathbb{E}_{z\sim G(z|y)}[\sum_{b\in B}\log p(y|z;\theta_b^p)] \tag{5}$$

However, since the set of benign clients is unknown, it also becomes impractical to utilize Eq. (5) to recover the ground-truth distribution. A more approachable idea is that, instead of learning based on local models with users' data, we extract the knowledge from global model with random label sequences. It is worth noting that, the global model is compromised between benign and malicious clients in early training rounds. Fortunately, with our data quality detection strategy, which is formulated later, the malicious clients will be gradually identified and the global model will finally get rid of the malicious clients to learn only from the clean data of the benign clients. Thus, the objective is changed as:

$$G^* = \underset{G:\hat{y}\to z}{argmax}\ \mathbb{E}_{\hat{y}\sim R}\mathbb{E}_{z\sim G(z|\hat{y})}[\log p(y|z;\theta^p)] \tag{6}$$

R is a function that generates $\hat{y}$, a random sequence of labels, which is considered as training samples.

Following the above reasoning, in order to extract the mapping of labels to latent feature space without causing data leakage, we propose a novel framework, which is based on a conditional generator G parameterized by ω and a global model parameterized by $\theta := [\theta^f; \theta^p]$. We aim to optimize the following objective:

$$\min_{\omega} J(\omega) := \mathbb{E}_{\hat{y}\sim R}\mathbb{E}_{z\sim G_\omega(z|\hat{y})}[l(h(z;\theta^p),\hat{y})] \tag{7}$$

Given arbitrary sequence of labels $\hat{y}$, the generator can yield feature representations $z \sim G_\omega(\cdot|\hat{y})$, which is consistent with the knowledge of global model, a global view of users' local data.

It is noticed that, given random label sequences, with only the predictor module θ^p of global model, Eq. (7) could be optimized, i.e., generator G can be trained on the server side, without extra burden on the client side. Moreover, as the generator is lightweight and easy to train, it would not incur much extra cost to the current FL system.

B. Data Quality Detection

The learned generator G and global model θ are then broadcasted to each client for data quality detection. The detection process is formulated as follows, which is simple but effective.

First, we define the data quality of user i as DQ_i

$$DQ_i := \frac{1}{|D_i|}\mathbb{E}_{x\sim D_i}\mathbb{E}_{z\sim G_\omega(z|y)}[Acc(argmax\ h(z;\theta^p), argmax\ h(f(x;\theta)))] \tag{8}$$

where $z \sim G_\omega(\cdot|y)$ are latent features corresponding to the labels of user data from a global view via generator G_ω and y is the label sequence of the local training data of user i. The function $ACC(a, b)$ counts the sum of identical elements in sequences a and b. $argmax\ h(z;\theta^p)$ are the predictions based on the latent features z corresponding to the user labels from the predictor module θ^p of global model, which are considered as the real labels. $argmax\ h(f(x;\theta))$ are the predictions based on the samples of client's local data, which are treated as the actual labels of client's local samples.

The detection strategy is based on the following insight: $argmax\ h(z;\theta^p) \neq argmax\ h(f(x;\theta))$ appears in local data where a fraction of labels are tampered, i.e., the quality parameters sent from malicious participants are worse compared to benign participants.

After completing the data quality detection of user i, the quality DQ_i is returned to the server. Subsequently, the server will detect malicious nodes according to these quality parameters $DQ := \{DQ_1, DQ_2, \ldots, DQ_N\}$. Specifically, the server clusters the quality parameters into two categories through K-means clustering [32] and identifies the client category. The clients with lower average quality are considered as the set of malicious nodes, while the other category are considered as the set of benign users.

It is worth noting that unlike [17], [18], [19], [21], [26], the proposed detection strategy does not require prior knowledge, without significant additional computational costs, privacy concern for retraining on users' local data and transmission for local updates. The server can thus effectively and promptly identify malicious participants before each round of training, and consequently restrict their participation in global aggregation. Moreover, the detection strategy can scale well to various large deep learning networks as it does not introduce much additional computational overhead.

C. Personalization for Local Training

In order to accurately recover a distribution over the latent feature space, we focus on the performance of the local model on training data. We adopt "base+personalization" [33] to construct a personalized model for each user rather than using traditional federated learning method that relies on a single global model to train on each participant's local data. The main difference between such personalized design and traditional federated learning is the learning objective. The purpose of traditional federated learning is to strive the global model to approach the ground-truth, while personalized federated learning is to enable each client's local model to approach the ground-truth. Many applications [34], [35], [36] have shown that a personalized model can better fit local training data than a compromised model.

Specifically, we have the design of one single global representation and a number of local heads for the personalization as follows. The user's local model is divided into two components, including base layer θ_i^f from the global model for learning features of local data (i.e., $\theta_i^f = \theta_i$) and personalized layer θ_i^p that keeps locally and outputs prediction results. Such a strategy is based on the intuition that a global feature representation is often shared by the data, while the statistical heterogeneity among the clients is concentrated in the labels. It can thus allow each client to optimize a personalized and low-dimensional classifier that accounts for the unique labeling of its local data [37]. With the personalized design, we can assemble real knowledge of each client over the latent feature space, from which the generator can extract accurate latent features.

Moreover, each user can obtain latent feature representations $z \sim G_\omega(\cdot|y)$ as augmented representations to introduce inductive bias for local training, reinforcing the generalization performance of their local models. Specifically, given $r(z|y) : y \rightarrow z$ is the conditional distribution from the generator, a user model θ_i can be regulated by utilizing features sampled from $r(z|y)$ to perform matching between the distributions derived from local training data and the generator. Therefore, the objective of regulation is to minimize the conditional KL-divergence between these two distributions:

$$\max_{\theta_i}\ \mathbb{E}_{y\sim p(y),z\sim r(z|y)}[\log p(y|z;\theta_i)] \equiv \min_{\theta_i} D_{KL}[r(z|y)||p(z|y;\theta_i)]. \tag{9}$$

where $p(y)$ is the prior distribution of local labels. Assuming y is the output, $p(z|y;\theta_i)$ is the probability that the feature input of θ_i is z. The proof of Eq. (9) can be found in Appendix C.

Thus, the objective of a local model $\theta_i := [\theta_i^f; \theta_i^p]$ turns to minimize the loss of predictions for both augmented samples and local data:

$$\min_{\theta_i}\ J(\theta_i) := L(\theta_i) + \mathbb{E}_{y\sim\hat{p}(y),z\sim G_\omega(z|y)}[l(h(z;\theta_i^p), y)] \tag{10}$$

In practice, Eq. (9) and the second term of a local model objective (Eq. (10)) are both optimized by empirical samples from the generator, which consists of two components:

$\{(z, y)|y \sim \hat{p}(y), z \sim G_\omega(z|y)\}$. Thus, we have:

$$\max_{\theta_i} \mathbb{E}_{y\sim\hat{p}(y),z\sim r(z|y)}[\log p(y|z;\theta_i)] \approx \min_{\theta_i} \mathbb{E}_{y\sim p(y),z\sim G_\omega(z|y)}[l(h(z;\theta_i^p), y)] \quad (11)$$

D. Optional Global Aggregation Strategy

It is noticed that our detection process has no additional requirements for the aggregation strategy, which enables it to be applicable in many existing FL frameworks. In this paper, we take FedAvg [7] as an example for illustration, which is a widely used classic benchmark framework. Specifically, it takes the amount of data per client as the weight to aggregate local model parameters. The objective is defined as follows,

$$\theta \leftarrow \sum_{k=1}^{K} \frac{|D_k|}{|D_K|}\theta_k. \quad (12)$$

where θ is the global model, $|D_K|$ is the total data volume of the selected client set K, $|D_k|$ and θ_k are the data volume and local model of user k, respectively.

Specifically, the global model θ consists of two components, i.e., the feature extraction layer θ^f and the prediction layer θ^p. θ^p is used to guide the training of the generator to recover an aggregated distribution over the feature space. θ^f is used as a shared component for the construction of the local personalized model.

E. The Algorithm of MCDFL

The process of MCDFL is shown in Algorithm 1. First, we initialize the generator ω and global model θ in the central server, which are subsequently broadcasted to each active participant. Then, each participant i computes its data quality DQ_i on local training data by Eq. (8) and sends it back to the server. After receiving all quality parameters, the central server can execute malicious detection based on DQ by K-means algorithm. After that, the server selects K clients participating in current round of training from benign client set B on demand. For each selected participant k, it performs local training using the personalized model constructed with local predictor θ_k^p and global shared component θ^f. θ_k is updated iteratively by Eq. (10). After the completion of local training, the clients will send θ_k back to the server. In the end, the server will perform global aggregation for local models and optimize the generator by Eq. (7).

Algorithm 1 MCDFL. Global Parameter θ, Local Parameters $\{\theta_k\}_{k=1}^K$; Generator Parameter w; Local Updating Step T, Learning Rate α, β

```
server initializes generator ω and global model θ
Repeat:
    server broadcasts ω, θ to all active user set A
    for all user i ∈ A in parallel do
        user i computes DQ_i                    ▷ Execute Eq. (8)
        user i sends DQ_i back to the server
    end for
    server performs K-means on DQ:={DQ_1,...,DQ_|A|}
    to get benign user set B
    server selects K users required for current training
    round from B
    for user k = 1,2...K in parallel do
        θ_k ← [θ^f; θ_k^p]
        for t = 1...T do
            θ_k ← θ_k − β∇_{θ_k} J(θ_k)          ▷ Optimize Eq. (10)
        end for
        user k sends θ_k back to the server
    end for
    server updates θ ← Aggregate(∑_{k=1}^K θ_k)
    ω ← ω − α∇_w J(ω)                           ▷ Optimize Eq. (7)
Until TRAINING STOP:
```

V. Experiments

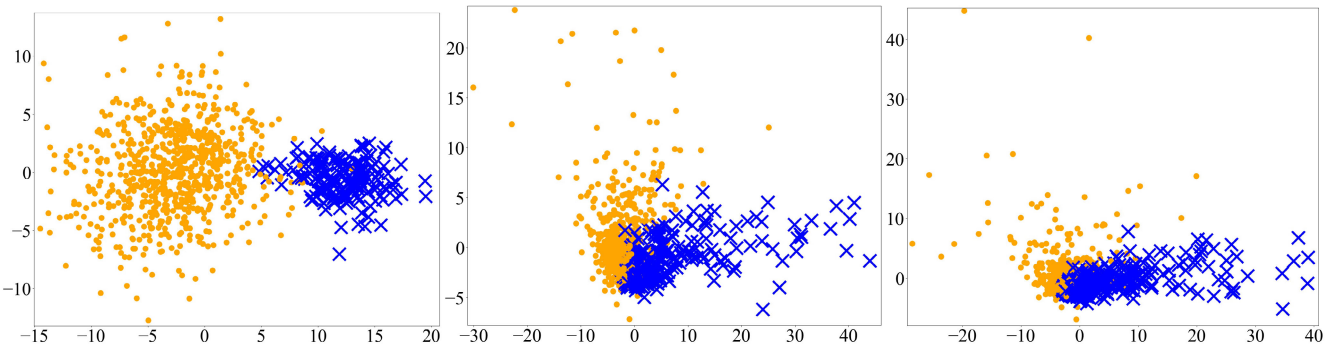
In this section, firstly, we present the setup for the experiments. Then we show the limitation of previous work and evaluate the performance of the proposed detection strategy on two datasets, i.e., Fashion-MNIST [38] and CIFAR-10 [39]. Following that, we compare the performance between pure FedAvg and FedAvg with MCDFL applied in label flipping attacks. At last, we present the execution time and additional storage overhead of MCDFL.

A. Datasets and Neural Networks

Our experiments are performed in an independent and identically distributed (i.i.d.) data distribution, i.e., the whole datasets are randomly assigned to each client, and the testing data is only used for the evaluation of local model.

Fashion-MNIST is a gray-scale image dataset that covers the front images of 70,000 different commodities from 10 categories, including dress, shirt, bag and so on. The complete dataset is divided into two parts: 60,000 images for training and 10,000 images for testing, and both are equally distributed to each client for training and testing, respectively. CIFAR-10 contains a total of 10 categories of color images with 6,000 images included per class, which are real objects in the real world, with not only noisy, but also different proportions and characteristics. The dataset is also divided into two components, 50,000 training images and 10,000 test images evenly distributed to each client for training and testing.

We adopt a convolutional neural network with six convolutional layers in experiments with CIFAR-10 and a convolutional neural network with two convolutional layers for Fashion-MNIST. The test accuracies of the models for CIFAR-10 and Fashion-MNIST in the centralized scenario are 79.90% and 91.87%, respectively [8]. Note that the standard model we use is sufficient, as our goal is to evaluate the optimization of our detection strategy rather than to achieve the best possible accuracy on these tasks. The generators used for the experiments on CIFAR-10 and Fashion-MNIST are both MLP architecture with one hidden layer, which take a one-hot vector $\hat{y}$ as input and produce the corresponding feature



(a) CIFAR-10 with $m = 20\%$ (b) CIFAR-10 with $m = 30\%$ (c) CIFAR-10 with $m = 40\%$

Fig. 3. Limitations of PCA and clustering based detection strategy [8] for the CIFAR-10 dataset under label flipping attacks. m is the proportion of malicious clients.

representation. The specific architecture of the networks and generators can be found in Appendix A.

B. Configurations

Unless otherwise mentioned, with one central server, we select 10 clients from $N = 100$ participants for global aggregation each round. The proportion of malicious clients in all participants is denoted by m. The updating step for local training is denoted by T. Specifically, we adopt $T = 25$ for Fashion-MNIST and $T = 5$ for CIFAR-10, where each step uses a mini batch with size $B = 32$. It is mentioned that, in the design of a robust defense strategy, "gradient drift" is a potential challenge since the update parameters may come from the learning of the actual benign clients (which is desirable) or the destruction of attackers (which is unexpected) [8]. Thus, for testing whether they will suffer from gradient drift, although both CNN models can converge within 100 training rounds, we set our detection experiments to run for $R = 200$ rounds in total. We set updating step $T = 20$ and a batch size of 32 for the generator, which is optimized in the central server.

C. Label Flipping Setup

For simulating the label flipping attacks in FL settings, we leverage the "source class → target class pairing" attack settings in [8], which represents adversarial attacks under various conditions, including

1) a pairing that the source class was frequently misclassified as the target class in non-poisoned federated setup,
2) a pairing where the source class was very infrequently misclassified as the target class,
3) a pairing between these two extremes.

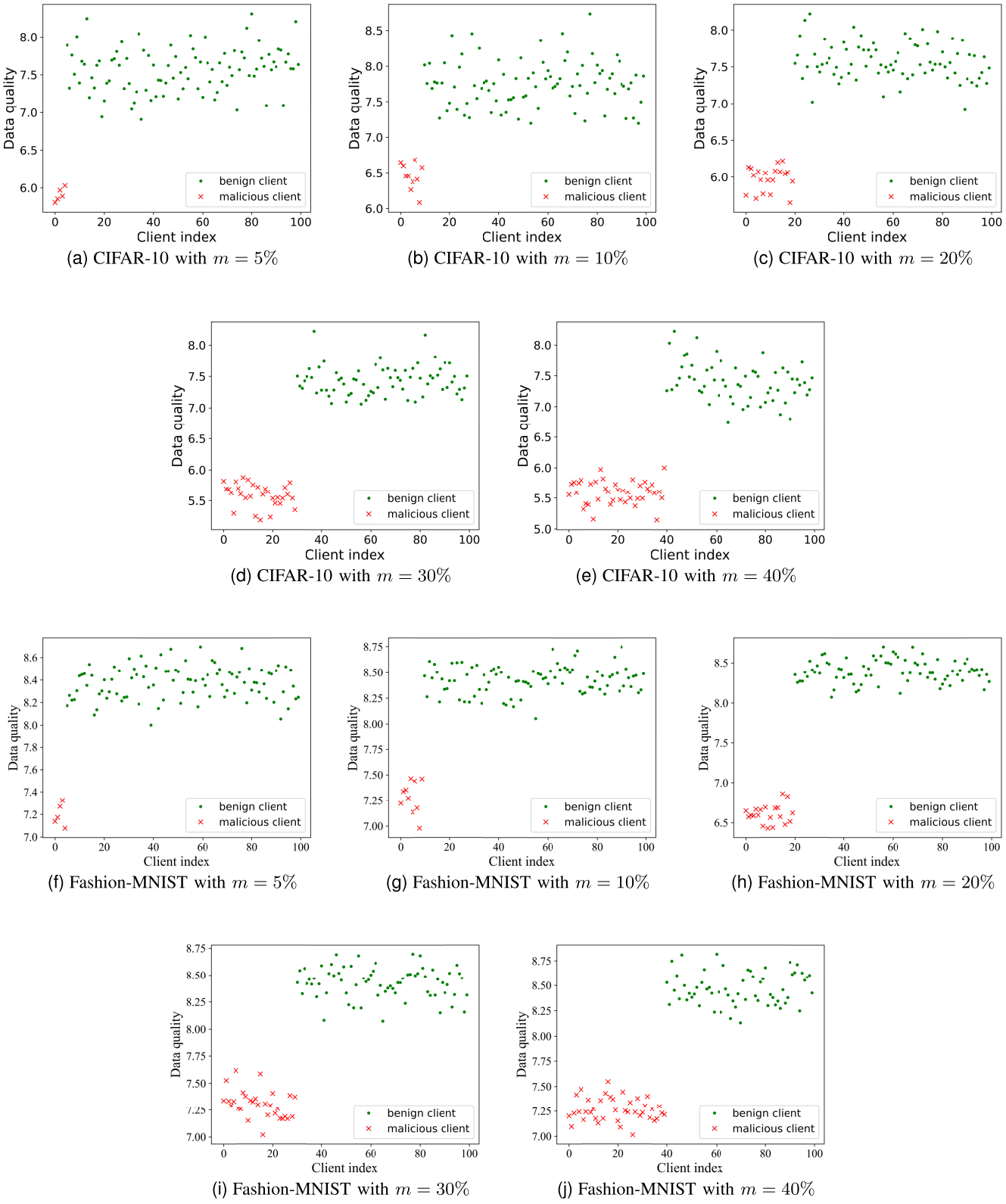
Specifically, corresponding to these three situations, for CIFAR-10 we run with (1) 5: dog → 3: cat, (2) 0: airplane → 2: bird, and (3) 1: automobile → 9: truck. For Fashion-MNIST we set (1) 6: shirt → 0: t-shirt/top, (2) 1: trouser → 3: dress, and (3) 4: coat → 6: shirt.

Experiments in [8] show that such an attack strategy works well in federated environment, even with a few malicious participants, it can effectively decrease the accuracy of the learned model. In experiments with CIFAR-10, once the proportion of malicious participants reaches 40%, there is a devastating drop in the recall of the source class, even drops to 0% and the accuracy of global model drops from 78.3% to 74.4%. Similar loss shown in experiments conducted on Fashion-MNIST also demonstrates the effectiveness of such attack. The recall of the source class drops by ∼10% at $m = 4\%$ and by ∼20% at $m = 10\%$. The empirical results above show that even a minor proportion of participants are dominated by the attacker, the global model can also be significantly affected. Moreover, the loss of the global model shows an upward trend as the rise of m, such loss can even be catastrophic.

D. Results of Malicious Clients Detection

We first investigate the malicious clients detection of the strategy powered by PCA and clustering technology [8] in Fig. 3. Specifically, we keep their original experimental settings and extend their experiments by setting the total number of clients as 100 and the proportion of malicious clients to be larger than 20%. Fig. 3 shows relevant gradients collected from local models with two dimensions, where the orange color and the blue color denote the gradients of the benign updates and the malicious updates, respectively. It is obvious that the two groups highly overlap in scenarios where the proportion of malicious clients exceeds 20%. Consequently, updates from benign and malicious clients are unable to be distinguished by any clustering-based detection strategy.

We then show the detection results of algorithm 1 under multiple scenarios in Fig. 4, including random attack setting and various proportions of malicious participants on CIFAR-10 and Fashion-MNIST. It is noticed that, the source class which is tampered is randomly selected by attackers and the detection results are randomly selected from $R = [20, 200]$. We present the detection results of the whole attack strategies under different conditions in Appendix B. As the experimental results show, the malicious nodes have worse data quality compared with benign ones', which place them into a significantly different cluster from the benign nodes. In extreme scenarios ($m = 5\%$ or $m = 40\%$), our defense strategy is still capable to identify all malicious participants, while the method of [8] has some malicious nodes to be mistaken as benign when $m \geq 20\%$ shown in Fig. 3. Compared to [8], our proposed method does not suffer from the overlapping problem



(a) CIFAR-10 with $m = 5\%$ (b) CIFAR-10 with $m = 10\%$ (c) CIFAR-10 with $m = 20\%$

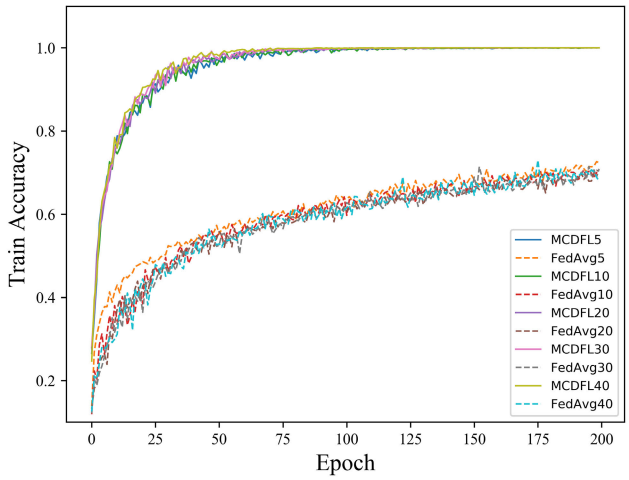
(d) CIFAR-10 with $m = 30\%$ (e) CIFAR-10 with $m = 40\%$

(f) Fashion-MNIST with $m = 5\%$ (g) Fashion-MNIST with $m = 10\%$ (h) Fashion-MNIST with $m = 20\%$

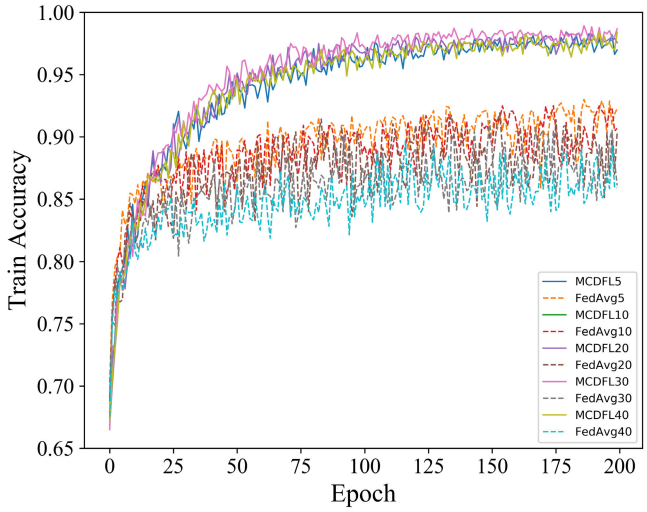
(i) Fashion-MNIST with $m = 30\%$ (j) Fashion-MNIST with $m = 40\%$

Fig. 4. Detection results overview given different adversarial settings. Plots represent relevant data quality collected from $N = 100$ participants with all global training rounds R>20.

when the proportion of malicious clients is higher than 20%. A long period of test suggests that effective identification between malicious and benign participants can be achieved in early global training epochs (global epoch around 20) and maintained in long-term global training (up to 200). Therefore, the server can perform such a detection before each round

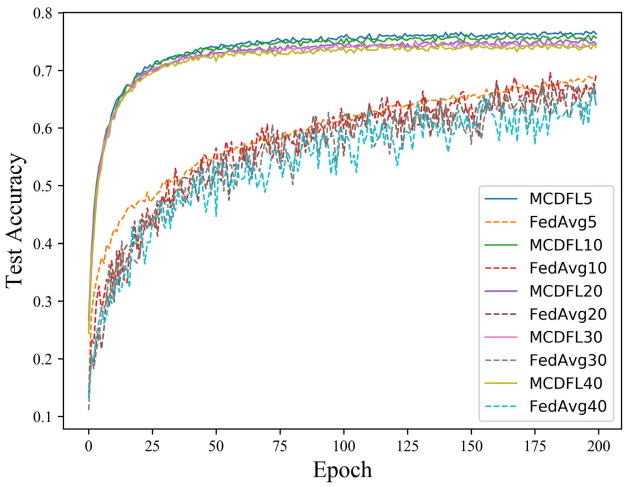


(a) training accuracy on CIFAR-10

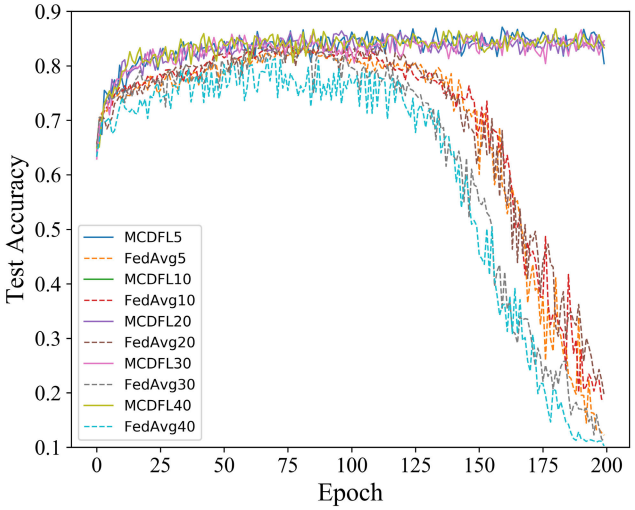


(b) training accuracy on Fashion-MNIST

Fig. 5. Averaged training accuracy over 10 selected clients on training data. MCDFL5 (10, 20, 30, 40) and FedAvg5 (10, 20, 30, 40) represent the training curves of the algorithm with 5 (10, 20, 30, 40) malicious users, respectively.



(a) test accuracy on CIFAR-10



(b) test accuracy on Fashion-MNIST

Fig. 6. Averaged testing accuracy over 10 selected clients on test data. MCDFL5 (10, 20, 30, 40) and FedAvg5 (10, 20, 30, 40) represent the testing curves of the algorithm with 5 (10, 20, 30, 40) malicious users, respectively.

TABLE I
ADDITIONAL STORAGE FOR THE GENERATOR

dataset	Fashion-MNIST	CIFAR-10
additional storage	0.8MB	4.13MB

of aggregation to identify and limit malicious participants to participate in global aggregation.

E. Performance Comparison

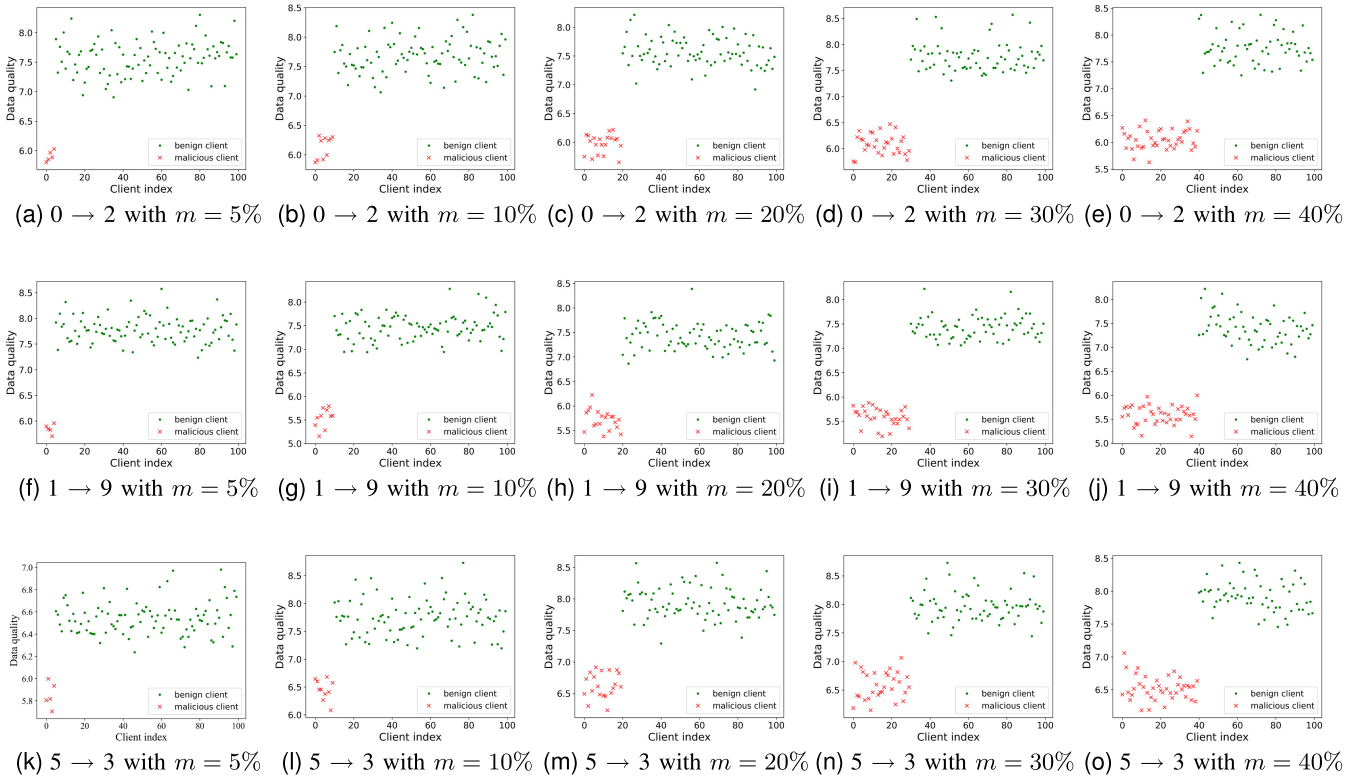
In this section, we explore the performance of the pure FedAvg and FedAvg with MCDFL applied under different malicious scenarios.

1) Local Knowledge Extraction: Fig. 5 shows the local learning curves on CIFAR-10 and Fashion-MNIST over different malicious proportions, which are the average of 10 selected clients. With our personalized paradigm, the training accuracy is around 98% on Fashion-MNIST and even up to 100% on CIFAR-10, which improved by 20%-points or even 30%-points more than the pure FedAvg. The results show that, with our personalized design, the model could accurately extract the knowledge about local training data from each client. Therefore, the distribution over the latent feature space we strive to recover is promising to guide the generator for proper learning and malicious detection.

2) Robustness of MCDFL: We show the accuracy curves on test data in Fig. 6, which are the average of 10 selected clients. In Fashion-MNIST and CIFAR-10, the test accuracy of the pure FedAvg decreases as the number of malicious clients increases, and the test curves also become more and more tortuous. What's worse, in the test of Fashion-MNIST, the model even suffers from gradient drift, which is manifested as a decrease in the model's test accuracy after about 100 global epochs. Such damage is caused by the update parameters

TABLE II
COMPUTATIONAL TIME ON CLIENT SIDE IN EACH TRAINING ROUND, USING FASHION-MNIST DATASET

Batch size		8	16	32	64	128
Computational Time (s)	Pure FedAvg	0.19±0.01	0.26±0.01	0.54±0.02	1.00±0.09	2.04±0.22
	FedAvg with MCDFL	0.34±0.02	0.43±0.01	0.68±0.01	1.33±0.12	2.23±0.06



(a) $0 \rightarrow 2$ with $m = 5\%$ (b) $0 \rightarrow 2$ with $m = 10\%$ (c) $0 \rightarrow 2$ with $m = 20\%$ (d) $0 \rightarrow 2$ with $m = 30\%$ (e) $0 \rightarrow 2$ with $m = 40\%$

(f) $1 \rightarrow 9$ with $m = 5\%$ (g) $1 \rightarrow 9$ with $m = 10\%$ (h) $1 \rightarrow 9$ with $m = 20\%$ (i) $1 \rightarrow 9$ with $m = 30\%$ (j) $1 \rightarrow 9$ with $m = 40\%$

(k) $5 \rightarrow 3$ with $m = 5\%$ (l) $5 \rightarrow 3$ with $m = 10\%$ (m) $5 \rightarrow 3$ with $m = 20\%$ (n) $5 \rightarrow 3$ with $m = 30\%$ (o) $5 \rightarrow 3$ with $m = 40\%$

Fig. 7. Detection results over all adversarial settings on CIFAR-10. Plots represent relevant data quality collected from $N = 100$ participants with all global training rounds R > 20. 0→2, 1→9, 5→3 represent the flips with the label airplane to bird, automobile to truck, and dog to cat, respectively.

from malicious participants. While in FedAvg with MCDFL applied, its test accuracy outperforms the pure FedAvg with a considerable margin. The stable testing curves also demonstrate that, under various degrees of noisy environment, our detection strategy is robust to quickly identify malicious participants and restrict them from global aggregation. Moreover, the capability of stable identification and flat-lying learning curves in a long period of rounds also illustrates that our defence could get rid of the problem of gradient drift.

F. Comments on the Costs

We point out that the generative model could be lightweight especially when the latent space is compact. In practice, we use the generator of an MLP architecture with one hidden layer, which takes up only a small amount of storage on the client side, as shown in Table I. The comparison of computational time on the client side is in Table II. It shows that, FedAvg with MCDFL applied requires only a bit additional storage and computational time to resist label flipping attacks. Empirical results above indicates that the generator is promising for recovering the distribution over a latent feature space to defend against label flipping attacks and lead to better generalization performance of the local model, which can trade off the allowable additional computational and storage costs.

VI. CONCLUSION

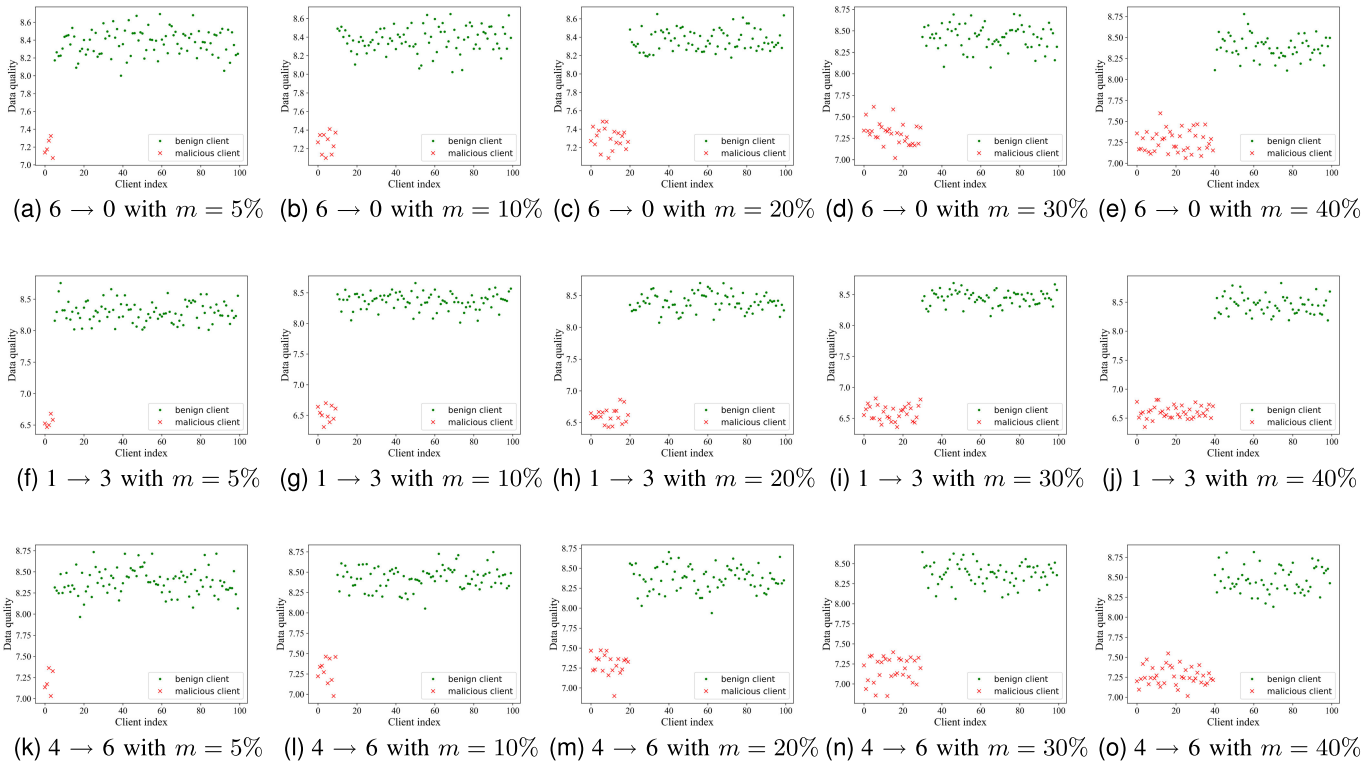
In this paper, we have proposed MCDFL, a federated learning system with data quality detection to defend against label flipping attacks. In practice, designed with the lightweight generator, MCDFL is capable to identify the adversaries and lead to better generalization performance of the local model, with neither considerable computational costs, nor any prior knowledge. Extensive empirical experiments were conducted over two benchmark datasets under various conditions. The experimental results verified that MCDFL is efficient and robust to defend against label flipping attacks, where the proportion of malicious clients is in the range of 5% and 40%.

As future work, we intend to evaluate and enhance the proposed method against other types of data poisoning attacks and further mitigate vulnerabilities in FL systems.

APPENDIX

A. Model Architecture

We design two neural networks for the datasets, i.e., Fashion-MNIST and CIFAR-10, as follows:



(a) $6 \rightarrow 0$ with $m = 5\%$ (b) $6 \rightarrow 0$ with $m = 10\%$ (c) $6 \rightarrow 0$ with $m = 20\%$ (d) $6 \rightarrow 0$ with $m = 30\%$ (e) $6 \rightarrow 0$ with $m = 40\%$

(f) $1 \rightarrow 3$ with $m = 5\%$ (g) $1 \rightarrow 3$ with $m = 10\%$ (h) $1 \rightarrow 3$ with $m = 20\%$ (i) $1 \rightarrow 3$ with $m = 30\%$ (j) $1 \rightarrow 3$ with $m = 40\%$

(k) $4 \rightarrow 6$ with $m = 5\%$ (l) $4 \rightarrow 6$ with $m = 10\%$ (m) $4 \rightarrow 6$ with $m = 20\%$ (n) $4 \rightarrow 6$ with $m = 30\%$ (o) $4 \rightarrow 6$ with $m = 40\%$

Fig. 8. Detection results over all adversarial settings on Fashion-MNIST. Plots represent relevant data quality collected from $N = 100$ participants with all global training rounds R > 20. 6→0, 1→3, 4→6 represent the flips with the label shirt to t-shirt/top, trouser to dress, and coat to shirt, respectively.

- Fashion-MNIST: The dataset is normalized with mean [0.5,] and standard deviation [0.5,]. Table III describes the architecture of the Convolutional Neural Network, where Conv2D represents 2D Convolutional Layer, BN represents Batch Normalization, and FC represents fully connected layer. Specifically, the last layer of the model, i.e., FC, is the predictor module θ^p, and the other are layers for feature extraction, which are denoted as θ^f. The architecture of the generator for Fashion-MNIST is shown in Table IV.
- CIFAR-10: The dataset is normalized with the mean of [0.485, 0.456, 0.406] and the standard deviation of [0.229, 0.224, 0.225]. These values are the mean and standard deviation of the dataset [40], which are common used in pre-processing, even in the scenario of Torchvision models [41]. Additionally, we perform data augmentation with default padding, random horizontal flipping, and a size of 32 for random cropping. The architecture of the CNN is shown in Table V. Specifically, the last component, i.e., FC plus log_Softmax, is the predictor module θ^p of the model, and the others are layers for feature extraction, which are denoted as θ^f. The architecture of the generator for CIFAR-10 is shown in Table VI.

B. Detailed Detection Results

We provide an overview of detection results that are randomly selected from R = [20, 200] over all adversarial settings mentioned in our work on CIFAR-10 and Fashion-MNIST in Fig. 7 and Fig. 8. The empirical results of MCDFL in a series of label flipping attacks show that adversaries who tamper with sample labels have generally worse data quality. Due to the discrimination strategy based on data quality, adversaries are classified as malicious participants in subsequent clustering procedure.

TABLE III
MODEL ARCHITECTURE FOR FASHION-MNIST

Module	Layer	Detail
θ^f	Conv2D + ReLu + BN	5 × 1 × 16
	Max Pooling	2 × 2
	Conv2D + ReLu + BN	5 × 16 × 32
	Max Pooling	2 × 2
θ^p	FC	1568 / 10

TABLE IV
GENERATOR FOR FASHION-MNIST

Layer	Detail
FC	10 / 128
FC	128 / 1568

C. Theoretical Derivations

Given $r(z|y) : y \rightarrow z$ is the conditional distribution from the generator, a user model θ_i can be regulated by utilizing features sampled from $r(z|y)$ to perform matching between the distributions derived from local training data and the generator. Therefore, the objective of regulation is to minimize the conditional KL-divergence between these two distributions:

$$\max_{\theta_i} \mathbb{E}_{y\sim p(y), z\sim r(z|y)}[\log p(y|z;\theta_i)] \equiv \min_{\theta_i} D_{KL}[r(z|y)||p(z|y;\theta_i)].$$

TABLE V
MODEL ARCHITECTURE FOR CIFAR-10

Module	Layer	Detail
θ^f	Conv2D + ReLu + BN	3 × 3 × 32
	Conv2D + ReLu + BN	3 × 32 × 32
	Max Pooling	2 × 2
	Conv2D + ReLu + BN	3 × 32 × 64
	Conv2D + ReLu + BN	3 × 64 × 64
	Max Pooling	2 × 2
	Conv2D + ReLu + BN	3 × 64 × 128
	Conv2D + ReLu + BN	3 × 128 × 128
	Max Pooling	2 × 2
θ^p	FC + log_Softmax	2048 / 10

TABLE VI
GENERATOR FOR CIFAR-10

Layer	Detail
FC	10 / 512
FC	512 / 2048

Proof: The KL-divergence can be expanded as:

$$\begin{aligned}
&D_{KL}[r(z|y)||p(z|y;\theta_i)] \\
&\quad\equiv \mathbb{E}_{y\sim p(y)}[\mathbb{E}_{z\sim r(z|y)}[\log \frac{r(z|y)}{p(z|y;\theta)}]] \\
&\quad= \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log r(z|y)] \\
&\qquad -\mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log p(z|y;\theta)] \\
&\quad= \underset{constant\ \ w.r.t\ \ \theta_i}{-H(r(z|y))} - \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log p(z|y;\theta)],
\end{aligned}$$

where $H(r(z|y))$ is the entropy of distribution $r(z|y)$, which is a constant that is not optimizable with respect to θ_i. Therefore, when optimizing θ_i, we aim to have

$$\begin{aligned}
&\min_{\theta_i} D_{KL}[r(z|y)||p(z|y;\theta_i)] \\
&\equiv \min_{\theta_i} -\mathbb{E}_{y\sim p(y),z\sim r(z|y)}[\log p(z|y;\theta_i)] \\
&\equiv \max_{\theta_i} \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log \frac{p(y|z;\theta_i)p(z)}{p(y)}] \\
&\equiv \max_{\theta_i} \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log p(y|z;\theta_i) + \log p(z) - \log p(y)] \\
&\equiv \max_{\theta_i} \mathbb{E}_{y\sim p(y)}\mathbb{E}_{z\sim r(z|y)}[\log p(y|z;\theta_i)],
\end{aligned}$$

assuming y is the output, $p(z|y;\theta_i) := \frac{p(y|z;\theta_i)p(z)}{p(y)}$ is the probability that the representation input of θ_i^f is z. To this end, we have completed the proof of Eq. (9).

REFERENCES

[1] Y. Zhao et al., "Privacy-preserving blockchain-based federated learning for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1817–1829, Feb. 2021.

[2] W. Zhang et al., "Dynamic-fusion-based federated learning for COVID-19 detection," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15884–15891, Nov. 2021.

[3] J. Passerat-Palmbach et al., "Blockchain-orchestrated machine learning for privacy preserving federated learning in electronic health data," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Nov. 2020, pp. 550–555.

[4] C. Galea and R. A. Farrugia, "Matching software-generated sketches to face photographs with a very deep CNN, morphed faces, and transfer learning," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 6, pp. 1421–1431, Jun. 2018.

[5] R. F. Nogueira, R. de Alencar Lotufo, and R. C. Machado, "Fingerprint liveness detection using convolutional neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1206–1213, Jun. 2016.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol. (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[8] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data poisoning attacks against federated learning systems," in *Proc. Eur. Symp. Res. Comput. Secur.*, Cham, Switzerland: Springer, 2020, pp. 480–501.

[9] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras, "When does machine learning FAIL? generalized transferability for evasion and poisoning attacks," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1299–1316.

[10] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Security Privacy*, vol. 17, no. 2, pp. 49–58, Mar./Apr. 2019.

[11] A. I. Newaz, A. K. Sikder, M. A. Rahman, and A. S. Uluagac, "A survey on security and privacy issues in modern healthcare systems: Attacks and defenses," *ACM Trans. Comput. Healthcare*, vol. 2, no. 3, pp. 1–44, Jul. 2021.

[12] M. Burruss, S. Ramakrishna, and A. Dubey, "Deep-RBF networks for anomaly detection in automotive cyber-physical systems," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Aug. 2021, pp. 55–60.

[13] E. Borgnia et al., "Strong data augmentation sanitizes poisoning and backdoor attacks without an accuracy tradeoff," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Jun. 2021, pp. 3855–3859.

[14] M. Zhang, L. Hu, C. Shi, and X. Wang, "Adversarial label-flipping attack and defense for graph neural networks," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 791–800.

[15] J. Zhang, B. Chen, X. Cheng, H. T. T. Binh, and S. Yu, "PoisonGAN: Generative poisoning attacks against federated learning in edge computing systems," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3310–3322, Mar. 2021.

[16] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*.

[17] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 601–618.

[18] R. Doku and D. B. Rawat, "Mitigating data poisoning attacks on a federated learning-edge computing network," in *Proc. IEEE 18th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2021, pp. 1–6.

[19] 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 *Proc. 8th Int. Conf. Dependable Syst. Their Appl. (DSA)*, Aug. 2021, pp. 551–559.

[20] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[21] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5650–5659.

[22] J. Steinhardt, P. W. W. Koh, and P. S. Liang, "Certified defenses for data poisoning attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 3520–3532.

[23] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[24] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: Information leakage from collaborative deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 603–618.

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

[26] N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez, and A. Blanco-Justicia, "Defending against the label-flipping attack in federated learning," 2022, *arXiv:2207.01982*.

[27] B. Wang et al., "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 707–723.

[28] B. Tran, J. Li, and A. Madry, "Spectral signatures in backdoor attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 8011–8021.

[29] A. Qayyum, M. U. Janjua, and J. Qadir, "Making federated learning robust to adversarial attacks by learning data and model association," *Comput. Secur.*, vol. 121, Oct. 2022, Art. no. 102827.
[30] Z. Ma, J. Ma, Y. Miao, X. Liu, K.-K.-R. Choo, and R. H. Deng, "Pocket diagnosis: Secure federated learning against poisoning attack in the cloud," *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3429–3442, Nov. 2022.
[31] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12878–12889.
[32] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist, Prob.*, vol. 1, Jun. 1967, pp. 281–297.
[33] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *Proc. 4th World Conf. Smart Trends Syst., Secur. Sustainability (WorldS)*, Jul. 2020, pp. 794–797.
[34] Q. Wu, X. Chen, Z. Zhou, and J. Zhang, "FedHome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2818–2832, Aug. 2022.
[35] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent IoT applications: A cloud-edge based framework," *IEEE Open J. Comput. Soc.*, vol. 1, pp. 35–44, 2020.
[36] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-IID data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–13.
[37] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 2089–2099.
[38] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[39] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Tech. Rep., 2009.
[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[41] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proc. 18th ACM Int. Conf. Multimedia*, Oct. 2010, pp. 1485–1488.

Yifeng Jiang is currently pursuing the M.E. degree in computer science with the Guangdong University of Technology (GDUT). His current research interests include federated learning and data poisoning.

Weiwen Zhang (Member, IEEE) received the B.E. degree in software engineering and the M.E. degree in computer science from the South China University of Technology (SCUT) in 2008 and 2011, respectively, and the Ph.D. degree in computer engineering from Nanyang Technological University (NTU), Singapore, in 2015. He was a Scientist at the Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), Singapore, a Research Associate at NTU, and a Visiting Scholar at Purdue University, USA. He is currently an Associate Professor with School of Computer Science and Technology, Guangdong University of Technology (GDUT), China. His research interests include cloud computing, mobile computing, big data analytics, and machine learning.

Yanxi Chen is currently pursuing the M.E. degree in computer science with the Guangdong University of Technology (GDUT). His current research interests include edge computing and federated learning.