Personalized Edge Intelligence via Federated Self-Knowledge Distillation

Hai Jin, *Fellow, IEEE*, Dongshan Bai, Dezhong Yao, *Member, IEEE*, Yutong Dai, Lin Gu, Chen Yu, and Lichao Sun

Abstract—*Federated Learning* (FL) is an emerging approach in edge computing for collaboratively training machine learning models among multiple devices, which aims to address limited bandwidth, system heterogeneity, and privacy issues in traditional centralized training. However, the existing federated learning methods focus on learning a shared global model for all devices, which may not always be ideal for different devices. Such situations become even worse when each edge device has its own data distribution or task. In this paper, we study personalized federated learning in which our goal is to train models to perform well for individual clients. We observe that the initialization in each communication round causes the forgetting of historical personalized knowledge. Based on this observation, we propose a novel *Personalized Federated Learning* (PFL) framework via self-knowledge distillation, named pFedSD. By allowing clients to distill the knowledge of previous personalized models to current local models, pFedSD accelerates the process of recalling the personalized knowledge for the latest initialized clients. Moreover, self-knowledge distillation provides different views of data in feature space to realize an implicit ensemble of local models. Extensive experiments on various datasets and settings demonstrate the effectiveness and robustness of pFedSD.

Index Terms—Edge computing, personalized federated learning, knowledge distillation

1 Introduction

Federated Learning (FL) [1] is an efficient distributed learning paradigm over heterogeneous edge computing environment. It is very promising when local data cannot be uploaded to cloud due to privacy, storage, or communication constraints. For example, IoT devices may generate massive amounts of data at the network edge, and medical applications may be forbidden from sharing private data for privacy concerns. Under FL setting, a number of edge devices collaboratively train a global model without transferring local data. Serving as a communication-efficient and privacy-preserving learning scheme, FL has shown its capability to facilitate real-world edge computing applications [2], [3], [4], such as healthcare [5], object detection [6], and natural language processing [7].

Unfortunately, learning a single global model may fail when the data distribution varies across devices or each edge device has different tasks [8], [9], [10]. For example, client data may originate from different devices or geographical locales, where the data is potentially *non-Independent and Identically Distributed* (non-IID) [11], [12]. In such a data-heterogeneous scenario, the single global model's performance for the individual client may be even worse than the local models trained without collaboration. In addition, for multitask applications, each edge device has different optimal directions. Considering the next-word prediction task on mobile devices[13]. Given the text *"I love playing,"* the next word suggestion should be different from one to another. A shared global model learned by the conventional federated learning scheme has a consistent training target, failing to give personalized suggestions. Personalization is also important in many other areas, such as healthcare. Different individuals, hospitals or regions usually have different lifestyles, activity patterns, and other health-related characteristics[14]. Therefore it is necessary to obtain personalized models for each client to maintain their specific knowledge while exploiting common information.

To handle the above heterogeneous challenges, *personalized federated learning* (PFL)[8], [9], [10] is proposed to deal with FL from a client-specific perspective. As shown in Fig. 1, current FL can be categorized into two groups: conventional federated learning and personalized federated learning. *Conventional federated learning*, such as FedAvg [1] and FedProx [12], aims to learn a strong and general shared model to find the global optima. In contrast, *personalized federated learning* aims to allow each client to train a personalized model that performs well on client-specific tasks. Unlike local-only training without federation, which lacks data, the personalized model in PFL can benefit from shared knowledge through collaborative training. Personalized FL not only needs personalization to fit the local data

- *Hai Jin, Dongshan Bai, Dezhong Yao, Lin Gu, and Chen Yu are with the National Engineering Research Center for Big Data Technology and System, Service Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: {hjin, carlbai, dyao, lingu, yuchen}@hust.edu.cn.*
- *Yutong Dai and Lichao Sun are with Lehigh University, Bethlehem, PA 18015 USA. E-mail: {yud319, lis221}@lehigh.edu.*

Manuscript received 25 April 2022; revised 7 October 2022; accepted 8 November 2022. Date of publication 28 November 2022; date of current version 19 December 2022.
This work was supported in part by the National Natural Science Foundation of China under Grant 62072204, and in part by the Fundamental Research Funds for the Central Universities under Grant HUST:2020kfyXJJS019. (Corresponding authors: Dezhong Yao and Lichao Sun.)
Recommended for acceptance by R. Prodan.
Digital Object Identifier no. 10.1109/TPDS.2022.3225185

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

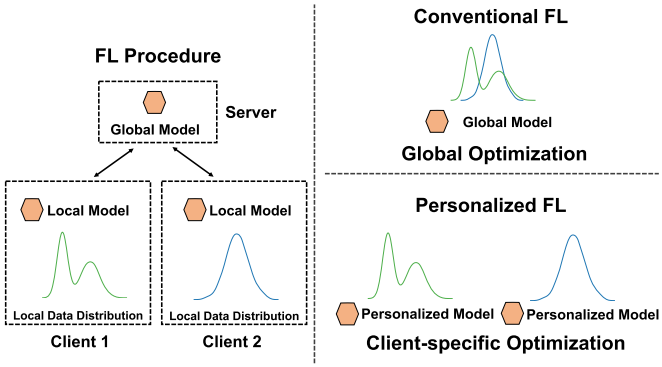


Fig. 1. The difference between conventional FL and personalized FL

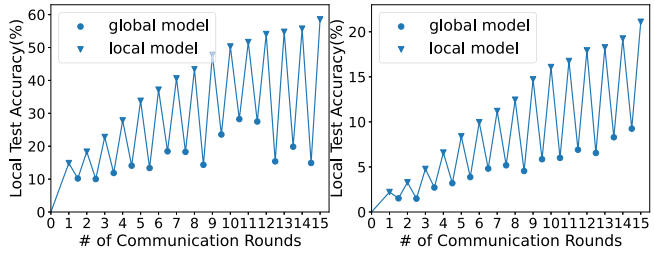


Fig. 2. The performance gap between the received global model and the last round local trained model. FedAvg is applied on CIFAR-10 and CIFAR-100 with 100 non-IID clients, and evaluated the local accuracy.

distribution but also requires generalization to exploit the common knowledge from collaboration.

The key challenge in PFL is to strike a careful balance between shared knowledge and local task-specific knowledge. To examine the relationship between personalization and generalization, we empirically compare performance between the global model and the local model. As shown in Fig. 2, the personalized performance of the global model is fairly lower than the local model (a detailed description of Fig. 2 is given in Section 2.2). In FL, the local model is initialized with the latest global model at the beginning of each communication round. Such initialization discards the historical local knowledge and causes severe performance degradation of the local model. The clients may have to train the local model from scratch in a new communication round. This phenomenon becomes more critical when the data distributions are more heterogeneous or the participation ratio is lower. As a result, we hypothesize that the initialization in FL upsets the balance between personalization and generalization.

Based on our observation, we propose a novel personalized Federated learning framework via local Self-knowledge Distillation, named pFedSD [1]. Specifically, pFedSD lets clients preserve their local trained models and regards them as personalized models at the end of each round. In the next round, the local models are initialized with the received global model and trained under the guidance of previous personalized models. The previous personalized model transfers historical personalized knowledge to the current local model by *Knowledge Distillation* (KD). Therefore, pFedSD avoids forgetting historical local knowledge and obtains a balanced trade-off between personalization and generalization. Moreover, the previous local model provides an additional view of local data in feature space. So we can regard our self-knowledge distillation as performing implicit ensemble together with KD[15]. Experiment results on various datasets and settings show that our method significantly improves personalized performance and convergence. The main contributions of this paper are as follows.

- We are the first to explore the personalized knowledge forgetting phenomena in PFL. In each training round, the clients forget the historical personalized knowledge after initializing with the aggregated model (Section 2.2).
- We propose an effective personalized FL framework via self-knowledge distillation to transfer the historical personalized knowledge and strike a better balance between personalization and generalization (Section 3). We further analyze the trade-off between personalization and generalization, and the benefits of self-distillation to personalized federated learning (Section 3.3).
- We conduct extensive experiments over various real-world datasets and setups. Compared with the current state-of-the-art FL methods, our method demonstrates superior efficiency and robustness in personalized local model training (Section 4).

2 Preliminaries

In this section, we first formalize the personalized FL problem. Then we present our empirical observation about the phenomena of forgetting in personalized FL, which motivates us to provide pFedSD.

2.1 Problem Statement

Our target is to collaboratively train personalized models for a set of clients in FL. In this paper, we focus on a supervised C-class classification task as the universal task setting. Consider there are K clients connected to a central server, and each client $k \in [K]$ maintains its local private dataset D_k. Each data sample is a pair (x, y), where x is the input and $y \in [1, C]$ is the label. The entire dataset is denoted as $D = \bigcup_{k=1}^{K} \{D_k\}$. The goal of the conventional FL system is to find a global model w that minimizes the total empirical loss over the entire dataset D

$$\min_{w} F(w) = \sum_{k=1}^{K} \frac{|D_k|}{|D|} F_k(w), \tag{1}$$

where $F_k(w)$ is the loss function of client k, defined as $F_k(w) := \mathbb{E}_{x_k \sim D_k}[f_k(w; x_k)]$ with $f_k(w; x_k)$ being the cross entropy loss. The global objective function $F(w)$ is the weighted average of the local objectives $F_k(w)$ over K clients. The classic *FedAvg* iterates between local training $w_k = \arg\min_w F_k(w)$ and global aggregation $w = \sum_{k=1}^{K} p_k w_k$ for multiple rounds of communication. In each communication round, a set of clients is selected to perform local training in parallel. The local model w_k of each participating client k is initialized by the global model w received from the server at the beginning of the local training phase. Then clients optimize w_k for several epochs and send back the local model to

1. Source code https://github.com/CGCL-codes/pFedSD

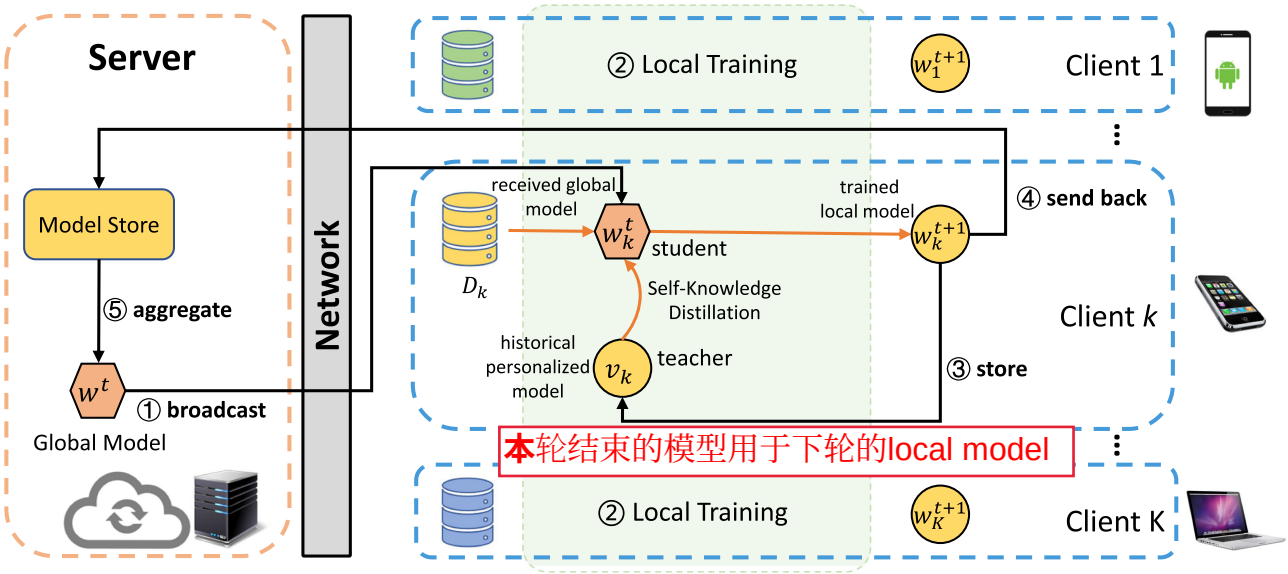


Fig. 3. Illustration of pFedSD workflow. Training is performed in rounds until a termination condition is met. The workflow includes 5 steps: ① The server selects a subset S^t of clients and broadcasts the global model w^t; ② The activated client $k \in S^t$ initializes the local model w_k^t with received global model, and then performs local training with self-knowledge distillation; ③ clients store the trained local model w_k^{t+1} as the teacher v_k for the next round; ④ clients send back the updated local model w_k^{t+1} to the server; ⑤ the server aggregates all received local models to obtain a new global model w^{t+1}.

the server. The server averages the trained local models of all participating clients to obtain global model w.

Due to the non-IID data distribution across clients, i.e., D_k has highly different distributions, the optimal global model w^* may not generalize well to the client whose local objective $F_k(w)$ significantly deviates from $F(w)$. The personalized performance of the global model may be even worse than the local training model without federation. Such clients would not have the will to join in FL and instead train their own models to minimize the local objectives. Personalized FL on client k learns a personalized model v_k to align with its own local objective

$$\min_{\{v_1,v_2,\ldots,v_K\}} \sum_{k=1}^{K} F_k(v_k). \tag{2}$$

2.2 Observation on Personalized Knowledge Forgetting

Forgetting Personalized Knowledge After Initialization. In communication round t, part of the clients with a participation ratio of $r \in (0, 1]$ are activated to perform E local epochs. Activated client k receives the latest global model from the server and regards it as the initial local model. The latest global model is aggregated from a set of selected clients, where these clients have different data distributions. Obviously, the latest downloaded model tends to perform well on the global distribution rather than the biased local distribution [1], [12]. Fig. 2 shows the personalized test accuracy gap between the global model and the previous local model. The personalized performance of the latest downloaded global model is much worse than the local model in the last round. At the beginning of each round, participating clients directly replace their local model with the global model, which causes a huge performance drop. The performance degradation becomes more severe in the more realistic setting (i.e., a smaller r, larger E, higher data heterogeneity). For instance, the server samples a subset S^t at round t, produces the global model w^{t+1}, and then the server samples another subset S^{t+1} at round $t+1$. If client $k \in S^{t+1}$, but $k \notin S^t$, the latest global model w^{t+1} is most likely unsuitable for client k due to the different data distributions. As a result, the client k needs to personalize the latest initialized local model w_k^{t+1} from scratch, leading to a slower convergence.

Similar observations are also examined in[16], [17], [18]. Suppose a client has not been selected to participate in local training for multiple rounds. This client needs more local updates to recall the previous personalized knowledge from the global starting point when it is selected next time. This phenomenon also reminds us of '*catastrophic forgetting*' [19] in continual learning. Although the vanilla FedAvg algorithm provides a strong generalization power to share common knowledge across clients, it hurts the personalization for a specific client to some extent, especially in a high data heterogeneity scenario. The core of personalized FL is to pursue a trade-off between generalization and personalization. Simply initializing the local model with the global model in FedAvg enhances the generalization power but discards previous personalized information. Based on the empirical observation, we hypothesize that preventing forgetting personalized knowledge could improve the performance and convergence in personalized FL.

3 Personalized Federated Self-Knowledge Distillation

In this section, we first introduce our core idea of pFedSD and analyze the compatibility of our proposed method with existing algorithms. Then, we explain the system design of our framework in detail. An overview of pFedSD workflow in the edge computing environment is illustrated in Fig. 3.

The FL process is based on the client-server framework. Note that we regard edge devices as the clients.

3.1 The Proposed pFedSD

Motivated by the analysis in Section 2.2, we propose *Personalized Federated Self-knowledge Distillation* (pFedSD). To preserve the historical personalized knowledge, we keep the personalized model v_k for supervising the local training in the next rounds at client k. Specifically, we update the personalized model v_k with the local model w_k in the current round to store personalized knowledge. To transfer the knowledge from the past model to the current local model, there are many schemes, such as knowledge distillation, parameter regularization. Here, we employ knowledge distillation from the most recently updated personalized model v_k during the client updates phase. The benefits of self-knowledge distillation are theoretically and experimentally discussed in Sections 3.3.2 and 4.4, respectively. To this end, the loss function $\Phi_k(w_k)$ of client k combines its own empirical risk $F_k(w_k)$ and the distillation loss

$$\Phi_k(w_k^t) = \underbrace{F_k(w_k^t)}_{CE\ Loss} + \underbrace{\lambda \mathcal{L}_{KL}(q(v_k)||q(w_k^t))}_{KD\ Loss}. \tag{3}$$

本地训练的同时做KD

Here the hyperparameter λ controls the contributions of knowledge distillation. $F_k(\cdot)$ denotes the *cross entropy* (CE) loss of client k. $\mathcal{L}_{KL}$ denotes the *Kullback-Leibler* (KL) divergence function between past personalized prediction $q(v_k)$ and current local prediction $q(w_k^t)$. The soft prediction q is calculated with softmax function $\sigma(\cdot)$ of logits z, i.e., $q = \frac{exp(z_c/\tau)}{\sum_c exp(z_c/\tau)}$(assume a C-class classification task), logits z is the output of the last fully connected layer, τ is the temperature hyperparameter of the softmax function.

The client k updates the local weights following the local objective $\Phi_k(w_k)$ (instead of $F_k(w_k)$) in pFedSD. So the local weights w_k is updated by running *Stochastic Gradient Descent* (SGD) as the following

$$w_k^t = w_k^t - \eta \nabla \Phi_k(w_k^t, v_k), \tag{4}$$

where η is the learning rate.

Algorithm 1. pFedSD Algorithm: Server Side

Input: participation ratio r, communication rounds T
Output: local personalized models $\{v_k\}_{k\in[K]}$

1: **procedure** ServerExecute
2: Initialize model w^0
3: **for** $t = 1, \ldots, T$ communication rounds **do**
4: Sample a subset S^t of clients of size rK (via Device Sampler)
5: Send global model w^t to all clients in S^t (via Communication Manager)
6: **for** each client $k \in S^t$ **in parallel do**
7: $w_k^{t+1} \leftarrow ClientLocalUpdate(k, w^t)$
8: **end for**
9: $w^{t+1} \leftarrow \sum_{k\in S^t} \frac{w_k^{t+1}}{|S|}$ (via Aggregator)
10: **end for**
11: **end procedure**

From a high level view, the backbone of pFedSD is actually a map-reduce process. The details of our method are summarized in Algorithm 1 and Algorithm 2. The procedure on the server side is outlined in Algorithm 1. The procedure on the client side is outlined in Algorithm 1. At the beginning of the whole procedure, the server randomly initializes the global model (Line 2). pFedSD iteratively runs T rounds of communication (Line 3). At each round, the server samples a subset S^t of clients with the fraction r to participate in the process of the current round (Line 4). Then the cloud server sends the global model to these active clients (Line 5). These clients perform local training in parallel and send the updated local model back to the server (Lines 6-8). At the end of the round, the server aggregates all received models to obtain a new global model (Line 9). The procedure on the client side is outlined in Algorithm 2. The clients first receive the global model sent by the server (Line 2). During the local training phase, the active client k overwrites the local model with the newly received global model (Line 3). The client updates the local model on the private dataset via Eq. (4) for E epochs (Lines 5-9). After local training, the client stores the personalized model with the updated local model (Line 10) and then returns the local model (Line 11).

Algorithm 2. pFedSD Algorithm: Client Side

Input: learning rate η, local batch size B, local epochs E
Output: local personalized models $\{v_k\}_{k\in[K]}$

1: **procedure** ClientLocalUpdatek, w^t
2: Receive the global model (via Communication Manager)
3: Initialize the local model (via Model Manager) $w_k^t \leftarrow w^t$
4: Compute # of local iterations $I_k = \lceil \frac{|D_k|}{B} \rceil$
5: **for** $i = 1, \ldots, E$ local epochs **do**
6: **for** $j = 1, \ldots, I_k$ local iterations **do**
7: $w_k^t \leftarrow w_k^t - \eta \nabla \Phi_k(w_k^t, v_k)$ ▷ Optimize Eq.(4) (via Training Optimizer)
8: **end for**
9: **end for**
10: Store personalized model v_k with w_k^t (via Model Store)
11: **return**local model w_k^t (via Communication Manager)
12: **end procedure**

vk distill 到wk

Compatibility With Other Algorithms. As pFedSD focuses on the local side, it is orthogonal to recent personalized FL methods. Here, we store the local models w_k^t as personalized models v_k to guide the local training process in the next communication round. v_k could also be replaced by other personalized models suggested in other algorithms. Moreover, the global model could be produced with momentum or in a cluster manner to get a more accurate or fine-grained generalization. Anyway, the nature of pFedSD makes it easily incorporated into advanced FL algorithms to evolve a more effective and robust personalized FL system. We will take a closer look at why it works in the next section. Without loss of generality, we analyze the basic version built upon FedAvg in this paper.

3.2 System Design

System Architecture. We give a high-level illustration of the system design and architecture in Fig. 4. It consists of a

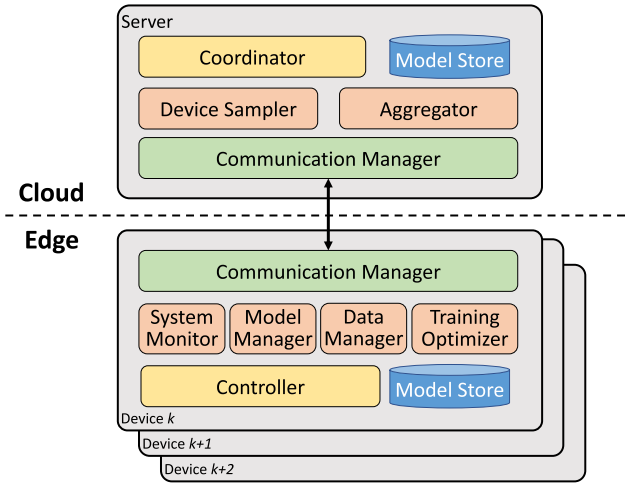


Fig. 4. An overview of pFedSD. The system includes two parts: a central cloud server and a set of edge devices. The modularized system design provides much flexibility and extensibility.

cloud server and a mass of edge devices. The FL system adopts a modular design and decouples the training, communication, and storage. It is flexible for the users to extend to new datasets, models, and algorithms. Another benefit is that the system enables module reuse and fair comparison for different FL algorithms. The cloud server has the following components.

- Coordinator is the top-level operator of the cloud server, which manages the whole FL process. It instructs other components in the server, manages a set of edge devices and orchestrates FL tasks across the edge devices. All the devices, modules, and events must be registered in the coordinator.
- Device Sampler selects a subset of edge devices to participate in each communication round. It provides support for the partially-participated FL. The naive client selection strategy allows a uniform sampling. This module could be extended with more fancy selection strategies to improve FL's quality.
- Aggregator is responsible for aggregating all received models to produce the global model. The input of the aggregator is clients' models or model updates. The output of the aggregator is the global model. It is triggered when the server either receives all the models of participating clients or meets a predefined time limit. The naive aggregate method is simply averaging the model parameters. Many papers propose various aggregation strategies to enhance the global model.
- Communication Manager is responsible for interacting between the server and clients. The interaction includes sending work request messages to clients, exchanging the model parameters and other auxiliary information for different FL algorithms. We implement this module based on MPI and Gloo[2].

The modules on the edge device are as follows:

- Controller is the core component on the client side which controls the local process and instructs other modules of the edge side.
- System monitor obtains the system statistics of the current device, such as battery and load information. The system information is reported to the controller and used to decide whether to accept the work request from the server.
- Model Manager prepares the local model. It initializes the local model with the global model at the beginning of local training and stores the local model as the personalized model at the end.
- Data Manager loads local training and testing datasets.
- Training Optimizer is responsible for performing local training for the clients. It could be designed specifically for different FL algorithms, such as adding a proximal term in the local objective. The major changes of pFedSD are implemented in this module.

System Overhead. Unlike other personalized FL approaches, pFedSD works well without introducing significant system overhead compared to vanilla FL. The soft predictions of v_k on local data could be obtained in two ways. One is to store the past predictions of v_k. The other is to hold the past model v_k on the local side. The former does not need to do duplicate forward propagation. It saves computational costs but requires more disk space to store previous predictions. Furthermore, it may not be suitable for the situation where local data updates all the time (such as streaming data). The latter may need more GPU memory to compute soft predictions but shows more flexibility. We can make choices from these two ways according to our tasks and local hardware resources. Moreover, some high availability systems may require the edge devices to be stateless for easy migration. Our method is convenient to fit such systems to meet the demands of fault tolerance and load balancing. We just need to maintain the personalized models on the server and deliver them along with the global model during the model download phase.

3.3 pFedSD Analysis

To offer more insight into the efficiency of our proposed pFedSD, we discuss the following two questions. First, we explain the trade-off between personalization and generalization of the personalized FL algorithm. Second, we explain why self-knowledge distillation works in our method.

3.3.1 Generalization-Personalization Trade-off

As we analyze above, the initialization at the beginning of each round causes the forgetting of past personalized knowledge and upsets the balance between generalization and personalization. The extreme of personalization is the local-only algorithm without collaboration. On the other hand, the extreme of generalization is the conventional FL algorithm, including FedAvg. Personalized FL can be considered an intermediate state between local-only and conventional FL, with better personalized performance than both.

Current personalized FL algorithms are all trying to pursue a better trade-off between personalization and

2. MPI is a high performance Message Passing Interface implementation for distributed computing. For the ease of reproducing the experiments, we also build a Gloo-backend version which is supported for PyTorch by default.

generalization. Initialization plays an important role in personalized FL, which influences personalized optimization in limited local steps. [9], [20], [21] initialize the local model w_k^t with the previous local model w_k^{t-1}. The initialized local model has strong personalization but a weak generalization. So these approaches train the local models with the regularization between the global model and the local models. The regularization transfers the common knowledge to local models and relieves over-fitting to scarce local data. Such methods reduce to the local-only algorithm when the weight of the regularization term is set to 0 and conventional FL when set to ∞. [18], [22], [23] decouple the model into personal and global parameters. These methods initialize the local models with a mixture of these two parts and train the local models without additional regularization. Our method initializes the local models with the global model as FedAvg, and trains the local models with the regularization between the current local model and the previous personalized model. Existing methods are a bit more local-only-based and give priority to personalization. Our method is a bit more FedAvg-based and gives priority to generalization.

3.3.2 Why Does Self-Knowledge Distillation Work ?

To help the clients recall the historical personalized knowledge, we employ self-knowledge distillation to transfer knowledge from the previous personalized model to the latest initialized local models. There are also some other choices to transfer the knowledge, such as using L2-norm to guide the local model in parameter space. To gain more insights into our proposed algorithm, we analyze why self-knowledge distillation works in personalized FL at a high level. Ablation studies in Section 4 also show the superiority of self-knowledge distillation over other regularizers.

Combining Ensemble and Knowledge Distillation Implicitly. We argue that the benefits of self-distillation in FL partially come from an implicit ensemble and knowledge distillation. Recent work[15] suggests that in many real-world datasets exists a "multi-view" structure where each class of data has multiple view features. However, not all of the data contain full view features. For instance, a car picture could be properly classified by recognizing its windshields, wheels or headlights. However, the car pictures taken from the side only contain the wheels feature. The data where multiple features exist are *multi-view data*, and the rest are *single-view data*. The multi-view data could be classified correctly with only part of the features and no longer provide gradients. Then the neural network memorizes the remaining data without learning any new features. For this reason, the individual network only learns a subset of the view features during the training process. Consider FL scenario, the local model w_k is initialized with the latest global model. w_k learns a subset of the features based on the generalized initialization during the local training phase. Moreover, the local model also learns the subset of features already learned by v_k in a knowledge distillation manner. This process can be viewed as "first ensemble learning w_k and v_k, then distilling the ensemble knowledge to v_k." In other words, the personalized models of pFedSD cover more view features of local data compared to other FL methods. Based on that, pFedSD improves the performance of personalized models and achieves faster convergence and better training stability.

TABLE 1
Statistics of the Datasets

Dataset	# Samples	# Class	Task
Fashion-MNIST	70,000	10	Image classification
CIFAR-10	60,000	10	Image classification
CIFAR-100	60,000	100	Image classification

4 Experiments

4.1 Experimental Setup

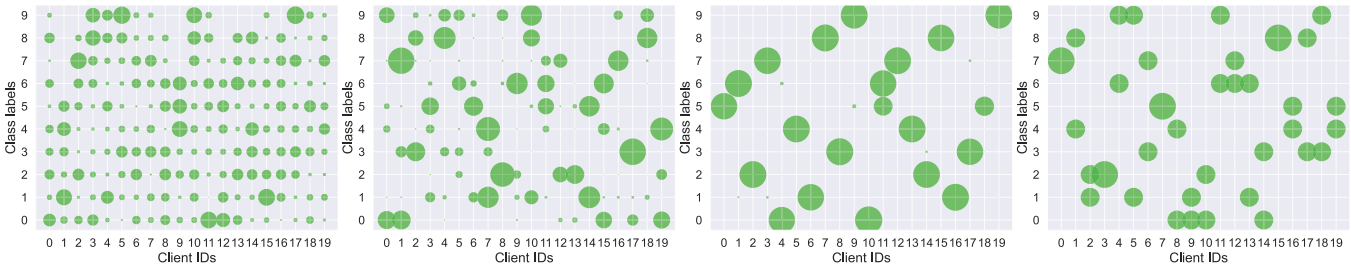
We assume two different FL scenarios: 1) $K = 20$ clients with $r = 100\%$ participation ratio; 2) $K = 100$ clients with $r = 10\%$ participation ratio. We run for $T = 50$ communication rounds and $T = 100$ communication rounds, respectively. Participating clients execute $E = 5$ epochs in every round of local training. For all methods, we record the average test accuracy of all clients for evaluation. We report the results with mean and standard deviation over three different random seeds.

Datasets and Tasks. We evaluate our proposed FL framework on different image classification tasks: Fashion-MNIST (FMNIST)[24] and CIFAR-10/100[25]. We give details of the datasets in Table 1. For each dataset, we employ two different non-IID data settings. (1) Pathological non-IID[1]: each client only holds at most s classes of samples. In other words, s is the number of shards per client and represents the maximum number of classes each client can have. As s decreases, the degree of data heterogeneity increases. (2) Dirichlet non-IID[26]: we use Dirichlet distribution $Dir(\alpha)$ to create disjoint heterogeneous client data. The value of α controls the extent of label skew. Smaller α leads to more non-IID data partitions, as it makes the distribution of $p_k(y)$ more biased for client k. We adopt a balanced partition strategy with Dirichlet distribution following [27]. Fig. 5 presents how data samples are scattered among 20 clients on the CIFAR-10 dataset. All datasets are split randomly by an 80-20 ratio for training and testing, respectively.

Model Structure. Following the previous evaluation settings [9], [28], [29], we use a simple CNN for Fashion-MNIST dataset following [28] and 5-layer CNNs for CIFAR dataset similar to [1].

Baselines. We compare pFedSD against SOTA FL methods which can be conveniently divided into two categories: (1) non-personalized FL methods (FedAvg[1], FedProx[12]), (2) personalized FL methods (FedPer[22], LG-FedAvg[28], pFedMe[9], FedFomo[29]). We give a brief introduction to these baselines as follows.

- FedAvg[1] is proposed in the pioneering work of federated learning. It simply averages the parameters of local models to learn a global model.
- FedProx[12] adds a proximal term to the local objective, which prevents the client's updates deviates too much from the global model. It effectively mitigates the issue of non-IID in conventional FL.



(a) Dirichlet $\alpha = 1$ (b) Dirichlet $\alpha = 0.1$ (c) Dirichlet $\alpha = 0.01$ (d) Pathological $s = 2$

Fig. 5. Illustration of statistical heterogeneity among 20 clients on CIFAR-10, where the x-axis indicates client IDs, the y-axis indicates class labels, and the size of dots indicates # of samples per class allocated to each client.

TABLE 2
Personalized Accuracy Overview With Dirichlet Distribution $\alpha = 0.1$

Datasets	Scale	FedAvg	FedProx	FedPer	LG-FedAvg	pFedMe	FedFomo	pFedSD
FMNIST	20 clients	90.15 ± 0.10	90.05 ± 0.13	96.30 ± 0.03	95.22 ± 0.22	93.24 ± 0.08	95.38 ± 0.04	$\mathbf{96.57 \pm 0.08}$
	100 clients	86.82 ± 0.39	76.78 ± 0.30	94.99 ± 0.17	91.41 ± 0.08	92.05 ± 0.08	92.06 ± 0.14	$\mathbf{95.97 \pm 0.06}$
CIFAR-10	20 clients	50.44 ± 0.68	50.50 ± 0.78	80.74 ± 0.47	78.61 ± 0.31	79.51 ± 0.28	79.29 ± 0.34	$\mathbf{82.08 \pm 0.46}$
	100 clients	49.04 ± 0.80	42.11 ± 0.95	78.89 ± 0.90	71.60 ± 1.52	74.62 ± 0.72	73.16 ± 0.65	$\mathbf{80.22 \pm 0.24}$
CIFAR-100	20 clients	32.24 ± 0.32	32.50 ± 0.55	52.12 ± 0.22	40.71 ± 0.05	38.42 ± 0.71	44.67 ± 0.37	$\mathbf{55.27 \pm 0.30}$
	100 clients	29.13 ± 0.18	14.27 ± 0.35	44.45 ± 0.64	21.61 ± 0.21	27.07 ± 0.25	26.48 ± 0.27	$\mathbf{48.91 \pm 0.77}$

- FedPer[22] decouples the neural network into the body and the head. It learns the entire network jointly but only aggregates the body to share a global representation, i.e., the body is shared but the head is personalized.
- LG-FedAvg[28] also proposed a parameter decoupling method. But it learns compact local representations for the individual client and global head across all clients, i.e., the body is personalized but the head is shared.
- pFedMe[9] uses the Moreau envelopes as clients' regularized loss function to control the distance between the personalized model and the global model.
- FedFomo[29] computes the first-order approximations of the optimal model weighted combination for each client. The client must maintain an additional validation set to determine how suitable other clients are towards its local objectives.

Hyperparameters Setting. We follow the universal hyperparameter settings of related work. We set the learning rate as 0.01 for all methods. We adopt the SGD optimizer with weight decay $1e-5$ and momentum 0.9 all through the experiments. The training batch size on all the tasks is $B = 64$.

We ran the baselines under the suggested settings in corresponding papers or code. For FedProx, the proximal term μ is selected from $\{0.001, 0.01, 0.1, 1\}$. The best μ of FedProx for Fashion-MNIST, CIFAR-10, and CIFAR-100 are $0.001, 0.01, 0.001$. For FedPer, we keep the last two layers as the personalized head for Fashion-MNIST and the last linear layer for CIFAR. For LG-FedAvg, we keep the last two layers as the shared head for all datasets. For pFedSD, the regularization parameter λ is selected from [15,20,30]. For FedFomo, we set the number of models downloaded $M = 5$, ϵ-greedy parameter $\epsilon = 0.3$ with 0.05 decay each round. For our method, we select λ from $\{0.1, 0.5\}$ and tune temperature τ from $\{1, 3\}$.

Implementation. We implement all baselines mentioned above in PyTorch. We simulate the server and a set of clients in a multiprocessing manner and adopt MPI as the communication backend. All experiments are conducted on a deep learning server equipped with four V100 GPUs.

4.2 Performance Overview

Accuracy Comparison. We compare the personalized top-1 test accuracy under the Dirichlet distribution $\alpha = 0.1$, as reported in Table 2. pFedSD consistently outperforms alternative methods across datasets and client setups. It outperforms the best baseline by up to 4.46%. The non-personalized FL algorithms (FedAvg and FedProx) perform poorly in most cases. Note that FedProx does not perform well compared to FedAvg in partial-participation scenarios by reason of pulling the local optimization direction to the global distribution. The recent personalized FL method FedFomo does not perform well in the common non-IID settings. We think the strict aggregation strategy makes FedFomo hard to benefit from other clients with different data distribution. We notice that FedPer shows a sub-optimal performance compared to our methods in most cases. We think FedPer draws strength from a shared representation. The benefits of learning a good representation have been discovered in many recent works [22], [23], [30]. pFedSD shows superior performance on more difficult tasks such as CIFAR-100. The performance of pFedSD on large-scale FL systems implies the scalability of our method, which is important in real-world deployment for edge scenarios.

We also show the personalized performance under the pathological distribution $s = 2$ for CIFAR-10 and Fashion-

TABLE 3
Personalized Accuracy Overview With Pathological Distribution

Datasets	Scale	FedAvg	FedProx	FedPer	LG-FedAvg	pFedMe	FedFomo	pFedSD
FMNIST	20 clients	75.71 ± 0.27	75.40 ± 1.06	99.42 ± 0.01	99.15 ± 0.04	98.76 ± 0.03	99.28 ± 0.01	$\mathbf{99.45 \pm 0.01}$
	100 clients	84.13 ± 0.44	74.38 ± 0.90	97.28 ± 0.12	95.70 ± 0.08	94.87 ± 0.03	96.14 ± 0.41	$\mathbf{97.42 \pm 0.04}$
CIFAR-10	20 clients	45.37 ± 0.53	45.86 ± 1.12	91.69 ± 0.10	92.21 ± 0.36	89.69 ± 0.39	91.66 ± 0.17	$\mathbf{92.52 \pm 0.24}$
	100 clients	43.52 ± 1.70	38.24 ± 1.39	86.66 ± 0.24	79.25 ± 1.63	80.47 ± 1.10	81.62 ± 0.54	$\mathbf{86.81 \pm 0.32}$
CIFAR-100	20 clients	31.92 ± 0.23	32.16 ± 0.32	55.57 ± 0.26	45.48 ± 0.63	38.66 ± 1.21	48.84 ± 0.11	$\mathbf{59.54 \pm 0.01}$
	100 clients	27.88 ± 0.35	13.32 ± 0.37	47.01 ± 0.50	18.96 ± 0.30	26.96 ± 0.30	25.21 ± 0.63	$\mathbf{50.99 \pm 0.43}$

MNIST, $s = 20$ for CIFAR-100, as reported in Table 3. pFedSD still outperforms other approaches in pathological non-IID distribution. Some personalized methods perform even worse than the non-personalized FL method (FedAvg) on CIFAR-100. We attribute the performance degradation of these methods to poor generalization. Both Dirichlet distribution $(\alpha) = 0.1$ and pathological distribution $(s) = 20$ provide more difficult tasks for the reason of more classes but fewer data per class for each individual. So the client needs more generalization power from FedAvg to exploit the common knowledge.

Communication Efficiency. Fig. 6 shows the personalized accuracy in each round during training across different datasets and settings. Compared to other baselines, pFedSD shows faster and more smooth convergence and consistently outperforms others. The growth speed of performance in pFedSD is nearly the same as FedPer in the early stages. But pFedSD can achieve a better result later, which benefits from self-knowledge distillation. In the 20-clients scenario on CIFAR-10 dataset, the learning curve of FedPer first reaches its highest and then oscillates in a downtrend, while pFedSD still keeps stable. We conjecture that the unstable performance of FedPer is caused by overfitting. Self distillation brings the regularization effect to relieve the overfitting of local data.

From Fig. 6, we can also observe that pFedSD spends the least communication rounds to reach the same accuracy. Table 4 shows how many communication rounds it takes to achieve the same accuracy as running FedPer for 100 rounds. Assume $K = 100$ clients with $r = 10\%$ participation ratio and the Dirichlet parameter $\alpha = 0.1$. The speedup of pFedSD is up to $5\times$ compared to FedPer. In general, pFedSD is more communication efficient than other methods due to the faster convergence.

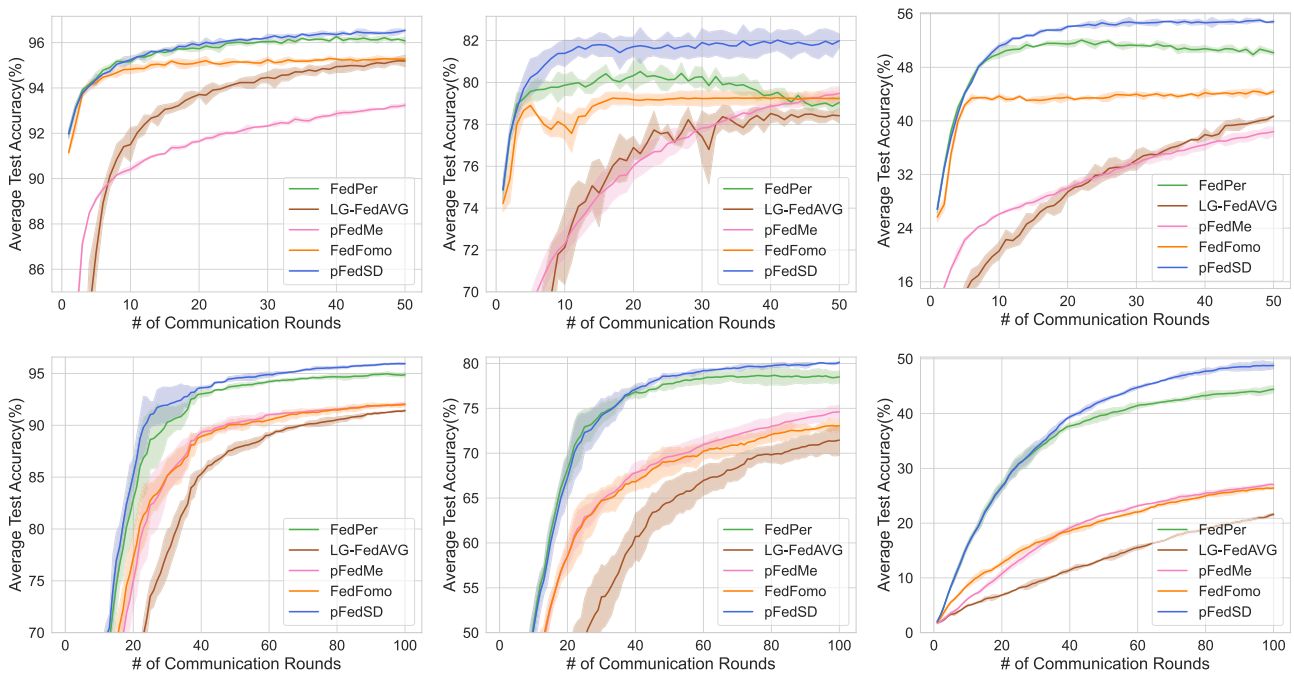
Fairness Among Clients. Due to the data heterogeneity, the personalized performance of different clients may vary significantly. To examine how fair the improvements of personalized performance are across clients, we study the performance of the personalized model held by every client in detail. Following the precedent works[21], [31], we compute the *Standard Deviation* (SD) across all clients for the 100-clients scenario in Table 5. A lower SD implies that the performance distribution of all clients is more uniform and the algorithm is fairer. The common goal of PFL is to reduce the variance while keeping a reasonable mean test accuracy. The SD of our method is the least on Fashion-MNIST and CIFAR-10. Although our method has a slightly higher SD on CIFAR-100, our test accuracy significantly outperforms others. We also show the overall distribution of the performance of all clients in Fig. 7. pFedSD always has more clients with higher testing accuracy on all datasets. In general, pFedSD achieves better fairness and results in more clients with higher accuracy.

4.3 Sensitivity Analysis

Impacts of Participation Ratio. As shown in Table 6, we explore different participation ratios r among $\{20\%, 60\%, 100\%\}$ for each communication round on the CIFAR-100 dataset. Our method outperforms all baselines clearly regardless of the setting of participation ratio. When r rises from 20% to 60%, the accuracy of all methods increases as there are more training rounds for each client overall. When r rises from 60% to 100%, the performances of some approaches, including pFedSD, drop for the reason that clients may overfit their local data. The performance of FedProx notably increases as r rises for the reason that the latest global model in the proximal term is more suitable to the global distribution.

Effects of Data Heterogeneity. We analyze different levels of data heterogeneity on the CIFAR-100 dataset. We control the degree of statistical heterogeneity by varying the concentration parameter α of Dirichlet distribution from $\{0.01, 0.1, 1\}$. For a smaller α, the data distributions among clients will be more unbalanced. As shown in Table 7, our method always achieves the best accuracy among three heterogeneity levels, especially in highly non-IID settings. When the data distribution becomes more balanced ($\alpha = 1$), our method still maintains a clear advantage while some other personalized FL methods perform even worse than non-personalized methods. The experiments show the effectiveness and robustness of pFedSD about the data heterogeneity levels.

Robustness of Model Architectures. We also conduct experiments on more popular model architectures. We evaluate our methods on ResNet-8[32] and MobileNetV2[33]. As shown in Table 8, pFedSD still outperforms the baselines on the modern models. Compared to simple CNN, FedProx narrows the gap to the FedAvg as the model complexity increases. LG-FedAvg shows poor performance on modern neural networks. Since ResNet and MobileNet have only one linear layer at the end of the model, and the head is defined as this linear layer, LG-FedAvg almost degenerates to the local-only training for only aggregating the head. FedPer still maintains its competitiveness on modern models. FedPer is more related to representation learning and LG-FedAvg is more related to linear decision boundary learning. The significant gap in performance between LG-



(a) Fashion-MNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 6. Average test accuracy of pFedSD compared to baselines in different number of communication rounds on various datasets. The non-IID Dirichlet parameter $\alpha = 0.1$. Shaded regions show the standard deviation over 3 trials with different random seeds. **Top** $K = 20$ clients with $r = 100\%$ participation ratio. **Bottom** $K = 100$ clients with $r = 10\%$ participation ratio.

FedAvg and FedPer shows the power of representation learning in PFL.

4.4 Ablation Studies

Effects of λ and τ. We introduce two hyperparameters in our method, one is coefficient λ and another is temperature τ. Fig. 8 shows the varying performance under different hyperparameters. We tried a set of λ selected from $\{0.01, 0.1, 0.5, 1\}$ and τ from $\{1, 3\}$.

pFedSD is robust to the hyperparameters on easier tasks such as Fashion-MNIST. When the task is easier, the local model recalls the personalized knowledge from the latest initialized point in fewer updates. In other words, the forgetting phenomenon is easy to overcome in such tasks. The performances when $\tau = 3$ are higher than $\tau = 1$ on all datasets. The higher temperature indicates more attention paid to negative labels during self-distillation. An appropriate temperature brings more informative supervisions. The λ controls the influence of self-distillation in local training, and we fix the value of λ in our experiments. Fig. 8 demonstrates that a large value of λ cannot always make better performance. $\lambda = 1$ means that the contribution of regular training and recalling historical knowledge becomes equal, which may cause performance degradation. As a result, a moderate λ is necessary to achieve the best accuracy.

TABLE 4
Number of Communication Rounds of pFedSD to Reach the Same Accuracy as FedPer

Method	Fashion-MNIST	CIFAR-10	CIFAR-100
FedPer	100	100	100
pFedSD	20(5×)	57(1.8×)	43(2.3×)

We also consider a dynamic strategy for tuning λ. Obviously, the personalized model v_k becomes closer to optimal with the communication rounds increasing. So the intuitive idea is to ramp up the λ during FL process to avoid the large contribution of distillation in early rounds. We leave the more fine-grained tuning strategy in future work.

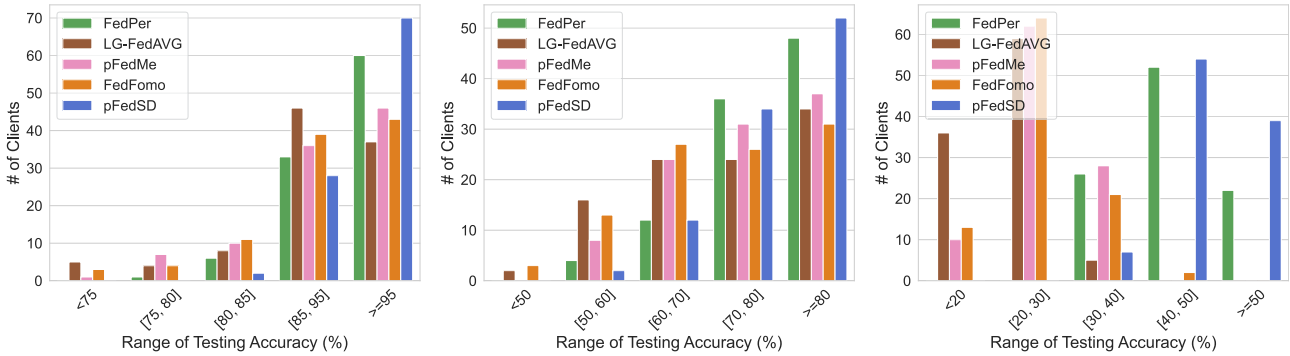
Different Regularizers. In addition to self knowledge distillation, there are other forms of knowledge transfer. We also conduct experiments to explore the effects of alternative methods. In particular, we consider the following objectives:

- apply an L2 regularizer between the local model w_k^t and the previous personalized model v_k.
- incorporate *Elastic Weight Consolidation* (EWC)[19] in the local training to prevent *'catastrophic forgetting.'* EWC leverages Fisher Information Matrix to restrict the changes of the model parameters important for previous tasks.
- replace KL divergence to *Mean Squared Error* (MSE) in distillation loss.

As shown in Table 9, L2 and EWC perform worst on all datasets, which demonstrates that self-distillation brings

TABLE 5
Average Accuracy and Standard Deviation across All Clients

Method	Fashion-MNIST	CIFAR-10	CIFAR-100
FedPer	94.69 ± 5.08	79.25 ± 10.20	44.45 ± 5.93
LG-FedAvg	91.31 ± 7.20	72.56 ± 12.26	21.78 ± 4.95
pFedMe	92.14 ± 6.82	75.43 ± 11.29	26.82 ± 5.07
FedFomo	91.96 ± 6.79	73.09 ± 11.91	26.43 ± 5.52
pFedSD	95.92 ± 3.88	80.17 ± 9.23	48.62 ± 6.00



(a) Fashion-MNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 7. The testing accuracy distribution across all clients.

additional benefits from data perspectives compared to parameter perspectives. The insights about why self-distillation works have been given in Section 3.3.2. The accuracy of MSE-based distillation is slightly higher than KL-based distillation on Fashion-MNIST and CIFAR-10, but nearly 1% worse on CIFAR-100.

4.5 System Performance Metrics

Although the main benefits of pFedSD are the improvement in accuracy and convergence speed, We evaluate the system metrics to show the robustness of our method. System performance metrics mainly focus on communication and computational efficiency[10]. Communication efficiency depends on the number of communication rounds and the communication cost between the server and the clients per round. Computational efficiency is evaluated by the number of floating point operations (FLOPs) and execution time.

We consider three kinds of overheads in FL system: execution time, network cost, and computational cost. We assume $K = 20$ clients with $r = 60\%$ participation ratio.

TABLE 6
Accuracy With Different Participation Ratio on CIFAR-100

Method	$r = 20\%$	$r = 60\%$	$r = 100\%$
FedAvg	28.09 ± 0.15	31.54 ± 0.35	32.24 ± 0.32
FedProx	16.52 ± 0.42	21.23 ± 0.46	32.50 ± 0.55
FedPer	53.25 ± 0.13	52.14 ± 0.65	52.12 ± 0.22
LG-FedAvg	29.53 ± 0.73	37.77 ± 0.08	40.71 ± 0.05
pFedMe	36.97 ± 0.52	38.64 ± 0.38	38.42 ± 0.71
FedFomo	42.42 ± 0.29	44.14 ± 0.21	44.67 ± 0.37
pFedSD	$\mathbf{55.24 \pm 0.55}$	$\mathbf{56.61 \pm 0.34}$	$\mathbf{55.27 \pm 0.30}$

TABLE 7
Accuracy With Different Data Heterogeneity on CIFAR-100

Method	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$
FedAvg	29.02 ± 0.43	31.54 ± 0.35	33.78 ± 0.33
FedProx	18.19 ± 0.44	21.23 ± 0.46	31.42 ± 0.43
FedPer	64.47 ± 0.43	52.14 ± 0.65	34.12 ± 0.27
LG-FedAvg	54.27 ± 0.64	37.77 ± 0.08	21.36 ± 0.58
pFedMe	46.97 ± 0.67	38.64 ± 0.38	24.95 ± 0.35
FedFomo	58.52 ± 0.37	44.14 ± 0.21	29.41 ± 0.03
pFedSD	$\mathbf{67.11 \pm 0.58}$	$\mathbf{56.61 \pm 0.34}$	$\mathbf{37.81 \pm 0.33}$

Execution Time. As shown in Table 10, we measure the execution time per round along with the percentage of time spent on communicating (downloading and uploading), local training, and working on the server (model aggregation and distribution). LG-FedAvg takes the least time as the clients only communicate the shared classifiers with the server. FedFomo takes the longest time due to the activated clients requiring additional downloads for other clients' models. pFedMe takes a little longer time than other algorithms for the reason that the clients need more computation to train the personalized models and local models at the same time. pFedSD takes comparable time to FedAvg and FedProx.

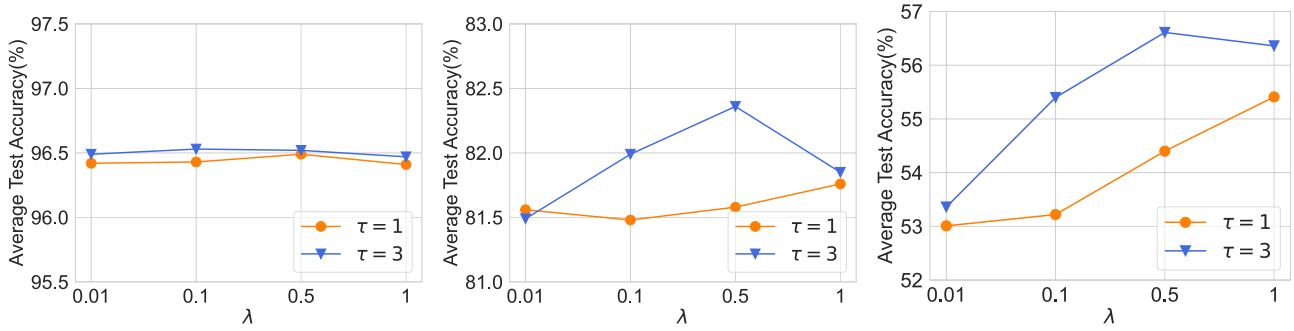
Under the premise of achieving the same accuracy, the total execution time equals the execution time per round times the communication rounds. Although pFedSD introduces a little time overhead in each round, it substantially accelerates the convergence and achieves the target accuracy with fewer rounds. As shown in Section 4.2, FedPer takes up to 5X communication rounds compared to pFedSD. Overall, our method reduces the total execution time to achieve the same target accuracy. From Table 10, the execution time of each round is mainly spent on local training rather than communication or the operations on the

TABLE 8
Accuracy With Different Model Architectures on CIFAR-100

Method	ResNet	MobileNet
FedAvg	23.57 ± 1.72	31.92 ± 1.10
FedProx	20.81 ± 0.66	31.36 ± 0.75
FedPer	57.20 ± 0.68	59.74 ± 0.38
LG-FedAvg	16.02 ± 0.26	18.26 ± 1.33
pFedMe	32.96 ± 0.98	42.88 ± 0.77
FedFomo	45.62 ± 0.31	41.87 ± 0.34
pFedSD	$\mathbf{59.74 \pm 0.90}$	$\mathbf{63.28 \pm 0.28}$

TABLE 9
Accuracy With Different Regularizers

Method	Fashion-MNIST	CIFAR-10	CIFAR-100
L2	96.43 ± 0.03	81.56 ± 0.34	52.76 ± 0.54
EWC	96.44 ± 0.05	81.64 ± 0.46	52.86 ± 0.32
MSE	$\mathbf{96.56 \pm 0.12}$	$\mathbf{82.41 \pm 0.40}$	55.70 ± 0.41
KL	96.53 ± 0.08	82.36 ± 0.70	$\mathbf{56.61 \pm 0.34}$



(a) Fashion-MNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 8. Effects of λ and τ.

server. This indicates that the execution time is primarily influenced by the computing power of the edge devices and the computational complexity of local training.

Communication Cost. As shown in Table 11, we measure the communication cost per client between the client and the server during a single round. The communication cost includes the total number of downloaded and uploaded bytes. For the vanilla FL scheme like FedAvg and FedProx, each activated client downloads the global model from the server at the beginning of the round and uploads the local model to the server at the end of the round. pFedSD and pFedMe do not introduce extra communication overhead. FedPer and LG-FedAvg transmit part of the model and thus reduces the communication cost. FedPer exchanges the body, and LG-FedAvg only exchanges the head. In our settings, the head only contains the last two layers of the model leading to a significant reduction of communication cost for LG-FedAvg. In FedFomo, each participating client needs to download M models of other clients to obtain the optimal model combination. So FedFomo brings M times the additional communication cost compared to FedAvg ($M = 5$ in our settings).

Computational Cost. We also measure the computational cost, in Table 11, of local training for each client at a single round. The FLOPs are counted as the sum of amounts for training via the FLOPs counting tool, pytorch-OpCounter[3]. The computational cost of FedAvg is 906.15 GFLOPs. LG-FedAvg and FedPer keep the same computational cost as FedAvg. Besides the normal forward and backward FLOPs, FedProx and pFedMe introduce the additional negligible computational cost to compute the regularization term. FedFomo needs to compute the weight vector for aggregating the clients' models before local training.

Our method obtains the soft target for KD in two schemes. One is to store the past predictions of the personalized model v_k. The other is to recompute the predictions when local training. The detailed analysis is presented in Section 3.2. For the former scheme, the client does not introduce any computational overhead compared to the standard FL algorithms. For the latter scheme, the client k needs to compute the forward pass of v_k for one epoch. Under our settings, the computational cost per round of pFedSD increases by 6.7% compared to FedAvg. However, the increase in computational cost per round is likely to be offset by the far fewer rounds required to reach the target accuracy (refer to Table 4).

3. https://github.com/Lyken17/pytorch-OpCounter

5 Related Work

5.1 Personalized FL

The essential incentive for clients to engage in personalized FL process is to obtain a better model than the local-only models on each client's data distribution. Most of the recent works preserve the traditional FL paradigm, which keeps a single global model on the server. These works customize a personalized model for each client from the global model and can be categorized into four types: local fine-tuning[16], [34], regularization[9], [20], [21], [35], model interpolation [36], meta learning[37], [38], and model mixture/parameter decoupling[6], [22], [23], [28], [30], [36], [39]. The methods mentioned above leverage the single global model to transfer group knowledge involving all clients. The sole global model may lose generalization diversity in some data distributions. Therefore, recent works keep multiple global models on the server or client side to achieve a fine-grained generalization. Cluster-based FL methods aim to cluster clients into several groups and develop global models corresponding to each group. FedAMP[40] maintains personalized cloud models for each client on the server to enforce stronger pair-wise collaboration between similar clients. FedFomo[29] allows clients to download models of other clients and use local validation data to calculate personalized weighted combinations. These methods perform well only in the scenario where the local data distributions among clients show a strong characteristic of clustering. And the latter needs additional validation data representing target distribution, which is unrealistic. Unlike the aforementioned works, our

TABLE 10
Average Time Per Round on CIFAR-100, and Percentage of Time on Communication (Comm), Local Training (Client), and Model Aggregation/Distribution on the Server (Server)

Method	Time (s)	Comm (%)	Client (%)	Server (%)
FedAvg	5.69	12.70	83.00	4.30
FedProx	6.26	11.91	84.69	3.39
FedPer	5.42	11.66	85.41	2.93
LG-FedAvg	5.34	13.14	85.75	1.11
pFedMe	9.31	8.11	90.20	1.69
FedFomo	38.78	4.14	95.58	0.28
pFedSD	7.23	8.27	90.63	1.10

TABLE 11
Communication Cost and Computational Cost Per Round on CIFAR-100

Method	Communication Cost (MB)	Computational Cost (FLOPs)
FedAvg	2.15	906.15G
FedProx	2.15	906.25G
FedPer	2.12	906.15G
LG-FedAvg	0.03	906.15G
pFedMe	2.15	906.35G
FedFomo	7.53	951.47G
pFedSD	2.15	906.15G / 966.56G (+6.7%)

method is more adaptable and robust to universal settings and does not need additional data.

5.2 Knowledge Distillation in FL

Knowledge Distillation for the neural network is first introduced by [41] and has shown remarkable success in many areas. By letting the student imitate the teacher's output, the knowledge can be transferred from teacher to student. Conventional KD exploits knowledge from a large and powerful teacher model to generate soft targets for a student network. A few recent papers[42], [43] propose self-distillation that a teacher model with the identical structure of the student model could improve the student's generalization ability over the teacher. It is worth emphasizing that our method is motivated by the idea of self-knowledge distillation but does not strictly follow its paradigm.

KD and its variants [44], [45] have been used as effective techniques in FL. Most approaches[7], [27] adopt KD to transfer knowledge from clients to the server, i.e., local models are teachers and global model is the student. In this way, the global model gains higher performance and more stable convergence. These approaches are sensitive to proxy data for performing KD on the server. Notably, it is not easy to get or generate such data (whether labeled or not). [46], [47], [48] apply co-distillation to FL by exchanging model output instead of model parameters to reduce communication cost and handle model heterogeneity. [46] exposes private data logits to other clients, whereas [47], [48] assume a public dataset among clients. FedGen[49] makes KD feasible for FL in a data-free manner where the server learns a generator to ensemble user information and passes it to clients to supervise local training. Besides, FML[50] allows clients to train local models mutually with the global model based on DML. FedGKT[51] combines FL with Split Learning and employs KD to facilitate dual knowledge transfer between edge and server. A recent preprint FedLSD[52] also applies the concept of self-distillation to obtain a better global model in a non-IID environment. However, FedLSD distills the knowledge from global model during local training, which looks more like a soft version of FedProx[12].

In brief, most of the previous works assume public/proxy data, which need delicate consideration and even prior knowledge of clients' private data. On the contrary, our method does not relax the assumption in standard FL system without introducing any additional data or local information. Furthermore, our approach makes full use of existing resources and could be easily integrated into popular personalized FL systems.

6 Conclusion

In this article, we propose a simple yet effective personalized federated learning framework, coined pFedSD, which aims to tackle the challenge of statistical heterogeneity in edge scenarios. Through empirical observations, we investigate the personalized knowledge forgetting phenomena caused by the initialization in FL. Our method makes full use of the historical personalized models via self-distillation, which relieves the forgetting and purses a better trade-off between personalization and generalization. Experimental results on real-world datasets demonstrate the effectiveness and robustness of pFedSD over the existing works.

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 *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[2] W. Y. B. Lim et al., "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Mar. 2022.

[3] Q. Ye, W. Lu, S. Hu, and X. Xu, "Resource optimization in wireless powered cooperative mobile edge computing systems," *Sci. China Inf. Sci.*, vol. 64, no. 8, pp. 1–10, 2021.

[4] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 59–71, Jan. 2021.

[5] J. Xing, J. Tian, Z. Jiang, J. Cheng, and H. Yin, "Jupiter: A modern federated learning platform for regional medical care," *Sci. China Inf. Sci.*, vol. 64, no. 10, pp. 1–14, 2021.

[6] W. Zhuang et al., "Performance optimization of federated person re-identification via benchmark analysis," in *Proc. ACM Int. Conf. Multimedia*, 2020, pp. 955–963.

[7] D. Sui, Y. Chen, J. Zhao, Y. Jia, Y. Xie, and W. Sun, "FedED: Federated learning via ensemble distillation for medical relation extraction," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2020, pp. 2118–2128.

[8] V. Smith, C. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4424–4434.

[9] C. T Dinh, N. Tran, and T. D. Nguyen, "Personalized federated learning with moreau envelopes," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 21394–21405.

[10] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2022.3160699.

[11] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018, *arXiv:1806.00582*.

[12] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2020, pp. 429–450.

[13] A. Hard, "Federated learning for mobile keyboard prediction," , 2018, *arXiv:1811.03604*.

[14] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *J. Healthcare Inform. Res.*, vol. 5, no. 1, pp. 1–19, 2021.

[15] Z. Allen-Zhu and Y. Li, "Towards understanding ensemble, knowledge distillation and self-distillation in deep learning," 2020, *arXiv:2012.09816*.

[16] T. Yu, E. Bagdasaryan, and V. Shmatikov, "Salvaging federated learning by local adaptation," 2020, *arXiv:2002.04758*.

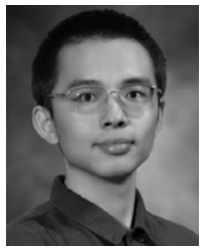
[17] J. Mills, J. Hu, and G. Min, "Multi-task federated learning for personalised deep neural networks in edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 630–641, Mar. 2022.

[18] B. Sun, H. Huo, Y. Yang, and B. Bai, "PartialFed: Cross-domain personalized federated learning via partial initialization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 23309–23320.
[19] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017.
[20] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," 2020, *arXiv:2002.05516*.
[21] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6357–6368.
[22] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," 2019, *arXiv:1912.00818*.
[23] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 2089–2099.
[24] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[25] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, Univ. Toronto, Toronto, Canada, 2009.
[26] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019, *arXiv:1909.06335*.
[27] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 2351–2363.
[28] P. P. Liang, "Think locally, act globally: Federated learning with local and global representations," 2020, *arXiv:2001.01523*.
[29] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, "Personalized federated learning with first order model optimization," in *Proc. Int. Conf. Learn. Representations*, 2021.
[30] J. Oh, S. Kim, and S.-Y. Yun, "Fedbabu: Towards enhanced representation for federated image classification," 2021, *arXiv:2106.06042*.
[31] T. Li, M. Sanjabi, A. Beirami, and V. Smith, "Fair resource allocation in federated learning," in *Proc. Int. Conf. Learn. Representations*, 2020.
[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[33] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[34] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, "Federated evaluation of on-device personalization," 2019, *arXiv:1910.10252*.
[35] F. Hanzely, S. Hanzely, S. Horváth, and P. Richtarik, "Lower bounds and optimal algorithms for personalized federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 2304–2315.
[36] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," 2020, *arXiv:2003.13461*.
[37] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 3557–3568.
[38] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," 2019, *arXiv:1909.12488*.
[39] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," 2020, *arXiv:2002.10619*.
[40] Y. Huang et al., "Personalized cross-silo federated learning on non-IID data," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 7865–7873.
[41] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[42] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3713–3722.
[43] K. Kim, B. Ji, D. Yoon, and S. Hwang, "Self-knowledge distillation with progressive refinement of targets," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6567–6576.
[44] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4320–4328.
[45] R. Anil, G. Pereyra, A. Passos, R. Ormándi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," in *Proc. Int. Conf. Learn. Representations*, 2018.
[46] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," 2018, *arXiv:1811.11479*.
[47] D. Li and J. Wang, "Fedmd: Heterogenous federated learning via model distillation," 2019, *arXiv:1910.03581*.
[48] J. Zhang, S. Guo, X. Ma, H. Wang, W. Xu, and F. Wu, "Parameterized knowledge transfer for personalized federated learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 10092–10104.
[49] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12878–12889.
[50] T. Shen et al., "Federated mutual learning," 2020, *arXiv:2006.16765*.
[51] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large CNNs at the edge," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 14068–14080.
[52] G. Lee, Y. Shin, M. Jeong, and S.-Y. Yun, "Preservation of the global knowledge by not-true self knowledge distillation in federated learning," 2021, *arXiv:2106.03097*.

Hai Jin (Fellow, IEEE) received the PhD degree in computer engineering from Huazhong University of Science and Technology, in 1994. He is a chair professor of computer science and engineering with the Huazhong University of Science and Technology (HUST) in China. In 1996, he was awarded a German Academic Exchange Service fellowship to visit the Technical University of Chemnitz in Germany. He worked with The University of Hong Kong between 1998 and 2000, and as a visiting scholar with the University of Southern California between 1999 and 2000. He was awarded Excellent Youth Award from the National Science Foundation of China, in 2001. He is a Fellow of CCF, and a life member of the ACM. He has co-authored more than 20 books and published more than 900 research papers. His research interests include computer architecture, parallel and distributed computing, big data processing, data storage, and system security.

Dongshan Bai is currently working toward the MS degree in computer science with the School of Computer Science and Technology, Huazhong University of Science and Technology, China. His main research interests include distributed machine learning and federated learning.

Dezhong Yao (Member, IEEE) received the PhD degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2016. He was a research fellow with Nanyang Technological University, Singapore between 2016 and 2019, and as a visiting scholar with the Carnegie Mellon University, Pittsburgh USA between 2010 to 2012. He is currently an associate professor with the School of Computer Science and Technology, HUST. His research interests include large-scale machine learning, federated learning, and distributed optimization.



Yutong Dai is working towards the PhD degree in industrial and systems engineering with Lehigh University, USA. His main research interests is in designing, analyzing and implementing algorithms for large scale non-convex non-smooth optimization problems arisen in machine learning and federated learning.

Lin Gu received the MS and PhD degrees in computer science from the University of Aizu, Fukushima, Japan, in 2011 and 2015, respectively. She is currently an associate professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, China. Her current research interests include serverless computing, network function virtualization, cloud computing, software-defined networking, and data center networking. She has authored 2 books and more than 40 papers in refereed journals and conferences in these areas. She is a senior number of CCF.

Chen Yu received the BS degree in mathematics from Wuhan University, Wuhan, China, in 1998, the MS degree in computer science from Wuhan University, Wuhan, China, in 2002, and the PhD degree in information science from Tohoku University, Sendai, Japan, in 2005. Since 2008, he has been with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, where he is currently a full Professor and a special research fellow, working in the areas of wireless sensor networks, ubiquitous computing, and green communications.

Lichao Sun received the PhD degree from the University of Illinois at Chicago, Chicago, IL, USA, in 2020. He is currently an assistant professor with Lehigh University, Bethlehem, PA, USA. His research interests include federated learning, reinforcement learning and deep learning. He has studied on various computer vision, natural language processing and graph applications. He has published more than 40 papers in top-tier conferences and journals, such as NeurIPS, KDD, AAAI, IJCAI, ICLR, CCS, *Usenix-Secuirty*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Mobile Computing* and *IEEE Transactions on Neural Networks and Learning Systems*.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.