Research Article

Personalized Federated Learning with Semisupervised Distillation

Xianxian Li,¹ Yanxia Gong¹, ,¹ Yuan Liang^{2,3} , and Li-e Wang¹

¹Guangxi Key Lab of Multi-Source Information Mining and Security, Guangxi Normal University, Guilin 541004, China

²School of Computer Science and Engineering, Beihang University, Beijing 100191, China

³Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

Correspondence should be addressed to Yuan Liang; liangyuan120@buaa.edu.cn

Received 28 September 2021; Accepted 26 November 2021; Published 17 December 2021

Academic Editor: Chien Ming Chen

Copyright © 2021 Xianxian Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Heterogeneous data and models pose critical challenges for federated learning. However, the traditional federated learning framework, which trains the global model by transferring model parameters, has major limitations; it requires that all participants have the same training model architectures, and the trained global model does not guarantee accurate projections for participants' personal data. To solve this problem, we propose a new federal framework named personalized federated learning with semisupervised distillation (pFedSD), which ensures the privacy of the participants' model architectures and improves the communication efficiency by transmitting the model's predicted class distribution rather than model parameters. First, the server adopts the adaptive aggregation method to reduce the weight of low-quality model predictions for the model's predicted class distributions uploaded by all clients, which helps to improve the quality of the aggregation of the prediction class distribution. Then, the server sends it back to the clients for local training to obtain the personalized model. We finally conducted experiments on different datasets (MNIST, FMNIST, and CIFAR10), and the results show that the model performance of pFedSD exceeds the latest federated distillation algorithms.

1. Introduction

In recent years, federated learning has become a popular machine learning paradigm [1–3]. In federated learning, a group of clients cooperate to train a global model without uploading local datasets. Each client can only access its data, which protects the privacy of participants' data in the training. Because of its advantages, federated learning has broad application prospects in medicine, finance, artificial intelligence, and other industries, and it has been a research hotspot in recent years. In the training of federated learning, the participants' storage model device memory and computing power may differ. For example, some participants' equipment performance may be better or their local datasets may be larger, which will require a slightly more complicated model architecture. However, if the local device performance is poor, and the local datasets are not very large, a relatively simple model architecture should be selected.

In addition, some participants had already trained on local data using a model of architecture before federated

training, so they did not need to start training from scratch when doing federated training with other participants. Some participants were even reluctant to share their models due to privacy [4]. For the above reasons, the model architectures selected by participants were also different. However, the traditional federated learning framework trains the global model by transferring model parameters between the server and clients, which fails to meet the need for each client to freely choose its own model architecture. Therefore, it is necessary to allow each client the freedom to choose the suitable model architecture for training.

In particular, in the medical and financial scenarios, due to regional and equipment differences, the patient data of each hospital or the user data of each insurance company are heterogeneous. However, the current federated learning focuses on learning local data from all clients to obtain the global model. However, due to the heterogeneity of clients in the actual scene, the performance of the trained global model is poor when it is run on clients. Furthermore, in reality, each participant (such as a hospital) has mostly unlabeled local data, and very little labeled

data. Therefore, how to train a suitable model (including model performance and model framework) for each participant under the constraint of privacy protection and semisupervised scenarios is an urgent problem to be solved.

In this paper, to address the above issues, we first propose personalized federated learning with semisupervised distillation (pFedSD), which aims at sharing the same unlabeled dataset on all clients, and then use this auxiliary unlabeled dataset to train with the client's private dataset. This provides each client with a personalization model whose architecture is completely designed by the client itself and unknown to the server and other clients.

The current methods combining federated learning and knowledge distillation directly average the model predictions (through softmax) uploaded by clients [4–6]. However, due to the data heterogeneity of clients and the variable quality of the models trained by clients, the importance of the model outputs uploaded by them also varies. Therefore, the model output uploaded by each client cannot be treated equally. Therefore, to ensure the quality of aggregation model prediction, this paper proposes an adaptive knowledge aggregation method based on Jensen–Shannon (JS) divergence, which improves the quality of aggregation model prediction by reducing the aggregation weight of low-quality models in each round, that is, reducing their contribution degree.

In addition, when the trained model is relatively complex, the transmission of model parameters will consume huge communication costs, and the transmission efficiency is very slow. pFedSD uses the right number of distilled datasets to improve communication efficiency in federated learning by uploading model predictions on clients and pseudolabel of shared unlabeled data back on the server. Moreover, Zhu et al. [7] and Yin et al. [8] proved that the original data could be recovered by using the stolen model gradient, and pFedSD could effectively defend against gradient attack by transmitting the model outputs between the server and clients. The contributions of this paper are summarized as follows:

- (i) We propose a new federal framework to meet the needs of each client to freely choose its model architecture for training, which is the first framework that combines federated learning and knowledge distillation in a semisupervised scenario to solve the problems of data and model heterogeneity.
- (ii) A method of dynamic knowledge aggregation is adopted on the server side. By reducing the weight of the low-quality models, the method improves the aggregation model's predicted class distribution and then improves the performance of local models.
- (iii) The clients upload the model's predicted class distributions, and the server returns the pseudolabels of the shared unlabeled data, which greatly improves the communication efficiency in federated learning.
- (iv) Experimental results on benchmark datasets and real datasets demonstrate that the performance of pFedSD is better than that of the existing combined distillation methods.

2. Preliminary and Related Work

2.1. Knowledge Distillation. The basic idea of knowledge distillation is to take the output of an extensive complex network and transmit it as knowledge to a small network [9]. In the training process, the small network can learn the information of the real labels of the data and can learn the relationship between different labels and can then be converted into a compact network. An extensive network is called a teacher network, and a small network is called a student network. However, traditional knowledge distillation algorithms require a pretrained teacher network, and the teacher network cannot obtain feedback from the student network during the training process. Zhang et al. [10] proposed a deep mutual learning method in which multiple networks are trained simultaneously. During the training process, each network not only receives supervision from real labels but also learns from the experiences of other networks to further improve the generalization ability of the model.

Federated learning based on knowledge distillation is a collaborative training of multiple clients. However, direct use of knowledge distillation in federated learning requires the same local data of each device, which is obviously not practical. To solve this problem, Li and Wang [4], Itahara et al. [5], Sun and Lyu [6], Chang et al. [11], and Hu et al. [12] introduced a public unlabeled dataset. Enables clients to complete federated learning training by distilling on the same data. Figure 1 shows the process of federated distillation, in which each client is both a student and a teacher. As students, they receive the model prediction distribution from the central server aggregation for distillation training, and as teachers, they provide their own model's predicted class distribution to the server-side aggregation to guide other clients in distillation training. However, these methods assume an available public unlabeled dataset or use part of the original data as public data, which introduces certain constraints and may compromise the privacy of user data. Unlike previous research efforts, pFedSD uses generators trained by clients on unlabeled data locally to obtain public data.

2.2. Semisupervised Learning. In most cases of the real world, there are very little data with labels. Since labeled data are complex and unlabeled data are easy to obtain, people use many unlabeled data to train with labeled data. This training method is called semisupervised learning. Semisupervised learning has achieved much in recent years [13–15]. The current optimal semisupervised learning method, FixMatch [14], integrates the previous methods and consistently regularizes weakly augmented and strongly augmented images of the same image, thus achieving better performance.

2.3. Personalized Federated Learning. The original purpose of federated learning is to collaborate with all participants to obtain an aggregation model [3]. However, when the private

FIGURE 1: The workflow of federated distillation.

data of clients present different distributions, that is, they are not independent and identically distributed (non-IID), which causes each client model to update in a different direction. Finally, the single global model aggregated by the server cannot be suitable for all clients. Even some clients get a global model that is worse than the model they train locally without participating in the collaboration. This weakens the clients' motivation to participate in collaborative training. The heterogeneous data problem of clients can be solved by using the personalized federated learning method, that is, training a personalized model for each client. At present, there are kinds of personalized federated learning methods [16–19]. These personalized federated learning methods are the same as the traditional federated learning framework, which trains the model by transferring model parameters between the server and clients. However, this will lead to each client getting the model with the same architecture at last, which ignores the local device capability of clients and has great limitations. FedPU [20] is slightly similar to our work in that he builds models from labeled and unlabeled datasets stored on clients. But the goal of FedPU is to train a global model, and it also follows the traditional federated learning framework that does not allow for the heterogeneity of local models.

The current works adopt the combination of knowledge distillation and federated learning to transfer model predictions between clients and the server rather than model parameters to ensure the privacy of the client model architectures [4–6, 10, 12]. However, these federated knowledge distillation methods directly average the model predictions uploaded by clients. Due to each client's different qualities of the model trained locally, the importance of the knowledge in the model predictions uploaded by them is different, so there should be a more appropriate aggregation method. Unlike previous research works, pFedSD obtains public unlabeled data by using generators trained by clients and assigns different aggregate weights to each client according to the similarity between the model prediction values of clients in the current round and the aggregated prediction value of the previous round. The aggregate

methods reduce the contribution of low-quality models; therefore, the quality of the aggregate model prediction is improved. As the clients distill the aggregate model prediction, the aggregate method finally improves the performance of the clients' personalized models.

3. Methodology

3.1. Federated Distillation Learning. Setting K clients for federated learning training, each client k has labeled datasets $D_k = \{(x_i, y_i)\}_{i=1}^{N_k}$ of size N_k that draw from distribution P_k . Here, $P_{i \in K} \neq P_{j \in K} (i \neq j)$. There are public data $D_{\text{public}} = \{(x_i, y_i)\}_{i=1}^m$ of size m shared to clients. Each client k designs its model w_k according to the communication capacity, storage capacity, and local dataset size of its own local device. When the communication round $t = 0$, for each client k , their optimization goal is

$$\arg \min_{w_k} \text{Cross Entropy}(f_k(x, w), y). \quad (1)$$

Then, model w_k is used to predict the public data to obtain p_k^t , which is then averaged by the server and sent back to the clients:

$$p_{\text{avg}}^t = \sum_{k=1}^K \frac{N_k}{\sum_{k \in K} N_k} p_k^t. \quad (2)$$

When the communication round $t > 0$, client k obtains p_{avg}^t and performs model update together with $(X_{\text{public}}, p_{\text{avg}}^t) \cup (x, y)$:

$$\arg \min_{w_k} \{ \text{Cross Entropy}(f_k(x, w), y) + KL(f_k(X_{\text{public}}, w), p_{\text{avg}}^t) \}. \quad (3)$$

However, public data may compromise the privacy of the original data. In addition, when the client data present non-IID, the quality of the model trained by each client is different, which leads to the different importance of the class distribution predicted by each client model. The aggregation

method of equation (2) cannot flexibly provide the aggregation weight for each client.

3.2. Problem Definition of pFedSD. While existing federated distillation methods are conducted in supervised settings, pFedSD is conducted in semisupervised settings. We define $k \in K$ clients with local datasets D_k . D_k contains labeled datasets $D_{l_k} = \{(x_i, y_i)\}_{i=1}^{N_{l_k}}$ and unlabeled datasets $D_{u_k} = \{(x_i)\}_{i=1}^{N_{u_k}}$; the D_{l_k} and D_{u_k} for each client k draw from different distributions, and $N_u \gg N_l$.

To observe the same data on clients, we share the same unlabeled data $D_{\text{public}} = \{(x_i, y_i)\}_{i=1}^m$ on each client, and $m \gg N_l$. The learning task of pFedSD is to obtain the personalized model $P(k)$ through federated training for K clients with different model architectures w_k and different local datasets D_k under D_{public} .

3.3. pFedSD Framework. Traditional federated learning trains the model by transferring model parameters on both the server and clients, but this method has many limitations. To address these limitations, we propose pFedSD, and the pFedSD framework is shown in Figure 2. There are labeled and unlabeled image datasets with different degrees (non, mild, moderate, severe) of Alzheimer's disease locally on clients. The process of the whole framework is played by a central server and multiple clients as the main role.

3.3.1. Generation of Public Unlabeled Data. Since knowledge distillation is observed on the same data samples, we share the same unlabeled data D_{public} on clients. Algorithm 1 shows the way to obtain D_{public} . First, each client k trains a generator G_k on the local unlabeled dataset and uploads it to the server together with random seeds seed_k of random noise control. Then, the server uses G_k ($k = 1, 2, \dots, K$) to generate data and mix it. After that, the server selects m generated samples of good quality as public unlabeled data and finally returns them to the clients.

3.3.2. Adaptive Aggregation Method. The role of the central server is mainly to aggregate the model outputs uploaded by clients. Since clients' data and model architectures are heterogeneous, the importance of their model outputs is different, so they cannot be treated equally.

In our pFedSD framework, a dynamic aggregation method in which the central server assigns weight to each client according to the distribution similarity between the model output uploaded by each client and the model output value of the previous round of aggregation. We use JS divergence to measure the similarity of distributions, with a value between 0 and 1. The smaller the similarity is, the smaller the weight will be. First, we calculate the auxiliary value ∂_k of the aggregate weight of client k participating in the training in each round, which can be written as follows:

$$\partial_k = \frac{\sum_{t=1}^{C_t} JS(p_{\text{global}}^{t-1}, p_k^t)}{JS(p_{\text{global}}^{t-1}, p_k^t) + \varepsilon}, \quad (4)$$

where C_t is the number of participants in the t -th communication round, $JS(p_{\text{global}}^{t-1}, p_k^t)$ is the JS divergence value of the model prediction p_k^t uploaded by each participant k in the t -th communication round, and the model prediction p_{global}^{t-1} aggregated in the last round. To prevent the above expression of ∂_k from becoming meaningless when $JS(p_{\text{global}}^{t-1}, p_k^t) = 0$, we add a term ε in the denominator; ε denotes a minimal value approaching 0. After the normalization of ∂_k , the aggregation weight of each client is obtained.

Then, the server obtains the aggregate output p_{global}^t by

$$p_{\text{global}}^t = \sum_{k \in C_t} \frac{\partial_k}{\sum_{k \in C_t} \partial_k} * p_k^t, \quad (5)$$

p_{global}^t is an $m * d$ matrix, where d is the number of classes in the dataset. Each row of p_{global}^t corresponds to the probability distribution of the class of each sample, from which we select the one with the highest probability as the pseudolabel of the sample and send it back to the client. Using this method greatly improves the efficiency of communication without affecting performance. By adopting this method, the aggregation weight of low-quality models can be reduced, and the quality of aggregation model prediction can be improved. We show the pseudocode of the aggregation process of the server in Algorithm 2.

3.3.3. Local Updates Based on Distillation. All clients only train on local data in communication round $t = 0$, which is the same as the previous federated distillation methods. In communication round $t > 0$, they use labeled loss on both shared public data and local labeled data and unlabeled loss on local unlabeled data to perform local model updates. We formulate the objective of each client k as

$$\arg \min_{w_k} L_{\text{local}} = \arg \min_{w_k} \{L_l + \beta L_u + \gamma L_{\text{distill}}\}, \quad (6)$$

where L_{local} is the total loss function of training, L_l is the cross-entropy loss between prediction $f_k(x_l, w)$ of model w of labeled data x_l and hard label y on client k , and L_{distill} is the cross-entropy loss between prediction $f_k(X_{\text{public}}, w)$, and pseudolabel $\hat{Y}$ on shared public unlabeled data X_{public} . L_u is the loss function of the unlabeled data. β and γ are fixed hyperparameters ($\beta = 1, \gamma = 1$), which denote the loss weights of L_u and L_{distill} , respectively. L_l and L_{distill} can be written as

$$L_l = \frac{1}{b_l} \sum_{x \in b_l} (f_k(x, w), y), \quad (7)$$

$$L_{\text{distill}} = \frac{1}{b_{\text{distill}}} \sum_{x \in b_{\text{distill}}} (f_k(X_{\text{public}}, w), \hat{Y}). \quad (8)$$

For L_u , we adopt the same consistency regularization method as in FixMatch [14]. For the same image data, the predictions of the model should not change significantly under minor perturbations. Specifically, weak data augmentation (flip or cropping) and strong data augmentation (image distortion) are performed on image data x to obtain x_w and x_s , respectively. Function $1(\cdot)$ is used to determine

FIGURE 2: pFedSD framework.

the class with the highest confidence probability greater than the threshold τ in the model output $f(x_w, w)$ and take it as the pseudolabel of x_w . According to consistency

regularization, the pseudolabel $\arg\max(f(x_w, w))$ for x_w should be consistent with the model output $f(x_s)$ for x_s , and L_u can be written as

$$L_u = \frac{1}{b_u} \sum_{x \in b_u} 1(\max(f(x_w, w)) > \tau) \Phi t(\arg \max(f(x_w, w)), f(x_s, w)), \quad (9)$$

where $\Phi(\cdot)$ represents the cross-entropy loss. By training the public data with the local data, each client gains global knowledge from the aggregated prediction and improves the generalization of the local model. We show the pseudocode of the clients trained locally in Algorithm 3.

4. Experiments

4.1. Experiment Settings

4.1.1. Datasets. We conduct experiments on three benchmark datasets, MNIST [21], FMNIST [22], and CIFAR10 [23]. For MNIST, we split the dataset into 60000 samples for training, 5000 samples for validation, and 5000 samples for testing. For the FMNIST and CIFAR10 datasets, we split the dataset into 50000 samples for training, 5000 samples for validation, and 5000 samples

for testing. Each client uses DCGAN [24] to train the generator, and the server randomly samples $m = 5000$ samples from all the generated samples as public unlabeled data.

4.1.2. Model. We conducted experiments under two scenarios: homogeneous models and heterogeneous models. To simulate client training with different model frameworks, similar to [12], we use neural networks with different layers and different numbers of neurons. Either 2 or 3 convolution layers can be selected, and the number of neural units in each layer can be chosen from 64, 128, 192, and 256. For the scenario of homogeneous models, we choose a three-layer convolution layer and a model framework of a fully connected layer. The neural output channels of the convolution layer are 128, 192, and 256.


```

Input: number of clients  $K$ , local unlabeled data  $D_{u_k}$  for each client  $k$ 
Output:  $X_{\text{public}}$ 
for  $k = 1, 2, \dots, K$  do
     $G_k, \text{seed}_k \leftarrow \text{GAN}(D_{u_k})$ 
    return  $G_k, \text{seed}_k$ 
end
Server ( $G_k, \text{seed}_k$ ):
Initialize  $D$ 
for  $k = 1, 2, \dots, K$  do
     $X_k \leftarrow G_k(\text{seed}_k)$ 
     $D = D \cup X_k$ 
end
Sample  $m$  examples  $\{x^1, x^2, \dots, x^m\}$  as  $X_{\text{public}}$  from  $D$ 
return  $X_{\text{public}}$  to clients

```

ALGORITHM 1: Generation of public unlabeled data.

```

Input: number of clients  $K$ , number of communication rounds  $T$ 
Output: Aggregated prediction  $\hat{Y}$ 
Assign public unlabeled datasets  $D_{\text{public}}$ 
for each round  $t = 0, 1, 2, \dots, T-1$  do
     $C_t \leftarrow$  select random  $\text{frac} * K$  clients from  $K$ 
    if  $t = 0$  then
        for  $k \in C_t$  do
             $p_k^t \leftarrow \text{ClientUpdate}(t)$ 
            Compute  $\partial_k$  by Equation (4)
        end
    else
        for  $k \in C_t$  do
             $p_k^t \leftarrow \text{ClientUpdate}(\hat{Y}, t)$ 
            Compute  $\partial_k$  by Equation (4)
        end
    end
    Compute  $p_{\text{global}}^t$  by equation (5)
     $\hat{Y} = \text{argmax}(p_{\text{global}}^t)$ 
end

```

ALGORITHM 2: Server execute.

```

Input: communication round  $t$ , local labeled data  $D_l$ , local unlabeled data  $D_{u_k}$  and public data  $D_{\text{public}}$  for each client  $k$ , number of
local epochs  $E$ , batch size  $b_l'$  of labeled data, batch size  $b_{\text{distill}}'$  of public data, batch size  $b_u'$  of unlabeled data, confidence threshold  $\tau$ ,
learning rate  $\eta$ , loss weight  $\beta$  and  $\gamma$ 
Output: model's predicted class distribution  $p_k^t$  Initialize the local model  $w_k^0$ 
 $B_l \leftarrow$  split  $D_l$  into batches of size  $b_l'$ 
 $B_{\text{distill}} \leftarrow$  split  $D_{\text{public}}$  into batches of size  $b_{\text{distill}}'$ 
 $B_u \leftarrow$  split  $D_u$  into batches of size  $b_u'$ 
if  $t = 0$  then
    for each local epoch  $e = 0, 1, 2, \dots, E-1$  do
        for each batch  $b_l \in B_l, b_{\text{distill}} \in B_{\text{distill}}$  and  $b_u \in B_u$  do
            Compute  $L_{\text{local}}$  by equation (1)
             $w_k^e = w_k^{e-1} - \eta \nabla L_{\text{local}}$ 
        end
    end
else
    for each local epoch  $e = 0, 1, 2, \dots, E-1$  do

```

ALGORITHM 3: Continued.

```

for each batch  $b_l \in B_l$ ,  $b_{\text{distill}} \in B_{\text{distill}}$  and  $b_u \in B_u$  do
  Compute  $L_{\text{local}}$  by equations (6)–(9)
   $w_k^e = w_k^{e-1} - \eta \nabla L_{\text{local}}$ 
end
end
end
 $p_k^t = f_k(X_{\text{public}}, w_k)$ 
return  $p_k^t$  to server

```

ALGORITHM 3: Client update.

4.1.3. Baselines. We use three algorithms, FedMD [4], DS-FL [5], and MHAT [12], which are personalized algorithms combining knowledge distillation and federated learning, as the comparison algorithm for pFedSD. In particular, similar to [6], we observe the effect of different distillation methods, pFedSD-soft and pFedSD-hot, on our algorithm. pFedSD-soft represents the use of predictive distribution information as public unlabeled data to train the model. The aggregated predictive distribution information contains the model predictive class distribution of each sample in the public unlabeled data. pFedSD-hot means that the class with the highest probability in the model prediction class distribution of each sample is used as the pseudolabel of the sample to train the model. The benchmark performance evaluation indicator of all methods is the top-1 test accuracy.

4.1.4. Implementation Details. We implemented pFedSD and FedMD, DS-FL, and MHAT using PyTorch in a semi-supervised scenario. For all datasets, we choose $K=10$ clients and randomly sample clients with $\text{frac}=0.8$ for each communication round. For the MNIST dataset, the total number of samples per client is 6000, and the labeled samples and unlabeled samples are 50 and 5950, respectively. For FMNIST, the total number of samples per client is 5000, and the labeled and unlabeled samples are 50 and 4950, respectively. For the CIFAR10 dataset, the total number of samples per client is 5000, and the labeled samples and unlabeled samples are 150 and 4850, respectively.

Client local data training with SGD and the hyperparameters are learning rate(lr)=0.01, momentum=0.9, weight decay= $5 * 10^{-4}$, $\beta=1$, $\gamma=1$, $\tau=0.8$, $b_l'=20$, $b_u'=100$, $b_{\text{distill}}'=100$, $T=100$, and $E=10$. Dirichlet distribution [25] is adopted for client data distribution, which uses α to control the non-IID degree of data. The smaller α is, the greater the non-IID degree of the data.

4.2. Experimental Results

4.2.1. Performance on Benchmark Dataset. Figures 3 and 4 visualize the performance comparison of pFedSD and baselines on the MNIST dataset under different distribution settings (IID and non-IID) and different scenarios (homogeneous models and heterogeneous models). It can be seen that the test performance of our method pFedSD outperforms the baselines under different settings. FedMD adopts the simplest aggregation scheme, which considers all participants'

model predictions as equally important and directly averages the model predictions uploaded by clients. Since the entropy value of the model prediction after aggregation is too high, it is not conducive to the distillation of the client. At the same time, DS-FL intensifies and reduces the entropy value of the model prediction of aggregation. As a result, the client distillation training learns more accurate information from the predicted distribution of the aggregation more quickly, so the performance of DS-FL is slightly better than that of FedMD in most cases. MHAT implements information aggregation by training an auxiliary model on the server.

In Figures 3 and 4, when the distributions of local data on clients are not so skewed and the quality of the trained models is not so different, MHAT is not much different from pFedSD. In contrast, when the local data distributions of clients are skewed and there is a large difference between the local models, the auxiliary model trained on the server fails to effectively aggregate the information, leading to a decline in performance. Due to the data heterogeneity of clients and so on, the quality of the models trained locally on clients is different, and the reason why our method, pFedSD, outperforms baselines each round is that the aggregation weight of the low-quality models is reduced, so that the quality of the aggregation model prediction is improved. In addition, in pFedSD, the pseudolabels aggregated by the client download server are trained as hard labels for the public unlabeled data along with the local labeled data, and pFedSD greatly improves the performance of the client local models, due to data augmentation.

In addition, we can see that there is little difference between the performance of pFedSD-soft and pFedSD-hot, and even a slight improvement in the performance of pFedSD-hot compared to pFedSD-soft.

4.2.2. Performance Evaluation in Different Scenarios and Different Datasets. Figure 5 shows the test performance evaluation of all the methods under different datasets and settings. As the degree of non-IID decreases gradually, the test performance of all methods also increases gradually, which is very reasonable. For all α values of different datasets, the performance of pFedSD is no lower than that of the baselines, which shows the superiority of our method.

Specifically, when $\alpha \leq 0.1$, there was little difference in test accuracy between each method, and even though pFedSD was slightly more accurate than the other methods, the difference was not significant. This may be because the knowledge learned by the local models of clients is limited,

FIGURE 3: Test performance of pFedSD compared with baselines for different non-IID settings (i.e., $\alpha = 1$, $\alpha = 10$) and IID setting (i.e., $\alpha = 100$) on MNIST and the same model structures. (a) $\alpha = 1$. (b) $\alpha = 10$. (c) $\alpha = 100$.

FIGURE 4: Test performance of pFedSD compared with baselines for different non-IID settings (i.e., $\alpha = 1$, $\alpha = 10$) and IID setting (i.e., $\alpha = 100$) on MNIST and different model structures. (a) $\alpha = 1$. (b) $\alpha = 10$. (c) $\alpha = 100$.

FIGURE 5: Continued.

FIGURE 5: The test performance evaluation of all the methods on MNIST (a, d), F-MNIST (b, e), CIFAR10 (c, f). (a–c) Homogeneous models. (d–f) Heterogeneous models.

and the quality of the predicted distribution aggregated by the aggregation method in pFedSD is not significantly different from that of the aggregation method in FedMD and DS-FL, while MHAT is significantly lower than pFedSD. In contrary, when $\alpha \geq 0.1$, the adaptive aggregation method in pFedSD improves the predictive distribution quality of aggregation by reducing the weight of the predictive distributions of the low-quality models and thus improves the learning performance of the local models of clients, because the importance of the knowledge learned from the local model of each client is different. It is clear from Figure 5 that pFedSD has a better test performance than other methods.

4.2.3. Comparison of Communication Efficiency for Different Federated Distillation Methods. Compared with the traditional federated learning framework, which trains models by transferring model parameters between the server and clients, federated distillation trains models by transmitting model prediction information, which greatly improves the communication efficiency when the models are more complex. However, in this part, we mainly focus on the communication efficiency comparison between pFedSD and other federated distillation methods. After approximately 15 rounds, with the same accuracy, pFedSD has fewer communication rounds than the baselines used in Figures 3 and 4.

Table 1 summarizes the number of parameters required per communication round for each method (FedMD, DS-FL, MHAT, and pFedSD). Our method, pFedSD-hot, requires the least number of parameters per communication round. Therefore, pFedSD-hot greatly reduces the communication cost, thus improving the communication efficiency in federated learning. We used pFedSD-hot as the distillation method for the local clients in our framework

4.2.4. Performance Impact of Varying Public Data Sizes. In this section, we study the influence of varying public data sizes m on the test performance of the algorithms. Figures 6(a) and 6(b) illustrate that with the increase in public data m , the performance of pFedSD also increases

slightly, where pFedSD is pFedSD-hot. Therefore, public dataset sizes m should be selected moderately. If m is too small, the performance will decline significantly. If m is too large, the performance is not great, but communication transmission costs increase.

Figures 6(c) and 6(d) show the performance comparison of each method with different sizes of public data in different scenarios. From the figure, we know that with the increase in public data m , the degree of increase of each method is not consistent. Our method is better than the other methods in both homogeneous and heterogeneous scenarios. Especially in heterogeneous scenarios, our method shows superior performance, which also verifies our method analysis.

4.2.5. Further Experimental Results and Analysis

(1) Experiments on the Real-World Dataset. We think that experiments on a real-world example would provide much stronger evidence for evaluating the proposed method. Therefore, we evaluate pFedSD on the COVID-19 X [26] dataset. The dataset contained images of three categories (Normal, Pneumonia, and COVID-19), with the training dataset containing 13,954 training images and the test dataset containing 1,579 test images. We used ResNet50 as a model to evaluate our approach and baselines. We assign each client 150 labeled images and the rest as unlabeled images. We randomly sample clients with $\text{frac} = 1$ for each communication round. The results are shown in Figure 7.

As shown in Figure 7, our method outperforms all baselines in scenarios with different data skew degrees, which is consistent with our previous observations. The experiments on the real scenario demonstrate the effectiveness of our approach.

(2) The Importance of Distillation for Local Update. In the following description, when the pFedSD appears alone, it refers to pFedSD-hot. To understand the importance of knowledge distillation in pFedSD, we evaluate the performance of both methods in pFedSD using knowledge distillation ($\beta=1, \gamma=1$) and without knowledge distillation ($\beta=1, \gamma=0$). Not using knowledge distillation in pFedSD

TABLE 1: Comparison of the number of parameters of pFedSD-hot and baselines transmitted between the server and clients in each communication round.

Methods	The number parameters each communication round $\times 10^3$
FedMD	100
DS-FL	100
MHAT	100
pFedSD-soft	100
pFedSD-hot	55

FIGURE 6: In the case of homogeneous and heterogeneous models, the influence of different public dataset sizes on the test performance of (a, b) pFedSD, and (c, d) all methods on MNIST with $\alpha = 10$, $\text{frac} = 0.8$, and $T = 100$. (a) Homo-models (pFedSD). (b) Hetero-models (pFedSD). (c) Homo-models (all). (d) Hetero-models (all). FIGURE 7: Test performance of pFedSD compared with baselines on COVID-19 X dataset. (a) $\alpha = 1$. (b) $\alpha = 10$. (c) $\alpha = 100$.

means that the client updates only locally. As shown in Figure 8, the test performance using knowledge distillation in pFedSD outperforms local only, which means that our proposed method can effectively capture the knowledge of other models to improve the performance of the local model.

(3) *The Impact of Using a Different Threshold for pFedSD.* To clarify the effects of different thresholds on pFedSD, we studied the results of different thresholds on pFedSD. The results are shown in Table 2; we found that a lower threshold

will lead to a lower pFedSD performance. This is because when the threshold is low, most of the unlabeled data will be incorrectly labeled, which results in a large amount of error that is consistent with the observations in FixMatch [6].

(4) *The Impact of Using Different Learning Rate Schedules for pFedSD.* Table 3 shows the results of our ablation study with different optimizers. We tried different learning rate parameters for different optimizers. As seen from Table 3, the SGD optimizer performs better than Adam in our proposed

FIGURE 8: Test performance of pFedSD compared with local only for different settings on MNIST and same model structures.

TABLE 2: Test performance comparison of different thresholds for FedSD on MNIST and same model structures.

τ	$\alpha = 10$	$\alpha = 0.1$
0.2	92.12	38.95
0.4	92.02	39.47
0.6	93.33	40.73
0.7	93.62	41.17
0.8	93.73	41.29
0.85	93.69	40.58
0.9	93.58	40.15
0.95	93.5	40.13

TABLE 3: Test performance comparison of different optimizer thresholds for FedSD on MNIST and the same model structures.

Optimizer	Learning rate	$\alpha = 10$	$\alpha = 0.1$
SGD	0.01	93.53	41.29
	0.03	87.46	40.71
	0.05	83.22	40.04
	0.001	91.18	41.03
	0.003	90.46	40.28
	0.005	87.66	40.11
	0.001	81.07	39.41
Adam	0.003	82.46	38.78
	0.005	82.42	38.66
	0.0001	86.36	39.65
	0.0003	86.19	39.03
	0.0005	85.82	38.98

method. For the same optimizer, it is evident that using different learning rates results in different pFedSD performances in Table 3. At the same time, the varying degree of the learning rate also leads to different degrees of performance.

5. Conclusions

In this article, to solve the heterogeneous problems of data and models in federated learning, we propose a personalized federated learning framework pFedSD based on adaptive

aggregation and semisupervised knowledge distillation, which provides personalized models for the clients by transferring knowledge of nonmodel parameters between the server and clients. To observe the same public data on clients, each client adopts a generative adversarial network to train a generator on local unlabeled data uploaded to the central server. According to the generator, the server generates synthetic samples of the clients' local data and takes high-quality samples as public data. Second, we use the local labeled and unlabeled clients' data to train with the public data to learn from other clients and then use the trained

model to predict the public data and upload model prediction to the server side. In particular, we propose an adaptive aggregation method based on JS divergence, which reduces the weight of the low-quality models on the server side and improves the quality of the aggregation model prediction, which enhances the performance of the personalized model on the client side. Finally, clients download the pseudo label information of the public data from the server and train it with the local data to obtain the personalization models. We demonstrate the superiority of the pFedSD test performance and communication efficiency in the experiments. In the future, we will try to explore methods other than knowledge distillation to solve data and model heterogeneity problems.

Data Availability

All data are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the Guangxi Natural Science Foundation (no. 2020GXNSFAA297075), the Guangxi “Bagui Scholar” Teams for Innovation and Research Project, the Guangxi Collaborative Innovation Center of Multi-Source Information Integration and Intelligent Processing, the Guangxi Key Laboratory of Trusted Software (no. KX202037), the Project of Guangxi Science and Technology (no. GuiKeAD 20297054), and the Guangxi Natural Science Foundation Project (no. 2020GXNSFBA297108).

References

- [1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning,” *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.
- [2] L. Yuan, Y. Guo, Y. Gong, C. Luo, J. Zhan, and Y. Huang, “An isolated data island benchmark suite for federated learning,” 2020, <https://arxiv.org/abs/2008.07257>.
- [3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, S. Ananda Theertha, and D. Bacon, “Federated learning: strategies for improving communication efficiency,” 2016, <https://arxiv.org/abs/1610.05492>.
- [4] D. Li and J. Wang, “Fedmd: Heterogenous Federated Learning via Model Distillation,” 2019, <https://arxiv.org/abs/1910.03581>.
- [5] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, “Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data,” 2021, <https://arxiv.org/abs/2008.06180>.
- [6] L. Sun and L. Lyu, “Federated Model Distillation with Noise-free Differential Privacy,” 2020, <https://arxiv.org/abs/2009.05537>.
- [7] L. Zhu, Z. Liu, and S. Han, “Deep leakage from gradients,” in *Neural Information Processing System*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, Eds., Springer, New York, NY, USA, 2019.
- [8] H. Yin, A. Mallya, A. Vahdat, J. M. Alvarez, J. Kautz, and Pavlo Molchanov, “See through gradients: image batch recovery via gradinversion,” 2021, <https://arxiv.org/abs/2104.07586>.
- [9] G. E. Hinton, Oriol Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015, <https://arxiv.org/abs/1503.02531>.
- [10] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, “Deep mutual learning,” in *CVPR*, pp. 4320–4328, 2018, <https://arxiv.org/abs/1706.00384>.
- [11] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr, “Cronus: robust and heterogeneous collaborative learning with black-box knowledge transfer,” 2019, <https://arxiv.org/abs/1912.11279>.
- [12] L. Hu, H. Yan, L. Li, Z. Pan, X. Liu, and Z. Zhang, “MHAT: an efficient model-heterogenous aggregation training scheme for federated learning,” *Information Sciences*, vol. 560, pp. 493–503, 2021.
- [13] S. Laine and T. Aila, “Temporal ensembling for semi-supervised learning,” 2017, <https://arxiv.org/abs/1610.02242>.
- [14] K. Sohn, D. Berthelot, N. Carlini et al., “Fixmatch: simplifying semi-supervised learning with consistency and confidence,” in *Neural Information Processing System*, L. Hugo, R. Marc’Aurelio, H. Raia, B. Maria-Florina, and H.-T. Lin, Eds., Springer, New York, NY, USA, 2020.
- [15] Q. Xie, Z. Dai, E. H. Hovy, T. Luong, and Q. Le, “Unsupervised data augmentation for consistency training,” in *Neural Information Processing System*, L. Hugo, R. Marc’Aurelio, H. Raia, B. Maria-Florina, and H.-T. Lin, Eds., Springer, New York, NY, USA, 2020.
- [16] C. T. Dinh, N. H. Tran, and T. D. Nguyen, “Personalized federated learning with moreau envelopes,” in *Neural Information Processing System*, L. Hugo, R. Marc’Aurelio, H. Raia, B. Maria-Florina, and H.-T. Lin, Eds., Springer, New York, NY, USA, 2020.
- [17] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, “Self-balancing federated learning with global imbalanced data in mobile systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 2021.
- [18] A. Fallah, A. Mokhtari, E. Asuman, and Ozdaglar, “Personalized federated learning with theoretical guarantees: a model-agnostic meta-learning approach,” in *Neural Information Processing System*, L. Hugo, R. Marc’Aurelio, H. Raia, B. Maria-Florina, and H.-T. Lin, Eds., Springer, New York, NY, USA, 2020.
- [19] V. Smith, C.-K. Chiang, M. Sanjabi, and S. Ameet, “Federated multi-task learning,” in *Neural Information Processing System*, I. Guyon, U. von Luxburg, S. Bengio et al., Eds., Springer, New York, NY, USA, 2017.
- [20] X. Lin, H. Chen, Y. Xu et al., “Federated Learning with Positive and Unlabeled Data,” <https://arxiv.org/abs/2004.10342>.
- [21] Y. Mizukami, K. Tadamura, J. Warrell, Li Peng, and S. Prince, “CUDA implementation of deformable pattern recognition and its application to MNIST handwritten digit database,” in *Proceedings of the 20th International Conference on Pattern Recognition, ICPR 2010*, vol. 23–26, pp. 2001–2004, Istanbul, Turkey, August 2010.
- [22] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: A Novel Image Dataset for Benchmarking Machine Learning Algorithms,” 2017, <https://arxiv.org/abs/1708.07747>.

- [23] A. Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*, University of Toronto, Toronto, Canada, 2012.
- [24] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proceedings of the International Conference Learning Representations*, Y. Bengio and Y. LeCun, Eds., , San Juan, Puerto Rico, May 2016.
- [25] H. Tzu-Ming Harry, H. Qi, and M. Brown, "Measuring the Effects of Non-identical Data Distribution for Federated Visual Classification," 2019, <https://arxiv.org/abs/1909.06335>.
- [26] L. Wang, Z. Q. Lin, and A. Wong, "Covid-net: a tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images," *Scientific Reports*, vol. 10, no. 1, pp. 19549–19612, 2020.