Parameterized Knowledge Transfer for Personalized Federated Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

In recent years, personalized federated learning (pFL) has attracted increasing attention for its potential in dealing with statistical heterogeneity among clients. However, the state-of-the-art pFL methods rely on model parameters aggregation at the server side, which require all models to have the same structure and size, and thus limits the application for more heterogeneous scenarios. To deal with such model constraints, we exploit the potentials of heterogeneous model settings and propose a novel training framework to employ personalized models for different clients. Specifically, we formulate the aggregation procedure in original pFL into a personalized group knowledge transfer training algorithm, namely, KT-pFL, which enables each client to maintain a personalized soft prediction at the server side to guide the others' local training. KT-pFL updates the personalized soft prediction of each client by a linear combination of all local soft predictions using a knowledge coefficient matrix, which can adaptively reinforce the collaboration among clients who own similar data distribution. Furthermore, to quantify the contributions of each client to others' personalized training, the knowledge coefficient matrix is parameterized so that it can be trained simultaneously with the models. The knowledge coefficient matrix and the model parameters are alternatively updated in each round following the gradient descent way. Extensive experiments on various datasets (EMNIST, Fashion_MNIST, CIFAR-10) are conducted under different settings (heterogeneous models and data distributions). It is demonstrated that the proposed framework is the first federated learning paradigm that realizes personalized model training via parameterized group knowledge transfer while achieving significant performance gain comparing with state-of-the-art algorithms.

1 Introduction

Federated Learning (FL) [1] has emerged as an efficient paradigm to collaboratively train a shared machine learning model among multiple clients without directly accessing their private data. By periodically aggregating parameters from the clients for global model updating, it can converge to high accuracy and strong generalization. FL has shown its capability to protect user privacy while there remains a crucial challenge that significantly degrades the learning performance, i.e., statistic heterogeneity in users' local datasets. Given the Non-Independent and Identically Distributed (Non-IID) user data, the trained global model often cannot be generalized well over each client [2–5].

To deal with the above issues, employing personalized models appears to be an effective solution in FL, i.e., personalized federated learning (pFL). Recent works regarding pFL include regularization-based methods [6–8] (pFedMe [6], L2SGD [7], FedAMP [8]), meta-learning-based Per-FedAvg [9] and cluster-based IFCA [10, 11]. However, in order to aggregate the parameters from all clients, it is inevitable for them to have identical model structure and size. Such constraints would prevent status

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

quo pFL methods from further application in practical scenarios, where clients are often willing to own unique models, i.e., with customized neural architectures to adapt to heterogeneous capacities in computation, communication and storage space, etc.

Motivated by the paradigm of Knowledge Distillation (KD) [12–16] that knowledge can be transferred from a neural network to another via exchanging soft predictions instead of using the whole model parameters, we seek to develop a novel training framework that can accommodate heterogeneous model structures for each client and achieve personalized knowledge transfer in each FL training round. To this end, we formulate the aggregation phase in FL to a personalized group knowledge transfer training algorithm dubbed KT-pFL, whose main idea is to allow each client to maintain a personalized soft prediction at the server that can be updated by a linear combination of all clients' local soft predictions using a knowledge coefficient matrix. The principle of doing so is to reinforce the collaboration between clients with similar data distributions. Furthermore, to quantify the contribution of each client to other's personalized soft prediction, we parameterize the knowledge coefficient matrix so that it can be trained simultaneously with the models following an alternating way in each iteration round.

We show that KT-pFL not only breaks down the barriers of homogeneous model restriction, which requires to transfer the entire parameters set in each round, whose data volume is much larger than that of the soft prediction, but also improves the training efficiency by using a parameterized update mechanism. Experimental results on different datasets and models show that our method can significantly improve the training efficiency and reduce the communication overhead. Our contributions are:

- To the best of our knowledge, this paper is the first to study the personalized knowledge transfer in FL. We propose a novel training framework, namely, KT-pFL, that maintains a personalized soft prediction for each client in the server to transfer knowledge among all clients.
- To encourage clients with similar data distribution to collaborate with each other during the training process, we propose the 'knowledge coefficient matrix' to identify the contribution from one client to others' local training. To show the efficiency of the parameterized method, we compared KT-pFL with two non-parameterized learning methods, i.e., TopK-pFL and Sim-pFL, which calculate the knowledge coefficient matrix on the cosine similarity between different model parameters.
- We provide theoretical performance guarantee for KT-pFL and conduct extensive experiments over various deep learning models and datasets. The efficiency superiority of KT-pFL is demonstrated by comparing our proposed training framework with traditional pFL methods.

2 Related Work

Personalized Federated Learning. Recently, various approaches have been proposed to realize personalized FL, which can be categorized into three types according to the number of global models applied in the server, i.e., single global model, multiple global models, no global model.

Single-global-model-based methods extended from the conventional FL algorithms like FedAvg [1] combine the optimization of the local models and global model, which consist of four different kinds of approaches: local fine-tuning [17–19], regularization [6, 7, 20], mixture [11, 21] and meta learning [9, 22]. The straightforward approach for personalization is the local fine-tuning, where all clients collaboratively train a global model at beginning and then customize the well-trained global model using local data with extra gradient descent steps. Another pFL way is to constrain the deviation between local model and global model by adding a regularized term in the objective function (e.g., pFedMe [6], L2SGD [7, 20]). Instead of adding regularized loss function, Mansour et al. [11] introduce a general approach for FL by mixing the global and local models, and three different personalization methods are proposed with generalization guarantees, namely, client clustering, data interpolation, and model interpolation. The third approach is also used in [21] for the adaptive personalized federated learning (ApFL) algorithm, which aims to find the optimal combination of the global model and the local models. Besides, meta-learning can be used in FL for better personalization by generating a highly-adaptable global model that can help to boost the model training with a few data samples. Inspired by Model-Agnostic Meta-Learning (MAML) [23], Fallah et al. [9] propose to learn personalized models for each client with convergence guarantees (Per-FedAvg).

All of the above mentioned pFL methods apply a single global model, and only allow limited customization to the model at the client side. Therefore, some researchers [8, 10, 11] propose to train multiple global models at the server, where similar clients are clustered into several groups and different models are trained for each group. Mansour et al. [11] provide generalization guarantees for multiple global models, while Ghosh [10] take a further step to establish the convergence theory regarding the training loss function. FedAMP [8] can be regraded as a special case of the clustered-based method that each client owns a personalized global model in the server side.

As a contrast, some research proposals haven't applied any global model to deal with the heterogeneity problem. Taking an example in [24], the authors use multi-task learning framework to design a new training method called MOCHA, to address the statistical and systematic heterogeneity challenges. However, multi-task learning based methods need to solve the data-local quadratic subproblem for every client, which is computationally intensive. It requires all clients to participate in every training round, which is inapplicable for large-scale clients.

Knowledge Distillation. Knowledge Distillation (KD) [25] is introduced to enhance the performance of a small student network by transferring the knowledge from a large teacher network. The principle is to update the student model to approximate the soft prediction output of the teacher model [26–30]. As KD is independent with model structure, some literature [31–33] are proposed to take advantage of such flexibility to implement collaborative learning via KD, i.e., distilling the knowledge of an ensemble of teacher models to a student model [32]. Most of these schemes construct an ensembling teacher by simply averaging the teachers' soft predictions, or by heuristically combining the output of the teacher models, which are hard to achieve optimal combination of teachers. In our framework, KD is used in a more efficient way that the weights of the clients' soft predictions are updated together with the model parameters during every FL training iterations.

3 Problem Formulation

We aim to collaboratively train personalized models for a set of clients applying different model structures in FL. Consider supervised learning whose goal is to learn a function that maps every input data to the correct class out of $\mathcal{C}$ possible options. We assume that there are N clients, and each client n can only access to his private dataset $\mathbb{D}_n := \{x_i^n, y_i\}$, where x_i is the i-th input data sample, y_i is the corresponding label of x_i, $y_i \in \{1, 2, \cdots, C\}$. The number of data samples in dataset $\mathbb{D}_n$ is denoted by D_n. $\mathbb{D} = \{\mathbb{D}_1, \mathbb{D}_2, \cdots, \mathbb{D}_N\}$, $N = \sum_{n=1}^{N} D_n$. In conventional FL, the goal of the learning system is to learn a global model $\mathbf{w}$ that minimizes the total empirical loss over the entire dataset $\mathbb{D}$:

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) := \sum_{n=1}^{N} \frac{D_n}{D} \mathcal{L}_n(\mathbf{w}), \text{ where } \mathcal{L}_n(\mathbf{w}) = \frac{1}{D_n} \sum_{i=1}^{D_n} \mathcal{L}_{CE}(\mathbf{w}; x_i, y_i), \tag{1}$$

where $\mathcal{L}_n(\mathbf{w})$ is the n-th client's local loss function that measures the local empirical risk over the private dataset $\mathbb{D}_n$ and $\mathcal{L}_{CE}$ is the cross-entropy loss function that measures the difference between the predicted values and the ground truth labels.

However, this formulation requires all local clients to have a unified model structure, which cannot be extended to more general cases where each client applies a unique model. Therefore, we need to reformulate the above optimization problem to breaks the barrier of homogeneous model structure. Besides, given the Non-IID clients' datasets, it is inappropriate to minimize the total empirical loss. To that end, we propose the following training framework:

Definition 1. Let $s(\mathbf{w}^n, \hat{x})$ denote the *collaborative knowledge* from client n, and $\hat{x}$ denote a data sample from a public dataset $\mathbb{D}_r$ that all clients can access to. Define the personalized loss function of client n as

$$\mathcal{L}_{per,n}(\mathbf{w}^n) := \mathcal{L}_n(\mathbf{w}^n) + \lambda \sum_{\hat{x} \in \mathbb{D}_r} \mathcal{L}_{KL}\left(\sum_{m=1}^{N} c_{mn} \cdot s(\mathbf{w}^m, \hat{x}), s(\mathbf{w}^n, \hat{x})\right), \tag{2}$$

where $\lambda > 0$ is a hyper-parameters, $\mathcal{L}_{KL}$ stands for Kullback–Leibler (KL) Divergence function and is added to the loss function to transfer personalized knowledge from a teacher to another. c_{mn} is the knowledge coefficient which is used to estimate the contribution from client m to n. The second term in (2) allows each client to build his own personalized aggregated knowledge in the server and enhance the collaboration effect between clients with large c. The concept of *collaborative knowledge* can either refer to soft predictions or model parameters depending on the definition in

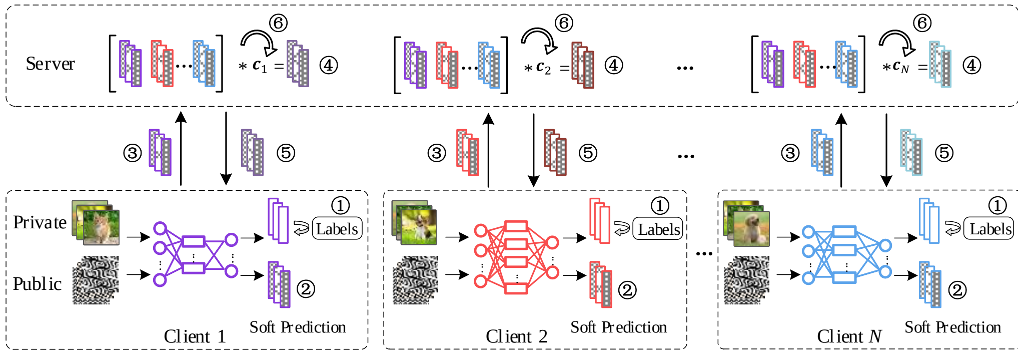


Figure 1: Illustration of the KT-pFL framework. The workflow includes 6 steps: ① local training on private data; ②, ③ each client outputs the local soft prediction on public data and sends it to the server; ④ the server calculates each client's personalized soft prediction via a linear combination of local soft predictions and knowledge coefficient matrix; ⑤ each client downloads the personalized soft prediction to perform distillation phase; ⑥ the server updates the knowledge coefficient matrix.

different situations. For example, $s(\mathbf{w}^n, \hat{x})$ can be deemed to be a soft prediction of the client n, which are calculated with the softmax of logits z^n, i.e., $s(\mathbf{w}^n, \hat{x}) = \frac{\exp(z_c^n/T)}{\sum_{c=1}^{C}\exp(z_c^n/T)}$, logits z^n is the output of the last fully connected layer on client n's model, T is the temperature hyperparameter of the softmax function.

Definition 2. We define $\mathbf{c} \in \mathbb{R}^{N \times N}$ as the *knowledge coefficient matrix*:

$$\mathbf{c} = \begin{Bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \cdots & c_{NN} \end{Bmatrix}. \tag{3}$$

Our objective is to minimize

$$\min_{\mathbf{w},\mathbf{c}} \mathcal{L}(\mathbf{w}, \mathbf{c}) := \sum_{n=1}^{N} \frac{D_n}{D} \mathcal{L}_{per,n}(\mathbf{w}^n) + \rho \|\mathbf{c} - \frac{\mathbf{1}}{N}\|^2, \tag{4}$$

where $\mathbf{w} = [\mathbf{w}^1, \cdots, \mathbf{w}^N] \in \mathbb{R}^{\sum_{n=1}^{N} d_n}$ is the concatenated vector of all weights, d_n represents the dimensions of model parameter $\mathbf{w}^n$. $\mathbf{1} \in \mathbb{R}^{n^2}$ is the identity matrix whose elements are all equal to 1. The second term in (4) is a regularization term that ensures generalization ability of the whole learning system. ρ is a regularization parameter that larger than 0.

4 KT-pFL Algorithm

In this section, we introduce the proposed KT-pFL algorithm, where the local model parameters and knowledge coefficient matrix are updated alternatively. To enable personalized knowledge transfer in FL, we train personalized models locally according to the related *collaborative knowledge*. Insert (2) into (4), and we can design an alternating optimization approach to solve (4), that in each round we fix either $\mathbf{w}$ or $\mathbf{c}$ by turns, and optimize the unfixed one following an alternating way until a convergence point is reached.

Update $\mathbf{w}$: In each communication round, we first fix $\mathbf{c}$ and optimize (train) $\mathbf{w}$ for several epochs locally. In this case, updating $\mathbf{w}$ depends on both the private data (i.e., $\mathcal{L}_{CE}$ on $\mathbb{D}_n, n \in [1, \cdots, N]$), that can only be accessed by the corresponding client, and the public data (i.e., $\mathcal{L}_{KL}$ on $\mathbb{D}_r$), which is accessible for all clients. We propose a two-stage updating framework for $\mathbf{w}$:

- *Local Training*: Train $\mathbf{w}$ on each client's private data by applying a gradient descent step:

$$\mathbf{w}^n \leftarrow \mathbf{w}^n - \eta_1 \nabla_{\mathbf{w}^n} \mathcal{L}_n(\mathbf{w}^n; \xi_n), \tag{5}$$

  where ξ_n denotes the mini-batch of data $\mathbb{D}_n$ used in local training, η_1 is the learning rate.

Algorithm 1 KT-pFL Algorithm

Input: Private dataset $\mathbb{D}_n$, public dataset $\mathbb{D}_r$, personalized model $\mathbf{w}^n$, $n = 1, \cdots, N$, regularization parameter ρ and learning rate η_1, η_2, η_3. Total communication rounds T.
Output: Trained personalized models $\mathbf{w} = [\mathbf{w}^1, \cdots, \mathbf{w}^N]$.

Initialize the model parameters $\mathbf{w}_0$ and knowledge coefficient matrix $\mathbf{c}_0$.
procedure SERVER-SIDE OPTIMIZATION
    Distribute $\mathbf{w}_0$ and $\mathbf{c}_0$ to each client
    for each communication round $t \in \{1, 2, ..., T\}$ **do**
        for each client n **in parallel do**
            $\mathbf{w}^n_{t+1} \leftarrow ClientLocalUpdate(n, \mathbf{w}^n_t, \mathbf{c}_{t,n})$
        Update knowledge coefficient matrix $\mathbf{c}$ via (7)
        Distribute $\mathbf{c}_{t+1}$ to all clients
procedure CLIENTLOCALUPDATE$(n, \mathbf{w}^n_t, \mathbf{c}_{t,n})$
    Client n receives $\mathbf{w}^n_t$ and $\mathbf{c}_n$ from the server
    for each local epoch i from 1 to E **do**
        for mini-batch $\xi_t \subseteq \mathbb{D}_n$ **do**
            Local Training: update model parameters on private data via (5)
    for each distillation step j from 1 to R **do**
        for mini-batch $\xi_{r,t} \subseteq \mathbb{D}_r$ **do**
            Distillation: update model parameters on public data via (6)
    return local parameters $\mathbf{w}^n_{t+1}$

- *Distillation*: Transfer knowledge from personalized soft prediction to each local client based on public dataset:

$$\mathbf{w}^n \leftarrow \mathbf{w}^n - \eta_2 \nabla_{\mathbf{w}^n} \mathcal{L}_{KL} \left(\sum_{m=1}^{N} \mathbf{c}_m^{*,T} \cdot s(\mathbf{w}^m, \xi_r), s(\mathbf{w}^n, \xi_r) \right), \tag{6}$$

  where ξ_r denotes the mini-batch of public data $\mathbb{D}_r$, and η_2 is the learning rate. $\mathbf{c}_m^* = [c_{m1}, c_{m2}, \cdots, c_{mN}]$ is the *knowledge coefficient* vector for client m, which can be found in m-th row of $\mathbf{c}$. Note that all *collaborative knowledge* and *knowledge coefficient matrix* are required to obtain the personalized soft prediction in this stage, which can be collected in the server.

Update c: After updating $\mathbf{w}$ locally for several epochs, we turn to fix $\mathbf{w}$ and update $\mathbf{c}$ in the server.

$$\mathbf{c} \leftarrow \mathbf{c} - \eta_3 \lambda \sum_{n=1}^{N} \frac{D_n}{D} \nabla_{\mathbf{c}} \mathcal{L}_{KL} \left(\sum_{m=1}^{N} \mathbf{c}_m \cdot s(\mathbf{w}^{m,*}, \xi_r), s(\mathbf{w}^{n,*}, \xi_r) \right) - 2\eta_3 \rho (\mathbf{c} - \frac{\mathbf{1}}{N}), \tag{7}$$

where η_3 is the learning rate for updating $\mathbf{c}$.

Algorithm 1 demonstrates the proposed KT-pFL algorithm and the idea behind it is shown in Figure 1. In every communication round of training, the clients use local SGD to train several epochs based on the private data and then send the *collaborative knowledge* (e.g., soft predictions on public data) to the server. When the server receives the *collaborative knowledge* from each client, it aggregates them to form the personalized soft predictions according to the knowledge coefficient matrix. The server then sends back the personalized soft prediction to each client to perform local distillation. The clients then iterate for multiple steps over public dataset [1]. After that, the knowledge coefficient matrix is updated in the server while fixing the model parameters $\mathbf{w}$.

Performance Guarantee. Theorem 1 provides the performance analysis of the personalized model when each client owns Non-IID data. Detailed description and derivations are deferred to Appendix.

Theorem 1. *Denote the n-th local distribution and its empirical distribution by $\mathcal{D}_n$ and $\hat{\mathcal{D}}_n$ respectively, and the hypothesis $h \in \mathcal{H}$ trained on $\hat{\mathcal{D}}_n$ by $h_{\hat{\mathcal{D}}_n}$. There always exist $c^*_{m,n}, m = 1, \ldots, N$, such that the expected loss of the personalized ensemble model for the data distribution $\mathcal{D}_n$ of client n is not larger than that of the single model only trained with local data: $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^N c^*_{m,n} h_{\hat{\mathcal{D}}_m}) \leq \mathcal{L}_{\mathcal{D}_n}(h_{\hat{\mathcal{D}}_n})$. Besides, there exist some problems where the personalized ensemble model is strictly better, i.e., $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^N c^*_{m,n} h_{\hat{\mathcal{D}}_m}) < \mathcal{L}_{\mathcal{D}_n}(h_{\hat{\mathcal{D}}_n})$.*

[1] Note that our method can work over both labeled and unlabeled public datasets.

Theorem 1 indicates that the performance of the personalized ensemble model under some suitable coefficient matrices is better than that of the model only trained on its local private data, which theoretically demonstrates the necessity of the parameterized personalization. However, it is challenging to find such matrices due to the complexity and diversity of machine learning models and data distributions. In this paper, our designed algorithm KT-pFL can find the desired coefficient matrix in a gradient descent manner, hence achieving the performance boost. Besides, we have the similar claim for the relationship between the personalized ensemble model and average ensemble model.

Remark 1. *There always exist* $c^*_{m,n}, m = 1, \ldots, N$*, such that the expected loss of the personalized ensemble model for the data distribution* $\mathcal{D}_n$ *of client* n *is not larger than that of the average ensemble model:* $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^{N} c^*_{m,n} h_{\hat{\mathcal{D}}_m}) \le \mathcal{L}_{\mathcal{D}_n}(\frac{1}{N}\sum_{m=1}^{N} h_{\hat{\mathcal{D}}_m})$*. Besides, there exist some problems where the personalized ensemble model is strictly better, i.e.,* $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^{N} c^*_{m,n} h_{\hat{\mathcal{D}}_m}) < \mathcal{L}_{\mathcal{D}_n}(\frac{1}{N}\sum_{m=1}^{N} h_{\hat{\mathcal{D}}_m})$.

5 Evaluations

5.1 Experimental Setup

Task and Datasets We evaluate our proposed training framework on three different image classification tasks: EMNIST [34], Fashion_MNIST [35] and CIFAR-10 [36]. For each dataset, we apply two different Non-IID data settings: 1) each client only contains two classes of samples; 2) each client contains all classes of samples, while the number of samples for each class is different from that of a different client. All datasets are split randomly with 75% and 25% for training and testing, respectively. The testing data on each client has the same distribution with its training data. For all methods, we record the average test accuracy of all local models for evaluation.

Model Structure: Four different lightweight model structures including LeNet [37], AlexNet [38], ResNet-18 [39], and ShuffleNetV2 [40] are adopted in our experiments. Our pFL system has 20 clients, who are assigned with four different model structures, i.e., five clients per model.

Baselines: Although KT-pFL is designed for effective personalized federated learning, it can be applied to both heterogeneous and homogeneous model cases. We first conduct experiments on heterogeneous systems where the neural architecture of the local models are different among clients. We compare the performance of KT-pFL to the non-personalized distillation-based method named FedDF [16][2] and other simple versions of KT-pFL including Sim-pFL and TopK-pFL. Sim-pFL calculates the knowledge coefficient by using cosine similarity between two local soft predictions. Instead of combining the knowledge from all clients, TopK-pFL obtains the personalized soft predictions only from K clients [3] who have higher value of cosine similarity.

To demonstrate the generalization and effectiveness of our proposed training framework, we further compare KT-pFL to FedAvg [1] and state-of-the-art pFL methods including Per-Fedavg [9], pFedMe [6] and FedAMP [8] under the homogeneous model setting. Note that Per-FedAvg is a MAML-based method which aims to optimize the one-step gradient update for its personalized model. pFedMe and FedAMP are two regularized-based methods. The former one obtains personalized models by controlling the distance between local model and global model, while the later one facilitates the collaboration of clients with similar data distribution.

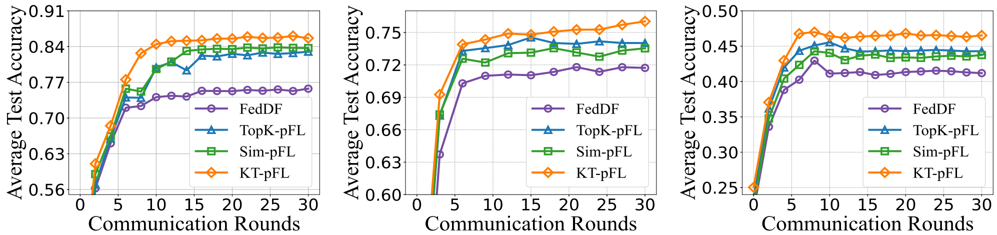
Implementation The experiments are implemented in PyTorch. We simulate a set of clients and a centralized server on one deep learning workstation (i.e., Intel(R) Core(TM) i9-9900KF CPU@3.6GHz with one NVIDIA GeForce RTX 2080Ti GPU).

5.2 Results

Subject to space constraints, we only report the most important experimental results in this section. Please refer to Appendix for the details of different Non-IID data settings on EMNIST, Fash-

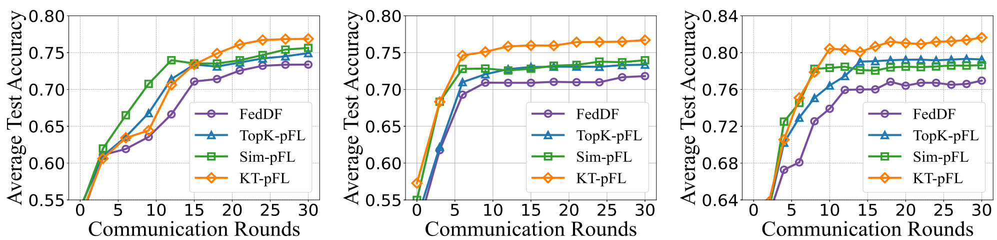
[2]FedDF exchanges model parameters with the server and performs knowledge transfer after averaging the local model parameters at the server side. In our heterogeneous setting, the local clients exchange soft predictions with the server, so we omit the parameters aggregation phase of FedDF in our experiments.

[3]In our experiments, we set K as 5.



(a) EMNIST (b) Fashion_MNIST (c) CIFAR-10

Figure 2: Performance comparison of FedDF, TopK-pFD, Sim-pFD, and KT-pFL in average test accuracy on three datasets (Non-IID case 1: each client contains all labels).



(a) EMNIST (b) Fashion_MNIST (c) CIFAR-10

Figure 3: Performance comparison of FedDF, TopK-pFD, Sim-pFD, and KT-pFL in average test accuracy on three datasets (Non-IID case 2: each client contains only two labels).

ion_MNIST and CIFAR10, the implementation details and the hyperparameter settings of all the methods, and also extra results about the convergence and robustness analysis.

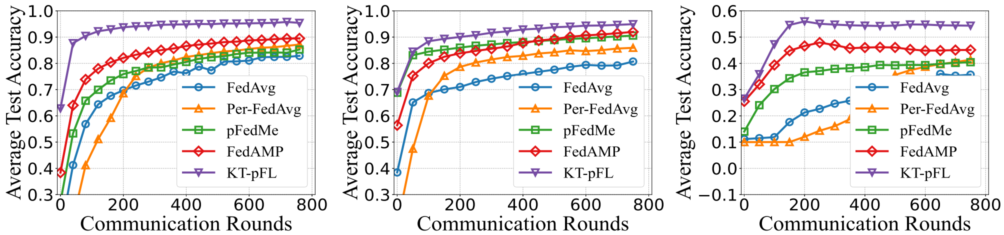
5.2.1 Performance Comparison

Heterogeneous FL. For all the methods and all the data settings, the batch size on private data and public data are 128 and 256, respectively, the number of local epochs is 20 and the distillation steps is 1 in each communication round of pFL training. Unless mentioned, otherwise the size of public data used in each communication round is 3000, the learning rate is set to 0.01 for EMNIST and Fashion_MNIST, and 0.02 for CIFAR-10.

Table 1: The comparison of final test accuracy on different datasets with heterogeneous models (i.e., Lenet, AlexNet, ResNet-18 and ShuffleNetV2).

Method	EMNIST (%)		FashionMNIST (%)		CIFAR-10 (%)	
	Non-IID_1	Non-IID_2	Non-IID_1	Non-IID_2	Non-IID_1	Non-IID_2
FedDF [16]	72.75	75.41	71.89	72.76	42.03	39.50
Sim-pFD	82.72	74.42	72.82	72.71	42.50	39.90
TopK-pFD	83.00	73.91	72.24	73.39	42.58	39.34
KT-pFL	**84.65**	**76.30**	**75.48**	**75.60**	**43.82**	**41.04**

Figure 2 and 3 show the curves of the average test accuracy during the training process on four different models with three datasets, which include the results of FedDF, Sim-pFL, TopK-pFL and KT-pFL. We also summarize the final average test accuracy in Table 1. In two cases of Non-IID settings, KT-pFL obtains comparable or even better accuracy performance than others. The performance of FedDF is worse than the other three methods. The reason is that taking the global aggregation of all local soft predictions trained on the Non-IID data from different clients produces only one global soft prediction, which cannot be well adapted to each client. The other two personalized methods, i.e., Sim-pFL and TopK-pFL, achieve comparably performance on most of cases with FedDF. Our proposed method KT-pFL has the best performance, because each client can adaptively aggregate all local soft predictions to form a personalized one instead of being restricted to a global soft prediction.



(a) EMNIST (b) Fashion_MNIST (c) CIFAR_10

Figure 4: Performance comparison of FedAvg, Per-Fedavg, pFedMe, FedAMP and KT-pFL in average test accuracy on three datasets. The Non-IID data setting: each client contains all labels. 20 clients with homogeneous models: CNN [1]. Learning rate: 0.005.

Table 2: The comparison of final test accuracy on different datasets with homogeneous models (i.e., the same CNN architecture as [1]).

Method	EMNIST (%)	Fashion_MNIST (%)	CIFAR-10 (%)
Fedavg [1]	85.86	80.69	35.69
Per-Fedavg [9]	86.16	85.91	41.37
pFedMe [6]	84.26	90.97	40.44
FedAMP [8]	88.51	90.79	40.11
KT-pFL	**94.40**	**93.93**	**52.35**

Homogeneous FL. Apart from heterogeneous system, we further extend KT-pFL to homogeneous model setting. To conduct fair comparison with baselines, we exchange model parameters with the server to perform knowledge transfer. Specifically, each client maintains a personalized global model at the server side and each personalized global model is aggregated by a linear combination of local model parameters and knowledge coefficient matrix. In this case, we compare the performance of homogeneous-version of KT-pFL with FedAvg, Per-Fedavg, pFedMe and FedAMP. Figure 4 and Table 2 show the curve of the average test accuracy during training and the final average test accuracy on three different datasets, respectively. For all datasets, KT-pFL dominates the other four methods on average test accuracy with less variance. Besides, the concept of FedAMP is similar to our proposed method KT-pFL. The difference is that the weights for each local model during personalized aggregation phase are updated in a gradient descent manner in KT-pFL.

5.2.2 Effect of hyperparameters

To understand how different hyperparameters such as E, R, ρ, and $|\xi_r|$ can affect the training performance of KT-pFL in different settings, we conduct various experiments on three dataset with $\eta_1 = \eta_2 = \eta_3 = 0.01$.

Effect of Local Epochs E**.** To reduce communication overhead, the server tends to allow clients to have more local computation steps, which can lead to less global updates and thus faster convergence. Therefore, we monitor the behavior of KT-pFL using a number of values of E, whose results are given in Table 3. The results show that larger values of E can benefit the convergence of the personalized models. There is, nevertheless, a trade-off between the computations and communications, i.e., while larger E requires more computations at local clients, smaller E needs more global communication rounds to converge. To do such trade-off, we fix $E = 20$ and evaluate the effect of other hyperparameters accordingly.

Effect of Distillation Steps R**.** As the performance of the knowledge transfer is directly related to the number of distillation steps R, we compare the performance of KT-pFL under different setting of distillation steps (e.g., R = 1, 2, 3, 5). The number of local epochs is set to 20. The results show that larger value of R cannot always lead to better performance, which means a moderate number of the distillation steps is necessary to approach to the optimal performance.

Effect of ρ**.** As mentioned in (4), ρ is the regularization term to do the trade-off between personalization ability and generalization ability. For example, larger value of ρ means that the knowledge

Table 3: The comparison of final test accuracy on different datasets with homogeneous models.

Dataset	# Local epochs (E)				# Distillation steps (R)			
	5	10	15	20	1	2	3	5
EMNIST (%)	90.15	**92.76**	91.03	90.15	**91.76**	91.40	91.18	91.54
Fashion_MNIST (%)	88.73	89.14	88.58	**89.42**	89.14	**89.90**	89.07	88.08
CIFAR-10 (%)	58.30	**59.38**	59.34	59.24	**59.24**	57.99	59.22	58.91

Table 4: The comparison of final test accuracy on different datasets with homogeneous models

Dataset	# Regularization parameter ρ				# Batch size of public data ($\|\xi_r\|$)			
	0.1	0.3	0.5	0.7	500	1000	3000	5000
EMNIST (%)	90.96	90.66	90.88	**91.76**	91.25	91.47	**91.66**	91.32
Fashion_MNIST (%)	89.49	89.30	**88.56**	89.14	88.08	89.40	**89.50**	89.14
CIFAR-10 (%)	59.08	58.52	**59.56**	58.40	**58.93**	58.79	58.91	58.40

coefficient of each local soft prediction should approach to $\frac{1}{N}$, and thus more generalization ability should be guaranteed. Table 4 shows the results of KT-pFL with different value of ρ. In most settings, a significantly large value of ρ will hurt the performance of KT-pFL.

Effect of $|\xi_r|$. Table 4 demonstrates the effect of batch size of public data $|\xi_r|$ used in local distillation phase. When the size of the mini-batch is increased, KT-pFL has higher average test accuracy. However, very large $|\xi_r|$ will not only slow down the convergence of KT-pFL but also incur higher computation time at the clients. During the experiments, the value of $|\xi_r|$ is configured to 3000.

5.2.3 Efficiency Evaluation

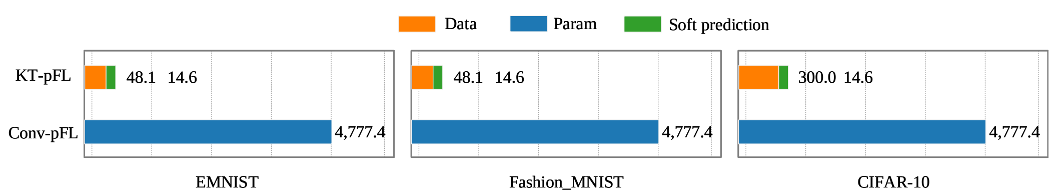


Figure 5: Communication Efficiency (X-axis units: MBytes) on three datasets. KL-pFL: soft prediction-based personalized federated learning; Conv-pFL: conventional parameter-based personalized federated learning. (20 models, 30 communication rounds for all datasets. In this experiment, the public dataset is stored on the server.)

To evaluate the communication overhead, we record three aspects information: Data (batch size used in distillation phase), Param (model parameters) and Soft Prediction without using data compression techniques. The results are shown in Figure 5. Compared with conventional parameter-based pFL methods, the communication overhead on soft prediction-based KT-pFL is far less than Conv-pFL.

Broader Impact

FL has been emerged as new paradigm to collaboratively train models among multiple clients in a privacy-preserving manner. Due to the diversity of users (e.g., statistical and systematic heterogeneity, etc.), applying personalization in FL is essential for future trend. Our method KT-pFL not only breaks the barriers of homogeneous model constraint, which can significantly reduce the communication overhead during training, but also improves the training efficiency via a parameterized update mechanism without additional computation overhead at the client side. This research has the potential to enable various devices to cooperatively train ML tasks based on customized neural network architectures.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2017.

[2] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems, MLSys*, 2020.

[3] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *Proceedings of International Conference on Machine Learning, ICML*, pages 4615–4625, 2019.

[4] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of International Conference on Machine Learning, ICML*, pages 5132–5143, 2020.

[5] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. Fair resource allocation in federated learning. In *Proceedings of 8th International Conference on Learning Representations, ICLR*, 2020.

[6] Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

[7] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020.

[8] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2021.

[9] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

[10] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

[11] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

[12] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[13] Daliang Li and Junpu Wang. FedMD: Heterogenous federated learning via model distillation. *arXiv*, oct 2019.

[14] Hongyan Chang, Virat Shejwalkar, Reza Shokri, and Amir Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

[15] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *arXiv preprint arXiv:2008.06180*, 2020.

[16] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

[17] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019.

[18] Johannes Schneider and Michail Vlachos. Personalization of deep learning. *arXiv preprint arXiv:1909.02803*, 2019.

[19] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019.

[20] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

[21] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[22] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[23] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML*, volume 70, pages 1126–1135, 2017.

[24] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. Federated multi-task learning. In *Proceedings of Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2017.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[26] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 4320–4328, 2018.

[27] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of 2019 IEEE/CVF International Conference on Computer Vision, ICCV*, pages 4794–4802, 2019.

[28] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 10687–10698, 2020.

[29] Chenglin Yang, Lingxi Xie, Siyuan Qiao, and Alan L Yuille. Training deep neural networks in generations: A more tolerant teacher educates better students. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, volume 33, pages 5628–5635, 2019.

[30] Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1355–1364, 2019.

[31] Guile Wu and Shaogang Gong. Peer collaborative learning for online knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, 2021.

[32] Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. Online knowledge distillation via collaborative learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 11020–11029, 2020.

[33] Ilai Bistritz, Ariana Mann, and Nicholas Bambos. Distributed distillation for on-device learning. In *Proceedings of Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020.

[34] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: extending MNIST to handwritten letters. In *Proceedings of 2017 International Joint Conference on Neural Networks, IJCNN*, 2017.

[35] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[36] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[37] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 1106–1114, 2012.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 770–778, 2016.

[40] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision, ECCV*, pages 116–131, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [N/A]
   (c) Did you discuss any potential negative societal impacts of your work? [No] We believe our work has the potential to be implemented in practical scenarios.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]