A Prototype-Based Knowledge Distillation Framework for Heterogeneous Federated Learning

Feng Lyu[1], Cheng Tang[1], Yongheng Deng[2], Tong Liu[3], Yongmin Zhang[1],Yaoxue Zhang[2]
[1]School of Computer Science and Engineering, Central South University, Changsha, China
[2]Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China
[3]Big Data Institute, Central South University, Changsha, China
{fenglyu, 214711035, csuliutong, zhangyongmin}@csu.edu.cn {dyh19@mails., zhangyx@}tsinghua.edu.cn

2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS) | 979-8-3503-3986-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDCS57875.2023.00091

***Abstract*—Federated learning (FL) is an emerging distributed machine learning paradigm, which has shown great potential in collaborative learning with privacy preservation. However, FL clients usually have disparate system resource capabilities (e.g., data, computation, and communication) for model training and aggregation, which can cause a series of system heterogeneity issues with performance degradation. To this end, we propose FedPKD, a Prototype-based Knowledge Distillation framework for FL. FedPKD integrates knowledge distillation and prototype learning with FL, which enables heterogeneous clients and the server to learn collaboratively, with different model architectures and resource capability adaptations. Specifically, FedPKD proposes to transfer dual knowledge of clients including the model output logits and prototypes to the server, and a prototype-based ensemble distillation mechanism is proposed to aggregate the logits and prototypes from clients, which can be used to train the server model with an unlabeled public dataset. The server model knowledge is then transferred back to clients to improve the performance of client models. Moreover, to improve learning performance and reduce communication overhead, we propose a prototype-based data filter mechanism to filter out the samples with low-quality knowledge. Extensive experiments under various settings demonstrate the superiority of FedPKD in learning performance and communication efficiency when compared to state-of-the-art benchmarks.**

***Index Terms*—Distributed IoTs, Federated Learning, Heterogeneity, Prototype-Based Knowledge Distillation**

I. Introduction

Federated learning (FL) is one of the popular distributed artificial intelligence paradigms, which supports distributed computing devices to collaboratively learn a machine learning model without committing their raw data to the server [1], [2]. In FL, each client trains the model with local data and then commits the model weights to the server, and the server then aggregates the model parameters from clients to update the server model and client models. Owing to its privacy-preserving manner, FL has been widely investigated and applied in extensive fields such as medical health [3]–[5], and smart city [6]–[8].

However, in practical application scenarios, system heterogeneity is a fundamental challenge for FL. Particularly, decentralized clients in FL usually have disparate system resources in terms of data, computation, communication, etc., while, the mainstream algorithms of FL like FedAvg [1] and FedProx [9] require all clients and the server to share the same structured model. Therefore, the size and complexity of the shared model are constrained by the resource-scarce clients, which cannot unleash the full potential of distributed learning. Moreover, when clients with distinct resource capabilities train identical models, it inevitably leads to performance gaps in training speed among clients, enlarging the FL training time.

Previous efforts to solve the system heterogeneity issues in FL can be taxonomized into three main classes: 1) client selection [10]–[13], selecting the clients with substantial computing and communication resources to participate in FL; 2) asynchronous learning [14]–[16], updating the global model in an asynchronous manner, which can save the waiting time for stragglers and thus accelerate the training time of FL; 3) submodel learning [17], [18], deriving a submodel for each client from the shared global model to accommodate its resource capabilities. Although these approaches alleviate the problems associated with system heterogeneity, the model architecture of clients is still constrained.

Different from the above three lines of works, in this paper, we aim to enable uniquely designed models between heterogeneous clients and the server, adapting to their system resource capabilities. To achieve it, recent studies have integrated the knowledge distillation (KD) techniques with FL [19]–[23]. Instead of transferring the model updates between clients and the server, in the KD-based FL approaches, clients and the server share a public dataset and transfer the model output knowledge to collaboratively learn client and server models. Specifically, FedMD [20] proposed to transfer client's model output logits to the server and use the aggregated logits to improve client models. FedDF [21] utilized KD to distill the aggregated knowledge of client models to the server model. FedGEMS [23] and FedET [22] enabled heterogeneous client models and larger server model via KD. In this way, each client and the server can learn different structured models in accordance with their system resource capabilities, which enhances the flexibility of FL. However, existing methods still have various limitations. For example, the performance of FedMD is limited in non-IID scenarios [19]. The server model architectures are restricted to client models in FedDF. FedGEMS requires a large labeled dataset shared between clients and the server. FedET requires a unified architecture for the representation layers of client models and server model.

To overcome the limitations of previous works, in this paper, we aim to design a flexible KD-based FL framework that can

2575-8411/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDCS57875.2023.00091

enable heterogeneous clients and the server collaboratively learn different structured models with a shared unlabeled public dataset. To achieve this goal, we have to address two key challenges. First, with preliminary experiments, we find that the learning performance of the KD-based FL method is limited in non-IID scenarios, since the quality of the aggregated knowledge from clients is undesirable. Therefore, the first challenge lies in improving the learning performance of the KD-based method in non-IID scenarios. Besides, the communication overhead of the KD-based method can be significant to reach a desirable learning performance, since the communication overhead is proportional to the sample size in the shared public dataset. Therefore, the second challenge comes from how to reduce communication overhead while guaranteeing learning performance.

To address the above challenges, we propose FedPKD, a prototype-based knowledge distillation framework for FL. In FedPKD, an unlabeled public dataset is shared among clients and the server to transfer knowledge between different structured client and server models. To improve the learning performance, FedPKD transfers dual knowledge between clients and the server, including not only the model output logits but also the prototypes of data classes. A prototype is the representation of one class in the feature space, and FedPKD uses prototypes to regularize the training of client and server models to improve learning performance. Specifically, each client first trains the model with local data and transfers the model output logits and prototypes to the server. Then, the server aggregates the logits and prototypes which are used to train the server model with the unlabeled public dataset. Besides, to improve learning performance and reduce communication overhead, FedPKD integrates a prototype-based data filtering mechanism to filter out the samples with low-quality knowledge. Finally, the output knowledge of the server model is transferred back to clients to update client models.

To evaluate the performance of FedPKD, we conduct extensive experiments in various settings and compare with state-of-the-art benchmarks. The experimental results show that under various non-IID settings, the proposed FedPKD outperforms the benchmarks in learning performance and communication efficiency for both client and server models. In addition, ablation experiments are performed to demonstrate the effectiveness of each component in FedPKD. Our major contributions can be summarized as follows.

- We propose a prototype-based knowledge distillation framework for FL, named FedPKD, which enables heterogeneous clients and the server learn different structured models in a collaborative manner.
- We devise a dual knowledge transfer mechanism, a prototype-based ensemble distillation, and a prototype-based data filtering mechanism, collectively improving learning performance and communication efficiency.
- We conduct extensive experiments and verify the superiority of the proposed framework in comparison with state-of-the-art methods.

II. Background And Motivation

A. Federated Learning And Knowledge Distillation

In federated learning, each client $c \in \{C\}$ trains its local model $\mathcal{M}_{\omega_c}$ with local data $\mathcal{D}_c$, and commits the local model updates ω_c to the server. Then, the server aggregates the received model updates with model aggregation algorithms like FedAvg [1]:

$$\omega_G = \frac{\sum_{c\in\{C\}} |\mathcal{D}_c|\omega_c}{\sum_{c\in\{C\}} |\mathcal{D}_c|}. \tag{1}$$

In this way, the global model $\mathcal{M}_{\omega_G}$ can be iteratively updated with ω_G. However, the model architecture between clients and the server is restricted to be identical, which limits the flexibility of FL and poses a series of system heterogeneity issues with performance degradation.

To enable different model architectures between heterogeneous clients and the server, the knowledge distillation technique provides a feasible way [24]–[26]. Typically, the outputs of a teacher model $\mathcal{M}_{\omega_t}$, i.e., logits, are transferred to the student model $\mathcal{M}_{\omega_s}$ through a shared dataset $\mathcal{D}_p$. The student model is then enabled to train with the one-hot labels of the dataset and the logits of the teacher model:

$$\begin{aligned}\min_{\omega_s} \sum_{(x_i,y_i)\in\mathcal{D}_p} &(\mathcal{L}_{KL}(\mathcal{M}_{\omega_s}(x_i), \mathcal{M}_{\omega_t}(x_i)) \\ &+\mathcal{L}_{CE}(\mathcal{M}_{\omega_s}(x_i), y_i)),\end{aligned} \tag{2}$$

where $\mathcal{L}_{KL}(\cdot,\cdot)$ is the kullback-leibler divergence, and $\mathcal{M}_{\omega_s}(x_i)$, $\mathcal{M}_{\omega_t}(x_i)$ are the model output logits of student model and teacher model on sample x_i.

Recently, some researches have applied knowledge distillation to federated learning [19]–[23]. In typical federated knowledge distillation, a public dataset $\mathcal{D}_p$ is shared between clients and the server, each client c first trains the local model with local data $\mathcal{D}_c$, and then transfers the logits $\mathcal{M}_{\omega_c}(x_i)$ of each sample $x_i \in \mathcal{D}_p$ to the server for aggregation. The aggregated logits are distilled to the server model [21], [22] or transferred back to clients and distilled to client models [19], [20]:

$$\min_{\omega} \sum_{x_i\in\mathcal{D}_p} \mathcal{L}_{KL}(\frac{1}{|C|}\sum_{c\in\{C\}} \mathcal{M}_{\omega_c}(x_i), \mathcal{M}_{\omega}(x_i)) \tag{3}$$

where $\mathcal{M}_{\omega}(x_i)$ is the outputs of the server model or client model on sample x_i. In this way, knowledge is transferred between client models and the server model via the public dataset and the logits, enabling them to train models with different architectures.

B. Motivation

Although knowledge is transferred between different-structured client models and the server model via the output logits on the shared public dataset, it is difficult to achieve satisfactory distillation performance, especially when the public dataset is unlabeled. To verify this, we evaluate the performance of the server model updated with the FedAvg

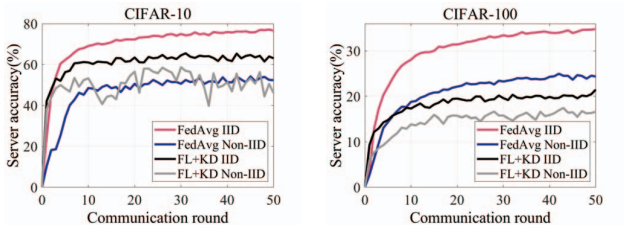
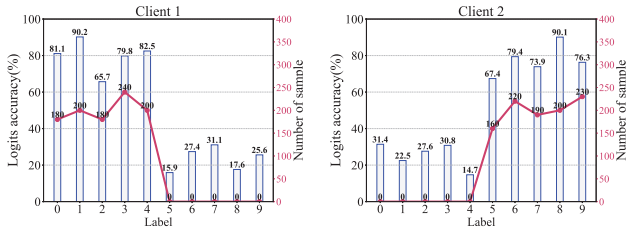


Fig. 1. Accuracy of server model trained with FedAvg and KD-based methods in IID and non-IID settings.

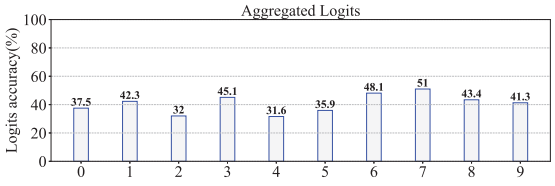
algorithm or the knowledge distillation method in IID or non-IID settings. Specifically, we randomly select 10000 samples from CIFAR-10 [27] or CIFAR-100 [27] dataset, and allocate them equally to each client for IID settings while following a Dirichlet distribution [28] (α = 0.3) for non-IID settings. Fig. 1 shows the server model accuracy trained in different settings, from which we can make two major observations. First, compared to the FedAvg method, the performance of the KD-based FL approach is deficient in both IID and non-IID settings. Besides, non-IID data can compromise the training performance significantly for both the FedAvg method and the KD-based FL method. This experiment demonstrates that it is challenging to achieve decent performance by simply distilling the aggregated logits knowledge to the server model, especially when the training data of clients are typically non-IID.

To improve the performance of the KD-based FL method, we first investigate the cause behind its poor performance in the typical non-IID scenario. Specifically, we set up two clients with the CIFAR-10 dataset, where the training data of client 1 is from classes 0-4 while the training data of client 2 is from classes 5-9. The two client models are respectively trained with the local data, then we select 10000 samples as the shared public dataset and measure the output logits accuracy of two client models. As shown in Fig. 2 (a), we can observe that the output logits accuracy of clients is closely related to their training data distribution. For example, client 1 has more samples from classes 0-4, and its output logits accuracy is higher for the samples with labels 0-4 while lower for other labels. This is because, with more training data, the learned model of client 1 is specialized in classes 0-4, while client 2 is specialized in classes 5-9. However, if we equally average the logits of two clients for each sample, the aggregated logits accuracy is undesirable as shown in Fig. 2 (b). Since the samples in the shared public dataset are unlabeled, the aggregated logits are served as the single supervised information for server model training, therefore, the low accuracy logits lead to poor training performance. Taken together, the experiments demonstrate that the aggregated logits quality is limited in non-IID scenarios, and it is challenging to achieve favorable model training performance using them alone.

In addition to the learning performance, the communication overhead is also one of the significant challenges in FL. In the methods of transferring model updates, the communication overhead is usually salient since the deep learning models contain a large number of parameters. In the KD-based FL



(a) Training data distribution and output logits accuracy of clients



(b) Accuracy of aggregated logits

Fig. 2. Training data distribution and logits accuracy of client 1 and client 2, and aggregated logits accuracy.

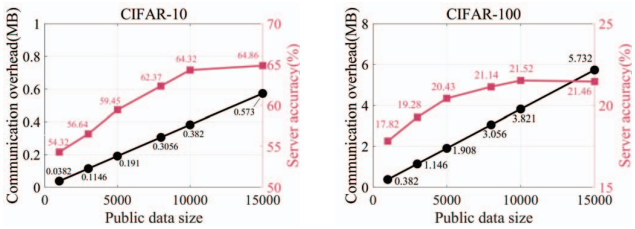


Fig. 3. Server model accuracy and communication overhead with different public dataset size.

methods, the communication overhead can be significant as well. Fig. 3 shows the communication overhead per client for transferring the model output logits under different sample sizes of the shared public dataset. It can be observed that the communication overhead is proportional to the number of samples in the public dataset for the KD-based method. As the number of samples increases, the communication overhead of transferring logits can be larger than transferring model updates (0.511MB). However, to guarantee the learning performance, a large dataset is essential, since Fig. 3 shows that the learning accuracy of the server model is increased with the public dataset size. Therefore, it is important to reduce the communication overhead of the KD-based FL method while guaranteeing the learning performance.

C. Challenges

From these pilot experiments, we can conclude that: 1) the learning performance of the KD-based FL method is limited in non-IID scenarios due to the low quality of the aggregated logits from clients; 2) the communication overhead of the KD-based FL method is significant for desirable learning performance. Therefore, our goal is to improve the learning performance of the KD-based FL method while reducing the

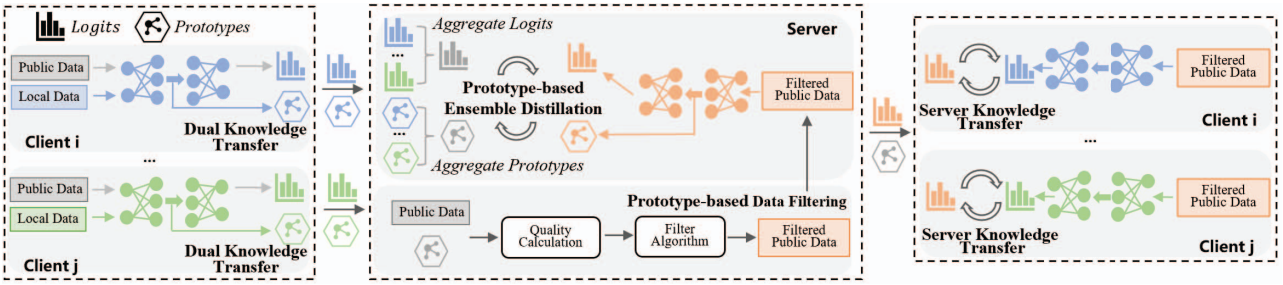


Fig. 4. System overview of FedPKD.

communication overhead. To achieve this, we will face with the following challenges:

a) **Improve the learning performance:** In the KD-based FL method, the knowledge quality distilled to the learned model plays a vital role in the training performance. However, it has been demonstrated that the quality of the aggregated logits from clients is quite limited when the data distribution of clients in FL is typically non-IID. How to effectively extract the model knowledge of clients and how to aggregate client's knowledge to derive high-quality knowledge for model training are nontrivial.

b) **Reduce the communication overhead:** The communication overhead of the KD-based FL method is increasing with the amount of data in the shared public dataset. However, a large shared public dataset is necessary for satisfactory learning performance, which can also result in expensive communication overhead. How to make a fine trade-off between the learning performance and the communication overhead is critical but challenging.

III. System Overview

To address the above challenges, we propose FedPKD, a prototype-based knowledge distillation framework for FL as illustrated in Fig. 4. FedPKD integrates four key components, i.e., *dual knowledge transfer*, *prototype-based ensemble distillation*, *prototype-based data filtering*, and *server knowledge transfer*. Specifically, to improve the learning performance, each client first trains with the local data and transfers dual knowledge to the server, including not only the model output logits of the public dataset but also the prototypes of the training dataset. The prototype for each class is the average of the feature space representation of the samples in that class [29], which can reflect the feature of each class. Then, the logits and prototypes as dual knowledge carriers are transferred to the server. On the server side, we design a prototype-based ensemble distillation mechanism to aggregate the logits and prototypes and distill the dual knowledge to the server model. Besides, to improve the training performance of the server model and reduce the communication overhead, we propose a prototype-based data filtering mechanism, which calculates the quality of the samples in the public dataset and selects the samples with high-quality knowledge for server model training. Finally, the server transfers the output logits of the server model as well as the global prototypes to each client. Each client further trains with the filtered public dataset and the knowledge from the server.

IV. Design of FedPKD

In this section, we introduce the design of the key components of FedPKD, including *dual knowledge transfer*, *prototype-based ensemble distillation*, *prototype-based data filtering*, and *server knowledge transfer*.

A. Client Local Training And Dual Knowledge Transfer

In FedPKD, we consider a setup with a N-class classification task in FL. Suppose there are C participating clients, which are connected to the server for model training. Each client $c \in \{C\}$ has its local training dataset $\mathcal{D}_c$, in which each data sample x_i has a corresponding label $y_i \in [0, N-1]$. In the first communication round, each client c performs local mini-batch stochastic-gradient descent (SGD) with its local dataset $\mathcal{D}_c$ to update its local model $\mathcal{M}_{\omega_c}$:

$$\min_{\omega_c} \sum_{(x_i,y_i)\in\mathcal{D}_c} \mathcal{L}_{CE}\left(\mathcal{M}^0_{\omega_c}(x_i), y_i\right), \tag{4}$$

where $\mathcal{L}_{CE}(\cdot,\cdot)$ is the cross-entropy loss function. $\mathcal{M}^0_{\omega_c}(x_i)$ is the model output on sample x_i. Then each client transfers the dual knowledge of the client model, i.e., the model output logits and prototypes to the server. Specifically, the knowledge of logits is the output of the last fully connected layer of the client model on the public dataset. We use $\mathcal{M}^t_{\omega_c}(x_i)$ to represent the output logits of client model $\mathcal{M}^t_{\omega_c}$ on each sample x_i in the public dataset $\mathcal{D}_p$. The knowledge of prototypes is the feature representations of client's local training dataset. For the client c, the prototype P^j_c of each class j is calculated with the average of the feature vectors of all data samples of class j in the local training dataset $\mathcal{D}_c$:

$$P^j_c = \frac{1}{|D^j_c|} \sum_{(x_i,y_i)\in D^j_c} \mathcal{R}_{\omega_c}(x_i), \tag{5}$$

where $\mathcal{R}_{\omega_c}$ is the middle layer of client model $\mathcal{M}^t_{\omega_c}$, $\mathcal{R}_{\omega_c}(x_i)$ is the output of $\mathcal{R}_{\omega_c}$ on sample x_i, and $\mathcal{D}^j_c$ is a subset of the data samples of class j in client's local dataset $\mathcal{D}_c$.

B. Prototype-based Ensemble Distillation and Data Filtering

On the server side, FedPKD aggregates the dual knowledge of all clients and utilizes the aggregated knowledge to guide the training of the server model. In addition, To reduce

Algorithm 1: Prototype-based data filtering.

Input: public dataset $(x_i, \widetilde{y}_i) \in \mathcal{D}_p$, server model $\mathcal{M}_{\omega_G}$, Selected-ratio sr, global prototype P^t, classes number N

Output: public data subset $\check{\mathcal{D}}_p$

for $n = 0, ..., N-1$ **do**
  $\mathcal{D}_p^n \leftarrow \{x_i \in \mathcal{D}_p | y_i = n\}$;
  Calculate the L2 distance of each sample $d(x_i) \leftarrow$ *Eq. (10)* **foreach** $x_i \in \mathcal{D}_p^n$;
  Sort samples in $\mathcal{D}_p^n$ in order of $d(x_i)$;
  Select $Sr|\mathcal{D}_p^n|$ samples with closest $d(x_i)$ as $\check{\mathcal{D}}_p^n$;
$\check{\mathcal{D}}_p \leftarrow \{\check{\mathcal{D}}_p^0, ..., \check{\mathcal{D}}_p^{N-1}\}$;
return $\check{\mathcal{D}}_p$;

communication overhead and further improve the learning performance, FedPKD filters out the samples with low-quality knowledge from the public dataset and utilizes the selected samples with high-quality knowledge to train the server model.

Aggregate logits. To aggregate the output logits of clients, FedPKD first calculates the logits quality of each sample, since the quality of client's output logits varies from samples. To this end, we calculate the variance $Var(\mathcal{M}_{\omega_c}^t(x_i))$ of each logits as the quality of the logits. For each sample, the larger the variance of its logits, the more confident the client model is in the prediction of that sample [30]. Therefore, we put more emphasis on the logits with high variances, and give higher weight in logits aggregation. The aggregated logits $\mathcal{S}^t(x_i)$ of sample x_i can be calculated by:

$$\mathcal{S}^t(x_i) = \sum_{c \in \{C\}, x_i \in \mathcal{D}_p} \beta_c^t(x_i) \mathcal{M}_{\omega_c}^t(x_i), \tag{6}$$

where the aggregation weight $\beta_c^t(x_i)$ is defined as:

$$\beta_c^t(x_i) = \frac{Var(\mathcal{M}_{\omega_c}^t(x_i))}{\sum_{k \in \{C\}, x_i \in \mathcal{D}_p} Var(\mathcal{M}_{\omega_k}^t(x_i))}. \tag{7}$$

Aggregate prototypes. Since a prototype is associated with a class, the local prototypes P_c^t of each client can overlap between classes. Taking the CIFAR-10 dataset as an example, if a client has training data for dogs, cats, and frogs, it will send prototypes of dogs, cats, and frogs to the server, while another client has training data for cats, airplanes, and trucks, then the prototype of cats will overlap between these two clients. Therefore, the server will aggregate the overlapped prototypes of clients to derive global prototypes. Formally, for each class j, the server selects the prototypes $\{P_c^{t,j}\}$ of the clients with class j, and aggregates them to derive a global prototype $P^{t,j}$ for class j:

$$P^{t,j} = \frac{1}{|C_j|} \frac{\sum_{c \in \{C_j\}} |\mathcal{D}_c^j| P_c^{t,j}}{\sum_{c \in \{C_j\}} |\mathcal{D}_c^j|}, \tag{8}$$

where C_j is the set of clients with training data samples from class j.

Prototype-based data filtering. The aggregated logits and prototypes can be used for server model training. To further improve the learning performance, we propose a prototype-based data filtering mechanism for FedPKD, which filters out the samples with low-quality logits and uses the selected samples for server model training. Specifically, the server first uses the global logits $\mathcal{S}^t(x_i)$ to predict a pseudo-label for each sample x_i in the public dataset:

$$\widetilde{y}_i = \underset{label \in [0, N-1]}{argmax} \ \mathcal{S}^t(x_i). \tag{9}$$

Then we aim to filter out the samples with low-quality with the help of the global prototypes. Since the prototype is a representation of a class in the feature space, the feature representation of a sample should ideally be close to its prototype. Therefore, we consider that if the feature representation of a sample has a large distance from its pseudo-label prototype, the sample is low-quality, or the sample's pseudo-label is likely to be wrong. To mitigate the side effects of the samples with low-quality, FedPKD introduces the prototype-based data filtering mechanism as illustrated in Algorithm 1. Specifically, the server calculates the L2 distance between the feature vector of each sample in the public dataset and its global prototype:

$$d(x_i) = L2(\mathcal{R}_{\omega_G}^t(x_i), P^{t,\widetilde{y}_i}). \tag{10}$$

Afterwards, for the samples $\mathcal{D}_p^j$ of each class j in the public dataset, the server would select $\theta\%$ samples with the closest distance from the global prototype $P^{t,j}$. After performing data filtering on all classes, the selected samples will be assembled into the dataset $\check{\mathcal{D}}_p$ for server model training. In this way, we select the samples with high-quality knowledge that are closer to the global prototype in the feature space.

Prototype-based ensemble distillation. With the global prototypes and the selected samples in the public dataset with their logits, a prototype-based ensemble distillation mechanism is used to update the server model. Specifically, the server calculates the kullback-leibler divergence loss between the aggregated logits $\mathcal{S}^t(x_i)$ and the server model output $\mathcal{M}_{\omega_G}^t(x_i)$, and the cross-entropy loss between the pseudo-label $\widetilde{y}_i$ and the model prediction $\mathcal{M}_{\omega_c}^t(x_i)$, then the distillation loss term in the loss function for server model training can be expressed as:

$$\mathcal{L}_{kd} = \frac{1}{|\check{\mathcal{D}}_p|} \sum_{(x_i, \widetilde{y}_i) \in \check{\mathcal{D}}_p} (\mathcal{L}_{KL}(\mathcal{S}^t(x_i), \mathcal{M}_{\omega_G}^t(x_i)) + \mathcal{L}_{CE}(\mathcal{M}_{\omega_G}^t(x_i), \widetilde{y}_i)). \tag{11}$$

For the global prototypes P^t, the server would calculate the feature vector $\mathcal{R}_G^t(x_i)$ of each sample in the public data subset $\check{\mathcal{D}}_p$. We expect to narrow the distance between the feature vector of each data sample $(x_i, \widetilde{y}_i) \in \check{\mathcal{D}}_p$ and the global prototype $P^{t,\widetilde{y}_i}$ to regularize the training of server model. Therefore, the prototype loss term in the loss function for the server model training is:

$$\mathcal{L}_p = \frac{1}{|\check{\mathcal{D}}_p|} \sum_{(x_i, \widetilde{y}_i) \in \check{\mathcal{D}}_p} \mathcal{L}_{MSE}(\mathcal{R}_{\omega_G}^t(x_i), P^{t,\widetilde{y}_i}). \tag{12}$$

Algorithm 2: Algorithm of FedPKD

Input: public dataset $\mathcal{D}_p$, private dataset $\mathcal{D}_c, c \in \{C\}$, server model $\mathcal{M}_{\omega_G}$, client model $\mathcal{M}_{\omega_c}, c \in \{C\}$, communication rounds T

Output: Trained Server model $\mathcal{M}_{\omega_G}$, Trained Client model $\mathcal{M}_{\omega_c}, c \in \{C\}$

for $t = 0, 1, \ldots, T-1$ **do**
  foreach *client* c **do in parallel**
    $\boldsymbol{w}_c^t \leftarrow$ `ClientPriTrain`$(c, e_{c,tr}, t)$;
    Send logits $\mathcal{M}_{\omega_c}^t(x_i)$ and prototype P_c^t to server;
  for *server* **do**
    Aggregate logits using Eq. (6);
    Aggregate prototype using Eq. (7);
    $\check{\mathcal{D}}_p \leftarrow$ **Algorithm 1**;
    $\boldsymbol{w}_G^t \leftarrow$ `ServerTrain`(e_s);
    Send server logtits $\mathcal{M}_{\omega_G}^t(x_i)$ and global prototype P^t to each client;
  foreach *client* c **do in parallel**
    Receive server logits and global prototype;
    $\boldsymbol{w}_c^t \leftarrow$ `ClientPubTrain`$(c, e_{c,p}, t)$;
return $\mathcal{M}_{\omega_G}$, $\mathcal{M}_{\omega_c}$;
Function `ClientPriTrain`(c, e_c, t)**:**
  for $e = 1, 2, \ldots, e_c$ **do**
    for *batch* $\boldsymbol{b} \in \{(x_i^k, y_i^k)\}_{x_i^k \in \mathcal{D}_c}$ **do**
      if $t = 0$ **then**
        $\boldsymbol{w}_c^t \leftarrow \boldsymbol{w}_c^t - \eta \nabla \mathcal{L}_c^0(\boldsymbol{w}_c^t; \boldsymbol{b})$; ▷ *in Eq. (4)*
      if $t > 0$ **then**
        $\boldsymbol{w}_c^t \leftarrow \boldsymbol{w}_c^t - \eta \nabla \mathcal{L}_c^1(\boldsymbol{w}_c^t; \boldsymbol{b})$; ▷ *in Eq. (16)*
  return $\boldsymbol{w}_c^t$
Function `ServerTrain`(e_s)**:**
  for $e = 1, 2, \ldots, e_s$ **do**
    for *batch* $\boldsymbol{b} \in \{(x_i, y_i, \mathcal{S}^t(x_i))\}_{x_i \in \check{\mathcal{D}}_p}$ **do**
      $\boldsymbol{w}_G^t \leftarrow \boldsymbol{w}_G^t - \eta \nabla \mathcal{L}_G(\boldsymbol{w}_G^t; \boldsymbol{b})$; ▷ *in Eq. (13)*
  return $\boldsymbol{w}_G^t$
Function `ClientPubTrain`(c, e_c, t)**:**
  for $e = 1, 2, \ldots, e_c$ **do**
    for *batch* $\boldsymbol{b} \in \{(x_i, y_i, \mathcal{M}_{\omega_G}^t(x_i))\}_{x_i \in \check{\mathcal{D}}_p}$ **do**
      $\boldsymbol{w}_c^t \leftarrow \boldsymbol{w}_c^t - \eta \nabla \mathcal{L}_c(\boldsymbol{w}_c^t; \boldsymbol{b})$; ▷*in Eq. (15)*
  return $\boldsymbol{w}_c^t$

where $\mathcal{L}_{MSE}(\cdot, \cdot)$ is the mean-squared error loss function, Finally, the server model is trained by minimizing the following optimization objective:

$$F(\omega_G) = \delta \mathcal{L}_{kd} + (1 - \delta)\mathcal{L}_p, \tag{13}$$

where δ is a preset parameter.

C. Server Knowledge Transfer

After training the server model, the server will transfer the knowledge of the server model output logits $\mathcal{M}_{\omega_G}^t(x_i)$ as well as the global prototypes P^t to clients to update client models. To reduce the communication overhead, the server only transfers the output logits of the samples $x_i \in \check{\mathcal{D}}_p$ to clients. When each client receives the logits $\mathcal{M}_{\omega_G}^t(x_i)$ from server model, it first yields a pseudo-label for each sample based on the server logits:

$$\widetilde{y}_i^s = \underset{label \in [0, N-1]}{argmax} \mathcal{M}_{\omega_G}^t(x_i). \tag{14}$$

To avoid overfitting of the client model due to supervised training with limited local training data, and to enhance the generalization performance of the client model, each client trains with the public data subset $\check{\mathcal{D}}_p$ and their knowledge received from the server. The optimization objective of training is:

$$\min_{\omega_c} \sum_{(x_i, \widetilde{y}_i^s) \in \check{\mathcal{D}}_p} (\gamma \mathcal{L}_{KL}(\mathcal{M}_{\omega_c}^t(x_i), \mathcal{M}_{\omega_G}^t(x_i)) + (1 - \gamma)\mathcal{L}_{CE}(\mathcal{M}_{\omega_c}^t(x_i), \widetilde{y}_i^s)). \tag{15}$$

It should be noted that the client training optimization objective Eq. (4) using local training data is only for each client in the first communication round. In the later communication round $t + 1$, each client would receive the global prototypes P^t from the server. we aim to further regularize the model training of each client by narrowing the distance between the client prototypes and the global prototypes. Therefore, for each $t+1$ communication round we set a new optimization objective for each client model:

$$\min_{\omega_c} \sum_{(x_i, y_i) \in \mathcal{D}_c} (\mathcal{L}_{CE}(\mathcal{M}_{\omega_c}^{t+1}(x_i), y_i) + \epsilon \mathcal{L}_{MSE}(R_{\omega_c}^{t+1}(x_i), P^{t, y_i})). \tag{16}$$

The complete algorithm of FedPKD is illustrated in Algorithm 2. First, each client c uses local training data for model training, then transfers the dual knowledge to the server. Afterwards, the server aggregates the dual knowledge, filters the public data via the prototype-based data filtering algorithm, and distills the aggregated knowledge to the server model. Finally, the server transfers its output logits and global prototypes to each client, which further trains with the filtered public dataset and the knowledge from the server.

V. Performance Evaluation

A. Evaluation Methodology

Benchmarks. We compare the performance of FedPKD with the following benchmarks: 1) *FedMD* [20], a KD-based FL framework allowing model heterogeneous settings; 2) *DS-FL* [19], a KD-based FL framework introducing an entropy-reduction aggregation method; 3) *FedET* [21], a heterogeneous ensemble knowledge transfer framework for training large models on the server; 4) *FedDF* [22], a robust model fusion algorithm with KD; 5) *FedAvg* [1], the classic FL algorithm transferring the model updates; 6) *FedProx* [9], an algorithm for data heterogeneous FL.

Dataset. We utilize two widely-used image classification datasets CIFAR-10 [27] and CIFAR-100 [27] to evaluate the performance of FedPKD. CIFAR-10 consists of 50,000 training images and 10,000 test images divided into 10 classes, while CIFAR-100 is similar to CIFAR-10 but is more complex

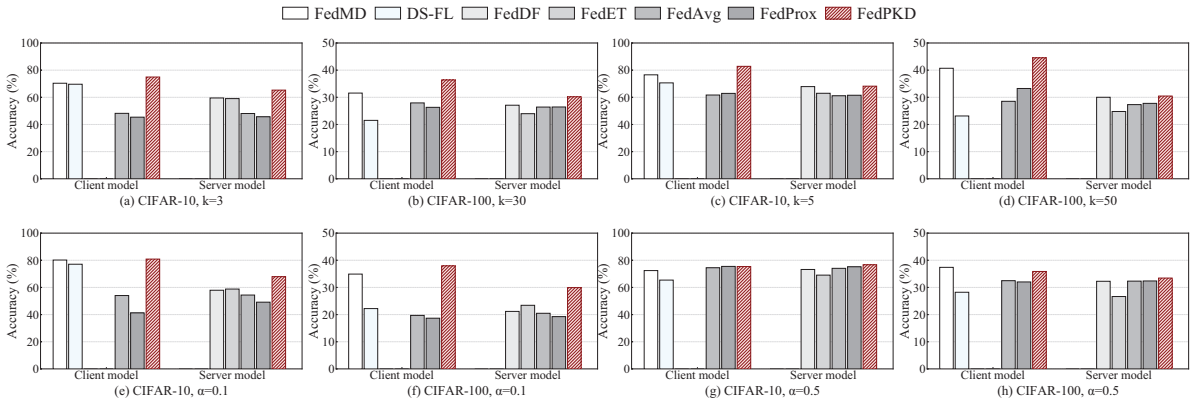
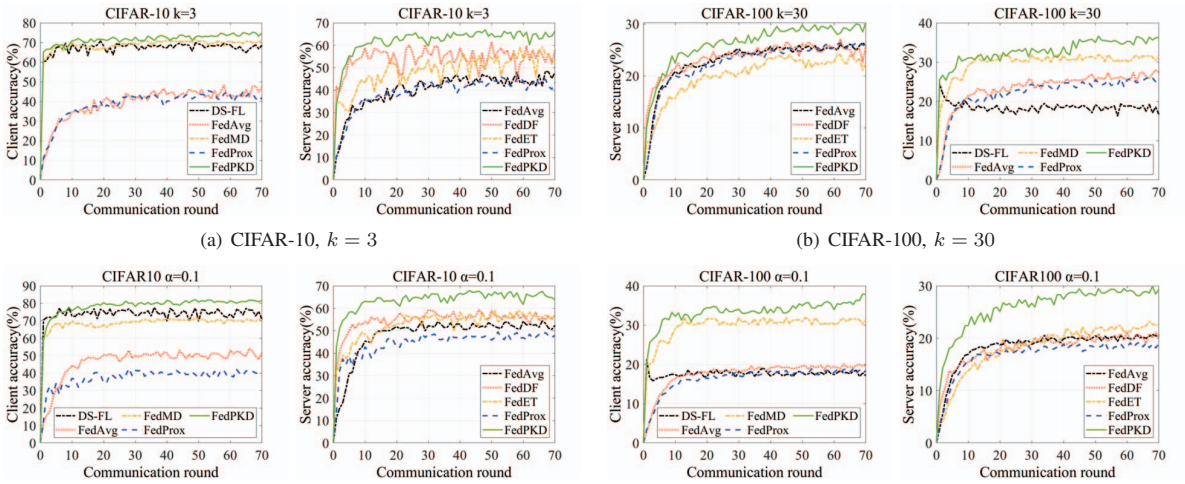


Fig. 5. Accuracy of FedPKD and benchmarks under different non-IID settings with homogeneous models between clients and server. There is no server model in FedMD and DS-FL, while FedDF and FedET are not focused on client model performance.



(a) CIFAR-10, $k = 3$ (b) CIFAR-100, $k = 30$

(c) CIFAR-10, $\alpha = 0.1$ (d) CIFAR-100, $\alpha = 0.1$

Fig. 6. Accuracy of FedPKD and benchmarks with homogeneous models under highly non-IID settings.

with 100 classes. For each dataset, we randomly partition the data into a public dataset with 5000 samples and a local training dataset for each client, the public dataset is unlabeled, and the local training dataset is divided and distributed to each client to set up the non-IID setting across clients. To this end, we use two different data partition methods: 1) Dirichlet distribution method [28], which assigns samples to clients following a Dirichlet distribution with $\alpha = 0.1$ and $\alpha = 0.5$. In this method, clients have a different number of samples per class; 2) Shards method [9], where the dataset is split into shards of size 20, and each client is assigned with 40 shards from k classes. We use $k = 3$ and $k = 5$ for CIFAR-10, while $k = 30$ and $k = 50$ for CIFAR-100. In these data partition methods, a smaller value of α or k means a higher non-IID degree across clients.

Models. In the evaluations, we set up homogeneous and heterogeneous model settings across clients. The client model is ResNet20 in homogeneous model settings, and in heterogeneous model settings, the client model is one of ResNet11, ResNet20, and ResNet29. For the server model, since the benchmarks of FedAvg, FedProx, and FedDF do not support heterogeneous models between clients and the server, we use ResNet20 with the same structure as client models in these methods and use ResNet56 in other benchmarks. Besides, it should be noted that there is no server model in the methods of FedMD and DS-FL.

Parameters. For the parameters of FedPKD, we set the size of local mini batch B = 32 using the Adam optimizer and the learning rate η = 0.001 for each client and server training. For the parameters epoch $e_{c,tr}$ in the local training of client models and epoch e_s in the server model training, we set $e_{c,tr} = 10$ for FedAvg and FedProx, $e_{c,tr} = 10$, $e_s = 20$ for FedMD and DS-FL, $e_{c,tr} = 10$, $e_s = 10$ for FedET, and $e_{c,tr} = 30$, $e_s = 5$ for FedDF. In FedPKD, we set epoch $e_{c,tr} = 15$ for the training with the local training dataset, set epoch $e_{c,p} = 10$ for the training with the public dataset, and set epoch $e_s = 40$ for the server training. We set the select ratio $\theta = 70\%$, the training parameters $\epsilon = \delta = \gamma = 0.5$, and the communication

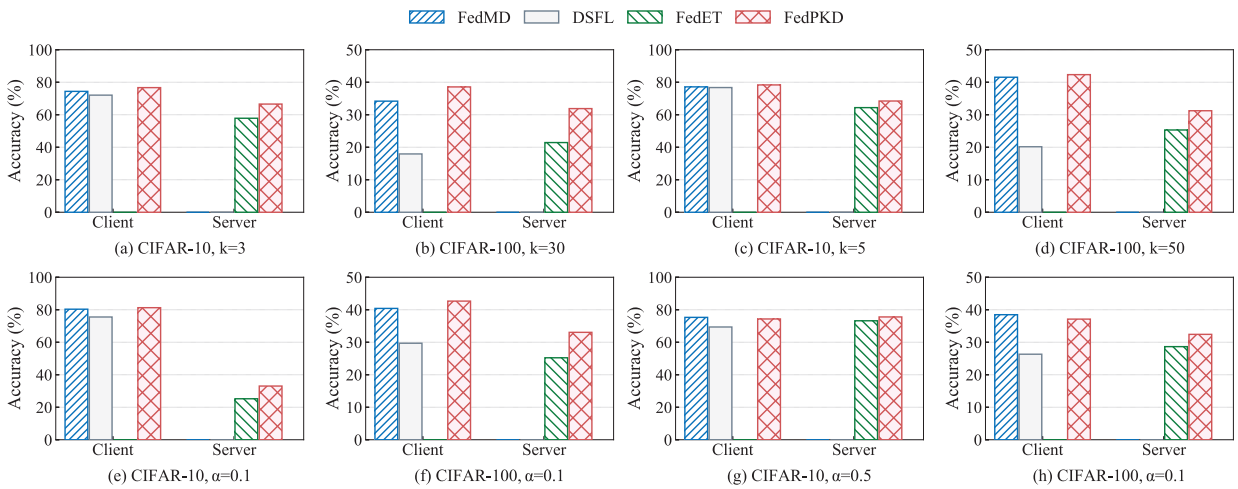


Fig. 7. Accuracy of FedPKD and benchmarks under different non-IID settings with heterogeneous models between clients and server.

round $T = 70$.

Metrics. We use the following metrics to evaluate the performance of FedPKD and benchmarks: 1) the server model accuracy S_{acc} tested on the global test dataset, which contains data samples from all classes, so the accuracy of the server model reflects the generalization performance of the server model; 2) the client model accuracy C_{acc} that tested on client's local test dataset, the local test dataset of each client has the same distribution as the local training dataset, so the accuracy on the local test dataset represents the personalized performance of each client model; 3) the communication efficiency, we calculate the total communication overhead consumed to reach the target model accuracy.

B. Performance Comparison

Accuracy in homogeneous model settings. We first compare the performance of FedPKD with benchmarks in homogeneous model settings. To this end, we set two different non-IID settings with two different data partition methods. For highly non-IID settings, we set $k = 3$, $k = 30$ in the shards method, and $\alpha = 0.1$ in the Dirichlet distribution method. For the weakly non-IID settings, we set $k = 5$, $k = 50$ in the shards method and $\alpha = 0.5$ in the Dirichlet distribution method. Fig. 5 shows the achieved test accuracy of FedPKD and benchmarks in different non-IID settings and different datasets in model homogeneous settings. We can observe that FedPKD achieves the highest server model accuracy in all settings, for example, with 16% improvement compared to the FedProx in CIFAR-10. In terms of client model accuracy, FedPKD achieves outperformed client model accuracy in most cases, which demonstrates that FedPKD enables efficient dual knowledge transfer between client models and the server model. In addition, there are some noteworthy observations. Specifically, with $\alpha = 0.5$, FedPKD achieves 75.36% client accuracy in CIFAR-10 while FedProx achieves higher accuracy of 75.48%, and FedMD slightly outperforms FedPKD by around 1.5% of client model accuracy in CIFAR-100. The possible reason is that, in weakly non-IID settings, the data distribution of clients is approaching a uniform distribution, and hence the training of client models is not limited by the variety and amount of the training data. Moreover, Fig. 6 shows the model accuracy changes of FedPKD and benchmarks with the number of communication rounds in weakly non-IID settings. It can be observed that the model accuracy of FedPKD outperforms the benchmarks significantly in highly non-IID settings, which verifies the superiority of FedPKD in highly non-IID settings.

Accuracy in heterogeneous model settings. We next compare the performance of FedPKD with the benchmarks that support model heterogeneity across clients, including the FedMD, DS-FL, and FedET methods. Fig. 7 shows the comparison results between FedPKD and the benchmarks. It can be observed that in most cases FedPKD outperforms the benchmarks in terms of both server model accuracy and client model accuracy, for example, with 20% improvement compared to the DS-FL in CIFAR-100, which demonstrates that FedPKD enables efficient dual knowledge transfer between heterogeneous client models and the server model. Moreover, we can also observe that in model heterogeneous settings, FedPKD achieves better performance than in model homogeneous settings under highly non-IID settings. This is because the knowledge of the relatively larger client models brings larger benefits of model performance, which also indicates the superiority of our proposed FedPKD.

Communication overhead. We compare the communication overhead of FedPKD and benchmarks achieving the target model accuracy in weakly non-IID settings. We set the target model accuracy as 60% for CIFAR-10 and 25% for CIFAR-100. As shown in Table I, we can see that FedPKD consumes less communication overhead compared to the benchmarks.

Table I. Communication overhead of FedPKD and benchmarks used to achieve the target accuracy in lower non-IID settings. There is no server model in FedMD and DS-FL, while FedDF and FedET are not focused on client model performance.

Communication overhead(MB)	$k = 5, k = 50$				$\alpha = 0.5$			
	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100	
	$C_{acc}(60\%)$	$S_{acc}(60\%)$	$C_{acc}(25\%)$	$S_{acc}(25\%)$	$C_{acc}(60\%)$	$C_{acc}(60\%)$	$C_{acc}(25\%)$	$S_{acc}(25\%)$
FedAvg	546.84	441	246.96	458.64	88.2	105.84	194.04	194.04
FedProx	582.12	493.9	123.48	599.26	105.84	105.84	229.32	211.68
FedDF	N/A	150.22	N/A	786.76	N/A	85.84	N/A	193.14
FedMD	22.92	N/A	190.8	N/A	11.46	N/A	152.8	N/A
DS-FL	26.48	N/A	N/A	N/A	19.12	N/A	191.2	N/A
FedPKD	**6.49**	**25.98**	**64.94**	**389.64**	**6.49**	**25.98**	**97.41**	**162.18**

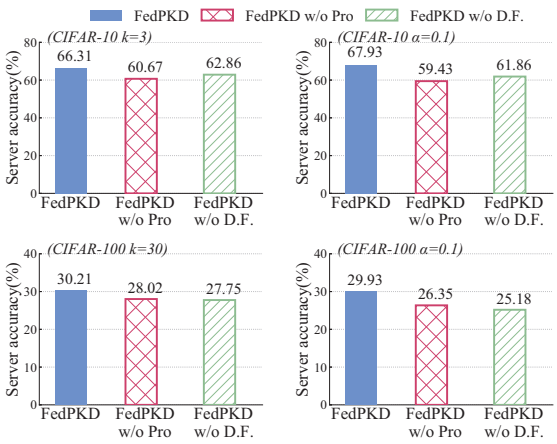


Fig. 8. Ablation experiment under highly non-IID settings: 1) w/o Pro: FedPKD without prototype; 2) w/o D.F.: FedPKD without prototype-based public data filtering.

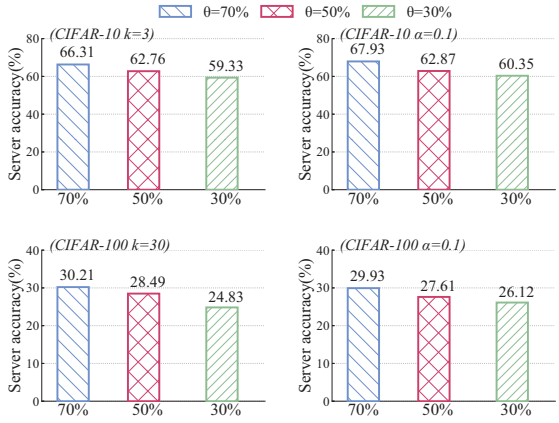


Fig. 9. Server model accuracy with different θ under highly non-IID settings.

Specifically, compared to the benchmarks with the lowest communication overhead (FedMD in CIFAR-10, FedDF in CIFAR-10, FedProx and FedMD in CIFAR-100, FedAvg and FedMD in CIFAR-100), FedPKD can achieve about 5.7 $\times$ less communication overhead than some benchmarks like FedMD. The experiment verifies that the proposed FedPKD has significant superiority in optimizing the communication efficiency of FL.

C. Ablation Experiments

We next perform ablation experiments to validate the effectiveness of the key components of FedPKD, including the prototype-based ensemble distillation mechanism and the prototype-based data filtering mechanism. Specifically, we set up the following benchmark methods: 1) FedPKD without prototypes, in which we remove the prototype loss items from the server model optimization objective; 2) FedPKD without prototype-based data filtering, in which we use the unfiltered public dataset for the server model training. As shown in Fig. 8, it can be observed that the server model accuracy approximately decreases by an average of 7% in CIFAR-10 and 2.5% in CIFAR-100. Besides, when we remove the prototype-based data filtering algorithm, the server model accuracy also has a decrease of about 5% on average in CIFAR-10 and 3.5% on average in CIFAR-100. The ablation experiments demonstrate the superiority of each component of FedPKD.

D. Robustness Experiments

Performance with θ. To explore the impact of θ on the performance of FedPKD, we show the model accuracy of FedPKD with different θ in Fig . 9, where a smaller θ means that more samples in the public dataset will be discarded. We can observe that the model accuracy is declining from $\theta = 70\%$ to $\theta = 30\%$, which indicates that discarding low-quality samples contributes to better model training performance.

Performance with δ. In server model training, a larger δ indicates that the server model training is more inclined to classifier learning and a smaller δ is more inclined to feature learning. We explore the impact of δ on the training performance of the server model and show the experimental results in Fig. 10. The results vary from datasets, in CIFAR-10, the server model achieves the best performance at δ = 0.5, while in CIFAR-100, the highest accuracy of the server model is achieved at δ = 0.1. It happens because the classification task of CIFAR-100 is more complex, where the server model

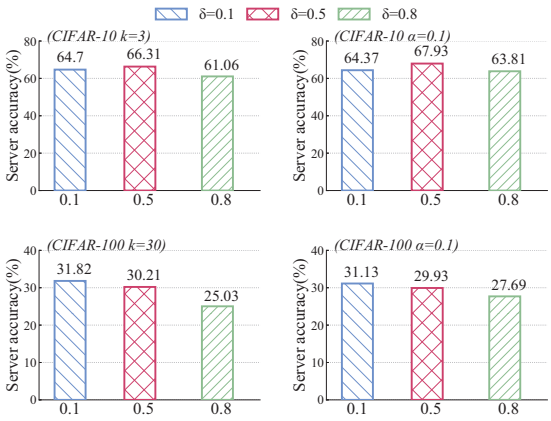


Fig. 10. Server model accuracy with different δ under highly non-IID settings.

might need to learn more knowledge from the feature space to achieve better performance.

VI. Related Works

A. Knowledge Distillation in FL

Recently, some studies have investigated the combination of federated learning and knowledge distillation to improve the performance of FL. Specifically, in [20], Li et al. proposed FedMD, which enables uniquely designed models across clients using transfer learning and knowledge distillation. To improve the learning performance of the KD-based method, Tao et al. proposed DS-FL in [19], which introduced an entropy-reduction aggregation method to aggregate the logits from clients. However, there is no model training on the server in these approaches. FedDF [21] proposed a robust model fusion method using ensemble distillation, which updates the server model with both the aggregated client model updates and the aggregated client model logits. However, transferring model updates constrains server model to be client models. FedGEMS [23] proposed to train heterogeneous models among clients and train larger model on the server, but it depends on a large labeled public dataset for knowledge transfer. An ensemble knowledge transfer method FedET was proposed in [22], where small models are trained on clients and the knowledge of client models is used to train a larger server model. However, this method requires a unified architecture for the representation layer of client and server models, besides, the communication overhead is large due to transferring the model parameters between clients and server. In contrast, our proposed FedPKD supports different structured models across clients and server with an unlabeled public dataset, and can achieve more competitive learning performance with lower communication overhead via dual knowledge transfer, prototype-based ensemble distillation, and prototype-based data filtering.

B. Prototype Learning in FL

The concept of prototype learning was first proposed in [29]. Previous researches on prototype learning mainly focused on few-shot learning [29], contrastive learning [31], [32], image retrieval [33], [34] and action recognition [35]. Recently, there have been some researchers [36], [37] applying prototype learning to FL to improve the learning performance of FL. For example, in [38], FedProc proposed a prototypical contrastive federated learning method to boost the learning performance under non-IID data. However, FedProc does not support heterogeneous model structures across clients because it transfers model parameters between clients and the server. FedProto [37] proposed to aggregate clients' prototypes and send them back to clients to regularize the training of client models. However, the server is only used for aggregating and transferring prototypes without training a server model. ProtoFSSL [39] introduced prototype learning into federated semi-supervised learning (FSSL), which uses prototypes to make pseudo labels for unlabeled data samples. However, the focus of ProtoFSSL is different from FedPKD. In contrast, the proposed FedPKD applies prototype learning to KD-based FL to improve the learning performance of client and server models, and to filter training data samples to reduce the communication overhead.

VII. Conclusion and Future Work

In this paper, we have proposed FedPKD, a prototype-based knowledge distillation framework for FL. FedPKD integrates knowledge distillation and prototype learning with FL, which allows heterogeneous clients and the server to learn different structured models in a collaborative manner. FedPKD enables efficient knowledge transfer between heterogeneous models via three novel mechanisms, i.e., dual knowledge transfer, prototype-based ensemble distillation, and prototype-based data filtering. Specifically, the model output logits and prototypes as dual knowledge carriers are transferred between client and server models; the logits and prototypes are aggregated and distilled to models via the prototype-based ensemble distillation mechanism; and the prototype-based data filtering mechanism filters out the training samples with low-quality knowledge to boost the model training performance and reduce communication overhead. We have conducted extensive experiments and demonstrated the superiority of FedPKD in comparison with benchmarks. For our future work, we will conduct more experiments with new datasets to evaluate its robustness, and further optimize FedPKD by enhancing its ensemble distillation and data filtering mechanisms to improve the learning performance and resource efficiency.

Acknowledgment

This research was supported in part by the National K&D Program of China (Grants. No. 2022YFC2009805, 2022YFF0604504), the National Natural Science Foundation of China (Grants No. 62002389), 111 project (No.B18059), the Young Elite Scientist Sponsorship Program by CAST (Grant No.YESS20200238), the Central South University Innovation-Driven Research Programme (Grant No.2023CXQD029), the Key Research and Development Program of Hunan Province of China (Grants No.2022GK2013), Natural Science Foundation of Hunan Province of China (Grants No.2021JJ20079), and the Young Talents Plan of Hunan Province of China (Grants No.2021RC3004).

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] S. Duan, D. Wang, J. Ren, F. Lyu, Y. Zhang, H. Wu, and X. Shen, "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey," *IEEE Communications Surveys & Tutorials*, 2022, doi: 10.1109/COMST.2022.3218527.

[3] D. Gao, C. Ju, X. Wei, Y. Liu, T. Chen, and Q. Yang, "HHHFL: Hierarchical heterogeneous horizontal federated learning for electroencephalography," *arXiv preprint arXiv:1909.05784*, 2019.

[4] Q. Liu, C. Chen, J. Qin, Q. Dou, and P.-A. Heng, "FedDG: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1013–1023.

[5] S. Lu, Y. Zhang, Y. Wang, and C. Mack, "Learn electronic health records by fully decentralized federated learning," *arXiv preprint arXiv:1912.01792*, 2019.

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

[7] B. Qolomany, K. Ahmad, A. Al-Fuqaha, and J. Qadir, "Particle swarm optimized federated learning for industrial iot and smart city services," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[8] Z. Zheng, Y. Zhou, Y. Sun, Z. Wang, B. Liu, and K. Li, "Applications of federated learning in smart cities: recent advances, taxonomy, and open challenges," *Connection Science*, vol. 34, no. 1, pp. 1–28, 2022.

[9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[10] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE international conference on communications (ICC)*. IEEE, 2019, pp. 1–7.

[11] Y. G. Kim and C.-J. Wu, "AutoFL: Enabling heterogeneity-aware energy efficient federated learning," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 183–198.

[12] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "OORT: Efficient federated learning via guided participant selection," in *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, 2021, pp. 19–35.

[13] Y. Deng, F. Lyu, J. Ren, H. Wu, Y. Zhou, Y. Zhang, and X. Shen, "AUCTION: Automated and quality-aware client selection framework for efficient federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1996–2009, 2021.

[14] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3654–3672, 2021.

[15] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 655–668, 2020.

[16] Y. Deng, F. Lyu, J. Ren, Y.-C. Chen, P. Yang, Y. Zhou, and Y. Zhang, "Improving federated learning with quality-aware user incentive and auto-weighted model aggregation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4515–4529, 2022.

[17] A. Li, J. Sun, P. Li, Y. Pu, H. Li, and Y. Chen, "Hermes: an efficient federated learning framework for heterogeneous mobile clients," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 420–437.

[18] Z. Xu, F. Yu, J. Xiong, and X. Chen, "Helios: Heterogeneity-aware federated learning with dynamically balanced collaboration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 997–1002.

[19] 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," *arXiv preprint arXiv:2008.06180*, 2020.

[20] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.

[21] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2351–2363, 2020.

[22] Y. J. Cho, A. Manoel, G. Joshi, R. Sim, and D. Dimitriadis, "Heterogeneous ensemble knowledge transfer for training large models in federated learning," *arXiv preprint arXiv:2204.12703*, 2022.

[23] S. Cheng, J. Wu, Y. Xiao, and Y. Liu, "FedGEMS: Federated learning of larger server models via selective knowledge fusion," *arXiv preprint arXiv:2110.11027*, 2021.

[24] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[25] J. Ba and R. Caruana, "Do deep nets really need to be deep?" *Advances in neural information processing systems*, vol. 27, 2014.

[26] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.

[27] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[28] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[29] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[30] C. Camacho-Gómez, S. Salcedo-Sanz, and D. Camacho, "A review on ensemble methods and their applications to optimization problems," *Applied Optimization and Swarm Intelligence*, pp. 25–45, 2021.

[31] J. Li, P. Zhou, C. Xiong, and S. C. Hoi, "Prototypical contrastive learning of unsupervised representations," *arXiv preprint arXiv:2005.04966*, 2020.

[32] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020.

[33] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1269–1277.

[34] H.-M. Yang, X.-Y. Zhang, F. Yin, and C.-L. Liu, "Robust classification with convolutional prototype learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3474–3482.

[35] N. Hoang, T. Lam, B. K. H. Low, and P. Jaillet, "Learning task-agnostic embedding of multiple black-box experts for multi-task model fusion," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4282–4292.

[36] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 713–10 722.

[37] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "Fedproto: Federated prototype learning over heterogeneous devices," *arXiv preprint arXiv:2105.00243*, 2021.

[38] X. Mu, Y. Shen, K. Cheng, X. Geng, J. Fu, T. Zhang, and Z. Zhang, "FedProc: Prototypical contrastive federated learning on non-iid data," *arXiv preprint arXiv:2109.12273*, 2021.

[39] W. Kim, K. Park, K. Sohn, R. Shu, and H.-S. Kim, "Federated semi-supervised learning with prototypical networks," *arXiv preprint arXiv:2205.13921*, 2022.