https://doi.org/10.1007/s11704-025-40840-4

RESEARCH ARTICLE

FedPD: personalized federated learning based on partial distillation

Xu YANG[1], Ji-Yuan FENG[1,2], Song-Yue GUO[1], Bin-Xing FANG[1], Qing LIAO[1,2]✉

1. School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China
2. Department of New Networks, Peng Cheng Laboratory, Shenzhen 518000, China

Received August 12, 2024; accepted February 17, 2025

E-mail: liaoqing@hit.edu.cn

© The Author(s) 2025. This article is published with open access at link.springer.com and journal.hep.com.cn

Abstract

In recent years, personalized federated learning (PFL) has gained widespread attention for its robust performance in handling heterogeneous data. However, most PFL methods require client models to share the same architecture, which is impractical in real-world scenarios. Therefore, federated distillation learning is proposed, which allows clients to use different architecture models for FL training. Nevertheless, these methods do not consider the importance of different distillation knowledge aggregated by the client knowledge, resulting in poor client collaboration performance. In this paper, we propose a novel personalized federated learning method based on partial distillation (FedPD) that assesses the relevance of the different distillation knowledge and ensemble knowledge for each client, thereby achieving selective knowledge transfer. Specifically, FedPD contains two key modules. One is the partial knowledge transfer (PKT) which uses the partial distillation coefficient to identify the importance of each distillation knowledge to select more valuable distillation knowledge. The other is the partial knowledge ensemble (PKE), which maintains a server model for each client to extract distillation knowledge to guide the client. Extensive experiments on real-world datasets in various experimental settings show that FedPD significantly improves client model performance compared to state-of-the-art federated learning methods.

Keywords

federated learning; knowledge distillation; model heterogeneity

1 Introduction

Federated learning (FL) jointly trains models based on the collaboration of various clients while protecting the data privacy of each client [1]. In FL, clients collaborate to train models without sharing local private data. During FL training, participating clients upload their model parameters trained on local data to the server, which then aggregates client models. Currently, FL is widely used in fields such as healthcare [2,3], smart city [4,5], and recommendation systems [6–8].

Although FL has been successful in practical applications, it still faces some challenges [8–14]. For example, considering the client's dataset is often non-independent and identically distributed (non-IID), personalized federated learning (PFL) is proposed to train a personalized local model for each client. The recent PFL works mainly include the regularization-based methods [15–17] that add a regularization term of the global model to improve the local model performance, similarity-based methods [18,19] that reinforce the collaboration between clients with similar data distributions, and model architecture-based methods [20–22] that utilize partial aggregation model parameters to improve local models. Despite the promising results achieved by these PFL methods, they require assuming that client models have the same architecture, which is impractical in the real world.

To break the constraints of the same architecture model, many federated distillation methods have been proposed. The federated distillation method uses the public database to achieve knowledge transfer between heterogeneous client models, thereby addressing the issue of model heterogeneity [23–28]. For example, FedMD [25] utilizes the client model to generate soft predictions for public dataset samples as local knowledge, then distills the average aggregated global knowledge to the local model. KT-pFL [28] enhances distillation by aggregating personalized global knowledge based on the similarity of local knowledge for each client. FedHeNN [26] takes the representations generated by distillation samples as local knowledge from clients, achieving client collaboration by aligning client knowledge with the averagely aggregated global knowledge. However, these methods do not take into account the importance of different distillation knowledge to the client, leading to inefficient collaboration among client models.

We propose a novel personalized federated learning method based

on partial distillation (FedPD), which uses partial distillation to selectively transfer knowledge based on the client data distribution, thereby enhancing the heterogeneous personalized model performance. FedPD tackles the challenges of model heterogeneity with two key modules. First, we propose the partial knowledge transfer (PKT) module which utilizes the partial distillation coefficient to identify the importance of each distillation sample, filtering global knowledge to benefit the client. Second, we propose the partial knowledge ensemble (PKE) module which extracts distillation knowledge for each client to improve client model performance. The main contributions are summarized as follows:

- We propose a novel personalized federated learning method called FedPD, which leverages partial distillation to achieve selective knowledge transfer, thereby improving the performance of client models.
- We propose the PKT module, which uses the partial distillation coefficient to measure the importance of different distillation knowledge, enabling more effective distillation for clients.
- We propose the PKE module, which integrates distillation knowledge for each client to guide their training, thereby enhancing client model performance.
- We evaluate the FedPD on various datasets and settings. Extensive experiments show that the proposed method achieves the best performance.

2 Related work

Personalized federated learning has attracted much attention in addressing non-IID challenges in FL. For example, some regularization-based methods improve client model performance by adding regularization terms about the global model during local training. Li et al. [15] proposed Ditto, which enhances client model performance by adding regularization terms between the client model and the global model. Dinh et al. [16] proposed pFedMe, which optimizes both the local and global models by adding L_2 loss between the client model and global model, thereby improving the generalization performance of the client model. Some similarity-based methods improve client model performance by encouraging collaboration among clients with similar data distributions. Ghosh et al. [29] proposed IFCA, which groups clients according to the minimum loss of client data on each group model, and the clients in the group share the model. Liu et al. [30] proposed PFA, which uses the distance between the privacy representations of local data to build a data similarity matrix for each client and groups clients according to the similarity matrix. Huang et al. [18] proposed FedAMP, which maintains a personalized cloud model based on similarity aggregation for each client on the server and trains a personalized model for each client based on these cloud models. Another type of solution is architecture-based methods, which improve client model performance by dividing the model into private and shared parts. Collins et al. [20] proposed FedRep, which splits the client model into a private classifier and a shared feature extractor, enhancing client models by sharing the global feature representation. Sun et al. [31] proposed PartialFed, where each client uses only part of the global model parameters, selected through an automatically generated strategy, to boost performance. Zhang et al. [22] proposed FedALA, which adaptively aggregates the global model and the local model according to each client's objective to initialize the local model, thereby achieving better client model performance. However, these approaches require all clients to use the same architecture model.

Recent research has started focusing on FL with different model architectures. Federated distillation learning is proposed to transfer knowledge by aligning local client knowledge with global knowledge, enabling clients to share useful information and improve model performance [32–34]. For example, Lin et al. [24] proposed FedDF, which aggregates models with the same architecture and uses average aggregated knowledge for distillation to update models with the same architecture. Makhija et al. [26] proposed FedHeNN that utilizes Centered Kernel Alignment (CKA) distance as the distillation loss to perform distillation, aiming to align client representations with the average aggregated global representations. Cho et al. [27] proposed FedET that uses distillation to transfer knowledge from multiple local models to a global model on an unlabeled proxy dataset. He et al. [35] proposed FedGKT that uses knowledge distillation to transfer client model knowledge to a large server model. Zhang et al. [28] proposed KT-pFL, which utilizes the knowledge collaboration matrix to aggregate personalized group knowledge for each client on the server, using this personalized global knowledge as guidance knowledge. Wang et al. [32] proposed DaFKD, which optimizes the aggregation of client predictions through a discriminator that identifies the correlation between distillation samples and client models, aligning the client's predictions accordingly during distillation. Tan et al. [36] used the average representation of local data samples as prototypes and introduced a prototype regularization term in the local loss function. He et al. [35] achieved knowledge transfer by minimizing the predictions of the server model and the soft predictions of the client. However, most existing federated distillation learning methods do not consider the importance of different global knowledge in local distillation, which may introduce redundant information for the client. In this paper, we propose FedPD, which selects valuable knowledge for distillation to adapt to local goals across different clients, thereby improving client model performance.

3 Methodology

3.1 Preliminary

We aim to collaboratively train personalized models for N total clients with different model architectures in FL. We consider each client n can only access to its local dataset $D_n = \{\boldsymbol{x}_i, y_i\}$, where $\boldsymbol{x}_i$ is the ith sample, with label y_i. $D = \cup_{n=1}^{N} D_n$ is composed of all client datasets, containing K classes. The goal of PFL is to learn a local model for each client, minimizing the total empirical loss of the dataset on the client. For the client n, we denote its personalized objective as:

$$\mathcal{F}_n := \mathbb{E}_{(\boldsymbol{x}_i, y_i)\sim D_i} L_n(\boldsymbol{x}_i, y_i; \boldsymbol{\omega}_n),$$

where $L_n(\cdot)$ and $\boldsymbol{\omega}_n$ are the local loss function and model parameter of client n. For all clients, their objective is:

$$\{\boldsymbol{\omega}_1, ..., \boldsymbol{\omega}_N\} = \arg\min \mathcal{G}(\mathcal{F}_1, ..., \mathcal{F}_N),$$

where $\mathcal{G}$ is a function that aggregates the local objectives $\{\mathcal{F}_n\}_{n\in[N]}$ of each client.

We refer to the features extracted by client models from public dataset samples as client knowledge, and the features extracted by the server model as server knowledge. The public dataset $\hat{D}$ is stored on the server and each client as in the typical federated distillation setting [25,28]. For the ith sample $\hat{\boldsymbol{x}}_i \in \hat{D}$, the distillation knowledge $\boldsymbol{z}_{n,i}$ and $\boldsymbol{z}^s_{n,i}$ extracted by the client model $\boldsymbol{\omega}_n$ and the server model $\boldsymbol{\omega}^s_n$ are defined as follows:

$$\boldsymbol{z}_{n,i} = f(\hat{\boldsymbol{x}}_i; \boldsymbol{\omega}^f_n),$$

$$\boldsymbol{z}^s_{n,i} = H(\hat{\boldsymbol{x}}_i; \boldsymbol{\omega}^s_n),$$

where $f(\cdot)$ is the feature extractor of client model $\boldsymbol{\omega}_n$ with parameters $\boldsymbol{\omega}^f_n$; $H(\cdot)$ is output function of server model with parameters $\boldsymbol{\omega}^s_n$. Note that the server model consists of the feature extractor part and output layer, without the classifier. The output of the server model is the server knowledge. All server models have the same architecture. The size of the last output layer corresponds to the length of the specific client knowledge. The local knowledge of client n is $Z_n = \{\boldsymbol{z}_{n,i}\}_{i=1}^{|\hat{D}|}$. Similarly, the server model extracts the distillation knowledge to guide client n as $Z^s_n = \{\boldsymbol{z}^s_{n,i}\}_{i=1}^{|\hat{D}|}$.

3.2 FedPD

In this section, we introduce the proposed FedPD in detail. The proposed FedPD consists of two key components: partial knowledge transfer (PKT) and partial knowledge ensemble (PKE). Figure 1 shows the workflow of FedPD. First, clients extract features from public dataset samples and upload them to the server. Second, the server uses PKE to ensemble knowledge from various clients and generate distilled knowledge for each client. Finally, each client uses PKT to assess the importance of the received server knowledge and update its local model parameters.

3.2.1 Partial knowledge transfer (PKT)

Global knowledge may contain useless information, such as missing class samples from other clients. Therefore, clients need to distinguish useful global knowledge based on its local model during distillation, rather than aligning with global knowledge indiscriminately. To address this, we propose the PKT module, which identifies the importance of each piece of distillation knowledge and only transfers partially useful distillation knowledge to different clients.

1) Partial distillation coefficient.

To identify the importance of different distilled knowledge, we introduce a partial distillation coefficient $\boldsymbol{\alpha}$ into the distillation loss. For client n, the partial distillation loss is defined as follows:

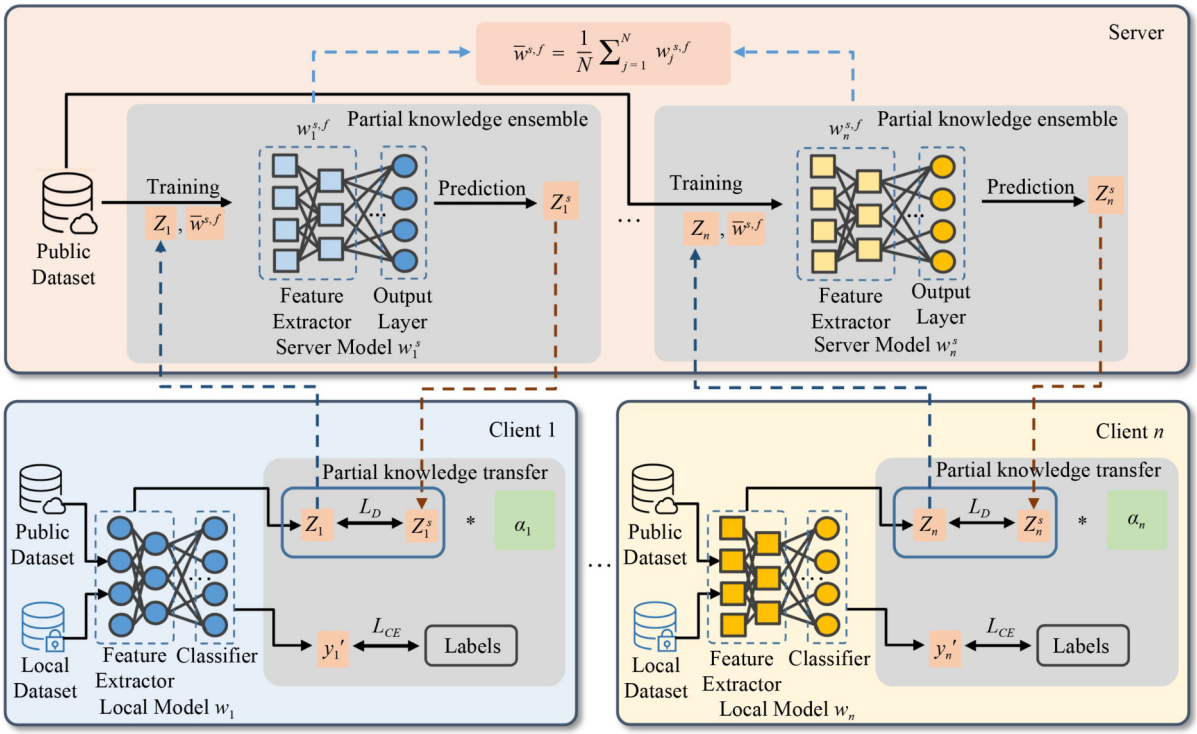


Fig. 1 The framework of FedPD. In each round, the client n selects distillation knowledge based on partial distillation coefficient α_n. The server model is trained based on the client knowledge of Z_n and the global basic model $\bar{\omega}^{s,f}$, and extracts the corresponding distillation knowledge Z^S_n for the client n

$$\min_{\omega_n,\boldsymbol{\alpha}_n} L_D(\omega_n,\boldsymbol{\alpha}_n)=\frac{\mathbf{1}}{|\hat{D}|}\sum_{i=1}^{|\hat{D}|}\boldsymbol{\alpha}_{n,i}L_1(f(\hat{\boldsymbol{x}}_i;\boldsymbol{\omega}_n^f),\boldsymbol{z}_{n,i}^s)+\frac{\tau}{2}\|\boldsymbol{\alpha}_n-\mathbf{1}\|^2, \quad (1)$$

where τ is a regularization hyperparameter greater than 0; $L_1(\cdot)$ stands for L_1 loss function. $\mathbf{1}$ is the vector of ones, where each element equals 1; $\boldsymbol{\alpha}_n$ represents the partial distillation coefficient vector for client n and $|\boldsymbol{\alpha}_n|=|\hat{D}|$, indicating the importance of each distillation knowledge. The regularization term $\|\boldsymbol{\alpha}_n-\mathbf{1}\|^2$ increases generalization and prevents clients from assigning too high weight to knowledge with large differences. The regularization term prevents client models from overfitting to the knowledge of their corresponding server models. It constrains the importance weights assigned by clients to different distillation knowledge to stay close to 1. Without this regularization term, clients tend to assign excessively high weights to knowledge that differs significantly from their local knowledge.

2) Optimization objective.

The objective of FedPD is to solve the joint optimization problem of clients with data and model heterogeneity. FedPD achieves collaboration among heterogeneous clients by introducing partial distillation loss while minimizing the loss of local training. The personalized loss function of client n is defined as follows:

$$\min_{\omega_n} L_n(\omega_n)=L_{CE}(\omega_n)+\lambda L_D(\omega_n;\boldsymbol{\alpha}_n), \quad (2)$$

where λ is a distillation hyperparameter. $L_{CE}(\cdot)$ is the cross-entropy loss function, defined as follows:

$$L_{CE}(\omega_n)=-\sum_{(\boldsymbol{x}_i,y_i)\sim D_n}\log p(y_i|\boldsymbol{x}_i;\omega_n),$$

where $p(y_i|\boldsymbol{x}_i;\omega_n)$ is the probability that the sample $\boldsymbol{x}_i$ is predicted to be y_i by client model ω_n.

3) Update partial distillation coefficient.

Furthermore, to adaptively adjust the partial distillation coefficients $\boldsymbol{\alpha}_n$ in each training round, we propose updating $\boldsymbol{\alpha}_n$ alternately with model parameters. This allows computing $\boldsymbol{\alpha}_n$ for different knowledge based on the model parameters. By adopting this alternating optimization method, the local model can effectively converge towards an optimal solution.

Specifically, we first fix ω_n on the client n to calculate the partial distillation coefficient $\boldsymbol{\alpha}_n$. According to Eq. (5), the update of $\boldsymbol{\alpha}_n$ is written as follows:

$$\boldsymbol{\alpha}_n\leftarrow\boldsymbol{\alpha}_n-\eta_{\boldsymbol{\alpha}_n}\nabla_{\boldsymbol{\alpha}_n}L_D(\omega_n,\boldsymbol{\alpha}_n), \quad (3)$$

where $\eta_{\boldsymbol{\alpha}_n}$ is the learning rate of $\boldsymbol{\alpha}_n$. After updating $\boldsymbol{\alpha}_n$, we fix $\boldsymbol{\alpha}_n$ to solve the model parameter ω_n.

4) Update client model.

Then the local model parameter ω_n is updated during local private data training and partial distillation. According to Eq. (6), the update of client n local model parameters is written as following:

$$\omega_n\leftarrow\omega_n-\eta_{\omega_n}\nabla_{\omega_n}L_n(\omega_n), \quad (4)$$

where η_{ω_n} is learning rate of ω_n. FedPD builds a positive feedback loop that can adaptively select more valuable distillation knowledge for the client model in each round and optimize local model parameters to improve overall model performance.

3.2.2 Partial knowledge ensemble (PKE)

Considering the challenge of data heterogeneity, a single aggregated global knowledge cannot meet the personalized needs of all clients, and significant biases may exist in local knowledge, potentially compromising the quality of linear aggregated global knowledge. As a result, generating individual global knowledge for each client is necessary. In this work, we propose the PKE module to partially ensemble distillation knowledge and extract distillation knowledge for each client to guide the client model training.

1) Server model.

First, we establish a corresponding server model for each client on the server side. All server models have feature extractors with the same architecture. The purpose of the server model is to extract knowledge from the corresponding client model and convert heterogeneous knowledge into a unified representation with the same architecture.

Then, we utilize the client n knowledge Z_n to train the server model ω_n^s, transferring the representation knowledge from the client model to the server model. Since the server models have the feature extractor with the same architecture, we can use the server models to achieve collaboration between heterogeneous client models. To collaborate with different clients and retain certain client information, a regularization term is introduced to the loss function. The loss function of the server model ω_n^s is defined as follows:

$$\min_{\omega_n^s} L(\omega_n^s)=\frac{1}{|\hat{D}|}\sum_{i=1}^{|\hat{D}|}L_s(H(\hat{\boldsymbol{x}}_i;\omega_n^s),\boldsymbol{z}_{n,i})+\mu L_R(\omega_n^{s,f},\bar{\omega}^{s,f}), \quad (5)$$

where μ is a server regularization hyperparameter; $\omega_n^{s,f}$ represents the feature extractor parameters of the server model ω_n^s; $\bar{\omega}^{s,f}$ is the global basic model which contains the global representation knowledge of all clients; $L_s(\cdot)$ is the loss function for server model training, and we use L_1 loss as the loss function; $L_R(\cdot)$ is the server regularization term used to achieve collaboration between $\omega_n^{s,f}$ and $\bar{\omega}^{s,f}$, which is defined as:

$$L_R(\omega_n^{s,f},\bar{\omega}^{s,f})=\|\omega_n^{s,f}-\bar{\omega}^{s,f}\|^2.$$

2) Global basic model.

The global basic model is defined as follows:

$$\bar{\omega}^{s,f}=\frac{1}{N}\sum_{n=1}^{N}\omega_n^{s,f}. \quad (6)$$

Based on previous work [20,37], the global basic model $\bar{\omega}^{s,f}$ aggregated by all model feature extractors naturally contains the global representation information of all clients. The regularization

term $L_R(\cdot)$ allows the server model to retain personalized preferences while approaching global representation knowledge and achieving the ensemble of partial global knowledge.

3) Update server model.

According to Eq. (10), the update of server model ω_n^s is as follows:

$$\omega_n^s \leftarrow \omega_n^s - \eta_{\omega_n^s} \nabla_{\omega_n^s} L(\omega_n^s), \tag{7}$$

where $\eta_{\omega_n^s}$ is learning rate.

After server models training, server model ω_n^s extracts partial global knowledge Z_n^s to guide the client n. Compared with the global knowledge generated by the linear combination of local knowledge, PKE captures more complex relationships between local knowledge and makes full use of the rich information within client knowledge.

We summarize all the procedures of FedPD on the server and client in Algorithm 1. Each client finally gets a personalized model after FL training. In each training round, clients first extract local knowledge for public distillation data (Line 4). Then the server partially ensembles the client knowledge and extracts guiding knowledge for each client (Line 5). Finally, the clients utilize local data to update model parameters based on the server knowledge guidance (Line 6). Partial distillation is implemented in two parts: first, the PKE module partially ensembles knowledge for each client (Lines 11−14), and second, the PKT module achieves selective distillation based on partial distillation coefficients (Lines 17−19). These two parts aim to filter valuable collaborative information for clients as much as possible, thereby enhancing collaboration effectiveness.

Algorithm 1 FedPD

Input: client model $\omega_1, ..., \omega_n$, server model $\omega_1^s, ..., \omega_N^s$, communication round T

Output: personalized client models $\omega_1, ..., \omega_N$

1 **for** $t = 1, 2, ..., T$ **do**
2 Server randomly selects participating clients
3 **for** *each client* $n = 1, ..., N$ ***in parallel*** **do**
4 $Z_n \leftarrow f(\omega_n^f;)$
5 $Z_n^s \leftarrow$ **ServerEnsemble**$(Z_n, \omega_n^s, \bar{\omega}^{s,f})$
6 **ClientUpdate**(Z_n^s, ω_n)
7 **end**
8 Calculate $\bar{\omega}^{s,f}$ for next round by Eq. (12)
9 **end**
10 **ServerEnsemble** $(Z_n, \omega_n^s, \bar{\omega}^{s,f})$:
11 **for** *each epoch* **do**
12 Train server model ω_n^s using Z_n and $\bar{\omega}^{s,f}$ by Eq. (13)
13 **end**
14 $Z_n^s \leftarrow F(\omega_n^s; \hat{D})$
15 **return** Z_n^s
16 **ClientUpdate** (Z_n^s, ω_n) :
17 **for** *each local epoch* **do**
18 Calculate partial distillation coefficient α_n using Z_n^s and ω_n by Eq. (8)
19 Update local model ω_n by Eq. (9)
20 **end**

4) Complexity analysis.

FedPD consists of two key modules: PKT and PKE. Assume that the size of the public dataset is M, the number of client model parameters is W, the number of server model parameters is P, and the feature size of each sample is d. When all clients participate in training, in the PKT module, the computational complexity for calculating the importance coefficient α_n is approximately $O(dMN)$ and the complexity of extracting client knowledge is about $O(WMN)$. In the PKE module, the computational complexity for model training and knowledge extraction is about $O(PMN)$. Therefore, the total computational complexity of FedPD is $O((d+W+P)MN)$. In each round, clients need to upload and download client knowledge and server knowledge respectively. The communication cost for each client in each communication round is about $2dM$.

4 Experiments

4.1 Experimental setup

4.1.1 Datasets

We evaluate the effectiveness of the proposed method through image classification tasks on these four datasets. The details are as follows:

- CIFAR-10 [38] dataset consists of 60, 000 32 × 32 images, containing 10 classes, each with 6, 000 images. The training set and test set have 50, 000 and 10, 000 images, respectively.
- CIFAR100 [38] is similar to CIFAR-10, but it has 100 classes. There are 60, 000 images in total, and the size of each image is 32 × 32 pixels. Each class contains 500 training images and 100 test images.
- EMNIST [39] is the handwritten character dataset derived from NIST Special Database 19, comprising 62 classes. Each image in the dataset represents a handwritten digit in a 28 × 28 format. There are 814, 255 images in total.
- Fashion-MNIST (FMNIST) [40] contains a training set of 60, 000 samples and a test set of 10, 000 samples. Each image is a 28 × 28 grayscale image containing 10 classes.

For each dataset, we randomly allocate data for each client. Following [20,41], we use the data of the first 10 classes of CIFAR100 and EMNIST. Since it is easy to collect various public data samples in the real world, We select 100 samples from each class as the public dataset samples. The public dataset samples do not appear in the clients' training or testing sets. Additionally, to evaluate the impact of different public datasets on the proposed method, we conduct experiments using samples from different classes of other datasets as public dataset samples (Subsection 4.4).

4.1.2 Data segmentation

Following previous works [24,27,32,42], we use Dirichlet distribution to allocate data to each client. To verify the impact of different data heterogeneity levels on model performance, we adopt two non-IID experimental settings of $\beta = 0.1$ and $\beta = 0.5$. The size of the Dirichlet distribution coefficient β controls the degree of data

heterogeneity. As shown in Fig. 2, the smaller β is, the higher the data heterogeneity is, and vice versa. Figure 2 illustrates the data distribution of 20 clients on FMNIST, where each column represents the number of samples of each class that the client has, and the size of the points represents the number of samples.

4.1.3 Models

We set up models of different scales depending on the difficulty of the client classification task. Following the settings of [28], for the CIFAR10 and CIFAR100 dataset, we select four different architectures of models LeNet [43], ResNet-18 [44], MobileNetv2 [45], and ShuffleNetV2 [46] as client models. The server models use ResNet-34 [44]. For the EMNIST and FMNIST datasets, we use four types of models as client models, including two MLP models and two CNN models, as in previous works [20,35,47]. And the server model is a larger-scale CNN model. The four types of client models are evenly assigned among the clients.

4.1.4 Comparison methods

We compare FedPD with six popular baseline methods that support heterogeneous model, including:

- FedMD [25] averages the predictions of distillation samples uploaded by the client as global knowledge and aligns local knowledge with global knowledge during the distillation process.
- FedDF [24] averages the model parameters of models with the same model architecture and updates the average global model based on the predictions of all clients.
- KT-PFL [28] uses the similarity of client soft predictions to aggregate other clients' predictions as the client's global knowledge.
- FedHeNN [26] utilizes CKA distance to align local knowledge with global knowledge based on the weighted average of local knowledge.
- FedProto [36] uses the average representation generated by each type of sample of local data as the prototype. The client introduces the regularization term of the prototype in the local update to update the model.
- FedHKD [42] aggregates each class's prototype and soft prediction as hyper-knowledge and adds hyper-knowledge regularization terms to the local model update.

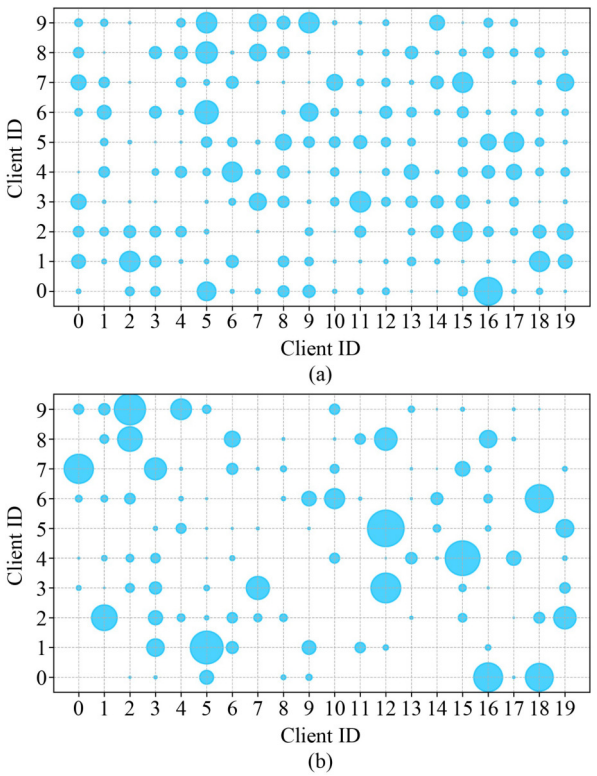


Fig. 2 Non-IID data distributions of the 20 clients on FMNIST. (a) $\beta = 0.5$; (b) $\beta = 0.1$

4.1.5 Implementation details

We evaluate FedPD under various experimental settings. For all datasets, the number of clients is set to 20 and 50. Considering that clients may not be able to participate in every training round in real-world scenarios, we randomly select only 20% of the clients to participate in FL training in each round. All experiments are conducted on eight NVIDIA Tesla V100 GPUs with 32Gb memory. The proposed FedPD is implemented on Pytorch 1.12 in Python 3.7 environment. All additional hyperparameters in the compared methods use their original settings. We run all methods for 200 communication rounds. We execute five local epochs of SGD with momentum to train the local model. For the CIFAR10 and CIFAR100 datasets experiments, we set the client local model learning rate $\eta_{\omega_n} = 0.005$ and batch size $b = 40$. For EMNIST and FMNIST datasets, client model learning rate $\eta_w = 0.01$ and batch size $b = 20$. In all experiments, for FedPD, the best performance is obtained by setting the server model learning rate $\eta_{\omega_n^s} = 0.001$, batch size $b = 40$, and setting the partial distillation coefficient α learning rate $\eta_{\alpha_n} = 0.05$. Same as work [20,21], we evaluate all methods by the average accuracy of the last 10 training epochs for all clients on the local test set.

4.2 Experimental results and analysis

4.2.1 Performance comparison

Table 1 shows the average client model test accuracy (ACC), Precision (PRE), Recall (REC), and AUC-ROC (AUC) of the comparison methods and FedPD in low data heterogeneity ($\beta = 0.5$) settings on the CIFAR10, CIFAR100, EMNIST, and FMNIST datasets. The best-performing method is marked in bold, and the second-best method is underlined. When the number of clients is 20, on the CIFAR10 dataset, FedPD outperforms the best baseline method KT-pFL by 2.78% in the ACC metric and outperforms FedHKD by 8.02% in the AUC metric. Additionally, our method outperforms the best baseline FedHKD by 1.74% and 2.23% in ACC and PRE metrics on the FMNIST dataset, respectively. When the number of clients is 50, FedPD outperforms the best method, FedHKD, by 6.93% and 4.57% in the ACC and REC metrics on the CIFAR10, respectively, and by 1.61% and 1.82% on FMNIST. This is because tasks on the CIFAR10 and CIFAR100 are more challenging compared to other datasets, leading to greater

Table 1 Test accuracy (ACC), Precision (PRE), Recall (REC), and AUC-ROC (AUC) (%) of FedPD and other baseline methods on CIFAR10, CIFAR100, EMNSIT, and FMNIST datasets and various heterogeneous settings with β=0.5

20 clients, β=0.5																
Method	CIFAR10				CIFAR100				EMNIST				FMNIST			
	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC
FedMD$_{[NIPS19]}$ [25]	41.24	25.90	41.30	65.94	45.89	23.90	36.91	67.55	80.30	73.33	80.47	97.69	71.64	63.35	71.73	92.67
FedDF$_{[NIPS20]}$ [24]	46.44	34.10	42.88	65.43	44.48	33.23	39.97	66.53	83.23	80.14	84.45	97.22	78.17	73.98	75.93	94.50
KT-pFL$_{[NIPS21]}$ [28]	47.03	25.97	41.09	66.02	43.85	20.54	33.24	64.86	82.76	79.02	83.33	97.79	76.06	71.82	75.19	93.23
FedHeNN$_{[ICML22]}$ [26]	45.07	32.04	45.07	68.91	45.32	35.11	45.54	75.25	86.03	84.81	85.83	97.35	78.62	77.10	78.90	94.55
FedProto$_{[AAAI22]}$ [36]	44.04	31.28	41.35	68.25	47.76	35.36	39.96	69.40	76.85	65.04	73.45	93.17	71.46	59.67	69.04	89.77
FedHKD$_{[ICLR23]}$ [42]	44.32	31.15	44.19	69.34	46.72	25.86	37.44	68.97	86.30	85.00	85.96	97.53	79.15	77.86	79.53	95.01
FedPD(Our)	**49.81** (2.78↑)	**40.37** (6.27↑)	**50.20** (5.13↑)	**77.36** (8.02↑)	**49.01** (1.25↑)	**38.12** (2.76↑)	**47.88** (2.34↑)	**77.11** (1.86↑)	**88.04** (1.74↑)	**87.06** (2.06↑)	**87.55** (1.59↑)	**97.84** (0.05↑)	**80.89** (1.74↑)	**80.09** (2.23↑)	**80.86** (1.33↑)	**95.63** (0.62↑)
50 clients, β=0.5																
Method	CIFAR10				CIFAR100				EMNIST				FMNIST			
	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC
FedMD$_{[NIPS19]}$ [25]	43.64	28.71	43.44	69.02	40.59	27.70	40.19	70.30	84.03	79.61	84.03	97.57	73.76	67.52	73.57	93.93
FedDF$_{[NIPS20]}$ [24]	43.57	33.71	44.07	73.70	43.29	31.93	43.69	70.93	87.34	86.40	87.60	97.22	77.67	76.91	77.69	94.58
KT-pFL$_{[NIPS21]}$ [28]	42.73	33.27	43.87	73.05	44.80	34.17	45.18	74.78	85.52	83.69	85.20	97.66	76.60	75.78	77.29	94.34
FedHeNN$_{[ICML22]}$ [26]	46.39	35.19	46.09	72.76	42.31	31.14	42.13	71.25	86.93	85.91	86.28	97.59	78.07	76.82	77.57	93.46
FedProto$_{[AAAI22]}$ [36]	46.40	36.21	45.25	72.45	42.27	28.84	41.23	70.26	86.40	84.90	84.33	97.05	78.43	77.76	78.25	94.95
FedHKD$_{[ICLR23]}$ [42]	46.72	36.97	47.50	73.73	42.38	29.63	41.68	70.81	87.35	87.39	87.67	97.68	78.79	78.90	79.20	95.47
FedPD(Our)	**53.65** (6.93↑)	**41.65** (4.68↑)	**52.07** (4.57↑)	**79.25** (5.52↑)	**52.01** (7.21↑)	**40.21** (6.04↑)	**50.27** (5.09↑)	**79.47** (4.69↑)	**88.78** (1.43↑)	**88.70** (1.31↑)	**88.63** (0.96↑)	**98.08** (0.40↑)	**80.40** (1.61↑)	**81.66** (2.76↑)	**80.72** (1.82↑)	**95.64** (0.17↑)

improvements.

Table 2 shows the results of the baseline methods and FedPD in high data heterogeneity ($\beta = 0.1$) settings on four datasets. When the number of clients is 20, FedPD outperforms the best baseline, FedHeNN, by 2.49% and 4.46% on CIFAR100, and surpasses FedHKD by 2.16% and 3.86% on FMNIST in terms of ACC and PRE, respectively. When the number of clients is 50, on the CIFAR100 dataset, FedPD outperforms the best baseline FedHKD by 5.44% in the ACC and outperforms FedHeNN by 6.83% in the AUC metric. FedPD consistently outperforms other methods in various heterogeneous settings because it allows clients to select more valuable knowledge during the distillation process, thereby enhancing model performance.

Table 3 shows the performance of FedPD under different non-IID settings. It can be seen that FedPD outperforms the best baseline methods by 2.78%, 2.05%, 3.30%, and 2.23% respectively under β=0.5, 0.1, 0.05, and 0.01. FedPD consistently achieved improvements across various experimental settings, demonstrating its effectiveness in addressing non-IID data challenges.

In addition to the client model heterogeneous setting, we also explore the performance of FedPD in the homogeneous model setting. We compare the performance of Scaffold [48], FedDC [49], and FedFed [50] on the FMNIST and EMNIST datasets with 20 clients. The results are shown in Table 4. It can be observed that FedPD outperforms other baseline methods in terms of the ACC metric and achieves better performance in most cases on other metrics, such as PRE, REC, and AUC.

4.2.2 Convergence evaluation

Figure 3 shows the average test accuracy curves during training on four datasets with high data heterogeneity ($\beta = 0.1$) and 20 clients, including all baseline methods and FedPD. The horizontal axis represents the communication rounds, and the vertical axis represents the test accuracy. As shown in Fig. 3(b), FedHeNN is the best baseline, but our method reaches a higher accuracy of 68.14%. As shown in Fig. 3(c), FedPD still surpasses the best baseline method FedHKD. This indicates that our method not only performs better but also converges faster. The reason is that FedPD adaptively selects distilled knowledge to adjust local models, alleviating the negative effects during the distillation process.

4.2.3 Impact of public dataset

Table 5 shows the impact of using different public datasets on FedPD performance. We compare the test accuracy of FedPD using different datasets as the public dataset. Note that even though the task dataset

Table 2 Test accuracy (ACC), Precision (PRE), Recall (REC), and AUC-ROC (AUC) (%) of FedPD and other baseline methods on CIFAR10, CIFAR100, EMNSIT, and FMNIST datasets and various heterogeneous settings with β=0.1

20 clients, β=0.1																
Method	CIFAR10				CIFAR100				EMNIST				FMNIST			
	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC
FedMD$_{[NIPS19]}$ [25]	66.91	53.06	66.58	76.86	63.15	50.96	63.58	79.98	89.85	86.33	90.48	98.54	85.47	79.28	85.28	96.16
FedDF$_{[NIPS20]}$ [24]	67.15	55.79	60.00	74.91	63.61	51.70	58.23	77.01	90.67	89.23	88.43	98.47	82.07	80.50	80.20	94.83
KT-pFL$_{[NIPS21]}$ [28]	67.80	53.02	65.56	74.25	62.37	48.28	61.75	74.46	91.59	88.91	91.64	98.80	87.56	83.59	87.02	96.33
FedHeNN$_{[ICML22]}$ [26]	67.89	58.62	67.62	76.70	65.65	53.68	63.36	78.97	92.26	90.90	92.34	98.59	87.73	83.51	86.78	96.45
FedProto$_{[AAAI22]}$ [36]	66.97	56.83	66.75	75.47	63.61	51.01	61.48	76.34	87.45	80.59	85.97	96.51	83.56	75.77	83.28	93.53
FedHKD$_{[ICLR23]}$ [42]	67.15	58.01	67.58	76.16	64.85	51.41	63.95	78.92	93.01	91.32	92.75	98.69	87.82	86.00	88.13	96.57
FedPD(Our)	**69.94** (2.05↑)	**62.75** (4.13↑)	**69.72** (2.10↑)	**79.45** (2.59↑)	**68.14** (2.49↑)	**58.14** (4.46↑)	**69.06** (5.11↑)	**83.68** (3.78↑)	**94.21** (1.20↑)	**93.74** (2.42↑)	**94.15** (1.40↑)	**98.91** (0.11↑)	**89.98** (2.16↑)	**89.86** (3.86↑)	**90.32** (2.19↑)	**97.02** (0.45↑)
50 clients, β=0.1																
Methods	CIFAR10				CIFAR100				EMNIST				FMNIST			
	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC
FedMD$_{[NIPS19]}$ [25]	68.69	57.42	68.86	73.99	67.32	54.87	66.67	74.47	90.86	87.47	90.83	97.96	87.14	83.22	87.34	95.35
FedDF$_{[NIPS20]}$ [24]	52.78	57.31	52.30	78.62	51.84	54.93	51.38	77.44	90.54	90.68	90.61	97.70	79.54	86.10	77.25	95.43
KT-pFL$_{[NIPS21]}$ [28]	69.53	59.55	69.43	71.32	66.01	55.62	65.98	70.18	92.04	90.21	91.99	98.37	88.30	85.79	88.35	95.89
FedHeNN$_{[ICML22]}$ [26]	72.10	64.08	71.77	77.29	68.17	58.19	68.23	77.47	92.54	90.73	92.56	97.87	88.72	87.12	88.77	95.82
FedProto$_{[AAAI22]}$ [36]	71.81	63.68	71.95	76.48	68.18	59.44	68.71	77.24	87.16	89.94	91.93	97.04	88.76	88.01	88.42	95.90
FedHKD$_{[ICLR23]}$ [42]	71.69	63.79	71.85	77.03	68.74	59.49	69.11	77.18	93.10	91.41	93.11	97.97	89.34	88.26	89.56	96.15
FedPD(Our)	**74.46** (2.36↑)	**69.65** (5.57↑)	**74.52** (2.57↑)	**82.58** (3.96↑)	**74.18** (5.44↑)	**69.60** (10.11↑)	**73.06** (3.95↑)	**84.30** (6.83↑)	**94.20** (1.10↑)	**93.83** (2.42↑)	**94.45** (1.34↑)	**98.27** (0.30↑)	**90.53** (1.19↑)	**90.62** (2.36↑)	**90.70** (1.14↑)	**96.42** (0.27↑)

Table 3 Test accuracy (%) of FedPD and other baseline methods on CIFAR10 datasets and various non-IID settings

Method	β=0.5	β=0.1	β=0.5	β=0.1
FedMD$_{[NIPS19]}$ [25]	41.24	66.91	76.25	90.25
FedDF$_{[NIPS20]}$ [24]	46.44	67.15	76.41	89.66
KT-pFL$_{[NIPS21]}$ [28]	47.03	67.80	77.41	91.25
FedHeNN$_{[ICML22]}$ [26]	45.07	67.89	77.65	92.51
FedProto$_{[AAAI22]}$ [36]	44.04	66.97	78.64	91.65
FedHKD$_{[ICLR23]}$ [42]	44.32	67.15	79.65	92.12
FedPD(Our)	**49.81** (2.78↑)	**69.94** (2.05↑)	**82.95** (3.30↑)	**94.74** (2.23↑)

and the public dataset are the same, the samples used for distillation do not appear in the client's private dataset. From Table 5, it is observed that replacing the public dataset in FedPD does not result in significant accuracy changes, with a specific decrease of 0.06% on EMNIST and 1.1% on FMNIST. This shows that FedPD does not rely heavily on unbiased public datasets and is robust to the setting of public datasets. But the performance increases by 0.81% on CIFAR100. The reason is that the partial distillation coefficient can adaptively assign different importance weights to different distillation knowledge, even if the public dataset samples differ significantly from local data.

4.2.4 Ablation study

To verify the effect of the PKT and PKE, we conduct the ablation experiments as shown in Table 6. The replacement of PKT is to set the same importance for all global knowledge, while the replacement of PKE is to averagely aggregate all client knowledge to obtain global knowledge. From Table 6, it can be observed that the accuracy of FedPD$_{\text{-V1}}$, which does not use both components, is the lowest at only 82.81%. FedPD$_{\text{-V2}}$ using only PKT, shows a 3.14% increase compared to baseline FedPD$_{\text{-V1}}$, while using only PKE increases accuracy by 2.52%. FedPD can increase the performance by 10% compared to the baseline by utilizing both components simultaneously. This indicates that: 1) the effectiveness of PKT in generating importance coefficients for different distillation knowledge; 2) the effectiveness of PKE in ensemble knowledge for clients. FedPD using both components simultaneously can achieve greater improvements.

Table 4 Test accuracy (ACC), Precision (Pre), Recall (REC), and AUC-ROC (AUC) (%) of FedPD and other baseline methods on EMNSIT, and FMNIST datasets and various homogeneous model settings

Method	β=0.5								β=0.1							
	EMNIST				FMNIST				EMNIST				FMNIST			
	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC	ACC	PRE	REC	AUC
Scaffold	85.50	88.70	85.26	**99.16**	74.10	72.83	74.11	95.56	89.35	88.45	89.64	98.95	82.77	78.71	82.10	92.55
FedDC	87.96	88.12	87.94	98.45	78.93	77.56	79.01	95.36	94.53	94.57	94.33	99.15	88.14	86.70	**88.51**	**96.98**
FedFed	88.66	**90.96**	**90.08**	98.64	76.05	73.78	75.80	95.33	93.36	93.19	93.65	99.08	86.99	85.32	87.57	96.23
FedPD	**89.07**	88.42	89.01	98.54	**80.58**	**80.23**	**80.35**	**95.76**	**94.78**	**94.85**	**94.91**	**99.31**	**88.44**	**88.02**	88.44	88.70

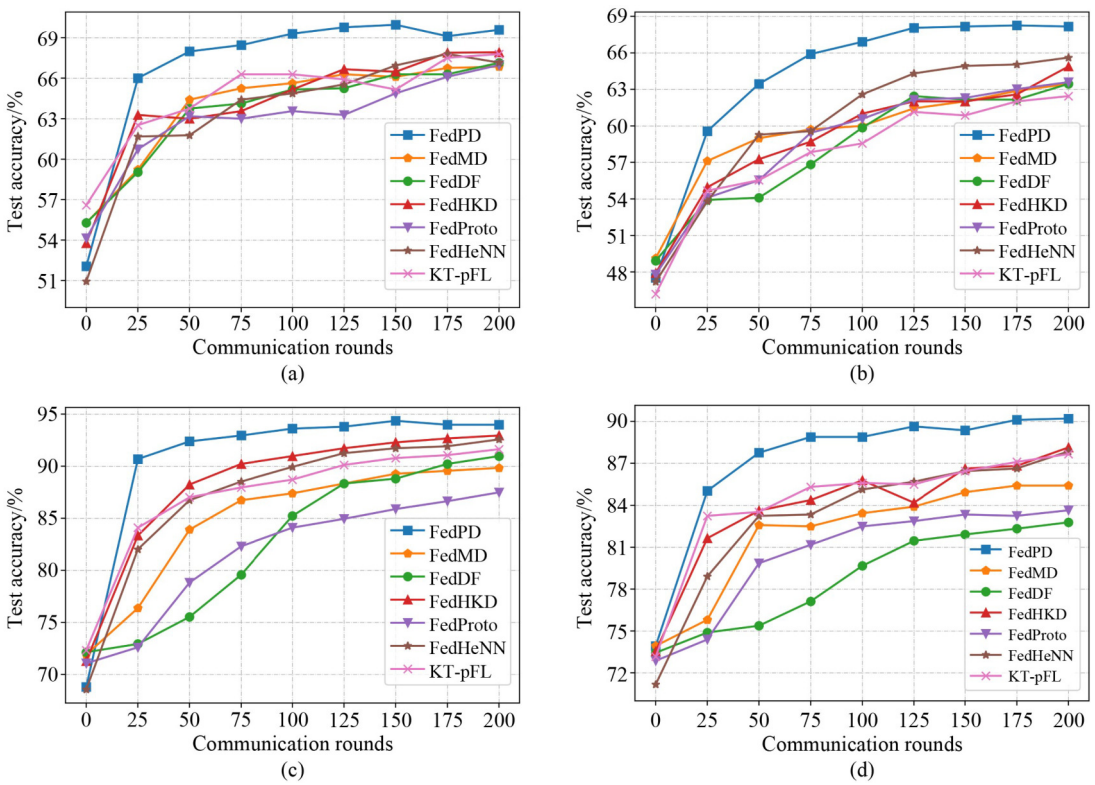


Fig. 3 Performance comparison in the accuracy of methods on different datasets, with 20 clients and $\beta = 0.1$. (a) CIFAR10; (b) CIFAR100; (c) EMNIST; (d) FMNIST

Table 5 Performance of FedPD under different settings of the public dataset

Task dataset	Public dataset	Test accuracy/%
CIFAR10	CIFAR10	49.81
	CIFAR100	47.16
CIFAR100	CIFAR100	49.01
	CIFAR10	49.82
EMNIST	EMNIST	87.98
	FMNIST	88.04
FMNIST	FMNIST	80.89
	EMNIST	79.79

Table 6 Ablation Study of FedPD in EMNIST dataset with 20 clients, imbalanced degree $\beta = 0.5$

Method	PKT	PKE	Test accuracy/%
$\text{FedPD}_{\text{-V1}}$	×	×	82.81
$\text{FedPD}_{\text{-V2}}$	√	×	85.95
$\text{FedPD}_{\text{-V3}}$	×	√	85.33
FedPD	√	√	88.04

4.2.5 Impact of hyperparameters

To evaluate the impact of hyperparameters on FedPD training, we conduct multiple experiments on four datasets.

Regularization parameter τ. As shown in Eq. (5), τ compromises the client model's personalization and generalization

capabilities. When τ increases, the partial distillation coefficients will approach 1, making the predictions closer to the server model's predictions. Table 7 shows the FedPD results for different τ values. The results indicate that an appropriate τ value needs to be selected to achieve better FedPD performance.

Server epoch *e*. The server model requires multiple epochs to learn client knowledge and perform partial knowledge ensemble to facilitate collaboration between clients. Therefore, we use various epoch settings to evaluate the impact of server epochs, with results shown in Table 8. The results show that a larger e is beneficial to the knowledge ensemble, but too large e is likely to damage FedPD performance.

Server regularization parameter μ. In Eq. (10), the server model's predictions are directly related to the server regularization parameter μ. The larger the μ, the more the knowledge extracted by the server model tends to average global knowledge, and vice versa. Table 9 shows the FedPD results under different μ settings. The results show that a larger μ does not bring better performance improvement, which means that a trade-off between local and global knowledge is needed to achieve the best performance.

Model learning rates. As the learning rate directly influences the model's training performance, we evaluate the performance of the client and server models using multiple learning rates η_{ω_n} and $\eta_{\omega_n^s}$ in Eqs. (9) and (13). Table 10 shows the performance of FedPD under different server model learning rates. It can be seen that a smaller server model learning rate will significantly affect the performance of FedPD. Table 11 shows the performance of FedPD under different client learning rates. It can be seen that an appropriate client model learning rate can achieve the best performance, and a lower or too high learning rate will damage the model performance.

Partial distillation coefficient learning rates η_{α_n}. We evaluate the performance of FedPD under different partial distillation coefficient α_n learning rate η_{α_n} in Eq. (8), as shown in Table 12. It can be observed that the proposed method is insensitive to the learning rate of the partial distillation coefficient. We select 0.05 as the learning rate for the partial distillation coefficient.

4.2.6 Case study

To demonstrate the effectiveness of FedPD, we conduct a case study as shown in Fig. 4. The separability of features extracted from each class is a key metric for evaluating the classification model. For

Table 7 Comparison of test accuracy (%) under different regularization parameter τ

Dataset	Regularization parameter τ			
	0.1	0.3	0.5	0.6
CIFAR10	47.02	48.35	**49.81**	48.61
CIFAR100	46.30	48.99	**49.01**	48.21
EMNIST	86.81	87.32	**88.04**	87.70
FMNIST	79.02	79.31	80.89	**80.91**

Table 8 Comparison of test accuracy (%) under different server epochs e

Dataset	Server epoch e			
	20	30	40	50
CIFAR10	49.31	**49.81**	49.73	48.39
CIFAR100	47.96	**49.01**	48.95	48.40
EMNIST	87.83	87.98	**88.04**	87.65
FMNIST	78.42	79.89	**80.89**	78.20

Table 9 Comparison of test accuracy (%) under different server regularization parameters μ

Dataset	Server regularization parameter μ			
	0.4	0.5	0.6	0.7
CIFAR10	48.79	49.35	**49.81**	48.68
CIFAR100	47.72	47.93	**49.01**	48.77
EMNIST	87.74	87.78	**88.04**	87.72
FMNIST	78.76	79.14	**80.89**	80.51

Table 10 Comparison of test accuracy (%) under different server model learning rates

Dataset	Server learning rates $\eta_{\omega_n^s}$			
	0.01	0.005	0.001	0.0005
CIFAR10	48.21	49.37	**49.81**	49.55
CIFAR100	47.99	48.57	**49.01**	46.86
EMNIST	87.30	87.32	**88.04**	76.63
FMNIST	80.40	**81.00**	80.89	80.80

Table 11 Comparison of test accuracy (%) under different client model learning rates

Dataset	Client learning rates η_{ω_n}			
	0.05	0.01	0.005	0.001
CIFAR10	48.92	49.19	**49.81**	45.37
CIFAR100	48.57	48.58	**49.01**	41.19
EMNIST	84.48	**88.04**	86.54	81.42
FMNIST	80.72	**80.89**	80.31	75.25

Table 12 Comparison of test accuracy (%) under different partial distillation coefficient learning rates η_{α_n}

Dataset	α_n learning rates η_{α_n}			
	0.1	0.05	0.01	0.005
CIFAR10	49.57	**49.81**	48.78	47.65
CIFAR100	47.62	**49.01**	47.55	47.21
EMNIST	87.29	**88.04**	87.59	87.79
FMNIST	80.35	**80.89**	80.55	80.67

client models, extracting more separable features benefits subsequent classification tasks. We show the features extracted from clients trained locally in Fig. 4(a). Features extracted from client models trained with averaged knowledge without FedPD are shown in Fig. 4(b). Features extracted from FedPD training are shown in Fig. 4(c). It can be seen that the features extracted by the client trained with FedPD are more distinguishable. The results also confirm the effectiveness of FedPD.

4.2.7 Scalability analysis

In this experiment, we evaluat the performance of FedPD under different client scales. Figure 5 shows the results on the FEMNIST and EMNIST datasets with $\beta = 0.1$ when the number of clients is 5, 20, 35, and 50. It can be observed that as the number of clients increases, the accuracy of FedPD improves and stabilizes. While computational costs rise with the number of clients, the accuracy

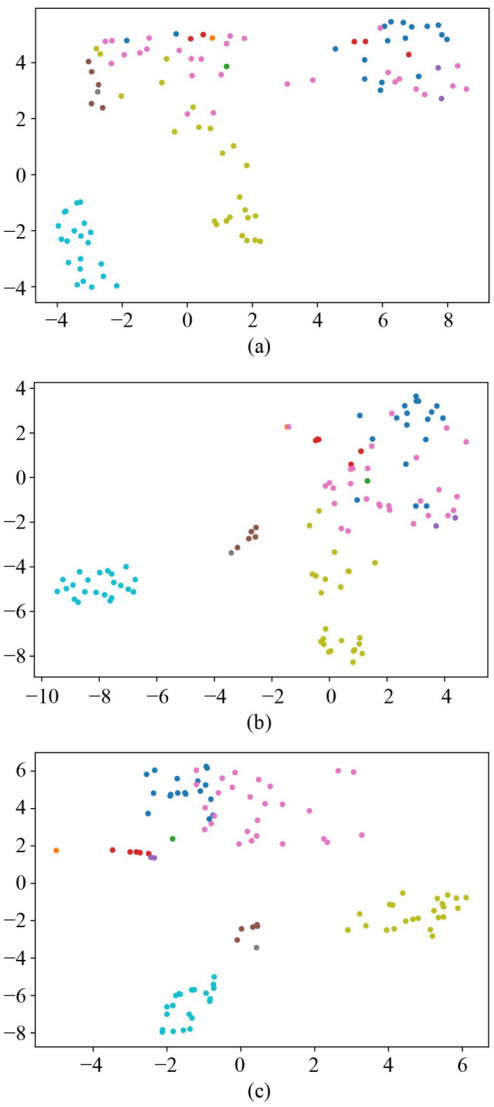


Fig. 4 t-SNE visualization of samples with (without) FedPD. (a) The features extracted from the 5th client model on the FEMNIST dataset with $\beta = 0.5$. (b) The features extracted by training client models with averaged client knowledge, without FedPD. (c) The features extracted with the FedPD

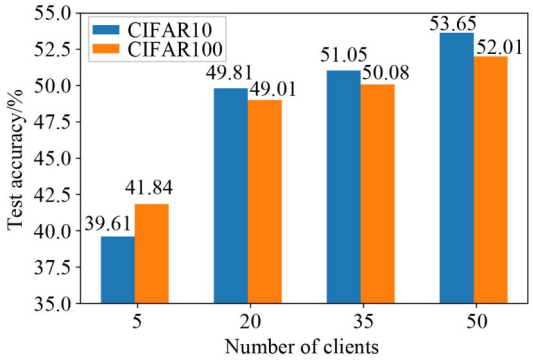


Fig. 5 Scalability study of FedPD with scales of clients on CIFAR10 and CIFAR100 datasets

improvement gradually slows. This is because the proportion of clients participating in each round of training remains unchanged, and the number of inactive clients increases in scenarios with more clients. These results effectively demonstrate the scalability and adaptability of FedPD to varying client scales in FL training.

5 Conclusion

In this paper, we proposed a novel FL method named personalized federated learning method based on partial distillation (FedPD) to address the challenges of data heterogeneity and model heterogeneity. In FedPD, we design two key modules, namely the partial knowledgetransfer (PKT) and partial knowledge ensemble (PKE), to collaborate and improve client model performance. PKT assigns different importance to distillation samples to improve distillation performance. PKE extracts personalized global knowledge for each client based on client knowledge. Furthermore, FedPD allows clients to utilize completely different architectural models without further constraints. Extensive experiments conducted on multiple datasets indicate that, compared to existing FL methods, FedPD consistently achieves higher accuracy in various settings involving multiple models and data heterogeneity.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 62076079) and the Guangdong Major Project of Basic and Applied Basic Research (No. 2019B030302002).

Competing interests

The authors declare that they have no competing interests or financial conflicts to disclose.

Open Access

This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made.

The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise

in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

[1] McMahan B, Moore E, Ramage D, Hampson S, Arcas B A Y. Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. 2017, 1273−1282

[2] Antunes R S, da Costa A C, Küderle A, Yari I A, Eskofier B. Federated learning for healthcare: systematic review and architecture proposal. ACM Transactions on Intelligent Systems and Technology (TIST), 2022, 13(4): 54

[3] Xu J, Glicksberg B S, Su C, Walker P, Bian J, Wang F. Federated learning for healthcare informatics. Journal of Healthcare Informatics Research, 2021, 5(1): 1−19

[4] Jiang J C, Kantarci B, Oktug S, Soyata T. Federated learning in smart city sensing: challenges and opportunities. Sensors, 2020, 20(21): 6230

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

[6] Yang L, Tan B, Zheng V W, Chen K, Yang Q. Federated recommendation systems. In: Yang Q, Fan L, Yu H, eds. Federated Learning: Privacy and Incentive. Cham: Springer, 2020, 225−239

[7] Wang Q, Yin H, Chen T, Yu J, Zhou A, Zhang X. Fast-adapting and privacy-preserving federated recommender system. The VLDB Journal, 2022, 31(5): 877−896

[8] Tan A Z, Yu H, Cui L, Yang Q. Towards personalized federated learning. IEEE Transactions on Neural Networks and Learning Systems, 2023, 34(12): 9587−9603

[9] Luo M, Chen F, Hu D, Zhang Y, Liang J, Feng J. No fear of heterogeneity: classifier calibration for federated learning with non-IID data. In: Proceedings of the 35th Conference on Neural Information Processing Systems. 2021, 5972−5984

[10] Ye M, Fang X, Du B, Yuen P C, Tao D. Heterogeneous federated learning: state-of-the-art and research challenges. ACM Computing Surveys, 2024, 56(3): 79

[11] Sun N, Wang W, Tong Y, Liu K. Blockchain based federated learning for intrusion detection for internet of things. Frontiers of Computer Science, 2024, 18(5): 185328

[12] Liu F, Zheng Z, Shi Y, Tong Y, Zhang Y. A survey on federated learning: a perspective from multi-party computation. Frontiers of Computer Science, 2024, 18(1): 181336

[13] Xu J, Wong R C W. Efficiently answering top-k window aggregate queries: calculating coverage number sequences over hierarchical structures. In: Proceedings of the 39th IEEE International Conference on Data Engineering. 2023, 1300−1312

[14] Dhasarathan C, Hasan M K, Islam S, Abdullah S, Khapre S, Singh D, Alsulami A A, Alqahtani A. User privacy prevention model using supervised federated learning-based block chain approach for internet of medical things. CAAI Transactions on Intelligence Technology, 2023

[15] Li T, Hu S, Beirami A, Smith V. Ditto: fair and robust federated learning through personalization. In: Proceedings of the 38th International Conference on Machine Learning. 2021, 6357−6368

[16] Dinh C T, Tran N H, Nguyen T D. Personalized federated learning with Moreau envelopes. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1796

[17] Li Q, He B, Song D. Model-contrastive federated learning. In: Proceedings of 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, 10708−10717

[18] Huang Y, Chu L, Zhou Z, Wang L, Liu J, Pei J, Zhang Y. Personalized cross-silo federated learning on non-IID data. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021, 7865−7873

[19] Ruan Y, Joe-Wong C. FedSoft: soft clustered federated learning with proximal local updating. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence. 2022, 8124−8131

[20] Collins L, Hassani H, Mokhtari A, Shakkottai S. Exploiting shared representations for personalized federated learning. In: Proceedings of the 38th International Conference on Machine Learning. 2021, 2089−2099

[21] Oh J, Kim S, Yun S Y. FedBABU: toward enhanced representation for federated image classification. In: Proceedings of the 10th International Conference on Learning Representations. 2022, 1−29

[22] Zhang J, Hua Y, Wang H, Song T, Xue Z, Ma R, Guan H. FedALA: adaptive local aggregation for personalized federated learning. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence. 2023, 11237−11244

[23] Li L, Gou J, Yu B, Du L, Tao Z Y D. Federated distillation: a survey. 2024, arXiv preprint arXiv: 2404.08564

[24] Lin T, Kong L, Stich S U, Jaggi M. Ensemble distillation for robust model fusion in federated learning. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 198

[25] Li D, Wang J. FedMD: heterogenous federated learning via model distillation. In: Proceedings of the 33rd Conference on Neural Information Processing Systems. 2019, 1–8

[26] Makhija D, Han X, Ho N, Ghosh J. Architecture agnostic federated learning for neural networks. In: Proceedings of the 39th International Conference on Machine Learning. 2022, 14860−14870

[27] Cho Y J, Manoel A, Joshi G, Sim R, Dimitriadis D. Heterogeneous ensemble knowledge transfer for training large models in federated learning. In: Proceedings of the 31st International Joint Conference on Artificial Intelligence. 2022, 2881−2887

[28] Zhang J, Guo S, Ma X, Wang H, Xu W, Wu F. Parameterized knowledge transfer for personalized federated learning. In: Proceedings of the 35th Conference on Neural Information Processing Systems. 2021, 10092−10104

[29] Ghosh A, Chung J, Yin D, Ramchandran K. An efficient framework for clustered federated learning. IEEE Transactions on Information Theory, 2022, 68(12): 8076−8091

[30] Liu B, Guo Y, Chen X. PFA: privacy-preserving federated adaptation for effective model personalization. In: Proceedings of the Web Conference 2021. 2021, 923−934

[31] Sun B, Huo H, Yang Y, Bai B. PartialFed: cross-domain personalized federated learning via partial initialization. In: Proceedings of the 35th Conference on Neural Information Processing Systems. 2021,

23309−23320
[32] Wang H, Li Y, Xu W, Li R, Zhan Y, Zeng Z. DaFKD: domain-aware federated knowledge distillation. In: Proceedings of 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 20412−20421
[33] Pfeiffer K, Rapp M, Khalili R, Henkel J. Federated learning for computationally constrained heterogeneous devices: a survey. ACM Computing Surveys, 2023, 55(14s): 334
[34] Gao D, Yao X, Yang Q. A survey on heterogeneous federated learning. 2022, arXiv preprint arXiv: 2210.04505
[35] He C, Annavaram M, Avestimehr S. Group knowledge transfer: federated learning of large CNNs at the edge. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1180
[36] Tan Y, Long G, Liu L, Zhou T, Lu Q, Jiang J, Zhang C. FedProto: federated prototype learning across heterogeneous clients. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence. 2022, 8432−8440
[37] Chen H Y, Chao W L. On bridging generic and personalized federated learning for image classification. In: Proceedings of the 10th International Conference on Learning Representations. 2022, 1−32
[38] Krizhevsky A. Learning multiple layers of features from tiny images. University of Toronto, Dissertation, 2009
[39] Cohen G, Afshar S, Tapson J, van Schaik A. EMNIST: extending MNIST to handwritten letters. In: Proceedings of 2017 International Joint Conference on Neural Networks. 2017, 2921−2926
[40] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. 2017, arXiv preprint arXiv: 1708.07747
[41] Achituve I, Shamsian A, Navon A, Chechik G, Fetaya E. Personalized federated learning with Gaussian processes. In: Proceedings of the 35th Conference on Neural Information Processing Systems. 2021, 8392−8406
[42] Chen H, Wang C, Vikalo H. The best of both worlds: accurate global and personalized models through federated learning with data-free hyper-knowledge distillation. In: Proceedings of the 11th International Conference on Learning Representations. 2023, 1−24
[43] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278−2324
[44] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. 2016, 770−778
[45] Sandler M, Howard A G, Zhu M, Zhmoginov A, Chen L C. MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018, 4510−4520
[46] Ma N, Zhang X, Zheng H T, Sun J. ShuffleNet V2: practical guidelines for efficient CNN architecture design. In: Proceedings of the 15th European Conference on Computer Vision. 2018, 122−138
[47] Qin Z, Deng S, Zhao M, Yan X. FedAPEN: personalized cross-silo federated learning with adaptability to statistical heterogeneity. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2023, 1954−1964
[48] Karimireddy S P, Kale S, Mohri M, Reddi S J, Stich S U, Suresh A T. Scaffold: stochastic controlled averaging for federated learning. In: Proceedings of the 37th International Conference on Machine Learning. 2020, 476
[49] Gao L, Fu H, Li L, Chen Y, Xu M, Xu C Z. FedDC: federated learning with non-iid data via local drift decoupling and correction. In: Proceedings of 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022, 10102−10111
[50] Yang Z, Zhang Y, Zheng Y, Tian X, Peng H, Liu T, Han B. FedFed: feature distillation against data heterogeneity in federated learning. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023, 2639



Xu YANG received his ME degree in computer science and technology from the Department of Computer Science and Technology, Harbin Institute of Technology (Weihai), China in 2021. He is currently working toward the PhD degree in the Department of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China. His research interests include federated learning, data mining, and machine learning.



Ji-Yuan FENG received his ME degree in computer technology from the Department of Cyberspace Security, Guangzhou University, China in 2022. He is currently working toward his PhD degree in the Department of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China. His research interests include federated learning, continual learning, and data mining.



Song-Yue GUO received his BE degree in Software Engineering from the School of Software, Shandong University, China in 2020. He is currently working toward his ME degree in the Department of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China. His research interests include machine learning, federated learning, and data mining.



Bin-Xing FANG received his MS degree in computer science and technology from Tsinghua University, China in 1984, and his PhD degree in computer science and technology from the Harbin Institute of Technology, China in 1989. He is currently a Fellow of the Chinese Academy of Engineering, China. His current research interests include computer networks, information and network security, and artificial intelligence security.



Qing LIAO received her PhD degree from Hong Kong University of Science and Technology, China in 2016. She is currently a professor with School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China. Her research interests include data mining, artificial intelligence, and information security, etc. She is a member of IEEE since 2013.