Dealing with imbalanced sources in heterogenous client distributions in federated learning

Arman Beikmohammadi[1], **Erfan Darzidehkalani**[2], **Mohammad Raziei**[3], **Amir Rezaei**[3,*]
[1] Department of Electrical and Computer Engineering,University of Florida, Gainseville, USA
[2]Data science center in health, University of Groningen, The Netherlands
[3]Department of Electrical Engineering, Amirkabir University of Technology, Tehran, Iran
*amirrezaei@aut.ac.ir

Abstract

Federated learning (FL) can let multiple clients collaboratively train a global model while keeping the raw data locally. Classical FL paradigm requires all clients to share an identical model structure, and thus cannot produce client side models to deal with the user heterogeneity, e.g., in data distribution. Recent literature have proposed many works to achieve client side FL, e.g., regularization-based pFedMe [1], FedAMP [2], meta-learning-based Per-FedAvg [3] and cluster-based IFCA [4]. However, all of them rely on model parameters aggregation at server side that employ same model structure and size. Solutions for collaboratively training client side models for multiple clients are missing from status quo, or difficult for deployment in practical scenarios. Inspired by the concept of knowledge distillation (KD) and its capability of transferring knowledge from a neural network to another via exchanging the logits output, he standard formulation of FL produces one shared model for all clients which ignores the personalization of each client, especially given the heterogeneity of data distribution for various clients.However, this architecture-agnostic training framework only develops a global output(/logits) for all the clients, and therefore, it does not adapt the personalization to each client during aggregation, especially for clients with heterogeneous data distributions.Although there are already plenty of work in efforts to achieve personlization in FL (e.g., regularization-based method), most of them rely on the model parameters in the server, which is impractical in FD. This paper is an attempt to deal with the aformentioned issue with an effectice collaortive algorithm focusing on client side of the network.

1 Introduction

Federated Learning (FL) [5] 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 [6–9].

To deal with the above issues, employing client side models appears to be an effective solution in FL, i.e., CSFL (pFL). Recent works regarding pFL include regularization-based methods [1, 10, 2] (i.e., pFedMe [1], L2SGD [10], FedAMP [2]), meta-learning-based Per-FedAvg [3] and cluster-based IFCA [4, 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 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, KD-based FL training methods have been studied [12, 14–19] to collaboratively train heterogeneous models in a privacy-preserving way. However, these works have neglected the further personalization requirement of FL clients, which can be well satisfied via the client side knowledge transfer for heterogeneous FL users.

In this paper, we seek to develop a novel training framework that can accommodate heterogeneous model structures for each client and achieve client side knowledge transfer in each FL training round. To this end, we formulate the aggregation phase in FL to a client side group knowledge transfer training algorithm dubbed MNFL, whose main idea is to allow each client to maintain a client side 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 client side 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 MNFL 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 client side knowledge transfer in FL. We propose a novel training framework, namely, MNFL, that maintains a client side 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 MNFL 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 MNFL and conduct extensive experiments over various deep learning models and datasets. The efficiency superiority of MNFL is demonstrated by comparing our proposed training framework with traditional pFL methods.

2 Related Work

client side FL with Homogeneous Models. Recently, various approaches have been proposed to realize client side FL with homogeneous local model structure, 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 and no global model.

single global model type is a close variety of conventional FL, e.g., FedAvg [5], that combine global model optimization process with additional local model customization, and consist of four different kinds of approaches: local fine-tuning [20–23], regularization (e.g., pFedMe [1], L2SGD [10, 24], Ditto [25]), hybrid local and global models [11, 26, 27] and meta learning [3, 28]. All of these pFL methods apply a single global model, and thus limit the customized level of the local model at the client side. Therefore, some researchers [2, 4, 11] propose to train multiple global models at the server, where clients are clustered into several groups according to their similarity and different models are trained for each group. FedAMP [2] and FedFomo [29] can be regarded as special cases of the clustered-based method that each client owns a client side global model at the server side. As a contrast, some literature waive the global model to deal with the heterogeneity problem [30, 31], such as multi-task learning based (i.e., MOCHA [31]) and hypernetwork-based framework (i.e., FedHN [30]). However, all these methods require aggregating the model parameters from the clients, who have to apply identical model structure and size, which hinders further personalization, e.g., employing client side model architectures for heterogeneous clients is not feasible.

Heterogeneous FL and Knowledge Distillation. To enable heterogeneous model architectures in FL, Diao et al. [32] propose to upload a different subset of global model to the server for aggregation with the objective to produce a single global inference model. Another way of personalization is to use Knowledge Distillation (KD) in heterogeneous FL systems ([12, 14, 15, 17–19, 33, 34]). The principle is to aggregate local soft-predictions instead of local model parameters in the server, whereby each client can update the local model to approach the averaged global predictions. As KD is independent with model structure, some literature [13, 16] are proposed to take advantage of such independence to implement client side FL with heterogeneous models at client sides. For example, Li et al. [13] propose FedMD to perform ensemble distillation for each client to learn well-client side models. Different from FedMD that exchanging soft-predictions between the clients and the server, FedDF [16] first aggregates local model parameters for model averaging at the server side. Then, the averaged global models can be updated by performing knowledge transfer from all received (heterogeneous) client models.

To sum up, 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 far away from producing 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 iteration.

3 Problem Formulation

We aim to collaboratively train client side 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. $\mathbb{D} = \{\mathbb{D}_1, \mathbb{D}_2, \cdots, \mathbb{D}_N\}$, $D = \sum_{n=1}^{N} D_n$. 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 break the barrier of homogeneous model structure. Besides, given the Non-IID clients' datasets, it is inappropriate to just minimize the total empirical loss (i.e., $\min_{\mathbf{w}^1, \cdots, \mathbf{w}^N} \mathcal{L}(\mathbf{w}^1, \cdots, \mathbf{w}^N) := \sum_{n=1}^{N} \frac{D_n}{D} L_n(\mathbf{w}^n)$). To that end, we propose the following training framework:

Definition 3.1. 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 client side 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 client side 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 client side aggregated knowledge in the server and enhance the collaboration effect between clients with large $\mathbf{c}$. The concept of *collaborative knowledge* can either refer to soft predictions or model parameters depending on the definition in 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 3.2. 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. Without the regularization term, a client with a completely different data distribution tends to set large values of the knowledge coefficient (i.e., equal to 1), and no collaboration will be conducted during training in this case. ρ is a regularization parameter that is larger than 0.

4 MNFL Algorithm

In this section, we introduce the proposed MNFL algorithm, where the local model parameters and knowledge coefficient matrix are updated alternatively. To enable client side knowledge transfer in FL, we train client side 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.

- *Distillation*: Transfer knowledge from client side 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 client side soft prediction in this stage, which can be collected in the server.

Update $\mathbf{c}$: 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 MNFL algorithm and the idea behind it is shown in Figure **??**. 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 client side soft predictions according to the knowledge coefficient matrix. The server then sends back the client side soft prediction to each client to perform local distillation. The clients then

Algorithm 1 MNFL Algorithm

Input: $\mathbb{D}, \mathbb{D}_r, \eta_1, \eta_2, \eta_3$ and T
Output: $\mathbf{w} = [\mathbf{w}^1, \cdots, \mathbf{w}^N]$

Initialize $\mathbf{w}_0$ and $\mathbf{c}_0$
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**
            $\mathbf{w}_{t+1}^n \leftarrow ClientLocalUpdate(n, \mathbf{w}_t^n, \mathbf{c}_{t,n})$
        Update knowledge coefficient matrix $\mathbf{c}$ via (7)
        Distribute $\mathbf{c}_{t+1}$ to all clients
procedure CLIENTLOCALUPDATE($n, \mathbf{w}_t^n, \mathbf{c}_{t,n}$)
    Client n receives $\mathbf{w}_t^n$ 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 $\mathbf{w}_{t+1}^n$

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 4.1 provides the performance analysis of the client side model when each client owns Non-IID data. Detailed description and derivations are deferred to Appendix A.

Theorem 4.1. *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 client side 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 client side ensemble model is strictly better, i.e., $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^N c^*_{m,n} h_{\hat{\mathcal{D}}_m}) < \mathcal{L}_{\mathcal{D}_n}(h_{\hat{\mathcal{D}}_n})$.*

Theorem 4.1 indicates that the performance of the client side 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 MNFL 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 client side ensemble model and average ensemble model.

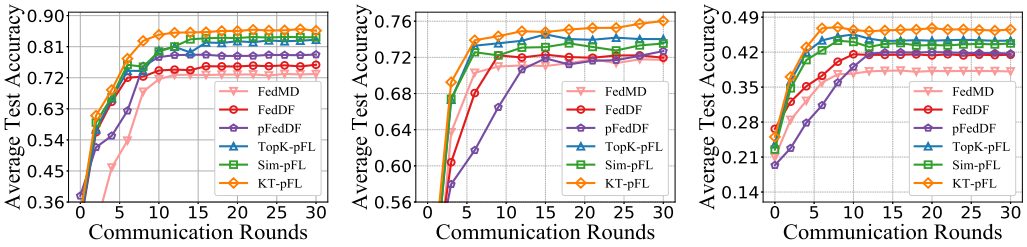
Remark 4.1. *There always exist $c^*_{m,n}, m = 1, \ldots, N$, such that the expected loss of the client side ensemble model for the data distribution $\mathcal{D}_n$ of client n is not larger than that of the average ensemble model: $\mathcal{L}_{\mathcal{D}_n}(\sum_{m=1}^N c^*_{m,n} h_{\hat{\mathcal{D}}_m}) \leq \mathcal{L}_{\mathcal{D}_n}(\frac{1}{N}\sum_{m=1}^N h_{\hat{\mathcal{D}}_m})$. Besides, there exist some problems where the client side 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 [35], Fashion_MNIST [36] and CIFAR-10 [37]. For each dataset, we apply two different Non-IID data settings: 1) each client only contains two classes of samples; 2) each

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



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

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

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 [38], AlexNet [39], ResNet-18 [40], and ShuffleNetV2 [41] 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 MNFL is designed for effective CSFL, 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 MNFL to the non-client side distillation-based methods: FedMD [13], FedDF [16] and the client side distillation-based method pFedDF[2] and other simple versions of MNFL 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 client side 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 MNFL to FedAvg [5] and state-of-the-art pFL methods including Per-Fedavg [3], Fedavg-Local FT[23], pFedMe [1], FedAMP [2], FedFomo[29] and FedHN[30] 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 client side model. pFedMe and FedAMP are two regularized-based methods. The former one obtains client side 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 B for the details of different Non-IID data settings on EMNIST, Fashion_MNIST and CIFAR10, the implementation details and the hyperparameter settings of all the methods, and also extra results about the convergence and robustness analysis. In our experiments, we run each experiment multiple times and record the average results.

5.2.1 Performance Comparison

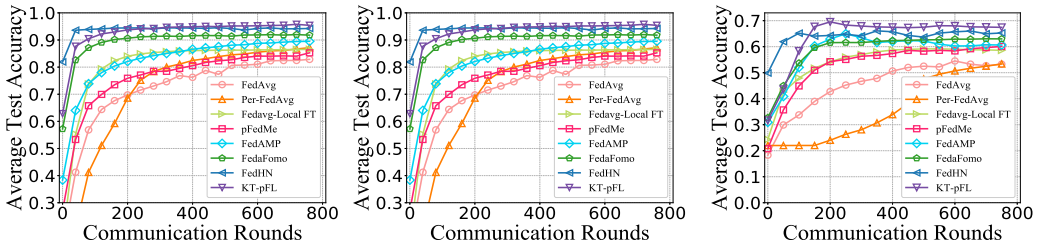
Heterogeneous FL. For all the methods and all the data settings, the batch size on private data and public data are 128 and 256, respectively, the number of local epochs is 20 and the distillation steps is 1 in each communication round of pFL training. Unless mentioned, otherwise the public data used

[2]We use the fused prototype models at the last round of FedDF as the pFedD's initial models. Each client fine-tunes it on the local private dataset with several epochs.

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

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

Method	EMNIST		Fashion_MNIST		CIFAR-10	
	Non-IID_1	Non-IID_2	Non-IID_1	Non-IID_2	Non-IID_1	Non-IID_2
FedMD [13]	71.37	70.15	68.02	66.10	39.14	35.56
FedDF [16]	71.92	72.79	70.32	70.83	40.19	37.81
pFedDF	78.59	75.72	72.69	73.45	42.83	39.54
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
MNFL	**84.65**	**76.30**	**75.48**	**75.60**	**43.82**	**41.04**



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

Figure 2: Performance comparison of FedAvg, Per-Fedavg, Fedavg-Local FT, pFedMe, FedAMP, FedFomo, FedHN and MNFL in average test accuracy on three datasets. The Non-IID data setting: each client contains all labels. 20 clients with homogeneous models: CNN [5]. Learning rate: 0.005.

for EMNIST and Fashion_MNIST is MNIST, and the public data used for CIFAR-10 is CIFAR-100. 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.

Figure 1 and **??** show the curves of the average test accuracy during the training process on four different models with three datasets, which include the results of FedMD, FedDF, pFedDF, Sim-pFL, TopK-pFL and MNFL. We also summarize the final average test accuracy in Table 1. In two cases of Non-IID settings, MNFL obtains comparable or even better accuracy performance than others. The performances of FedMD, FedDF, pFedDF are 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 client side methods, i.e., Sim-pFL and TopK-pFL, achieve comparably performance on most of cases with pFedDF. Our proposed method MNFL has the best performance, because each client can adaptively aggregate all local soft predictions to form a client side one instead of being restricted to a global soft prediction.

Homogeneous FL. Apart from heterogeneous system, we further extend MNFL 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 client side global model at the server side and each client side global model is aggregated by a linear combination of local model parameters and knowledge coefficient matrix (Appendix C). In this case, we compare the performance of homogeneous-version of MNFL with FedAvg, Per-Fedavg, Fedavg-Local FT, pFedMe, FedAMP, FedFomo and FedHN. Figure 2 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, MNFL dominates the other seven methods on average test accuracy with less variance. Besides, the concept of FedAMP and FedFomo is similar with our proposed method MNFL. The difference is that the weights for each local model during client side aggregation phase are updated in a gradient descent manner in MNFL.

To verify the efficiency of MNFL on large-scale FL system, we conduct experiments on 100 clients and apply the same client selection mechanism as other baselines. Specifically, in MNFL, only partial of elements in the knowledge coefficient matrix $\mathbf{c}$ will be updated in each communication round on

Table 2: The comparison of final test accuracy (%) on different datasets with different number of homogeneous models (i.e., the same CNN architecture as [5]). For large-scale FL system with 100 clients, we set the client sampling rate as 0.1.

Method	EMNIST		Fashion_MNIST		CIFAR-10	
	20 clients	100 clients	20 clients	100 clients	20 clients	100 clients
Fedavg [5]	85.86	76.84	80.69	72.36	51.94	43.56
Per-Fedavg [3]	86.16	82.87	85.91	77.93	55.61	49.82
Fedavg-Local FT [23]	86.59	84.55	90.17	80.93	42.88	50.91
pFedMe [1]	84.26	86.14	90.97	84.02	62.79	52.46
FedAMP [2]	88.51	85.63	90.79	84.72	60.75	51.26
FedFomo [29]	92.58	90.08	88.33	87.76	63.11	55.73
pFedHN [30]	94.27	92.21	92.22	89.14	65.31	57.38
MNFL	**94.40**	**92.50**	**93.93**	**90.37**	**66.96**	**58.29**

Table 3: The comparison of final test accuracy on different setting of local epochs E and distillation steps R.

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

large-scale FL systems. From the results in Table 2, it is obvious that our proposed method can work well both on small-scale and large-scale FL systems.

5.2.2 Effect of hyperparameters

To understand how different hyperparameters such as E, R, ρ, and $|\mathbb{D}_r|$ can affect the training performance of MNFL in different settings, we conduct various experiments on three datasets 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 MNFL 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 client side 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 MNFL 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 Eq (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 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 MNFL with different value of ρ. In most settings, a significantly large value of ρ will hurt the performance of MNFL.

Effect of different public dataset. Besides, we compare the performance of our proposed method on different public datasets (Table 5). From the experimental results, we can observe that different settings of the public dataset have little effect on the training performance.

Table 4: The comparison of final test accuracy on different setting of regularization parameter ρ.

Dataset	# Regularization parameter ρ			
	0.1	0.3	0.5	0.7
EMNIST (%)	90.96	90.66	90.88	**91.76**
Fashion_MNIST (%)	89.49	89.30	**89.56**	89.14
CIFAR-10 (%)	59.08	58.52	**59.56**	58.40

Table 5: Performance of MNFL under different setting of public dataset (Task: Fashion_MNIST; Unlabeled Open Dataset: divided from Fashion_MNIST).

Public dataset	Test Accuracy
MNIST	89.14 ($\pm$ 0.15)
EMNIST	88.76 ($\pm$ 0.21)
Unlabeled Open Dataset	89.03 ($\pm$ 0.11)

Effect of $|\mathbb{D}_r|$. Table 6 demonstrates the effect of public data size $|\mathbb{D}_r|$ used in local distillation phase. When the size of the public data is increased, MNFL has higher average test accuracy. However, very large $|\mathbb{D}_r|$ will not only slow down the convergence of MNFL but also incur higher computation time at the clients. During the experiments, the value of $|\mathbb{D}_r|$ is configured to 3000.

5.2.3 Efficiency Evaluation

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 3. Compared with conventional parameter-based pFL methods, the communication overhead on soft prediction-based MNFL is far less than Conv-pFL.

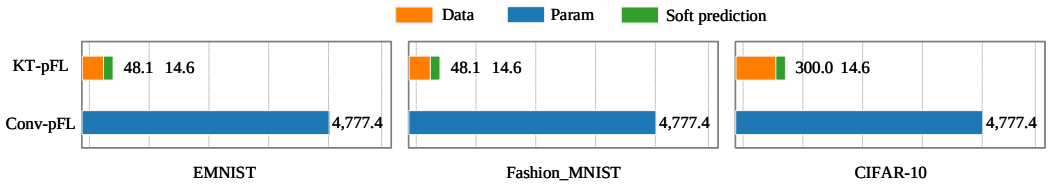


Figure 3: Communication Efficiency (X-axis units: MBytes) on three datasets. Soft prediction-based CSFL; Conv-pFL: conventional parameter-based CSFL. (20 models, 30 communication rounds for all datasets. In this experiment, the public dataset is stored on the server.)

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 MNFL 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.

Table 6: The comparison of final test accuracy on different setting of the public data size $|\mathbb{D}_r|$.

Dataset	# Size of public data ($\|\mathbb{D}_r\|$)					
	10	100	500	1000	3000	5000
EMNIST (%)	56.19	73.44	91.25	91.47	**91.66**	91.32
Fashion_MNIST (%)	50.80	67.43	88.08	89.40	**89.50**	89.14
CIFAR-10 (%)	39.08	47.70	**58.93**	58.79	58.91	58.40

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 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] Hong-You Chen and Wei-Lun Chao. Fedbe: Making bayesian model ensemble applicable to federated learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[18] Yanlin Zhou, George Pu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. Distilled one-shot federated learning. *CoRR*, abs/2009.07999, 2020.

[19] Lichao Sun and Lingjuan Lyu. Federated model distillation with noise-free differential privacy. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021*, pages 1563–1570. ijcai.org, 2021.

[20] 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.

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

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

[23] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *CoRR*, abs/2002.04758, 2020.

[24] 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.

[25] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

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

[27] Matthias Reisser, Christos Louizos, Efstratios Gavves, and Max Welling. Federated mixture of experts. *arXiv preprint arXiv:2107.06724*, 2021.

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

[29] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. Personalized federated learning with first order model optimization. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.

[30] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021*, volume 139, pages 9489–9502. PMLR, 2021.

[31] 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.

[32] Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, 2021.

[33] Neel Guha, Ameet Talwalkar, and Virginia Smith. One-shot federated learning. *CoRR*, abs/1902.11175, 2019.

[34] Chaoyang He, Salman Avestimehr, and Murali Annavaram. Group knowledge transfer: Collaborative training of large cnns on the edge. *arXiv preprint arXiv:2007.14513*, 2020.

[35] 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.

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

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.