DM-PFL: Hitchhiking Generic Federated Learning for Efficient Shift-Robust Personalization

Wenhao Zhang
State Key Laboratory of Software Development Environment, Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing,
Beihang University, Beijing, China
garyzhang@buaa.edu.cn

Zimu Zhou
School of Data Science,
City University of Hong Kong,
Hong Kong, China
zimuzhou@cityu.edu.hk

Yansheng Wang
State Key Laboratory of Software Development Environment, Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing,
Beihang University, Beijing, China
arthur_wang@buaa.edu.cn

Yongxin Tong
State Key Laboratory of Software Development Environment, Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing,
Beihang University, Beijing, China
yxtong@buaa.edu.cn

ABSTRACT

Personalized federated learning collaboratively trains client-specific models, which holds potential for various mobile and IoT applications with heterogeneous data. However, existing solutions are vulnerable to distribution shifts between training and test data, and involve high training workloads on local devices. These two shortcomings hinder the practical usage of personalized federated learning on real-world mobile applications. To overcome these drawbacks, we explore efficient shift-robust personalization for federated learning. The principle is to hitchhike the global model to improve the shift-robustness of personalized models with minimal extra training overhead. To this end, we present DM-PFL, a novel framework that utilizes a dual masking mechanism to train both global and personalized models with weight-level parameter sharing and end-to-end sparse training. Evaluations on various datasets show that our methods not only improve the test accuracy in presence of test-time distribution shifts but also save the communication and computation costs compared to state-of-the-art personalized federated learning schemes.

CCS CONCEPTS

• **Computing methodologies → Learning paradigms**.

KEYWORDS

Federated Learning; Sparse Training; Personalization; Robustness

ACM Reference Format:
Wenhao Zhang, Zimu Zhou, Yansheng Wang, and Yongxin Tong. 2023. DM-PFL: Hitchhiking Generic Federated Learning for Efficient Shift-Robust Personalization. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599311

1 INTRODUCTION

Federated Learning (FL) [19, 48] is an emerging paradigm that enables multiple clients to train models collaboratively without directly accessing their local data. It enables a wide spectrum of novel mobile and IoT applications including personal voice assistant [24], image recognition [16], next-word prediction on keyboard [46], human activity recognition [27], etc.

Data heterogeneity is a primary challenge when deploying FL to real-world mobile and IoT applications, where each client (often a resource-limited platform such as a smartphone, a smart-watch, a smart speaker, etc.) holds data partitions whose distributions differ from each other. For high-quality services on such personalized data, training client-specific models is necessary, which is known as personalized FL [11, 30, 46]. Unlike the generic FL [29, 33] that learns a global model for all clients, personalized FL trains customized models catered to better fit client-specific data distributions for high test accuracy at each client [46].

Despite extensive personalized FL proposals [7, 9, 28, 46], two drawbacks impede their practical usage for mobile applications.

- *Vulnerable to Test-Time Distribution Shifts.* Most personalized FL solutions [7, 9, 28, 46] implicitly assume that the training and test dataset at each client share the same distribution. This assumption often breaks for mobile applications, where user-specific training and testing are conducted in diverse contexts and environments. For example, the human activity recognition model on a smart-watch may be mainly trained for activities in the residential areas but tested for activities in the business areas. The model should yield consistent accuracy for user activities across diverse contexts and environments.
- *High Local Training Workload for Mobile Devices.* The clients in mobile and IoT applications are resource-constrained. A few personalized FL algorithms aim to reduce the communication cost [25, 30], yet the computation overhead of local

training remains an efficiency bottleneck. This is because existing personalized FL methods [9, 12, 28, 42] often demand additional training for personalization, which induces higher computation costs.

In response, we explore *efficient shift-robust personalization* for FL, which aims to train personalized models that are robust to distribution shifts between training- and test-time with low local training overhead. Our idea is to harness the global model that often emerges during personalized FL to improve the robustness to test-time shift through weight-level parameter sharing, and inject end-to-end sparsity into the federated learning process. Realizing this idea is challenging because existing schemes to combat distribution shifts [41, 44, 52] may induce significant overhead, and it is unknown how to enable sparse federated training dedicated to shift-robust personalization.

In this paper, we propose DM-PFL, (Dual Masked Personalized Federated Learning), an efficient and shift-robust personalized FL framework. With a novel dual-masking design, we managed to train both a global model and a set of personalized models for each client for shift-robust personalization without extra training workload. DM-PFL is also featured with a new federated training framework that allows end-to-end sparse training of both the global and personalized models. Our main contributions and results are:

- To the best our of knowledge, this is the first work exploring the *efficient shift-robust personalization problem*, an overlooked issue in practical FL for mobile and IoT applications.
- We propose DM-PFL, a new framework that efficiently hitchhikes generic FL for shift-robust personalization. We further design an extension DM-PFL+ to adaptively ensemble the global model and the personalized model for improved shift-robustness at test-time.
- Extensive experiments on various datasets show that we can outperform state-of-the-art personalized FL schemes [4, 7, 9, 28, 30, 46] by up to 31.67% in terms of test accuracy in presence of severe distribution shifts, and save the communication and computation cost by 26.6% and 37.1%.

2 PROBLEM STATEMENT

This section provides a quick review of generic and personalized federated learning (Sec. 2.1) and then presents the efficient shift-robust personalization problem (Sec. 2.2).

2.1 Federated Learning Basics

Federated learning allows distributed clients to collaboratively train models via the coordination of a central server. The objective is to improve the model accuracy at each client by exploiting data from other clients without direct access to their local datasets. The standard approach is *generic* FL, which trains a *global* model with the expectation to perform well for all clients by optimizing performance on the aggregation of large amounts of data across clients. However, since a client's local data distribution may notably deviate from the aggregated (global) data distribution, an averaged model can be sub-optimal. Accordingly, *personalized* FL is preferred in case of high data heterogeneity, which trains *client-specific* models to better fit the clients' local data distribution. Specifically, generic and personalized FL are defined below.

Generic FL. Given clients $\{1, 2, \ldots, C\}$ with local training datasets $\{D_1, D_2, \ldots, D_C\}$, generic FL aims to train a global model θ_g with the following objective.

$$\min_{\theta_g} l(\theta_g) = \sum_{c=1}^{C} \frac{|D_c|}{|D|}\mathbb{E}[\mathcal{L}(\theta_g; D_c)] \tag{1}$$

where θ_g is the global model parameters to be learned, $|D_c|$ is the size of client c's training dataset, $|D| = \sum_{c=1}^{C} |D_c|$ is the total size of training dataset, and $\mathbb{E}[\mathcal{L}(\theta_g; D_c)]$ is the expected empirical risk computed using model θ_g when data is sampled from client c's training dataset D_c.

Personalized FL. Given clients $\{1, 2, \ldots, C\}$ with local training datasets $\{D_1, D_2, \ldots, D_C\}$, personalized FL aims to train a set of personalized models $\{\theta_1, \theta_2, \ldots, \theta_C\}$, where personalized model θ_c for client c, with the following objective.

$$\min_{\theta_1,\theta_2,\ldots,\theta_C} l(\theta_1, ..., \theta_C) = \sum_{c=1}^{C} \frac{|D_c|}{|D|}\mathbb{E}[\mathcal{L}(\theta_c; D_c)] \tag{2}$$

where a key distinction from Eq.(1) is that the expected empirical risk $\mathbb{E}[\mathcal{L}(\theta_c; D_c)]$ is calculated with the personalized model θ_c rather than the global model θ_g on training dataset D_c.

Discussions. Both generic and personalized FL aim to train models in presence of data heterogeneity. Personalized FL is more challenging because *(i)* it assumes a higher level of data heterogeneity, which is the primary motivation of personalized FL [17]; and *(ii)* it often incurs higher costs, as it usually requires additional training to personalize the models [9, 11, 28, 42]. In this work, we explore *efficient personalized FL algorithm design*, yet focus on *an overlooked aspect of data heterogeneity*, as explained next.

2.2 Efficient Shift-Robust Personalized FL

Despite extensive research on personalized FL [2, 7, 9, 11, 12, 28, 39], we argue that there are two opportunities for improvement in terms of robustness to data heterogeneity and training efficiency.

- Prior studies [9, 12, 28, 39] mainly handle data heterogeneity *across* clients yet ignore that *within* clients. However, the data within clients can also be heterogeneous in the form of *distribution shifts between training- and test-time*. Enabling shift-robustness in personalized FL would deliver consistent user-specific test-time accuracy across diverse contexts or environments.
- Enforcing model sparsity proves effective to reduce the training workload by learning a sparse sub-network [10, 34]. Existing work [5, 35] has shown the feasibility of sparse training in generic FL. We aim to bring sparse training to the more challenging problem of personalized FL, so as to deploy personalized FL to low-resource devices.

We articulate the above two opportunities into the efficient shift-robust personalized FL problem below.

Definition 1 (Efficient Shift-Robust Personalized FL). *Given clients $\{1, 2, \ldots, C\}$ with local training datasets $\{D_1, D_2, \ldots, D_C\}$, we aim to train personalized models $\{\theta_1, \theta_2, \ldots, \theta_C\}$ with sparsity ratio $\mathbb{S}$ for each client c, with the objective to minimize the expected empirical risk on the unknown local test datasets $\{\tilde{D_1}, \tilde{D_2}, \ldots, \tilde{D_C}\}$, where*

$\tilde{D_c}$ can deviate from D_c in distribution. Formally, our objectives and constraints are as follows.

$$\min_{\theta_1,\theta_2,\ldots,\theta_C} l(\theta_1,...,\theta_C) = \sum_{c=1}^{C} \frac{|D_c|}{|D|}\mathbb{E}[\mathcal{L}(\theta_c;\tilde{D_c})] \\ s.t. \quad \|\theta_c\|_0 \le |\theta_c| * (1-\mathbb{S}), \forall c \in \{1,2,\ldots,C\} \tag{3}$$

where $\|\theta_c\|_0$ and $|\theta_c|$ are the L_0 norm and the total number of parameters in personalized model θ_c, respectively. $\mathbb{S}$ is a given sparsity ratio (the percentage of zero parameters for all clients).

We make the following notes on the efficient shift-robust personalized FL problem.

- Unlike previous studies [9, 12, 39] that implicitly assume the same distribution of D_c and $\tilde{D_c}$, we allow distribution shift between the local training dataset D_c and the local test dataset $\tilde{D_c}$. Hence, we account for data heterogeneity both across and within clients.
- We can assume both $\{D_c\}$ and $\{\tilde{D_c}\}$ are sampled from a meta-distribution, which is common in cross-device FL with large numbers of participants [51].
- For ease of presentation, we will use D as either a dataset or its distribution interchangeably if there is no confusion.
- We mainly improve training efficiency by enabling sparse training on clients. Given a sparsity rate $\mathbb{S}$, both the communication and the computation cost in federated learning will roughly reduce by $\mathbb{S}$ percent since they are proportional to the number of parameters [5, 10].

Challenges. Algorithms for the optimizations in Eq.(3) face two non-trivial challenges.

- *How to enable shift-robust personalized FL with minimal extra training efforts?* Existing methods to improve the robustness to distribution shifts [41, 44, 52] often induce considerable overhead. For example, they may generate specific datasets for extra training [44] or require test-time training on auxiliary tasks [41, 52].
- *How to design sparse training schemes dedicated to shift-robust personalized FL?* Existing sparse training strategies for generic FL [5, 35] are not directly applicable for Eq.(3) because generic FL only optimizes the *average* performance of a *single* model whereas we simultaneously optimize the performance of *multiple* models on different datasets that are potentially *shifted.*

3 METHOD

This section presents DM-PFL (Dual Masked Personalized Federated Learning), our efficient and shift-robust personalized FL framework.

3.1 DM-PFL Overview

This subsection shows an overview of DM-PFL. It trains shift-robust personalized models by hitchhiking generic FL without extra training overhead, and it is featured with a novel dual-mask-based scheme for end-to-end sparse training. We explain the high-level design rationales and the workflow of DM-PFL below.

Rationales. The design rationales of DM-PFL are two-fold.

- **Hitchhike Generic FL**. We train both a personalized model θ_c that optimizes Eq.(2) for each client c, and a global model θ_g that optimizes Eq.(1) and ensemble them into a final model for inference. Since the global model is trained on the aggregated dataset $D = \cup_{c=1}^{C} D_c$, it is likely to perform well on $\{\tilde{D_c}\}$ if the distribution shifts are seen in the local training datasets of other clients (Note that we can assume both $\{D_c\}$ and $\{\tilde{D_c}\}$ are sampled from the same meta-distribution [51]). Furthermore, training and maintaining additional global weights may only incur minimal extra overhead because we can leverage the locally trained personalized models to construct a global model (see our dual masking scheme in Sec. 3.2).
- **Mask-based Sparse Training**. We adopt mask-based training because it not only allows us to reduce the communication and computation cost by introducing sparsity [5], but also offers a natural option for fine-grained parameter sharing among multiple models, which is crucial because we aim to train both personalized and global models on clients with high efficiency. Parameter sharing across models may also help reduce training overhead than training the two models separately.

Workflow. We implement the above rationales into DM-PFL, a new personalized FL framework. It consists of a novel dual masking mechanism (Sec. 3.2), a dual-masked sparse training algorithm (Sec. 3.3), and an adaptive inference scheme (Sec. 3.4). The dual masking mechanism injects sparsity into the models and allows weight-level parameter sharing between the personalized and the global model. Under such a mechanism our dual-masked sparse training algorithm can perform end-to-end sparse training of both models efficiently and robustify the personalized models. The adaptive inference scheme can further enhance shift-robustness by dynamically ensembling the learned dual models for inference.

We introduce the principles and details of each module below.

3.2 Dual Masking Mechanism

We first propose a masking mechanism for joint training of personalized and global models under the sparsity constraint.

For our masking scheme, we maintain a global mask m_g and a set of personalized masks $\{m_c\}_{c=1}^{C}$. During training, we learn a global sparse model θ_g for all the clients, as well as a set of local sparse models $\{\theta_c\}_{c=1}^{C}$ defined as follows.

$$\theta_g = w_g \odot m_g,\ \theta_c = w_g \odot (m_g \cap m_c) + w_c \odot (m_c - m_g) \tag{4}$$

where w_g and w_c are the corresponding global and personalized weights in the common dense architecture shared by θ_g and θ_c.

We make the following notes on our masking scheme.

- The global model θ_g and the personalized model θ_c share the same dense architecture, while their sparsity is separately specified by two categories of masks m_g and m_c.
- For a given client, its personalized model θ_c and the global model θ_g share parameters at the weight-level, *i.e.*, controlled by the overlap between m_c and m_g.
- The personalized model θ_c differs in both mask positions and weights across clients. Each client inherits a different portion of the global weights w_g according to its overlap in

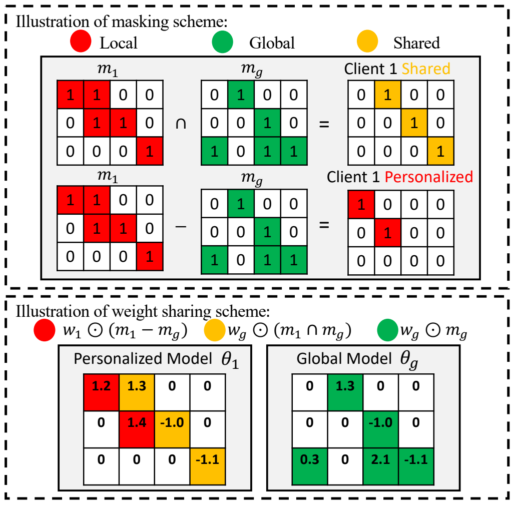


Illustration of masking scheme:
Local
Global
Shared
m_1
m_g
Client 1 Shared
Client 1 Personalized
Illustration of weight sharing scheme:
$w_1 \odot (m_1 - m_g)$
$w_g \odot (m_1 \cap m_g)$
$w_g \odot m_g$
Personalized Model θ_1
Global Model θ_g

Figure 1: An illustration of the dual masking scheme.

mask positions with the global mask, *i.e.*, $m_g \cap m_c$. For the remaining positions $m_c - m_g$ in its personalized mask, each client inherits weights from its own w_c.

Fig. 1 illustrates our masking scheme for Client 1. The personalized mask m_1 and global mask m_g are marked in red and green, respectively. The global model and the personalized model for Client 1 share the same weights at positions $m_g \cap m_1$ as marked in yellow, while the personalized model for Client 1 also has the client-specific masks and weights at $m_1 - m_g$ as marked in red.

Our dual masking scheme enables *(i)* weight-level parameter sharing between the personalized and the global model, and *(ii)* both the mask positions and weights in the personalized models to differ across clients. Our dual masking design offers more flexibility compared with coarse-grained parameter sharing schemes such as [4, 7, 40], while still allowing efficient and effective federated training of sparse dual models, as explained next.

3.3 Dual-Masked Sparse Training Algorithm

In this subsection, we introduce the federated training scheme of the global sparse model θ_g and the personalized sparse models θ_c, where $c = 1, 2, \ldots, C$, with the objectives specified in Eq.(1) and Eq.(2), and θ_c are defined as in Eq.(4). It demands effective and efficient learning of four parameter sets: m_c, m_g, w_c, and w_g for all the clients. In essence, we need to carefully coordinate the training of *(i)* masks vs. weights; as well as *(ii)* the global model vs. the personalized models. In response, we propose a novel two-phased sparse federated training pipeline, as explained below.

3.3.1 Two-Phased Training Pipeline. Fig. 2 shows our sparse federated training scheme. It consists of two phases by first training masks and then refining weights, while each phase partially learns the parameters of both the global and the personalized models. The two phases are conducted iteratively for high accuracy.

- *Phase 1: Train Dual Masks (details in Sec. 3.3.2).* It aims to learn which weights to share between the global and personalized models for each client by federated training of the global mask m_g and personalized masks m_c. At a high level, we search and adjust mask locations based on weight importance (by magnitude or gradient information), as in prior sparse federated training schemes [5, 8]. However, our mask training phase differs in the mechanisms that allow flexible transitions between the global and personalized masks during federated training. Specifically, in each round, we first optimize w_c under m_c, then perform mask readjustment on m_c during local training. We then obtain w_g and m_g at the server by aggregating parameters from local training, which are then disseminated to the clients for training the masks in the next round.
- *Phase 2: Refine Dual Weights (details in Sec. 3.3.3).* We take the weights from Phase 1 as initialization and refine the global weight w_g as well as the personalized weights w_c under fixed masks m_g and m_c. At a high level, we first refine w_g and then w_c to hitchhike the global weights to improve the shift-robustness of the personalized models. Specifically, we first train w_g under the fixed mask $w_g \odot m_g$ to obtain the global model θ_g by federated training with all clients, and then fix $w_g \odot (m_g \cap m_c)$ to refine $w_c \odot (m_c - m_g)$ for each client locally.

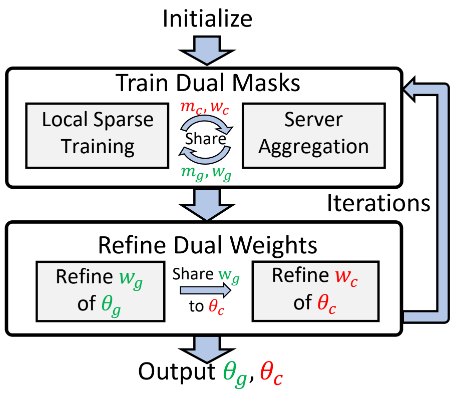


Figure 2: Two-phased sparse training pipeline.

Discussions. Our dual-masked sparse training algorithm has the following features.

- We orchestrate the training of dual masks and dual weights. The weights guide the search of the masks, while the masks determine the position of weights as well as whether they are shared between the global and personalized models.
- We efficiently train both the global and the personalized models without doubling the training workload. Training θ_c locally is equivalent to training different parts of the θ_g, and training θ_g is equivalent to training the shared overlapping weights that make θ_c shift-robust.
- To the best of our knowledge, we are the first to train both global and personalized models interactively with end-to-end sparse training.

We explain the two training phases in detail below.

3.3.2 Train Dual Masks. The training of dual masks m_g and m_c is conducted iteratively via *local sparse training* and *server aggregation.*

Algorithm 1: Train Dual Masks

```
Input: Clients' masks $m_1, ..., m_C$, weights $w_1, ..., w_C$, global
       mask $m_g$, global weight $w_g$
Output: Clients' masks $m_1, ..., m_C$, weights $w_1, ..., w_C$,
        global mask $m_g$, global weight $w_g$
1  for round $t$ do
2      Sample clients $\mathbb{C}$ for this round;
3      for each client $c \in \mathbb{C}$ do
4          Client $c$ receives $m_g^{t-1}$ and $m_g^{t-1} \odot w_g^{t-1}$;
5          $w_c^t \leftarrow w_c^{t-1}$;
6          $w_c^t \odot (m_g^{t-1} \cap m_c) \leftarrow w_g^{t-1} \odot (m_g^{t-1} \cap m_c)$;
7          for each epoch $e$ from 1 to $E$ do
8              $g_c \leftarrow \nabla\mathcal{L}(w_c \odot m_c^t; D_c)$;
9              $w_c^t \leftarrow w_c^t - \eta g_c \odot m_c$;
10         if readjust masks then
11             Prune $\alpha_s$-proportion of $m_c^t$ with smallest weight
                 magnitude;
12             Regrow $\alpha_s$-proportion of $m_c^t$ with largest
                 gradient magnitude;
13         Client $c$ uploads $w_c^t, m_c^t$ to server;
14     Server receives $w_c^t, m_c^t$ for all clients in $\mathbb{C}$;
15     Server performs aggregation, updates $w_g^t$ and $m_g^t$;
```

Concretely, the dual masks training procedure at each round t is as follows (see Algorithm 1).

- *Local Sparse Training.* Given m_g^{t-1} and the corresponding global weight w_g^{t-1} received from the server (Line 4), each client c keeps its personalized weights unchanged (Line 5) and sets weights at the overlapping position $m_g \cap m_c$ to w_g^{t-1} (Line 6). It then conducts E epochs of local training under its sparse mask m_c (Line 7-9). Afterward, each client can readjust its mask m_c via pruning and regrowing [5, 10], with a readjustment ratio of α_s (Line 10-12). Our method slightly differs from the original pruning and regrowing strategy [5, 10] in that we allow the pruned mask to regrow back in the same round, which allows the amounts of readjusted masks to differ across clients.
- *Server Aggregation.* Unlike existing FL that only aggregates weights in a dense model, we need to aggregate both w_g and m_g from w_c and m_c for the next round (Line 13-15). Accordingly, we design a new aggregation scheme below.
  - *Aggregate* w_g. We update w_g^t as Eq.(5).

$$w_g^t \odot (\cup_{c\in\mathbb{C}} m_c) \leftarrow \frac{\sum_{c\in\mathbb{C}} |D_c| * w_c^t \odot m_c^t}{\sum_{c\in\mathbb{C}} |D_c| * m_c^t} \tag{5}$$

    For positions that are already covered by the sampled clients' masks, we update w_g^t by the mask-wise average of weights adjusted by the dataset size. For other positions, we keep the same weight w_g^{t-1} as in the last round. This way, the server will not immediately discard the weights and positions in the previous round.
  - *Aggregate* m_g. After computing w_g^t, we select the top $1 - \mathbb{S}$ percent of positions with the largest weights to create the new global mask m_g^t. We will ignore positions selected by no more than 30% clients to avoid using positions important only for a small portion of clients in the global mask m_g^t.

Algorithm 2: Refine Global Weights

```
1  for round $t$ do
2      Sample clients $\mathbb{C}$ for this round;
3      for each client $c \in \mathbb{C}$ do
4          Client $c$ receives $w_g^{t-1} \odot m_g$;
5          $w^t \leftarrow w_g^{t-1} \odot m_g$;
6          for each epoch $e$ from 1 to $E$ do
7              $g_c \leftarrow \nabla\mathcal{L}(w^t \odot m_g; D_c)$;
8              $w^t \leftarrow w^t - \eta g_c \odot m_g$;
9          Client $c$ uploads $w^t$ to server;
10     Server receives $w^t$ for all clients in $\mathbb{C}$;
11     Update $w_g^t$ with standard weight aggregation;
```

Algorithm 3: Refine Personalized Weights

```
1  for each client $c$ do
2      Client $c$ receive $m_g$ and $m_g \odot w_g$;
3      $w_c \leftarrow w_g \odot (m_g \cap m_c)$;
4      for each local epoch $e$ do
5          $g_c \leftarrow \nabla\mathcal{L}(\theta_c; D_c)$;
6          $w_c^t \leftarrow w_c^t - \eta g_c \odot (m_c - m_g)$;
```

Discussions. A key feature of our dual mask training scheme is to allow m_g and m_c to dynamically evolve, which enables personalized and shared weights to transform into each other during training. We highlight this property via two toy examples in Fig. 3.

- *From personalized weight to shared weight (marked by blue boxes in Fig. 3).* Most clients find this position important throughout their local training process, so in the round 2 server aggregation, the position is covered by the global mask and turns into a shared weight afterward.
- *From shared weight to personalized weight (marked by purple boxes in Fig. 3).* The server finds this position is unimportant for most clients in round 2 based on the aggregated weight magnitude and pruned it in the global mask, so the position turns into a personalized weight for C_K in round 3.

3.3.3 Refine Dual Weights. Given the four sets of parameters obtained in Sec. 3.3.2, we now refine w_g and w_c. Specifically, we first refine the global weights w_g in the federated setting and then refine the personalized weights w_c locally at each client. Algorithm 2 and Algorithm 3 illustrate the training of w_g and w_c, respectively.

- *Refine Global Weights* w_g (Algorithm 2). We refine w_g following a similar process as FedAvg [33] yet with masks on, *i.e.*, $m_g \odot w_g$. Specifically, in each round t, clients receive $m_g \odot w_g^{t-1}$ from the server (Line 4-5), perform E epochs of local training (Line 6-8), and upload the new weights w^t to the server (Line 9). The server then performs standard

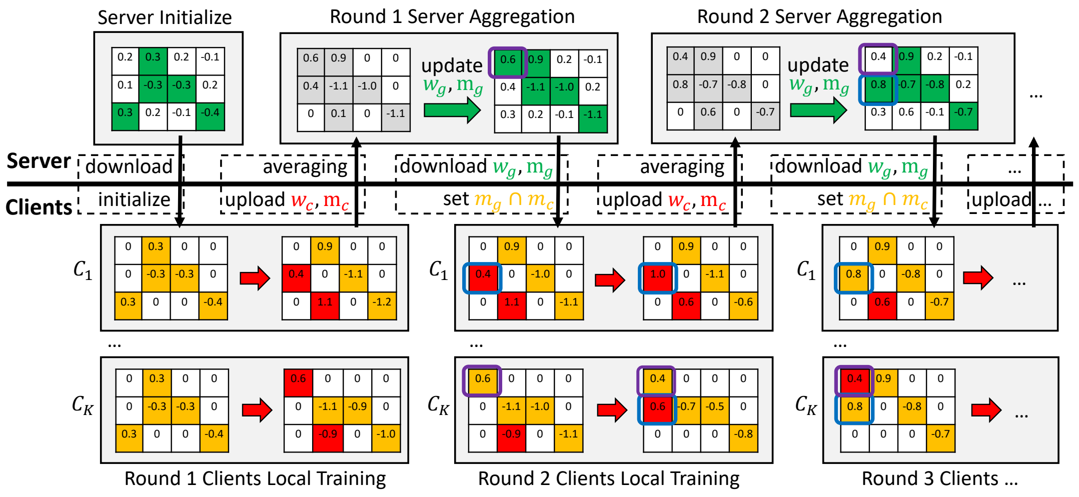


Figure 3: An example of dual mask training.

weight aggregation as [33] to get the new global weight w_g^t for the next round.

- *Refine Personalized Weights* w_c (Algorithm 3). Each client sets the weights at the shared positions $m_c \cap m_g$ with the global weights w_g (Line 2-3), and then locally retrains its personalized weights w_c for positions $m_c - m_g$ (Line 4-6).

Discussions. The workload of our weight training scheme is less than that of training w_g and w_c independently. With our dual masking design, refining the global weight w_g can be viewed as refining the shared weights of personalized models for each client. Accordingly, we only need to train the personalized weights w_c at the personalized positions rather than all the weights in the model when performing personalization.

3.4 Adaptive Inference

We can directly use θ_c for shift-robust personalization for inference at client c. However, we may further utilize θ_g to improve the shift-robustness, which adaptively ensembles θ_g and θ_c based on the estimated degree of test-time shift (see Algorithm 4). We adopt a lightweight test-time shift estimation method using a combination of two metrics: the *entropy of output logits* [13] and the cosine similarity of the outputs from θ_g and θ_c.

Specifically, we first compute the entropy of the model's output to estimate how well θ_c and θ_g fit the given input x(Line 1-2). To estimate the degree of shifts, the cosine similarity of the outputs that will be lower when encountering shifted input is computed (Line 3). The base entropy levels (*i.e.*, the average entropy of models' output logits when inputs are not shifted) of $BE(\theta_c)$ and $BE(\theta_g)$ are also considered, to avoid θ_c being over-confident. $BE(\theta_c)$ is typically lower than $BE(\theta_g)$, and we choose θ_g with higher $BE(\theta_g)$ when the cosine similarity of outputs is lower (Line 4-7).

Discussions. We make two notes on the inference using models obtained by our dual-masked sparse federated training scheme.

Algorithm 4: Adaptive Inference

```
Input: Personalized model $\theta_c$, global model $\theta_g$, base
       entropy levels $BE(\theta_c), BE(\theta_g)$, testing sample $(x, y)$
       from local testing dataset $\tilde{D}_c$
Output: One-hot prediction results for local test dataset $y^*$
1 $E_c \leftarrow entropy(\theta_c(x))$;
2 $E_g \leftarrow entropy(\theta_g(x))$;
3 $Sim \leftarrow cosine\_similarity(\theta_c(x), \theta_g(x))$;
4 if $E_c - (1 - Sim)BE(\theta_c) < E_g - (1 - Sim)BE(\theta_g)$ then
5     $y^* \leftarrow argmax(\theta_c(x))$;
6 else
7     $y^* \leftarrow argmax(\theta_g(x))$;
8 end
```

- The personalized model θ_c trained by our scheme is already more shift-robust than existing personalized FL algorithms, as shown in our evaluations (Sec. 4.2.1).
- The adaptive inference method in Algorithm 4 mainly serves as a proof-of-concept. One may apply more advanced strategies to perform model ensemble [47] and OOD-detection [15] for more accurate test-time shift adaptation.

4 EVALUATION

4.1 Experimental Setup

4.1.1 Compared methods. We present the experimental results of various generic FL and personalized FL algorithms. For ease of presentation, we denote generic FL and personalized FL as GFL and PFL. For GFL, we report results of : FedAvg [33], FedProx [29] and FedDST [5]. For PFL, we report results of: FedAvg+FT [46], LG-FedAvg [30], FedPer [4], FedRep [7], Ditto [28] and APFL [9]. DM-PFL+ is the proposed method with adaptive inference. More detailed introductions to the compared methods can be found in the appendix A.1.1.

Table 1: Accuracy with different degrees of shift on CIFAR10 in Dir(0.3) non-IID setting.

Type	Methods	w/o shift	w/ 20% shift	w/ 40% shift	w/ 60% shift	w/ 80% shift	w/ 100% shift	Averaged
Local	Local	61.86±9.53	54.11±11.66	46.23±13.78	38.38±15.84	30.70±17.97	22.93±20.07	42.37±14.81
GFL	FedAvg [33]	60.16±8.21	59.84±8.84	59.53±9.49	59.20±10.14	58.88±10.78	58.56±11.42	59.36±9.81
	FedProx [29]	58.61±10.60	58.52±10.55	58.43±10.5	58.33±10.46	58.24±10.41	58.15±10.36	58.38±10.48
	FedDST [5]	60.13±6.76	59.25±8.70	58.89±8.84	58.52±8.97	58.15±9.11	57.79±9.24	58.79±8.60
PFL	FedAvg+FT [46]	79.33±8.68	72.12±10.72	65.34±12.68	58.41±14.67	51.50±16.69	44.60±18.68	61.88±13.69
	LG-FedAvg [30]	69.50±13.29	60.53±14.52	51.09±15.76	41.83±16.96	32.80±18.20	23.62±19.42	46.56±16.36
	FedPer [4]	76.57±12.27	68.08±14.45	59.69±16.53	51.02±18.67	42.54±20.74	34.13±22.87	55.34±17.59
	FedRep [7]	76.85±11.31	67.40±13.00	58.49±14.80	49.28±16.51	40.24±18.24	31.20±19.95	53.91±15.63
	Ditto [28]	79.70±8.59	73.17±10.24	67.56±11.68	61.40±13.22	55.11±14.74	49.04±16.25	64.33±12.45
	APFL [9]	78.41±9.92	69.97±11.81	62.03±13.78	53.50±15.75	45.42±17.69	37.13±19.63	57.74±14.76
Ours	DM-PFL	**79.93±8.61**	**75.36±9.55**	**70.75±10.53**	66.09±11.46	61.38±12.39	56.79±13.34	68.38±10.98
	DM-PFL+	77.22±9.31	73.81±8.80	70.41±8.30	**66.97±7.79**	**63.61±7.27**	**59.61±8.57**	**68.61±8.34**

4.1.2 Datasets. We conduct experiments on three image classification datasets MNIST [23], FEMNIST [6], CIFAR10 [22], and a smartphone HAR dataset [3]. Different non-IID settings are simulated, including *(i)* Dirichlet distribution-based [16], *(ii)* pathological [33], and *(iii)* realistic [6]. For Dirichlet distribution-based and pathological non-IID settings, we further partition the dataset so that the data quantity at each client also differs. For the realistic non-IID setting, the data quantity at each client naturally varies.

4.1.3 Metrics. We assess the effectiveness and efficiency of different algorithms with the following metrics.

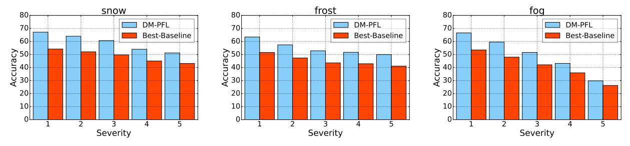
- Communication Cost: It measures the average total size (MB) of parameters uploaded/downloaded during training.
- Computation Cost: It counts the average total number of floating point operations (FLOP) per client.
- Accuracy without Shift: It measures the accuracy performance on local testing datasets that have the same distribution as local training datasets.
- Accuracy with Shift: It measures the accuracy when the data distribution is shifted at test time. We test different kinds and degrees of shifts.

More experimental settings are provided in our appendix A.1.

4.2 Main Experimental Results

4.2.1 Effectiveness Results. We first report test accuracy varying different degrees/kinds of shift and non-IIDness.

Accuracy under Different Degrees of Shifts. For ease of presentation, we represent the degree of shift by the proportion of shifted data, where the shifted data is randomly sampled from the hypothesized unified meta-distribution [51]. Table 1 reports the results on CIFAR10 in Dir(0.3) non-IID setting. It can be seen that most PFL algorithms including ours can outperform GFL by about 20% without shift, which verifies the effectiveness of personalization schemes. With the increase of shift degree, the performance of PFL goes down significantly while GFL remains relatively stable. But our algorithms DM-PFL and DM-PFL+ still show superiority over both PFL and GFL. The average advantages over the best GFL and PFL baselines are 9.25% and 4.28% respectively. With 100% shift, DM-PFL+ can outperform PFL by 10.57%. Meanwhile, it can even outperform GFL by 1.05%, as our θ_g has strong generalization ability which can be attributed to the in-time over-parameterization effect



(a) snow (b) frost (c) fog

Figure 4: Accuracy under different kinds of shifts with 5 levels of severity on CIFAR10-C. More results under different kinds of shifts can be found in the appendix A.2.2.

[32] brought by our dual-masked training scheme. We can also observe that DM-PFL+ can perform better than DM-PFL with higher shift degrees, which validates the effectiveness of adaptive inference. Results under different degrees of shift on realistic datasets FEMNIST and HAR are also reported and can be found in A.2.1.

Accuracy under Different Kinds of Shifts. We run experiments on CIFAR10-C [14] to further evaluate the shift-robustness of our personalization algorithm. The CIFAR10-C dataset contains images from CIFAR10 that are corrupted or perturbated under various settings with 5 different severity levels. We utilize CIFAR10-C as the testing dataset and run the experiment on the models trained on CIFAR10 Dir(0.3). The performance of DM-PFL and the best performance of all PFL baselines are reported in Fig. 4 and in A.2.2. DM-PFL consistently outperforms PFL baselines by 1.32% to 11.96% across different kinds of shifts with 5 levels of severity. It verifies that DM-PFL has strong shift-robustness compared with other baselines and is resilient to various kinds of shifts.

Accuracy in Different Non-IID Settings. We further compare our algorithm with other PFL baselines in different Non-IID settings. Results on CIFAR10, HAR, FEMNIST are shown in Table 2 and Table 3. Results on non-IID settings simulated on MNIST can be found in A.2.3. Comparing with the best PFL baseline in various non-IID settings, our methods have slight advantages without shift. As the client's data distribution becomes more heterogeneous, our methods have larger advantages in terms of shift-robustness. With 100% degree of shift and highly heterogeneous data distribution of clients, DM-PFL and DM-PFL+ can outperform the best PFL baseline by at most 26.49% and 31.67%.

Table 2: Accuracy with and w/o shift on CIFAR10 in Dir(0.1) and pathological non-IID settings.

Non-IID	Dir(0.1)		Pathological	
Methods	w/o	w/ 100%	w/o	w/ 100%
FedAvg+FT [46]	89.00±11.89	23.85±25.32	89.76±6.78	20.47±31.26
LG-FedAvg [30]	85.24±13.90	17.38±24.76	84.41±7.34	15.54±31.69
FedPer [4]	88.82±12.63	22.76±30.10	89.17±6.84	16.91±34.98
FedRep [7]	89.19±10.48	21.60±27.40	89.69±6.15	17.24±34.97
Ditto [28]	88.20±9.12	39.79±26.91	89.22±5.53	22.46±28.03
APFL [9]	89.18±9.76	22.15±29.28	89.41±6.87	21.48±34.69
DM-PFL	**89.29±11.65**	45.24±23.54	**90.06±5.59**	48.95±24.20
DM-PFL+	88.16±12.46	**47.33±16.53**	88.59±7.65	**54.13±23.44**

Table 3: Accuracy with and w/o shift on HAR and FEMNIST in realistic non-IID setting.

Dataset	HAR		FEMNIST	
Methods	w/o	w/ 100%	w/o	w/ 100%
FedAvg+FT [46]	97.38±5.53	92.84±6.97	98.85±5.23	96.86±9.76
LG-FedAvg [30]	98.16±9.12	80.12±11.23	93.15±8.86	62.44±19.70
FedPer [4]	98.07±3.54	94.52±5.50	98.75±6.78	97.44±5.61
FedRep [7]	97.91±3.11	90.14±8.22	96.49±6.42	84.16±13.97
Ditto [28]	97.52±4.03	94.53±6.17	98.71±4.48	96.72±7.31
APFL [9]	95.51±5.62	87.47±10.19	98.49±3.76	96.55±8.05
DM-PFL	**98.41±2.72**	95.37±4.05	**99.04±3.80**	97.86±5.28
DM-PFL+	97.93±3.46	**95.58±4.76**	98.62±4.63	**98.10±5.01**

Table 4: Averaged total communication and computation cost per client for training on CIFAR10 with $\mathbb{S} = 0.5$.

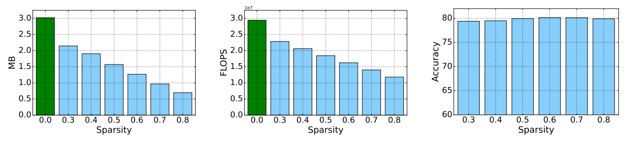
Methods	Comm (MB)		FLOP (1e12)	
Local	0	0%	6.12	100%
FedAvg [33]	90.75	100%	6.12	100%
FedProx [29]	90.75	100%	6.12	100%
FedDST [5]	46.44	51.2%	3.90	63.7%
FedAvg+FT [46]	90.75	100%	6.32	103%
LG-FedAvg [30]	60.71	66.9%	6.12	100%
FedPer [4]	90.53	99.7%	6.12	100%
FedRep [7]	90.53	99.7%	8.28	131%
Ditto [28]	90.75	100%	12.27	201%
APFL [9]	90.75	100%	12.26	200%
DM-PFL	36.58	40.3%	3.91	63.9%

4.2.2 Efficiency Results. In Table 4 we present the averaged total communication and computation cost per client for training on CIFAR10 with sparsity $\mathbb{S} = 0.5$. We will explore efficiency varying different sparsity ratios in ablation studies(Sec. 4.3.1).

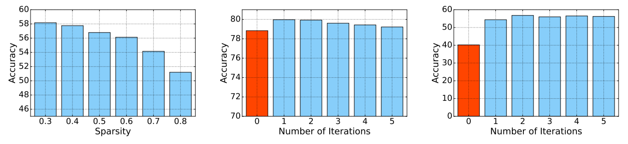
Communication Costs. Table 4 shows that comparing with FedAvg, we can save 59.7% of communication cost. Comparing with another communication efficient PFL method LG-FedAvg, we can save the communication cost by 26.6%. We also compare with FedDST, which is a GFL method with sparse training. We observe that with the same sparsity ratio, DM-PFL can still decrease the communication cost by 10.9%. It is because that our method does not need to upload/download parameters every round when refining personalized weights.

Computation Costs. Comparing with FedAvg, we save 36.1% of computation cost, which is roughly the same as FedDST. All the PFL baselines have no lower computation cost than FedAvg. APFL and Ditto even double the computation cost because they have to maintain both the personalized and the global models separately. Comparing with them, our method DM-PFL also obtains a well-performed global model but requires much lower computation cost, which verifies that we efficiently hitchhike the global model without extra training overhead.

4.3 Ablation Studies



(a) Impact of sparsity on communication cost **(b) Impact of sparsity on computation cost** **(c) Impact of sparsity on accuracy w/o shift**



(d) Impact of sparsity on accuracy w/ 100% shift **(e) Impact of iterative training on performance w/o shift** **(f) Impact of iterative training on performance w/ 100% shift**

Figure 5: Ablation results on impact of sparsity ratio (a)-(d) and iterative training (e)-(f).

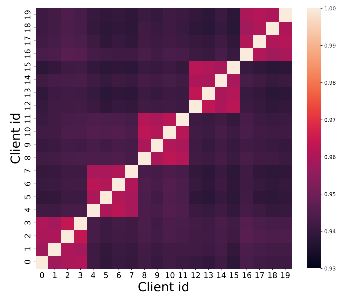
4.3.1 Impact of Sparsity Ratio. To investigate how different sparsity ratios will affect the effectiveness and efficiency of DM-PFL, we run experiments on CIFAR10 Dir(0.3) with sparsity ratios ranging from 0.3 to 0.8, which is a common setting in other sparse FL algorithms [5, 8, 25].

Impact on Efficiency. Fig. 5(a) shows how different sparsity ratios impact the upload/download size of parameters per round. We can see that the reduction of communication cost is roughly proportional to the degree of sparsity. Fig. 5(b) shows the results of FLOPS needed per sample. The reduction in computation cost is also proportional to the degree of sparsity, but slightly lower, since our sparsity setting follows the ERK distribution [10] which assigns lower sparsity to layers with fewer redundant parameters.

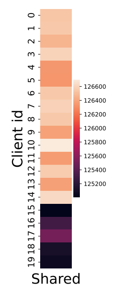
Impact on Effectiveness. Fig. 5(c) and (d) show the impact of different sparsity ratios on accuracy with and without shifts. We can observe that the performance with shift is more vulnerable with the sparsity ratio going higher. The results verify that balancing the trade-off between efficiency and shift-robustness is still challenging, since a higher sparsity ratio could easily result in under-parameterized models.

4.3.2 Impact of Iterative Training. Next, we study how the two-phased iterative training scheme affects the performance of our method. In Fig. 5 (e) and (f) we show how DM-PFL performs under different iterative training schedules. We divide $T = 300$ rounds equally into each phase and test the performance of the algorithm with different numbers of iterations. We also test the performance of the algorithm without iteration (*i.e.*, the algorithm only trains for 300 rounds in the first phase and skips the refining weights phase) at the same time, which we denote as $iter = 0$. We can see that without iterative training, the accuracy decreases by 1.13% without shift and by 14.14% with 100% shift. The results verify the effectiveness of our two-phased training scheme.

4.3.3 Analysis of the Learned Masks. To better explain how our dual masking scheme works, we conduct the following experiments on CIFAR10.



(a) Between m_c



(b) Between m_c and m_g

Figure 6: Analysis of the learned masks.

Effectiveness of Personalized Masks. We divide the 20 clients into 5 groups. Clients share similar data distribution within groups. In Fig. 6 (a) we show the similarity between the learned personalized masks, and it validates that the similarity of masks m_c is strongly correlated with the similarity of local training data distribution, hence proving that the learned masks can successfully adapt to its local data distribution for personalization.

Effectiveness of Weight Sharing. We divide the 20 clients as follows: clients 0 to 14 have similar data distribution to the global data distribution, and clients 15 to 19 have heterogeneous data distribution. Fig. 6 (b) shows the number of shared weights $\|m_c \cap m_g\|_0$ in the fully-connected layer between the global and personalized models. We can observe that clients with less heterogeneous data will share more parameters with the global model, while clients with higher heterogeneity will share fewer parameters. It validates the effectiveness and flexibility of DM-PFL in terms of weight-level parameter sharing.

5 RELATED WORK

Generic and Personalized FL. FedAvg [33] is the standard algorithm for generic FL. To combat data heterogeneity in FL, many generic FL algorithms improve over FedAvg, such as using regularization terms or control variables to correct drifts in local training [1, 20, 29], performing server aggregation with weight matching [45, 49] or knowledge distillation [31], etc. However, the problem of data heterogeneity remains, as training a single global model that generalizes well to heterogeneous clients is difficult [46, 50].

In response, personalized FL that adopts different strategies to train client-specific models is proposed for better performance. Methods such as local fine-tuning [46] and meta-learning [2, 11] train a single global model for all clients and then perform personalized local training on the client side, while multitask learning-based [39, 42] and clustering-based [12, 38, 53] methods group the clients and train multiple global models accordingly. Another more related category includes model interpolation methods [9, 28] and layer-wised weight sharing methods [4, 7, 30] which we compared empirically. For more details, we refer the readers to comprehensive surveys in [19, 43].

Sparsity in FL. Sparse training can save computation and communication costs in FL. Some studies employ compression techniques [21, 37] to lower the communication cost. But they do not sparsify the models during training, and thus cannot save the computation cost. Dense-to-sparse pruning techniques can be combined with federated learning [18, 25, 26] to improve communication efficiency during training and computation efficiency during inference. However, these methods first train dense models and then prune the local models, which will increase the total computation costs for training on mobile devices. A few recent studies [5, 8, 35] apply end-to-end sparse training to FL, which can save both the total communication and computation costs during training. For example, ZeroFL [35] adopts sparse weight activation training [36] to on-device FL, and empirically shows that sparsification in FL is more challenging than in centralized settings. FedDST [5] utilizes dynamic sparse training [10] in generic FL setting. It performs mask readjustment locally, but the server will synchronize all clients with the same weights and masks at the start of each round. DisPFL [8] is a decentralized (peer-to-peer) sparse FL algorithm. It has similar mask readjusting process to [5], but uses the sparse masks at the clients for personalization.

We also employs the dynamic sparse training technique in FL for end-to-end sparsification, but our focus is to improve the shift-robustness efficiently with the proposed dual masking mechanism.

6 CONCLUSION

This work presents DM-PFL, a novel personalized federated learning framework for efficient shift-robust personalization. It hitchhikes the global model learned under the control of a dual masking mechanism to make the personalized models less vulnerable to test-time distribution shifts and enable end-to-end sparse training. Evaluations show that DM-PFL can improve on both the test accuracy with and without test-time distribution shifts and save communication and computation costs. We envision our work will facilitate the practical adoption of personalized federated learning in mobile and IoT applications.

ACKNOWLEDGMENTS

We are grateful to anonymous reviewers for their constructive comments. This work is partially supported by National Science Foundation of China (NSFC) under Grant No. U21A20516 and 62076017, the Beihang University Basic Research Funding No. YWF-22-L-531, the Funding No. 22-TQ23-14-ZD-01-001, the CityU APRC grant No. 9610633, and WeBank Scholars Program. Yongxin Tong is the corresponding author.

REFERENCES

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. 2020. Federated learning based on dynamic regularization. In *Proceedings of the International Conference on Learning Representations*.

[2] Durmus Alp Emre Acar, Yue Zhao, Ruizhao Zhu, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. 2021. Debiasing model updates for improving personalized federated training. In *Proceedings of the International Conference on Machine Learning*. PMLR, Virtual Event, 21–31.

[3] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz, et al. 2013. A public domain dataset for human activity recognition using smartphones. In *Esann*, Vol. 3. 3.

[4] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. 2019. Federated learning with personalization layers. *CoRR* abs/1912.00818 (2019).

[5] Sameer Bibikar, Haris Vikalo, Zhangyang Wang, and Xiaohan Chen. 2022. Federated dynamic sparse training: Computing less, communicating less, yet learning better. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, Vancouver, British Columbia, Canada (Virtual Conference), 6080–6088.

[6] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. Leaf: a benchmark for federated settings. *CoRR* abs/1812.01097 (2018).

[7] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting shared representations for personalized federated learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, Virtual Event, 2089–2099.

[8] Rong Dai, Li Shen, Fengxiang He, Xinmei Tian, and Dacheng Tao. 2022. Dispfl: Towards communication-efficient personalized federated learning via decentralized sparse training. In *Proceedings of the International Conference on Machine Learning*. PMLR, Baltimore, Maryland, USA, 4587–4604.

[9] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. 2020. Adaptive personalized federated learning. *CoRR* abs/2003.13461 (2020).

[10] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the lottery: Making all tickets winners. In *Proceedings of the International Conference on Machine Learning*. PMLR, Virtual Event, 2943–2952.

[11] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 3557–3568.

[12] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 19586–19597.

[13] Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised Learning by entropy minimization. In *Advances in Neural Information Processing Systems*. Vancouver, British Columbia, Canada, 529–536.

[14] Dan Hendrycks and Thomas G. Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. In *Proceedings of the International Conference on Learning Representations*. New Orleans, LA, USA.

[15] Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of the International Conference on Learning Representations*. Toulon, France.

[16] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *CoRR* abs/1909.06335 (2019).

[17] Yihan Jiang, Jakub Konečný, Keith Rush, and Sreeram Kannan. 2019. Improving federated learning personalization via model agnostic meta learning. *CoRR* abs/1909.12488 (2019).

[18] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. 2022. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[19] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, et al. 2021. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210.

[20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, Virtual Event, 5132–5143.

[21] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: strategies for improving communication efficiency. *CoRR* abs/1610.05492 (2016).

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

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

[24] David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. 2019. Federated learning for keyword spotting. In *IEEE international conference on acoustics, speech and signal processing*. IEEE, 6341–6345.

[25] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. 2021. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *Proceedings of the Annual International Conference on Mobile Computing and Networking*. ACM, New Orleans, Louisiana, USA, 420–437.

[26] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. 2021. Fedmask: joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*. ACM, Coimbra, Portugal, 42–55.

[27] Chenglin Li, Di Niu, Bei Jiang, Xiao Zuo, and Jianming Yang. 2021. Meta-har: Federated representation learning for human activity recognition. In *Proceedings of the Web Conference 2021*. 912–922.

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

[29] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org.

[30] Paul Pu Liang, Terrance Liu, Ziyin Liu, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Think locally, act globally: federated learning with local and global Representations. *CoRR* abs/2001.01523 (2020).

[31] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 2351–2363.

[32] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. 2021. Do we actually need dense over-parameterization? In-time over-parameterization in sparse training. In *Proceedings of the International Conference on Machine Learning*. PMLR, Virtual Event, 6989–7000.

[33] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[34] Hesham Mostafa and Xin Wang. 2019. Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *Proceedings of the International Conference on Machine Learning*. PMLR, Long Beach, California, USA, 4646–4655.

[35] Xinchi Qiu, Javier Fernández-Marqués, Pedro Porto Buarque de Gusmão, Yan Gao, Titouan Parcollet, and Nicholas Donald Lane. 2022. Zerofl: efficient on-device training for federated learning with local sparsity. In *Proceedings of the International Conference on Learning Representations*.

[36] Md Aamir Raihan and Tor Aamodt. 2020. Sparse weight activation training. *Advances in Neural Information Processing Systems*, 15625–15638.

[37] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. 2020. FetchSGD: communication-efficient federated learning with sketching. In *Proceedings of the International Conference on Machine Learning*. PMLR, Virtual Event, 8253–8265.

[38] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2021. Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems* 32, 8 (2021), 3710–3722.

[39] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. 2017. Federated multi-task learning. In *Advances in Neural Information Processing Systems*. Long Beach, CA, USA, 4424–4434.

[40] Benyuan Sun, Hongxing Huo, Yi Yang, and Bo Bai. 2021. Partialfed: Cross-domain personalized federated learning via partial initialization. *Advances in Neural Information Processing Systems*, 23309–23320.

[41] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the International Conference on Machine Learning*. PMLR, Virtual Event, 9229–9248.

[42] Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 21394–21405.

[43] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2022. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[44] Florian Tramèr and Dan Boneh. 2019. Adversarial training and robustness for multiple perturbations. *Advances in Neural Information Processing Systems*, 5858–5868.

[45] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris S. Papailiopoulos, and Yasaman Khazaeni. 2020. Federated learning with matched averaging. In *Proceedings of the International Conference on Learning Representations*.

[46] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. 2019. Federated evaluation of on-device personalization. *CoRR* abs/1910.10252 (2019).

[47] Xinyi Wang, Yulia Tsvetkov, Sebastian Ruder, and Graham Neubig. 2021. Efficient test time adapter ensembling for low-resource language varieties. In *Findings of the Association for Computational Linguistics: EMNLP*. Association for Computational Linguistics, Virtual Event, 730–737.
[48] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: concept and applications. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (2019), 1–19.
[49] Fuxun Yu, Weishan Zhang, Zhuwei Qin, Zirui Xu, Di Wang, Chenchen Liu, Zhi Tian, and Xiang Chen. 2021. Fed2: Feature-aligned federated learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2066–2074.
[50] Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. 2020. Salvaging federated learning by local adaptation. *CoRR* abs/2002.04758 (2020).
[51] Honglin Yuan, Warren Richard Morningstar, Lin Ning, and Karan Singhal. 2022. What do we mean by generalization in federated learning?. In *Proceedings of the International Conference on Learning Representations*.
[52] Marvin Zhang, Sergey Levine, and Chelsea Finn. 2021. MEMO: Test time robustness via adaptation and augmentation. *CoRR* abs/2110.09506 (2021).
[53] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. 2021. Personalized federated learning with first order model optimization. In *Proceedings of the International Conference on Learning Representations*.

A APPENDIX

A.1 Additional Experiment Details

A.1.1 Compared Methods. The compared methods include:

- **Local**: Each client trains its model only with its local data.
- **FedAvg** [33]: It is the first GFL algorithm that learns a global model by weighted averaging of the local models.
- **FedProx** [29]: It is another GFL algorithm that adds a proximal regularization term to FedAvg.
- **FedDST** [5]: It is the state-of-the-art sparse FL algorithm that learns a sparse global model for all clients.
- **FedAvg+FT** [46]. It fine-tunes the global model obtained by FedAvg for personalization.
- **LG-FedAvg** [30]: It is a PFL scheme that shares the weights of the global head while learning different local feature extractors. It also improves communication efficiency.
- **FedPer** [4]: It is a layer-level weight sharing PFL algorithm that allows clients to share a global feature extractor while learning different local heads. It performs local updates on the feature extractor and local heads simultaneously.
- **FedRep** [7]: It is similar to FedPer in terms of weight sharing, but conducts local updates on the feature extractor and local heads separately.
- **Ditto** [28]: It additionally trains a global model to regularize the local updates of the personalized model for fairness among clients and robustness against attacks.
- **APFL** [9]: It trains both the global and the personalized models on each client, and adaptively tunes the interpolation rate between the global and personalized models' weights for adaptive personalization.
- **DM-PFL**: It uses the personalized model θ_c learned by our dual-masked sparse training algorithm for inference.
- **DM-PFL+**: It is the proposed method with adaptive inference on the global and the personalized models θ_g and θ_c.

A.1.2 Federation Configuration. We set the number of clients $C = 100$ for MNIST/CIFAR10, $C = 30$ for HAR, and $C = 500$ for FEMNIST (clients with only few samples are excluded). The join ratio of clients is set to 0.1 for all datasets. We fix the training rounds $T = 300$.

A.1.3 Datasets Partition. In the Dirichlet distribution-based setting, we use the parameter α to determine the label distribution and data quantity skew of each client. Specifically, a C-dimensional vector is drawn from $\mathrm{Dir}(\alpha)$ for each class label, and the data belonging to that class has been assigned proportionally to each client based on the drawn vector.

In the pathological distribution setting, each client is provided with samples from only two classes. The data quantity for each client is sampled using a uniform distribution, such that, in terms of data quantity, 90% of the clients hold half of the total samples while the remaining 10% of clients hold the other half.

For the HAR and FEMNIST datasets, which represent realistic non-IID settings, the label and feature distributions naturally differ among different smartphone users and writers, respectively. The number of data samples for each client also varies naturally.

In all the aforementioned settings, each client's data is randomly split into a training set and a test set, with a fraction of 0.25 being used for the test set.

A.1.4 Hyperparameters. Learning rate η are tuned in $\{0.1, 0.05, 0.01\}$. We set weight decay to 0.0005 and learning rate decay to 0.99 if can improve performance, otherwise are set to zero. For FEMNIST, MNIST, and CIFAR10 the local batch size is 64 and for HAR it is 10. For FEMNIST, MNIST, and CIFAR10 the number of local steps is $E = 5$, and for HAR it is $E = 3$.

For sparsity-based methods FedDST and DM-PFL, model sparsity is fixed to 0.5 if not specified.

For methods that require additional epochs to train the global and local models separately like APFL [9], FedRep [7], and Ditto [28], the additional training steps is of the same number as local training steps.

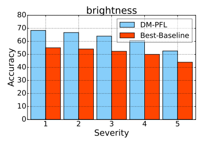
For baseline methods that require specific additional hyperparameters, we follow the setting below.

- FedProx [29]: Proximal regularization term is set to $\mu = 0.05$;
- FedDST [5]: Mask readjustment ratio is set to $\alpha_s = 0.01$, mask readjust interval is set to 10 by default;
- APFL [9]: The initial interpolation rate is set to $\alpha = 0.5$;
- Ditto [28]: The regularization term between the global and personalized model is set to $\lambda = 0.1$.

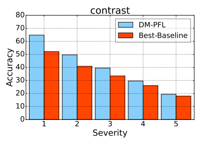
A.1.5 Model configurations. The models we use are lightweight for low-resource devices and are not over-parameterized specifically. We use similar model configurations as [33] for MNIST, FEMNIST, CIFAR10 datasets. For the HAR dataset, we employ DNN as [25].

Table 5: Accuracy with different degrees of shift on HAR.

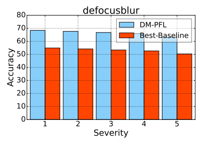
Type	Methods	w/o shift	w/ 20% shift	w/ 40% shift	w/ 60% shift	w/ 80% shift	w/ 100% shift	Average
Local	Local	91.95±16.31	89.00±16.65	85.55±16.94	82.28±17.27	79.17±17.58	76.02±17.90	83.99±17.11
GFL	FedAvg [33]	95.34±5.32	95.33±5.25	95.33±5.19	95.33±5.13	95.33±5.16	95.33±5.36	95.33±5.24
	FedProx [29]	95.12±5.44	95.11±5.42	95.10±5.41	95.09±5.39	95.08±5.38	95.07±5.36	95.10±5.40
	FedDST [5]	95.05±5.01	94.84±5.15	94.66±5.26	94.45±5.39	94.26±5.52	94.06±5.64	94.55±5.33
PFL	FedAvg+FT [46]	97.38±5.53	96.43±5.84	95.60±6.12	94.66±6.4	93.75±6.68	92.84±6.97	95.11±6.26
	LG-FedAvg [30]	98.16±9.12	94.67±9.56	91.03±9.99	87.41±10.4	83.71±10.8	80.12±11.23	89.18±10.18
	FedPer [4]	98.07±3.54	97.37±3.94	96.64±4.32	95.94±4.72	95.24±5.11	94.52±5.50	96.30±4.52
	FedRep [7]	97.91±3.11	96.29±4.15	94.84±5.16	93.19±6.19	91.67±7.21	90.14±8.22	94.01±5.67
	Ditto [28]	97.52±4.03	96.91±4.46	96.32±4.89	95.70±5.31	95.12±5.75	94.53±6.17	96.02 ± 5.10
	APFL [9]	95.51±5.62	93.94±6.55	92.21±7.45	90.69±8.35	89.10±9.28	87.47±10.19	92.29±7.45
Ours	DM-PFL	**98.41±2.72**	**97.78±3.00**	**97.16±3.27**	**96.58±3.52**	95.98±3.79	95.37±4.05	**96.88±3.39**
	DM-PFL+	97.93±3.46	97.44±3.72	96.98±3.99	96.51±4.24	**96.05±4.51**	**95.58±4.76**	96.75±4.11



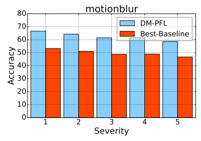
(a) brightness



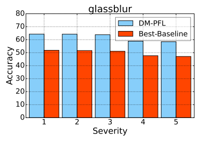
(b) contrast



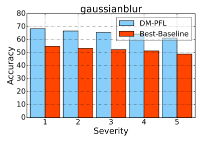
(c) defocus



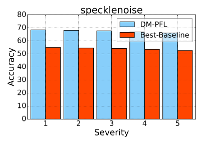
(d) motionblur



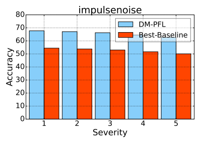
(e) glassblur



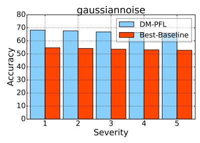
(f) gaussianblur



(g) specklenoise



(h) impulsenoise



(i) gaussiannoise

Figure 7: More results under different kinds of shifts.

A.1.6 Implementation. All experiments are implemented with PyTorch 1.10 and conducted on a workstation with AMD Ryzen 9 5950X 16-Core Processor CPU, NVIDIA GeForce RTX 3090 GPU.

A.2 More Experiment Results

A.2.1 Different Degrees of Shifts on HAR/FEMNIST. We also report the results under different degrees of shifts on HAR and FEMNIST in Table 5 and Table 6. On both datasets, we consistently outperform all PFL baselines. Our methods might not always outperform GFL methods when with extreme degrees of shift because the shift-robustness of our methods depends on the global model θ_g learned, which might not always have a significant advantage over the global models obtained by the GFL methods. Nevertheless, such a result does not affect our conclusion and instead illustrates the rationality of hitchhiking the GFL to improve the shift-robustness of PFL.

A.2.2 Different Kinds of Shifts on CIFAR10-C. We report more results under different kinds of shifts with different levels of severity in Fig. 7. We can observe that the results here are similar to the results shown in Fig. 4.

A.2.3 Different Non-IID settings on MNIST. We report the test accuracy on MNIST dataet in different non-IID settings in Table 7. The results on MNIST are similar to results shown in Table 2.

A.3 Further Discussions

Due to the length limitations of the paper, certain contents and discussions were not included. We will provide further discussions in https://github.com/garyzhang99/DM-PFL.

Table 6: Accuracy with different degrees of shift on FEMNIST.

Type	Methods	w/o shift	w/ 20% shift	w/ 40% shift	w/ 60% shift	w/ 80% shift	w/ 100% shift	Average
Local	Local	91.19±9.93	84.88±12.02	78.42±14.07	72.38±16.1	65.82±18.25	59.63±20.39	75.39±15.13
GFL	FedAvg [33]	98.33±4.99	98.33±5.01	98.32±5.03	98.32±5.04	**98.31±5.06**	**98.31±5.08**	98.32±5.04
	FedProx [29]	98.21±4.73	98.21±4.72	98.21±5.32	98.21±5.32	98.20±5.52	98.20±5.67	98.21±5.21
	FedDST [5]	98.16±5.49	98.15±5.34	98.14±5.19	98.12±5.05	98.11±4.89	98.10±4.75	98.13±5.12
PFL	FedAvg+FT [46]	98.85±5.23	98.45±6.14	98.07±7.03	97.66±7.89	97.26±8.84	96.86±9.76	97.86±7.48
	LG-FedAvg [30]	93.15±8.86	87.03±11.02	80.99±13.27	74.61±15.47	68.57±17.55	62.44±19.70	77.80±14.31
	FedPer [4]	98.75±6.78	98.49±6.54	98.24±6.31	97.98±6.09	97.72±5.85	97.44±5.61	98.10±6.20
	FedRep [7]	96.49±6.42	94.01±7.91	91.62±9.44	89.10±10.89	86.65±12.50	84.16±13.97	90.34±10.19
	Ditto [28]	98.71±4.48	98.32±5.05	97.92±5.60	97.52±6.19	97.12±6.78	96.72±7.31	97.72±5.90
	APFL [9]	98.49±3.76	98.11±4.60	97.72±5.47	97.33±6.31	96.96±7.19	96.55±8.05	97.53±5.90
Ours	DM-PFL	**99.04±3.80**	**98.80±4.09**	**98.57±4.39**	**98.34±4.68**	98.10±4.97	97.86±5.28	**98.45±4.54**
	DM-PFL+	98.62±4.63	98.52±4.71	98.42±4.79	98.31±4.86	98.20±4.93	98.10±5.01	98.36±4.82

Table 7: Accuracy with and w/o shift on MNIST in Dir(0.3), Dir(0.1) and pathological non-IID setting.

Non-IID	Dir(0.3)		Dir(0.1)		Pathological	
Methods	w/o	w/ 100%	w/o	w/ 100%	w/o	w/ 100%
FedAvg+FT [46]	98.91±3.64	97.05±4.36	99.11±4.75	92.93±11.37	99.74±0.56	85.26±13.26
LG-FedAvg [30]	97.01±2.71	60.18±27.16	98.43±2.86	34.96±32.48	99.05±6.43	19.73±39.47
FedPer [4]	99.08±1.39	69.06±27.57	99.17±2.85	52.81±33.85	99.72±0.55	25.65±36.89
FedRep [7]	99.10±1.22	75.58±23.08	99.28±3.61	50.10±32.78	99.61±0.76	19.90±39.81
Ditto [28]	99.11±1.05	96.87±2.32	99.20±1.21	93.86±13.58	99.69±0.61	90.70±14.21
APFL [9]	99.22±1.14	97.14±2.52	99.35±1.10	95.47±4.82	99.70±0.61	68.93±25.97
DM-PFL	**99.24±1.17**	97.76±2.12	**99.37±1.01**	96.18±3.39	**99.78±0.72**	93.80±8.31
DM-PFL+	99.05±1.41	**98.22±1.96**	99.10±1.62	**97.04±2.55**	99.30±0.75	**96.17±4.71**