BRIDGING GENERAL AND PERSONALIZED FEDERATED LEARNING THROUGH SELECTIVE MODEL INTEGRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Personalized federated learning (PFL) achieves high performance by assuming clients only meet test data locally, which does not meet many generic federated learning (GFL) scenarios. In this work, we theoretically show that PMs can be used to enhance GFL with a new learning problem named Selective FL (SFL), which involves optimizing PFL and model selection. However, storing and selecting whole models requires impractical computation and communication costs. To practically solve SFL, inspired by model components that attempt to edit a sub-model for specific purposes, we design an efficient and effective framework named *Hot-Pluggable Federated Learning* (`HPFL`). Specifically, clients individually train personalized *plug-in modules* based on a shared backbone, and upload them with a *plug-in marker* on the server *modular store*. In inference stage, an accurate selection algorithm allows clients to identify and retrieve suitable plug-in modules from the modular store to enhance their generalization performance on the target data distribution. Furthermore, we provide differential privacy protection during the selection with theoretical guarantee. Our comprehensive experiments and ablation studies demonstrate that `HPFL` significantly outperforms state-of-the-art GFL and PFL algorithms. Additionally, we empirically show `HPFL`'s remarkable potential to resolve other practical FL problems such as continual federated learning and discuss its possible applications in one-shot FL, anarchic FL, and FL plug-in market. Our work is the first attempt towards improving GFL performance through a selecting mechanism with personalized plug-ins.

1 INTRODUCTION

The performance of generic federated learning (GFL) (Brendan McMahan et al., 2016) suffers from data heterogeneity (Brendan McMahan et al., 2016; Kairouz et al., 2019; Li et al., 2020b; Karimireddy et al., 2019; Tang et al., 2022b; Chen & Chao, 2021), where clients have different data distributions. Personalized federated learning (Smith et al., 2017; Collins et al., 2021; Chen & Chao, 2021) (PFL) assumes that clients only need to inference on local test data, which has similar distributions to local training datasets. To this end, PFL prioritize fitting on local datasets while absorbing knowledge from the global training data. Due to this property, PFL gets rid of data heterogeneity, as the training convergence is not severely disturbed by the client drift (Li et al., 2020b; Karimireddy et al., 2019; Kairouz et al., 2021; Chen & Chao, 2021).

Table 1: Test accuracy on GFL and PFL of personalized models, with ResNet-18 and CIFAR-10.

Algorithm	FedAvg		FedPer		FedRep		FedRoD	
Problem	GFL	PFL	GFL	PFL	GFL	PFL	GFL	PFL
Accuracy	81.5	92.5	74.1	95.8	85.1	95.6	85.3	94.3

However, in real-world scenarios, FL users may encounter test data different from local training data (Liu et al., 2020; Luo et al., 2019; Hsu et al., 2020; Tang et al., 2022b), but which may appear in other training data. For example, when one traveling abroad, the personal map app might recommend entirely different restaurants from their residence. In such situation, models trained on local restaurant and personal data can make better recommendations [1]. In GFL, clients encounter test data of others, instead of only their own test data as in PFL (Smith et al., 2017; Collins et al.,

[1]More real-world examples of GFL-PM problems are provided in Appendix F.1.

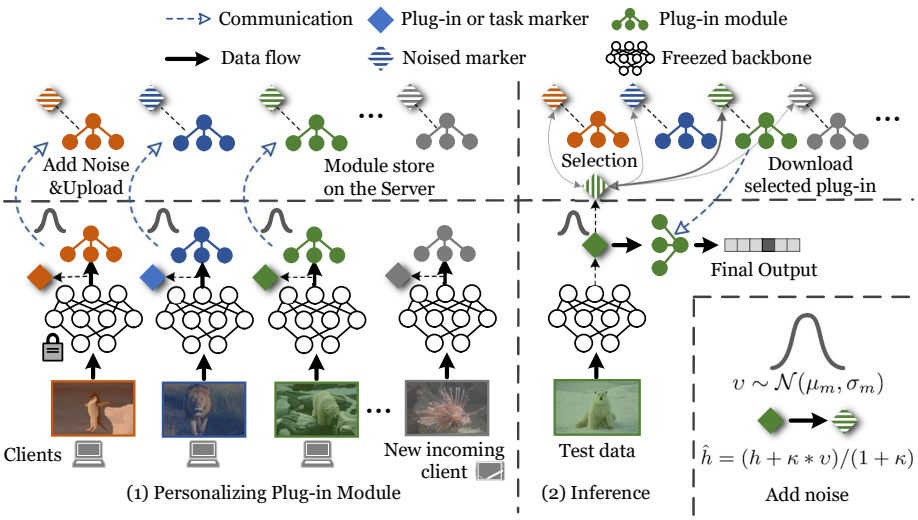


Figure 1: The framework of HPFL.

2021; Chen & Chao, 2021). In such realistic cases, PFL algorithms lose their general performance, as they prioritize fitting local datasets with the personalized models. As Table 1 shows, advanced PFL algorithms FedPer (Arivazhagan et al., 2019b), FedRep (Collins et al., 2021) and FedRoD (Chen & Chao, 2021) perform well in PFL, but performance collapses in GFL, where personalized clients encounter all test data. This performance gap motivates us following fundamental questions:

> *Whether a global model (GM) is compulsorily needed in GFL? Is it possible to enhance GFL with personalized models (PMs) trained in PFL?*

Considering that PMs are already solutions of PFL, intuitively, we formulate a new learning problem called Selective FL (SFL) bridging GFL and PFL (Equation 5), and prove the solution of SFL can achieve better performance than solution of GFL. Both GFL and PFL can be seen as special cases of SFL, whose core idea is to select suitable PMs for inference on clients according to incoming test data. However, the naive solution to SFL leads to privacy concerns, large system overheads and poor scalability. To this end, inspired by model components (Shah et al., 2024; Olsson et al., 2022), we propose a general and effective framework named *Hot-Pluggable Federated Learning* (HPFL) to practically solve SFL.

As shown in Figure 1, HPFL splits the model into two parts: a backbone (also called feature extractor) and a plug-in module. The backbone can be trained using any FL algorithm or initialized as a pre-trained backbone. Clients train plug-ins based on local datasets and upload them with the according *plug-in markers* to the server store. During inference, test data passes the backbone, and a suitable plug-in is selected to complete the inference. There are two ways to implement retrieving plug-ins in HPFL, α: Clients upload *task markers* to the server and select the appropriate module; β: Clients download all *plug-in markers* to select. α is suitable for situations where clients have limited computation ability, as it selects and completes final inference on the server; While β reduces the computation burden on the server. To protect the privacy, we provide differential privacy protection on communicated features during the selection with theoretical guarantee.

Our contributions are summarized as follows:

- We identify a substantial gap between GFL and PFL, and formulate a new problem SFL to bridge them together to address this performance gap (Section 2). As far as we know, this is the first work that enhances GFL through learning, sharing and selecting plug-ins, instead of classic paradigm with one single model.
- We propose a general, efficient and effective framework HPFL, which practically solves SFL (Section 3). And we add noise on communicated markers to provide differential privacy protection with theoretical guarantee (Section 3.4).
- We conduct comprehensive experiments and ablation studies on four datasets and three neural networks to demonstrate the effectiveness of HPFL (Section 5).

- We show the remarkable potential of HPFL in federated continual learning (Section 5.4) and discuss HPFL's possible applications in one-shot FL, anarchic FL and FL plug-in market (Section 6).

2 SELECTIVE FL: IMPLEMENTING GENERIC FL FROM PERSONALIZED FL

2.1 GENERIC FL

The GFL aims to make M clients collaboratively learn a global model parameterized as θ. Each client has its local data distribution $\mathcal{D}_m$. Thus, the local objective function $\mathcal{L}_m(\theta)$ on client m is also different. The global optimization object of GFL is defined as (Karimireddy et al., 2019; Woodworth et al., 2020; Tang et al., 2022b):

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}_G(\theta) = \sum_{m=1}^{M} p_m \mathcal{L}_m(\theta) = \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(f(\theta, \xi_m), \xi_m), \tag{1}$$

where $\xi_m \sim \mathcal{D}_m$ is the data sampled from $\mathcal{D}_m$, $f(\theta, \xi_m)$ is the prediction, d is the number of model parameters, $p_m > 0$ and $\sum_{m=1}^{M} p_m = 1$. Usually, $p_m = \frac{n_m}{N}$, where n_m denotes the number of client m's samples and $N = \sum_{m=1}^{M} n_m$. ***GM*** refers to the model obtained from optimizing GFL.

2.2 PERSONALIZED FL

Different from the object function of GFL, the PFL aims to learn multiple personalized models which fit well on different datasets individually: (Li & Wang, 2019; Chen & Chao, 2021; Li et al., 2021c):

$$\min_{\Omega, \theta_1, ..., \theta_M} \mathcal{L}_P(\Omega, \theta_1, ..., \theta_M) = \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(f(\theta_m, \xi_m), \xi_m) + \mathcal{R}(\Omega, \theta_1, ..., \theta_M), \tag{2}$$

where $\mathcal{R}$ is a regularizer (Chen & Chao, 2021) that varies with different algorithms, Ω is used to collaborate clients. We call each obtained locally personalized model θ_m as ***PM***.

2.3 WHEN PM MEETS GFL

In practice, PMs of clients may meet test data from other clients. Therefore, the learned PMs $\theta_1, ..., \theta_M$ need to perform well on all local data $\mathcal{D}_1, ..., \mathcal{D}_M$. We formulate the corresponding optimization goal with PMs in GFL scenario (GFL-PM) is:

$$\min_{\Omega, \theta_1, ..., \theta_M} \mathcal{L}_{P-G}(\Omega, \theta_1, ..., \theta_M) = \frac{1}{M} \sum_{i=1}^{M} \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(f(\theta_i, \xi_m), \xi_m) + \mathcal{R}(\Omega, \theta_1, ..., \theta_M), \tag{3}$$

which can be seen as a combination of GFL (Eq. 1) and PFL (Eq. 2): each PM is optimized to minimize the ℓ on all $\mathcal{D}_m$, $m \in 1, ..., M$. When not personalize θ_i on $\mathcal{D}_i$, Eq. 3 is reduced to GFL . And if each client's PM only needs to perform well on its local data, Eq. 3 turns into PFL.

One may think that there is no need to endow PMs with global generalization performance because one can optimize GFL to obtain a GM that generalizes well on all local datasets $\{\mathcal{D}_m, m \in \{1, ..., M\}\}$. However, theoretically and empirically, optimization of GM is difficult (Karimireddy et al., 2019; Woodworth et al., 2020) under communication cost and data heterogeneity constraints. Additionally, PMs' performance on local test data (PM on PFL) is usually significantly better than that of GM on global test data (GM on GFL) (Chen & Chao, 2021; Collins et al., 2021).

However, PMs after PFL usually cannot achieve better performance on unseen data distributions than GM in GFL (Chen & Chao, 2021). FedRoD (Chen & Chao, 2021) simultaneously optimizes $\mathcal{L}_G$ and $\mathcal{L}_P$, aiming to learn models that perform well both in GFL and PFL. This shares a similar spirit of optimizing GFL-PM problem (Eq. 3). However, PMs obtained from FedRoD remain a trade-off between minimizers of PFL and GFL. It is challenging to obtain model parameters that are both minimizers of GFL and PFL simultaneously. Next, we show that GFL-PM can be naturally transformed into a Selective FL (SFL) problem (Eq. 5), which involves optimizing PFL and a model selection problem (Eq. 6 in section 2.4). And the solution of SFL could serve as the minimizer of both GFL and PFL.

2.4 SELECTIVE FL

Successful personalization on client m means the following equation (Chen & Chao, 2021; Kairouz et al., 2019; Tan et al., 2022a).

$$\mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(f(\theta_i, \xi_m), \xi_m) \geq \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(f(\theta_m, \xi_m), \xi_m), i \neq m, \tag{4}$$

which means that for any client m, its PM outperforms than all PMs of other clients (Chen & Chao, 2021). Now, we are ready to state the following theorem (proof in Appendix B.1).

Theorem 2.1. *With Equation 4 and the PMs obtained from optimizing Equation 2 as:* $\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl} = \arg\min_{\Omega, \theta_1, ..., \theta_M} \mathcal{L}_P(\Omega, \theta_1, ..., \theta_M)$*, we have*

$$\mathcal{L}_{P-G}(\Omega, \theta_1, ..., \theta_M) \geq \mathcal{L}_P(\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}).$$

Remark 2.2. Theorem 2.1 implies that $\mathcal{L}_{P-G}$ is lower bounded by the minimum of $\mathcal{L}_P$.

Theorem 2.1 shows that the minimum of PFL is the lower bound of the GFL problem. Intuitively, this inspires us to think about exploit PMs trained in PFL to enhance GFL. Note that Equation 4 comes from the selective property of PFL, i.e. optimized PMs have the best performance on its local trained datasets. In light of this, the direct solution is to select trained PMs according to the test data. Thus, we propose the Selective FL (SFL) problem as the following:

$$\min_{\mathcal{H}} \mathcal{L}_S(\Theta, \mathcal{H}) = \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(\widehat{f}(\Theta, \xi_m, \mathcal{H}), \xi_m) \tag{5}$$

$$s.t.\ \widehat{f}(\Theta, \xi_m, \mathcal{H}) = f(\theta_s^{pfl}, \xi_m),\ s = S(\Theta, \xi_m, \mathcal{H}) \tag{6}$$

where $\Theta = \{\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}\} = \arg\min_{\Omega, \theta_1, ..., \theta_M} \mathcal{L}_P(\Omega, \theta_1, ..., \theta_M)$, S is called selection function that outputs the model index to select a model from the PMs based on the input ξ_m and the auxiliary information $\mathcal{H}$, which is exploited to select plug-in module, e.g. noisy feature for calculating distance metrics like Maximum Mean Discrepancy (MMD) will be illustrated in Section 3. Now, we can state the following theorem to illustrate that we can solve problem 3 by SFL (proof in Appendix B.2):

Theorem 2.3. *With Equation 4,* $\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl} = \arg\min_{\Omega, \theta_1, ..., \theta_M} \mathcal{L}_P(\Omega, \theta_1, ..., \theta_M)$ *and the* $\mathcal{H}^*$ *that guarantees* $\theta_m^{pfl} = \theta_{s(\Theta, \xi_m, \mathcal{H})}$*, we have*

$$\mathcal{L}_{P-G}(\Omega, \theta_1, ..., \theta_M) \geq \mathcal{L}_P(\Theta) = \mathcal{L}_S(\Theta, \mathcal{H}^*). \tag{7}$$

Remark 2.4. Theorem 2.3 shows that if we can accurately select θ_m^{pfl} out of all PMs when meeting data samples $\xi_m \sim \mathcal{D}_m$, the solution of SFL is also the lower bound of GFL-PM (Eq. 3). Therefore, solving SFL means that clients achieve performance in GFL as high as in PFL.

3 HPFL: HOT-PLUGGABLE FEDERATED LEARNING

In this section, we will first illustrate that directly selecting PM faces some fatal obstacles, including the large system overheads and privacy concerns in Section 3.1. Then, we introduce the design of `HPFL` in Section 3.2 with the Algorithm 1. Lastly, the selection method is introduced in Section 3.3.

3.1 PROBLEMS OF DIRECTLY SELECTING PM

With PMs $\Theta = \{\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}\} = \arg\min_{\Omega, \theta_1, ..., \theta_M} \mathcal{L}_P(\Omega, \theta_1, ..., \theta_M)$, an intuitive idea is to choose PM i based on the similarity between its local data $\mathcal{D}_i$ and the input data $\xi_m \sim \mathcal{D}_m$, thus the selection function 6 is implemented as: $s = S_\xi(\Theta, \xi_m, \mathcal{H}) = \arg\min_{i \in \mathcal{M}} d(\mathcal{D}_i, \xi_m)$, where $d(\cdot, \cdot)$ is any distance measure, then infer as $f(\theta_s^{pfl}, \xi_m)$. However, accessing data of other clients will cause **privacy concerns**. Moreover, communicating the whole model parameter θ_m is impractical due to **large system overhead**, especially for large language models and many clients.

3.2 DESIGN OF HPFL

Training the complete model θ. First, `HPFL` obtains a model θ that performs well (not as good as PMs in PFL) on all client datasets with any GFL algorithm. Thus, the model θ owns a backbone g that can extract general features from all client datasets. Due to the limited space, we chose the classic GFL algorithm FedAvg (McMahan et al., 2017) in our experiments. Future works can explore other advanced GFL algorithms to learn a better θ.

Training personalized plug-in module θ_m^ρ. Usually, after training, early layers of a model learn more general features than late layers (Yosinski et al., 2014; Asano et al., 2020), while late layers are more specific to some particular datasets. Inspired by this, HPFL decomposes the model as $f_m = \rho \circ g$ for each client m. As shown in Figure 1, g is a feature extractor, and ρ is a model head that outputs the final model prediction.

Clients can design a new personal plug-in module ρ_m (or say model head) different from the original head ρ, based on different computation characteristics. Then, with the frozen general feature extractor g, each client individually trains personalized ρ_m on local data $\mathcal{D}_m$ by optimizing:

$$\min_{\theta_m^\rho} \mathcal{L}_P(\theta_m) = \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(\rho_m \circ g(\xi_m), \xi_m). \tag{8}$$

Now, each client obtains a PM $f_m = \rho_m \circ g$, which enhances the generalization performance of $\rho_m \circ g$ on $\mathcal{D}_m$, which is usually better than original GM $f = \rho \circ g$ due to the personalization. Thus, the θ_m^{pfl} in SFL problem 5 can be constructed by θ^g and θ_m^ρ, inference becomes as $f(\theta_m^{pfl}, \xi_m) = \rho_m \circ g(\xi_m)$.

Algorithm 1 HPFL.

Initialization: server distributes the initial model θ^0 to all clients.
1. Training the complete model θ:
for each round $r = 0, 1, \cdots, R$ **do**
  server samples a set of clients $\mathcal{S}_r \subseteq \{1, ..., M\}$.
  server communicates θ^r to clients $m \in \mathcal{S}_r$.
  for each client $m \in \mathcal{S}^r$ **in parallel do**
    $\mathcal{C}_m^{r+1} \leftarrow$ LocalTraining$(\mathcal{D}_m, \theta^r)$ (GFL) .
  end for
  $\theta^{r+1} \leftarrow$ ServerUpdate$(\mathcal{C}_m^{r+1} | m \in \mathcal{S}^r)$ (GFL).
end for

2. Training personalized plug-in module θ_m^ρ:
for each client $m \in \mathcal{M}$ **in parallel do do**
  Clients share and freeze the θ^g,
  Clients design personalized θ_m^ρ.
  Training θ_m^ρ with object function 8 (PFL).
  Obtaining plug-in marker $\mathcal{H}_m$ (e.g. noised features explained in Section 3.3 in detail.) for **plug-in selection**.
  Upload θ_m^ρ and $\mathcal{H}_m$ to server.
end for
Server stores θ_m^ρ and $\mathcal{H}_m$.

HPFL Inference$(\theta^g, \mathcal{D}_{test})$**:**
$i \leftarrow$ SelectPlugIn$(\mathcal{D}_{test}, \theta^g, \mathcal{H})$.
Get output $\leftarrow \rho_i \circ g(\xi | \xi \sim \mathcal{D}_{test})$.

Inference and selecting plug-in module. In HPFL, we define some plug-in marker $\mathcal{H}_m$ as auxiliary information in Equation 5 that will be exploited to select plug-in module . When training θ_m^ρ, $\mathcal{H}_m$ are collected by clients and uploaded to the server. Note that as a general framework, HPFL does not limit the specific form of $\mathcal{H}_m$, which depends on the selection method. As the first attempt in this paradigm, We introduce a distance-based selection method in Section 3.3.

3.3 Selection Methods

Decomposing the model also avoids accessing the raw data $\xi_m \sim \mathcal{D}_m$. With the shared feature extractor g, we can select the ρ_m based on the intermediate features $h_m = g(\xi_m)$ rather than ξ_m itself to avoid leading raw data. Several studies have exploited the sharing of intermediate features to improve FL (He et al., 2020a; Lin et al., 2020a; Luo et al., 2021; Liang et al., 2020).

Distance based methods. Intuitively, now that each ρ_m is trained based on local features h_m, we only need to compare the similarity between h_m and $h_{test} = g(\xi_{test})$, where ξ_{test} is the data that needs testing. Now, the select problem turns from Equation 6 into:

$$S_{dist}(d, h_{test}, \hat{h}_1, ..., \hat{h}_M) = \arg\min_{m \in \mathcal{M}} d(\hat{h_m}, \hat{h}_{test}), \tag{9}$$

in which $\hat{h}_m$ and $\hat{h}_{test}$ are noised h_m and h_{test}, which are illustrated in the next section. In this selection method, the plug-in marker $\mathcal{H}_m = \hat{h}_m$. In HPFL, we utilize Maximum Mean Discrepancy (MMD) distance (Long et al., 2017) to measure the distance between plug-in markers and noised features of test data (task marker). We also exploit other distance measures like SVCCA (Raghu et al., 2017), CKA (Kornblith et al., 2019) and out-of-distribution confidence based selection methods and provide results in Appendix D.

3.4 Privacy Protection

Differential Privacy. In HPFL, plug-in markers (noised features of training data) and task markers (noised features of test data) are shared for selecting. To protect the privacy, following differential

privacy (DP) (Abadi et al., 2016; Balle & Wang, 2018; Wu et al., 2022; Yang et al., 2024), we add Gaussian noises as $\hat{h} = (h + \kappa * \upsilon)/(1 + \kappa)$ for both h_m and h_{test} where $\upsilon \sim \mathcal{N}(\mu_m, \sigma_m)$, in which $\hat{h}_m = (h_m + \kappa * \upsilon)/(1 + \kappa)$, where $\upsilon \sim \mathcal{N}(\mu_m, \sigma_m)$ is the noise to enhance privacy protection. The μ_m and σ_m are mean and variance of features h_m, κ is a coefficient controlling the relative magnitude between Gaussian noise and the features. The following theorem shows that the raw data is protected by our noise mechanism with (ϵ, δ)-DP. The detailed proof of Theorem 3.1 is shown in Appendix B.3.

Theorem 3.1. *For the procedure of obtaining and sharing markers $H(x_m) = (g(x_m) + \kappa * \upsilon)/(1 + \kappa)$, (ϵ, δ)-DP holds if the ϵ, δ conforms to any of the two conditions: (1) $\forall \epsilon, \delta \in (0, 1), \epsilon \geq O\left(\frac{\sqrt{2ln(1.25/(\kappa * \sigma_m))}}{\delta}\right)$; (2) $\forall 0 < \delta < 1/2 - e^{-3\epsilon}/\sqrt{2\pi\epsilon}, \epsilon \geq O(\frac{1}{2(\kappa * \sigma_m)^2})$.*

Model Inversion Attack. Besides theoretical analysis of DP protection `HPFL`, we also empirically verify the safety of sharing noised plug-in markers against the model inversion attack (Zhao et al., 2020). The failed reconstruction (in Appendix E.2) of raw data demonstrate that `HPFL` can defend model inversion attacks successfully.

4 Related Works

Generic Federated Learning. To address the data heterogeneity problem, FedProx (Li et al., 2020b) and MOON (Li et al., 2021b) propose to add regularization terms to mitigate the negative effect caused by data heterogeneity. Some methods explicitly or implicitly modify uploaded gradients to alleviate the gradient dissimilarity (Wang et al., 2020; Karimireddy et al., 2019; Tang et al., 2024). Some works share intermediate features (Jeong et al., 2018; Hao et al., 2021; Tang et al., 2024) or extra data (Tang et al., 2022b) to reduce client drift. There are also a bunch of work utilizing Knowledge Distillation (KD) (Hinton, 2015) such as FedDF (Lin et al., 2020b) and Fed-ET (Cho et al., 2022). Different from these works, we attempt to enhance the GFL performance with personalized models.

Personalized Federated Learning. PFL exploits personalizing client models to better suit local heterogeneous training data. Meta-learning (Fallah et al., 2020), knowledge distillation (Yu et al., 2020b; Li & Wang, 2019), adaptive regularization and model mixtures (Hanzely & Richtárik, 2020; Dinh et al., 2020; Deng et al., 2020) are used to enhance personal knowledge learning of models. Some works (Liang et al., 2020; Li et al., 2021a) allow clients to learn different PM structures. KNN-per (Marfoq et al., 2022) constructs PMs by replacing classifiers with non-parametric methods based on local datasets. FedRep (Collins et al., 2021) and FedRoD (Chen & Chao, 2021) propose to learn a global feature extractor and personalized classifiers. While FedRoD conducts inference with different classifiers, they are manually switched according to the prior knowledge about the source of test data, which is also impractical in real-world FL. All of these works only consider PMs in PFL settings, which is impractical in real-world FL, because clients might meet various test data. To address this problem, `HPFL` select suitable PMs according to the test data during the test time,

Test-time adaptation & domain adaptation methods in FL. Some works (Peng et al., 2019; Liu et al., 2021) focus on generalizing a federated model trained on multiple source domains to unseen target domains. FedTHE (Jiang & Lin, 2023) discussed test-time distribution shift of PMs, which is similar to but different from generalizing on global test data. These methods enhance federated models by better training schemes, which is orthogonal to our method. FedTHE & FedTHE+ (Jiang & Lin, 2023) discuss test-time distribution shift, which is similar to our problem setting. However, we narrow down the category of distribution shift to apply to the GFL setting and perform much better in our proposed circumstance, while their method mainly aims at dealing with unknown distribution shift. This is also the differences between our work and all methods applying TTA directly on local client training, therefore we only experimentally compare FedTHE and our work, as FedTHE significantly outperforms this type of works. Different from them, `HPFL` is the first FL framework that flexibly selects PMs for inference. Due to the limited space, we leave a more detailed discussion of the literature in Appendix A.

Table 2: Experiment results. Noisy coefficient κ=1. §: we focus more on GFL setting. Numbers in **ForestGreen** highlight highest values in GFL setting. *: FedAvg fine-tunes the whole model instead of partial model as in `HPFL`; FedSAM fine-tunes partial model as in `HPFL`; For these two methods, we only list the best performance in $E_p = 1$ & 10 and denote which epochs get the values in subscript. Plug-in selection is implemented with MMD. E_p denotes the epoch of fine-tuning. We provide error bars of our experiments in Table 11.

Clients	10 (sample 50% each round)						100 (5% each round)					
Non-IID	Dir(0.1)			Dir(0.05)			Dir(0.1)			Dir(0.05)		
Test Set	GFL§		PFL	GFL§		PFL	GFL§		PFL	GFL§		PFL
Method/Model	GM	PM	PM	GM	PM	PM	GM	PM	PM	GM	PM	PM
CIFAR-10												
FedAvg $E_p = 1$ & 10^*	81.5	-	92.8$_{(10)}$	62.4	-	96.1$_{(1)}$	73.6	-	91.6$_{(10)}$	47.9	-	93.4$_{(10)}$
FedPer	74.1	40.9	95.8	58.7	27.3	96.4	44.5	20.6	89.7	24.0	14.3	89.9
FedRoD	85.3	41.6	94.3	67.6	26.8	**96.9**	74.0	20.1	87.4	66.7	15.6	91.2
FedRep	85.1	51.3	95.6	**73.2**	30.2	85.3	66.5	27.4	89.3	59.2	20.4	89.1
PerFedMask $E_p = 5$	57.8	23.4	83.1	31.8	15.1	83.1	53.8	15.6	82.1	35.0	12.5	87.6
FedTHE	**86.2**	64.4	93.4	68.7	31.1	85.7	**75.0**	23.8	90.8	**66.9**	44.5	90.1
FedSAM $E_p = 1$ & 10^*	84.5	47.8$_{(1)}$	**96.0**	65.4	33.2$_{(1)}$	96.7$_{(10)}$	50.4	36.6$_{(1)}$	90.3$_{(10)}$	36.6	23.3$_{(1)}$	91.3$_{(10)}$
`HPFL` $E_p = 1$	81.5	95.4	95.4	62.4	96.0	96.0	73.6	**88.6**	94.9	47.9	**82.2**	93.9
`HPFL` $E_p = 10$	81.5	**95.7**	95.7	62.4	**96.3**	96.3	73.6	85.7	**95.7**	47.9	81.8	**95.3**
FMNIST												
FedAvg $E_p = 1$ & 10^*	86.0	-	98.2$_{(10)}$	**76.1**	-	99.1	**90.2**	-	97.8$_{(10)}$	**86.1**	-	98.4$_{(10)}$
FedPer	73.5	39.0	87.5	64.1	27.5	99.1	69.0	29.1	95.9	44.8	22.6	96.8
FedRoD	87.4	44.1	98.1	72.5	29.3	98.9	88.9	47.0	98.5	84.8	35.3	98.2
FedRep	87.0	43.0	97.5	74.7	39.5	98.0	88.2	72.4	97.9	84.4	59.6	98.3
PerFedMask $E_p = 5$	80.1	30.8	95.8	47.6	27.1	96.9	89.3	23.0	93.5	91.9	21.3	96.5
FedTHE	87.9	69.4	96.8	70.7	55.4	98.5	88.5	83.1	97.5	84.7	74.6	97.6
FedSAM $E_p = 1$ & 10^*	**89.3**	53.8$_{(1)}$	**98.5**	**77.2**	36.6$_{(1)}$	**99.4**	86.3	76.9$_{(1)}$	98.5$_{(10)}$	85.2	70.8$_{(1)}$	98.6$_{(10)}$
`HPFL`(MMD) $E_p = 1$	86.0	98.3	98.3	**76.1**	99.0	99.1	**90.2**	97.6	97.9	**86.1**	**81.4**	98.1
`HPFL`(MMD) $E_p = 10$	86.0	**98.4**	98.4	**76.1**	**99.1**	99.2	**90.2**	**97.9**	**98.8**	**86.1**	74.1	**98.7**
CIFAR-100												
FedAvg $E_p = 1$ & 10^*	69.1	-	79.5$_{(1)}$	65.3	-	80.9$_{(10)}$	**59.7**	-	66.7$_{(10)}$	47.9	-	75.1$_{(10)}$
FedRoD	69.4	32.5	77.2	**67.0**	23.6	78.5	52.8	11.2	55.4	**48.4**	7.3	66.3
FedRep	68.4	42.6	72.4	65.0	37.3	81.2	47.9	18.6	56.5	43.3	14.1	65.3
PerFedMask $E_p = 5$	47.3	7.0	40.0	49.4	7.0	39.7	41.7	3.8	35.8	42.1	3.6	35.2
FedTHE	**69.9**	24.8	74.9	**67.0**	18.3	79.6	53.3	14.8	61.2	**48.4**	13.2	70.3
FedSAM $E_p = 1$ & 10^*	68.4	57.4$_{(1)}$	85.6$_{(10)}$	64.1	43.0$_{(1)}$	**88.8**$_{(10)}$	41.3	27.3$_{(1)}$	71.1$_{(10)}$	34.8	18.4$_{(1)}$	77.3$_{(10)}$
`HPFL`(MMD) $E_p = 1$	68.6	**74.8**	83.3	65.3	**75.8**	87.4	**59.7**	**63.8**	81.2	47.9	**72.3**	84.1
`HPFL`(MMD) $E_p = 10$	68.6	72.2	**85.7**	65.3	73.9	**88.8**	**59.7**	55.7	**84.1**	47.9	70.9	**86.4**
Tiny-ImageNet-200												
FedAvg $E_p = 1$ & 10^*	56.5	-	69.5$_{(1)}$	54.9	-	75.3$_{(1)}$	47.2	-	67.5$_{(10)}$	42.1	-	68.9$_{(10)}$
FedPer	16.3	0.5	0.5	13.4	0.5	0.5	2.4	1.8	23.5	1.3	25.1	1.0
FedRoD	**57.5**	26.1	68.5	55.3	12.9	52.9	48.6	49.3	9.6	43.7	5.9	53.7
FedRep	56.1	28.7	55.4	54.5	31.8	69.6	46.4	18.6	52.5	40.3	12.8	58.6
PerFedMask $E_p = 5$	26.9	6.6	35.9	23.2	4.2	31.3	29.9	1.9	23.5	18.7	1.6	32.6
FedTHE	57.4	19.0	64.3	**55.5**	17.4	75.7	**49.1**	15.9	63.0	**44.5**	9.2	64.0
FedSAM $E_p = 1$ & 10^*	57.0	48.6$_{(1)}$	**75.1**$_{(10)}$	55.0	42.1$_{(1)}$	**78.2**$_{(10)}$	43.8	30.4$_{(1)}$	69.3$_{(10)}$	38.0	21.1$_{(10)}$	72.0
`HPFL`(MMD) $E_p = 1$	56.5	**51.9**	70.8	54.9	58.5	74.7	47.2	**50.7**	71.3	42.1	**47.1**	74.7
`HPFL`(MMD) $E_p = 10$	56.5	50.9	73.7	54.9	**58.8**	77.0	47.2	48.0	**73.2**	42.1	43.9	**76.5**

5 EXPERIMENTS

5.1 EXPERIMENT SETUP

Federated Datasets and Models. We conduct experiments on four commonly used image classification datasets in FL, including CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), Fashion-MNIST (Xiao et al., 2017), and Tiny-ImageNet (Le & Yang, 2015), with Latent Dirichlet Sampling (Dir) partition method (α = 0.1, 0.05) to simulate data heterogeneity following (He et al., 2020b; Li et al., 2021b; Luo et al., 2021; Tang et al., 2022b). We also evaluate the scalability of our proposed methods with different numbers of clients (M = 10, 100). We implement our algorithm and experiments based on the popular FL framework FedML (He et al., 2020b; Tang et al., 2023a). We train ResNet-18 (He et al., 2016), MobileNet (Howard et al., 2017) and a simple-CNN on all datasets. We run all algorithms for 1000 communication rounds, with 1 local epoch per round. Hyper-parameters and more details are explained in Appendix C.

Baselines and Metrics. We compare `HPFL` with GFL algorithms FedAvg (McMahan et al., 2017), FedSAM (Qu et al., 2022); advanced PFL algorithms including FedPer (Arivazhagan et al., 2019a), FedRep (Collins et al., 2021), PerFedMask (Setayesh et al., 2023); FedRoD (Chen & Chao, 2021) both for GFL and PFL; and a test-time adaption method FedTHE (Jiang & Lin, 2023). For all algorithms,

we validate the learned global model (GM) on the global test dataset (GFL), the personalized models (PM) on the personalized dataset (PFL), and PMs on GFL. More specifically, our new GFL-PM test setting is: for all clients, we randomly assign the local test data encountered by the clients with equal probability, i.e. $\forall i, j \in \{1, ..., M\}, Pr(\mathcal{D}_i^{test} = \mathcal{D}_j^{test,PFL}) = 1/M$, where $\mathcal{D}_j^{test,PFL}$ is test data IID with local training data on client j as local test data in PFL. More details about metrics are stated in Appendix C.

5.2 Experiment Results

`HPFL` consistently outperforms baselines in PM on GFL while comparable with classic PFL methods in classic personalized setting. As shown in Table 2, in **GFL-PM** setting, `HPFL` excels above all methods and most by a large margin, even surpasses accuracies in GFL-GM in most cases, while baselines perform poorly due to a lack of adaption to test data. We attribute the significant performance gain to adaptation to test data implemented with precise plug-in selection, which we discuss in Section 5.3. It is worth noting that FedTHE also attempts to adapt its model using test data, but only with the ensemble of its locally personalized and global classifier, thus ignores knowledge from other clients and underperforms `HPFL`. In terms of **GFL-GM accuracy**, `HPFL` actually shares the same GM with GFL backbone training method (in our case, i.e. FedAvg), so its GFL-GM accuracy is exactly the same as that of FedAvg and outperforms the classic PFL algorithms focusing on PFL performance like FedPer (Arivazhagan et al., 2019a). As for **PFL-PM accuracy**, our proposed method `HPFL` reports comparable results to the PFL baselines.

`HPFL` maintains fairly excellent robustness against non-IID degree. As shown in Table 2, the accuracy of `HPFL` is not only highest in GFL-PM, but also increases when the heterogeneity increases from $Dir(0.1)$ to $Dir(0.05)$ in a similar way as in PFL-PM in some cases. From this phenomenon, we infer that `HPFL` exploits local information from clients to ensemble a model in the form of plug-ins. The server holds these local information in the form of plug-ins instead of fusing these local knowledge in a single model, thus prevents the original local information from being corrupted in model aggregation as it occurs in highly heterogeneous data, and maintains a robustness against non-IID, which is a common issue in Federated Learning.

`HPFL` has excellent scalability in terms of performance in accuracy. `HPFL` adopts a one-client-one-plug method to better modify final inference models according to the data distribution of clients' local data. In this way, `HPFL` has inherent ability to allow more clients to come and go freely in the FL system. From Table 2, we observe that other PFL methods met extreme problems when dealing with the situation that the number of clients was larger ($\mathcal{M}$=100), with most of the accuracies lower than 30% on CIFAR-10, 20% on CIFAR-100. However, though with a little decay in accuracy, `HPFL` is still applicable in the situation where the system included larger number of clients.

Table 3: Results with different architectures.

Architecture	Mobilenet			Simple-CNN		
Method/Model	GM	PM	PM	GM	PM	PM
FedAvg	55.7	-	92.3	64.6	-	85.4
FedPer	53.7	10.0	10.0	44.1	27.6	85.5
FedRoD	76.3	36.1	92.3	67.1	28.8	83.5
FedRep	74.1	35.8	85.0	54.6	10.0	10.0
PerFedMask	13.0	19.0	76.4	31.5	10.0	50.5
FedTHE	76.3	45.4	82.7	67.1	45.1	70.9
`HPFL`	55.7	**92.8**	92.8	64.6	**87.8**	87.8

A generalized framework applicable to different model architecture. As a general FL framework, `HPFL` can be seamlessly applied to model architectures where parameter decoupling is available. We deploy it on three different model architectures (ResNet-18, MobileNet (Howard et al., 2017), and a simple-CNN structure whose architecture is the same as simple-CNN in (Tang et al., 2022b)), and `HPFL` outperforms baselines we use in the main experiment with all of the architectures, showing that `HPFL` can be extensively employed in different FL systems and improve their performance of GFL and adaptation ability to new clients. Results are in Table 3. Moreover, `HPFL` can exploit backbones trained with all kinds of GFL algorithms. An ablation study on GFL methods used to learn feature extractor of `HPFL` is demonstrated in Appendix D.6.

A win-win deal: Efforts to protect privacy is not contradictory to the performance of `HPFL`. In `HPFL`, clients share plug-in markers with the server, which may raise privacy concern. To protect clients from the risk of data breaches

Table 4: Accuracy of different κ

κ	0	1	10	100	1000
Accuracy	95.4	95.4	95.4	95.4	95.4

during communication or improper storage on the server, we add noise to the plug-in markers. However, we surprisingly found that noise will not damage the performance of HPFL as shown in Table 4. We attribute the robustness toward noise to robust selection method of HPFL, which we study later in Section 5.3. Discussions and experimental results about the privacy risk against HPFL are shown in Appendix E.

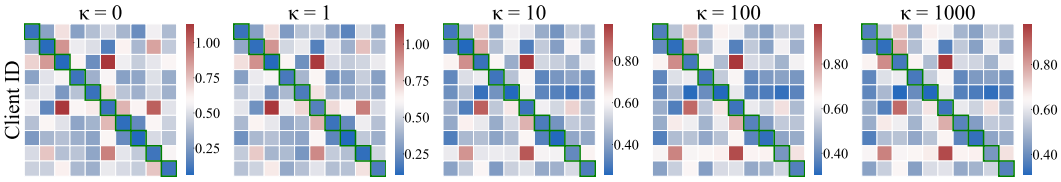


Figure 3: Selection score maps with different noise coefficients. Blocks with green anchor mean the corresponding client selects the plug-in and download it. Blocks with green anchor lying in diagonal indicate that clients choose plug-ins of themselves when met their own test data, which conforms to the aim of selection methods. X-axis represents Plug-in ID.

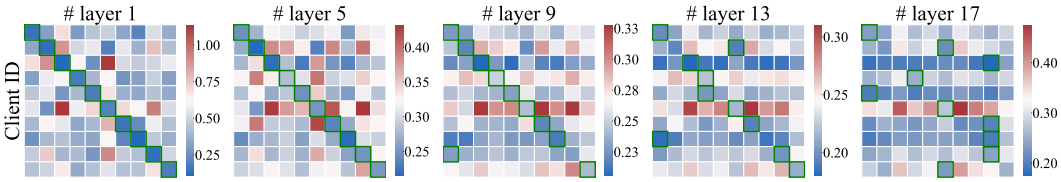


Figure 4: Selection score maps with different numbers of layers in ResNet-18 plug-ins own. X-axis represents Plug-in ID.

5.3 Selection Accuracy

The more flexible the models are, the better? As shown in Figure 2, the accuracy of HPFL continuously decreases with the increasing number of plug-in layers, we propose two possible reasons leading to the phenomenon: (1) local clients' samples are not sufficient for training big-scale plugs, resulting severe overfitting issue, and (2) The selection methods may not be suitable for markers from middle layers. However, according to Table 2, we believe that fine-tuning larger plug-ins does not lead to such a performance degradation, because FedAvg fine-tunes on the whole model without significant performance loss. Therefore, it is natural to give attention to the potential trouble large plug-ins may cause in plug-in selection. In Section 5.3, we conduct experiments to testify the speculation that the performance loss when increasing the plug-in layer is mainly due to the degradation of plug-in selection. Due to the page limit, we aim to provide an intuitive explanation in Appendix D.1.

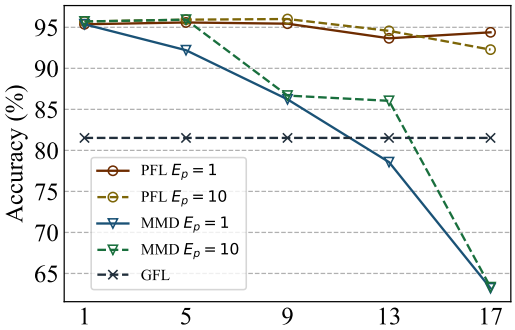


Figure 2: Accuracy with different numbers of plug-in layers. X-axis represents the number of layers in ResNet-18 for plug-ins.

Plug-in selection plays an important role in HPFL, so here we study how it gets affected by the magnitude of noise added on features and the number of plug-in layers. Experiments in this section are carried out with α=0.1, $\mathcal{M}$=10 on CIFAR-10 dataset, we include the results of additional configurations in Appendix D.3.

We observed the expected phenomenon conforming to our conjecture in Section 5.2 that it is harder for selection methods to correctly select plug-ins with more layers. With the increasing number of plug-in layers, the score map gradually changes. However, until it actually influences the result of selection, the performance of HPFL gets unaffected.

Observed from Figure 3, despite the slight variation in the heatmaps of MMD score with the noise coefficient, selecting plug-in with the lowest MMD score instead of combining plug-ins with MMD score adds robustness towards noise to HPFL. We shows the accuracies under different κ in Table 4.

5.4 Federated Continual Learning

Federated continual learning (FCL) (Yoon et al., 2021) is a new problem where clients join FL training after initial training. The trained model must retain previous dataset knowledge and perform well on data from newly arrived clients. HPFL can address the forgetting problem of FCL by preserving previous training knowledge in a personalized plug-in and providing it for client inference as shown in Table 5. It is an application of HPFL on the temporal scale, where clients collaboratively learn models that generalize well over time. We present more details about the experiment and discussion in Appendix D.5.

Table 5: Results of FCL

	GM	PM		GM	PM
Naive FCL	69.5	58.4	FCL under HPFL	62.2	**80.9**

6 Applications

Federated Continual Learning. As discussed in Section 5.4, HPFL effectively addresses the forgetting problem in FCL by preserving knowledge without loss and retrieving when needed.

One-shot FL. With an average backbone such as a pre-trained model, HPFL can train plug-ins in a single communication round, immediately proceeding to inference. This approach also accommodates new clients joining the FL system.

Anarchic FL. HPFL supports the dynamic in anarchic FL (Yang et al., 2022), where clients join and leave unpredictably. It operates without the need for immediate aggregation, thus allows clients to train and upload plug-ins asynchronously without disturbing server operations or model convergence with stale updates.

FL plug-in market. HPFL provides the possibility of constructing a more free and transparent model market, and customers can have better confidence knowing the plug-in they are purchasing is able to meet their requirements with a fair plug-in selection mechanism. Plug-in providers can obtain commercial benefits from this market.

7 Limitations

Scalability of Plug-In Selection. When the number of clients increase to thousands or more, selecting plug-in module will consume a lot of time and computing resources. The proposed plug-in selection methods in this paper need to communicate and select from $\mathcal{M}$ sets of plug-in ϕ. Future works may consider to cluster and merge M_c plug-in modules together, reducing the number of plug-ins as $\frac{M}{M_c}$. If we set $M_c = M$, the HPFL becomes the FedAvg of GFL. The challenge here is to design an appropriate aggregation method, which can make the merged plug-in performs well on according test datasets. Moreover, we can also think about skipping some plug-ins to reduce the computing overhead. This needs clients to provide more representative attributes of the plug-ins.

Accurate Plug-in Selection. In many experiments, our selection method choose optimal or nearly However, as an initial trial, our proposed plug-in selection methods select sub-optimal plug-ins in some circumstances as shown in Figure 4, 12 and 9 etc.. Future works may consider to design more accurate and robust selection methods.

8 Conclusion

In this paper, we explore how to improve the generalization performance when PMs meet test data from other clients. We formalize the SFL to bridge the GFL and PFL together. Then, We propose HPFL to practically solve the SFL. We verify the effectiveness and robustness of HPFL through comprehensive experiments. And we further experimentally verify the remarkable potential of HPFL to resolve other practical FL problems like FCL. Future work can consider to explore new plug-in selection methods, or applying HPFL into more FL related problems. Applying HPFL to foundation model is also an interesting future direction, especially with the Parameter-Efficient Fine-Tuning (PEFT) techniques, such as LoRA, adapter or prefix tuning.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019a. URL `http://arxiv.org/abs/1912.00818`.

Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019b. URL `http://arxiv.org/abs/1912.00818`.

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. In *ICLR*, 2020.

Borja Balle and Yu-Xiang Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pp. 394–403. PMLR, 2018.

Wenxuan Bao, Tianxin Wei, Haohan Wang, and Jingrui He. Adaptive test-time personalization for federated learning. *Advances in Neural Information Processing Systems*, 36, 2024.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv e-prints*, art. arXiv:1602.05629, February 2016.

Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. *arXiv preprint arXiv:2004.11791*, 2020.

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.

Hong-You Chen and Wei-Lun Chao. On bridging generic and personalized federated learning for image classification. In *International Conference on Learning Representations*, 2021.

Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitrios Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI) Main Track*, 2022.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2089–2099. PMLR, 18–24 Jul 2021.

Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning, 2020.

Canh T. Dinh, Nguyen H. Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes, 2020.

Xuefeng Du, Zhaoning Wang, Mu Cai, and Sharon Li. Towards unknown-aware learning with virtual outlier synthesis. In *ICLR*, 2022. URL `https://openreview.net/forum?id=TW7d65uYu5M`.

Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pp. 1–12. Springer, 2006.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 16937–16947. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251. PMLR, 2019.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 33:19586–19597, 2020.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. pp. 63–77. Springer-Verlag, 2005.

Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models, 2020.

Weituo Hao, Mostafa El-Khamy, Jungwon Lee, Jianyi Zhang, Kevin J Liang, Changyou Chen, and Lawrence Carin Duke. Towards fair federated learning with zero-shot data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3310–3319, 2021.

Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. In *Advances in Neural Information Processing Systems 34*, 2020a.

Chaoyang He, Songze Li, Jinhyun So, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pp. 76–92. Springer, 2020.

Rui Huang, Andrew Geng, and Yixuan Li. On the importance of gradients for detecting distributional shifts in the wild. In *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=fmiwLdJCmLS.

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. *NeurIPS*, 2018.

Liangze Jiang and Tao Lin. Test-time robust personalization for federated learning. *arXiv preprint arXiv:2205.10920*, 2022.

Liangze Jiang and Tao Lin. Test-time robust personalization for federated learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. *arXiv preprint arXiv:1910.06378*, 2019.

Julian Katz-Samuels, Julia B Nakhleh, Robert Nowak, and Yixuan Li. Training OOD detectors in their natural habitats. In *ICML*, Proceedings of Machine Learning Research. PMLR, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

John Langford, Alexander J. Smola, and Martin Zinkevich. Slow learners are fast. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, pp. 2331–2339, Red Hook, NY, USA, 2009. Curran Associates Inc. ISBN 9781615679119.

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015. URL https://api.semanticscholar.org/CorpusID:16664790.

Ang Li, Jingwei Sun, Binghui Wang, Lin Duan, Sicheng Li, Yiran Chen, and Hai Li. Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 68–79, 2021a. doi: 10.1145/3453142.3492909.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021b.

Tan Li, Linqi Song, and Christina Fragouli. Federated recommendation system via differential privacy. *arXiv preprint arXiv:2005.06670*, 2020a.

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*, volume 2, pp. 429–450, 2020b. URL https://proceedings.mlsys.org/paper/2020/file/38af86134b65d0f10fe33d30dd76442e-Paper.pdf.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *ICML*, 2021c.

Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *NeurIPS*, 2020a.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in neural information processing systems*, 33:2351–2363, 2020b.

Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space, 2021.

Yang Liu, Anbu Huang, Yun Luo, He Huang, Youzhi Liu, Yuanyuan Chen, Lican Feng, Tianjian Chen, Han Yu, and Qiang Yang. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 13172–13179, 2020.

Zelei Liu, Yuanyuan Chen, Han Yu, Yang Liu, and Lizhen Cui. Gtg-shapley: Efficient and accurate participant contribution evaluation in federated learning. *ACM Trans. Intell. Syst. Technol.*, 13(4), may 2022. ISSN 2157-6904.

Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pp. 2200–2207, 2013.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pp. 2208–2217. PMLR, 2017.

Jiahuan Luo, Xueyang Wu, Yun Luo, Anbu Huang, Yunfeng Huang, Yang Liu, and Qiang Yang. Real-world image datasets for federated learning. *arXiv preprint arXiv:1910.11089*, 2019.

Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

Othmane Marfoq, Giovanni Neglia, Richard Vidal, and Laetitia Kameni. Personalized federated learning through local memorization. In *International Conference on Machine Learning*, pp. 15070–15092. PMLR, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Kang Loon Ng, Zichen Chen, Zelei Liu, Han Yu, Yang Liu, and Qiang Yang. A multi-player game for studying federated learning incentive schemes. In *IJCAI International Joint Conference on Artificial Intelligence*, pp. 5279, 2020.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. Federated adversarial domain adaptation. In *International Conference on Learning Representations*, 2019.

Zhe Qu, Xingyu Li, Rui Duan, Yao Liu, Bo Tang, and Zhuo Lu. Generalized federated learning via sharpness aware minimization. In *International Conference on Machine Learning*, pp. 18250–18280. PMLR, 2022.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. *arXiv e-prints*, art. arXiv:1606.04671, June 2016.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8):3710–3722, 2020.

Mehdi Setayesh, Xiaoxiao Li, and Vincent W.S. Wong. Perfedmask: Personalized federated learning with optimized masking vectors. In *Proc. of International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, May 2023.

Harshay Shah, Andrew Ilyas, and Aleksander Madry. Decomposing and editing predictions by modeling model computation. *arXiv preprint arXiv:2404.11534*, 2024.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *ICML*, 2021.

MyungJae Shin, Chihoon Hwang, Joongheon Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Xor mixup: Privacy-preserving data augmentation for one-shot federated learning. *arXiv preprint arXiv:2006.05148*, 2020.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321, 2015.

Rachael Hwee Ling Sim, Yehong Zhang, Mun Choon Chan, and Bryan Kian Hsiang Low. Collaborative machine learning with incentive-aware model rewards. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *ICML*, Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022.

Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–17, 2022a.

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

Yue Tan, Chen Chen, Weiming Zhuang, Xin Dong, Lingjuan Lyu, and Guodong Long. Is heterogeneity notorious? taming heterogeneity to handle test-time shift in federated learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Zhenheng Tang, Shaohuai Shi, Bo Li, and Xiaowen Chu. Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication. *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–13, 2022a. doi: 10.1109/TPDS.2022.3230938.

Zhenheng Tang, Yonggang Zhang, Shaohuai Shi, Xin He, Bo Han, and Xiaowen Chu. Virtual homogeneity learning: Defending against data heterogeneity in federated learning. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 21111–21132. PMLR, 17–23 Jul 2022b.

Zhenheng Tang, Xiaowen Chu, Ryan Yide Ran, Sunwoo Lee, Shaohuai Shi, Yonggang Zhang, Yuxin Wang, Alex Qiaozhong Liang, Salman Avestimehr, and Chaoyang He. Fedml parrot: A scalable federated learning system via heterogeneity-aware scheduling on sequential and hierarchical training. *arXiv preprint arXiv:2303.01778*, 2023a.

Zhenheng Tang, Yuxin Wang, Xin He, Longteng Zhang, Xinglin Pan, Qiang Wang, Rongfei Zeng, Kaiyong Zhao, Shaohuai Shi, Bingsheng He, et al. Fusionai: Decentralized training and deploying llms with massive consumer-level gpus. *arXiv preprint arXiv:2309.01172*, 2023b.

Zhenheng Tang, Yonggang Zhang, Shaohuai Shi, Xinmei Tian, Tongliang Liu, Bo Han, and Xiaowen Chu. Fedimpro: Measuring and improving client update in federated learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the third ACM international workshop on edge systems, analytics and networking*, pp. 61–66, 2020.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7611–7623, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/564127c03caab942e503ee6f810f54fd-Paper.pdf`.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.

Blake E Woodworth, Kumar Kshitij Patel, and Nati Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33:6281–6292, 2020.

Q. Wu, K. He, and X. Chen. Personalized federated learning for intelligent iot applications: A cloud-edge based framework. *IEEE Open Journal of the Computer Society*, 1:35–44, 2020. doi: 10.1109/OJCS.2020.2993259.

Zhaomin Wu, Qinbin Li, and Bingsheng He. A coupled design of exploiting record similarity for practical vertical federated learning. *Advances in Neural Information Processing Systems*, 35: 21087–21100, 2022.

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

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous Federated Optimization. *arXiv e-prints*, pp. arXiv:1903.03934, March 2019.

Haibo Yang, Xin Zhang, Prashant Khanduri, and Jia Liu. Anarchic federated learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 25331–25363. PMLR, 17–23 Jul 2022.

Zhiqin Yang, Yonggang Zhang, Yu Zheng, Xinmei Tian, Hao Peng, Tongliang Liu, and Bo Han. Fedfed: Feature distillation against data heterogeneity in federated learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Sk7KsfW0-`.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12073–12086. PMLR, 18–24 Jul 2021.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014.

Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. A sustainable incentive scheme for federated learning. *IEEE Intelligent Systems*, 35(4): 58–69, 2020a. doi: 10.1109/MIS.2020.2987774.

Tao Yu, Eugene Bagdasaryan, and Vitaly Shmatikov. Salvaging federated learning by local adaptation. *arXiv preprint arXiv:2002.04758*, 2020b.

Rongfei Zeng, Chao Zeng, Xingwei Wang, Bo Li, and Xiaowen Chu. Incentive mechanisms in federated learning and game-theoretical approach. *IEEE Network*, pp. 1–7, 2022. doi: 10.1109/MNET.112.2100706.

Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? In *International Conference on Learning Representations*, 2020.

Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. In *ICML*. JMLR, 2017.

Yibo Zhou. Rethinking reconstruction autoencoder-based out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7379–7387, June 2022.

APPENDIX

A MORE RELATED WORK

A.1 GENERIC FEDERATED LEARNING

The convergence problem of FL with high non-IID data distribution has always been an important problem in improving the performance of models trained with FL. To resolve this problem, FedProx (Li et al., 2020b) and MOON (Li et al., 2021b) propose to add new model regularization terms to mitigate the client drift caused by data heterogeneity. There are also some methods modifying the uploaded gradient to alleviate the dissimilarity of gradients (Wang et al., 2020; Karimireddy et al., 2019; Tang et al., 2022a; 2023b). With a level of privacy protection, some works propose to share intermediate features (Jeong et al., 2018; Hao et al., 2021) or extra data (Tang et al., 2022b; Shin et al., 2020; Lin et al., 2020a) to reduce the gradient variance.

A.2 PERSONALIZED FEDERATED LEARNING

Different from the GFL methods that aim to directly reduce the gradient dissimilarity, PFL exploits the heterogeneous data to personalize client models to better suit the local training data.

Recently, several works have proposed to apply Model-Agnostic Meta-Learning (Finn et al., 2017) to Federated Learning for faster adaptation on local training data in clients. The Model-Agnostic Meta-Learning (Finn et al., 2017) (MAML) aims to meta-learn a global model, which will be broadcasted to different users to learn a local model adapted to different datasets. Per-FedAvg (Fallah et al., 2020) makes use of MAML to learn personalized models more efficiently. It first finds an initial global shared model with second-order gradient information, and then the global model is fine-tuned by local models with only several iterations to suit the local datasets.

Knowledge distillation is also used to promote efficient local adaptation of personalized models (Yu et al., 2020b). Specifically, a federated teacher model G_T and an adapted student model G_S are defined with the same structure. G_S is initialized with G_T, which has been trained through a common dataset shared across clients. And G_S is trained by local private datasets. However, in this method, the global model G_T won't get optimized as time goes on. It is more like a local fine-tuning technology rather than federated learning. Some works (Hanzely & Richtárik, 2020; Dinh et al., 2020; Deng et al., 2020) utilize some regularization and adaptive model mixture to learn personalized models. FedMD (Li & Wang, 2019) proposes a federated learning framework based on knowledge distillation using a shared dataset, on which clients transfer knowledge through mimicking the outputs of other client models. With knowledge distillation, it allows clients to independently design their own model architectures with their local private datasets.

In addition to the expected performance of personalized models, there are also works aiming at addressing the problems personalized models may meet when applied in reality. Ditto (Li et al., 2021c) adds the regularizer measuring the difference between personalized models and the global model into the objective functions to guarantee both the fairness and robustness of personalized models.

Apart from the usability of personalized models, the accessibility of personalized models is also a key consideration when it comes to real-world applications. Considering the situations where clients have heterogeneous environments like datasets, hardware, software, and the Internet, there are too many unpredictable situations in the real world blocking the access of personalized models. To solve these problems, some works (Wu et al., 2020; Li et al., 2021a) propose to allow clients to learn different personalized model structures. LotteryFL (Li et al., 2021a) proposes to let clients individually learn a lottery model, which is a subset of the global model. During the communication, these lottery models will be shared between servers and clients. Without the requirement of communicating a global model, this method can significantly reduce the communication cost in its training process. pFedHN (Shamsian et al., 2021) also makes clients learn a sub-model based on the global model.

Recently, there have also been many works exploring personalizing parts of models instead of the whole model to improve the performance of personalized models. LG-FedAvg (Liang et al., 2020) proposes to share the upper layers (model head) in the DNN and personalize the bottom layers (base model), which will not be averaged during the training. It utilizes personalized base models to output different local features in different clients, on which the global model head will be collaboratively trained through the FedAvg. Conversely, FedRep (Collins et al., 2021) proposes to learn a global feature extractor and personalized classifiers. FedRoD (Chen & Chao, 2021) proposes a two-predictor framework in which clients train different model heads to switch between GFL and PFL.

Different from their work, Our framework considers a more challenging FL setting, i.e. every client may meet OOD test data from other clients. Moreover, instead of improving the performance of the model itself, we consider more about how clients collaborate to handle the unpredictable test data.

A.3 Incentive Mechanism

The purpose of FL collaboration among clients is the improvement of model performance on the test data. Therefore, it is important to know how much performance gain can be obtained after FL collaboration (Ghorbani & Zou, 2019; Liu et al., 2022; Sim et al., 2020). Furthermore, there should be a well-designed incentive mechanism (Ng et al., 2020; Yu et al., 2020a; Zeng et al., 2022) that motivates clients to join FL. Our modular store essentially provides a market economy to let clients autonomously choose and download the needed plug module. The higher the generalization performance of the plug-in module, the more favorable it is. Therefore, the incentive mechanism of the modular store is naturally connected with the practical benefits of the plug-in module.

A.4 Federated Continual Learning

Continual learning (CL) (Kirkpatrick et al., 2017) is to learn different tasks sequentially. Some former tasks are inaccessible after training. Thus, when training subsequent tasks, the machine learning model may forget previous tasks. EWC (Kirkpatrick et al., 2017) finds the model parameters that are good for both previous and subsequent tasks using the Fisher Information Matrix. Progressive Neural Network approach (Rusu et al., 2016) is to increasingly construct the model during the training. Thus, the newly added parameters can learn the new tasks, while the old parameters can remember the old tasks. DEN (Yoon et al., 2018) dynamically decides the model capacity to learn a compact overlapping knowledge sharing among tasks.

Federated Continual Learning (FCL) (Yoon et al., 2021) is a new problem where, after FL training on some clients, there are some other clients that come and join the FL training. The trained model needs to avoid forgetting the previous dataset while performing well on the later dataset with data from newly arrived clients. We use a simple example to show that `HPFL` is naturally suitable to address the forgetting problem of FCL. Our plug-in can not only be seen as a personalized part of the model helping clients do inference on test data but also considered as a container preserving knowledge obtained from training. So it is natural to think we can store the knowledge in the previous dataset and access it whenever we are in need. In fact, it can be seen as an application of `HPFL` on the temporal scale. Most of the works in FL talk about many clients in a single period of time, i.e. Federated Learning in the spatial scale. FCL itself can be seen as a problem that happens at Federated Learning within the temporal scale: clients from different times collaboratively learn models that can generalize well on circumstances varied with time. We experimentally verified the potential of `HPFL` to address the forgetting problem of FCL in Section 5.4. Details about that experiment and more discussion are presented in Appendix D.5.

A.5 ASYNCHRONOUS FL

Asynchronous FL (Async-FL) (Xie et al., 2019) means to ease the constraint of the synchronous communication mechanism of classic federated optimization schemes (McMahan et al., 2017). In Async-FL, clients may download the global model from and return gradients to the server at different times. Thus, the server may receive a stale model update, causing unstable convergence. Such a staleness problem has long existed in the distributed machine learning area (Langford et al., 2009; Zheng et al., 2017). Stale updates are usually controlled by some staleness coefficients (Xie et al., 2019) or compensated by (Zheng et al., 2017) other newer gradients. Anarchic FL (Yang et al., 2022) can be seen as a more extreme version of Async-FL. In Anarchic FL, clients can decide to download and upload the models at any time, not controlled by the server at all. To this end, `HPFL` naturally allows this kind of working paradigm since once an average backbone, which can be obtained from pre-trained models or summoning several active clients to train, is accessible, any aggregation operation is not in demand for `HPFL`, so the server doesn't rely on timely respond of client and won't be disturbed by stale model update. Once a plug-in is updated by the client, the plug-in can be utilized to do inference on appropriate test data without concern that the parameter of the model will change over time.

A.6 TEST-ADAPTATION & DOMAIN ADAPTATION METHODS IN FL

There also emerge works that aim to adapt or generalize to new unseen clients with seen or unseen data distribution. FADA (Peng et al., 2019) utilize domain adaptation to tackle with seen target distribution. However, their method requires target domain data to train an adversarial model and thus cannot handle the situation where the target domain is unknown. FedDG (Liu et al., 2021) first proposed a novel setting where a federated model trained on multiple distributed source domains is required to generalize on unseen target domains. However, these methods all aim to train a unified global model for adaptation or generalization to new clients. As far as we know, `HPFL` is the first FL framework to directly exploit PMs to achieve this goal. TTPFL (Bao et al., 2024) proposed using unlabelled test data from new clients to personalize global model by adaptively learning each module in the personalized models on unlabelled test data. However, we pay little attention to how to obtain personalized model, which is done simply with fine-tuning in our experiments, instead we concentrate on how to make use of all clients' personalized models collaboratively. Therefore, personalized models learned in (Bao et al., 2024) can also be utilized in `HPFL` to further boost its performance.

B PROOF

B.1 LOWER BOUND OF PM WITH GFL

Theorem B.1. *With Equation 4 and the PMs obtained from optimizing Equation 2 as:* $\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl} = \arg\min_{\Omega,\theta_1,...,\theta_M} \mathcal{L}_P(\Omega, \theta_1, ..., \theta_M)$*, we have*

$$\mathcal{L}_{P-G}(\Omega, \theta_1, ..., \theta_M) \geq \mathcal{L}_P(\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}).$$

Proof.

$$\begin{aligned}
\mathcal{L}_{P-G}(\Omega, \theta_1, ..., \theta_M) &= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\ell(f(\theta_i, \xi_m), \xi_m) + \mathcal{R}(\Omega, \theta_1, ..., \theta_M) \\
&\geq \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\ell(f(\theta_m, \xi_m), \xi_m) + \mathcal{R}(\Omega, \theta_1, ..., \theta_M) \\
&= \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\ell(f(\theta_m, \xi_m), \xi_m) + \mathcal{R}(\Omega, \theta_1, ..., \theta_M) \\
&\geq \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\ell(f(\theta_m^{pfl}, \xi_m), \xi_m) + \mathcal{R}(\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}) \\
&= \mathcal{L}_P(\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}),
\end{aligned} \tag{10}$$

which completes the proof. □

B.2 THE EQUIVALENCE BETWEEN SFL AND PFL

Theorem B.2. *With Equation 4,* $\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl} = \arg\min_{\Omega,\theta_1,...,\theta_M} \mathcal{L}_P(\Omega, \theta_1, ..., \theta_M)$ *and the* $\mathcal{H}^*$ *that guarantees* $\theta_m^{pfl} = s(\Theta, \xi_m, \mathcal{H})$*, we have*

$$\mathcal{L}_{P-G}(\Omega, \theta_1, ..., \theta_M) \geq \mathcal{L}_P(\Theta) \geq \mathcal{L}_S(\Theta, \mathcal{H}^*). \tag{11}$$

Proof.

$$\begin{aligned}
\mathcal{L}_S(\Theta, \mathcal{H}^*) &= \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(\widehat{f}(\Theta, \xi_m, \mathcal{H}^*), \xi_m) \\
&= \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(f(\theta_m^{pfl}, \xi_m), \xi_m) \\
&= \mathcal{L}_P(\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}) - \mathcal{R}(\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}) \\
&\leq \mathcal{L}_P(\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl}),
\end{aligned} \tag{12}$$

combining with Theorem 2.1, which completes the proof. □

B.3 DIFFERENTIAL PRIVACY

In this section, we prove that our protection scheme in Section 3.3 can provide (ϵ, δ)-DP Privacy guarantee for the transmitted markers, and will not leak the information about the raw data. To prove this conclusion, we first introduce the Gaussian mechanism of Differential Privacy (Dwork et al., 2014):

Theorem B.3. $\forall(\epsilon, \delta) \in (0, 1)$*, the Gaussian mechanism* $M(x) = f(x) + N(0, \sigma^2)$ *provides* (ϵ, δ)*-DP privacy protection with*

$$\delta = \Delta_2 \sqrt{2ln(1.25/\sigma)}.$$

According to [5], the traditional Gaussian mechanism can be extended to Theorem 4 to support $\epsilon > 1$.

Theorem B.4. $\forall\epsilon > 0$ *and* $0 < \delta < 1/2 - e^{-3\epsilon}/\sqrt{2\pi\epsilon}$*, the Gaussian mechanism* $M(x) = f(x) + N(0, \sigma^2)$ *provides* (ϵ, δ)*-DP privacy protection with*

$$\sigma \geq \Delta_2/\sqrt{2\epsilon}.$$

With Theorem B.3, we can estimate the magnitude of Gaussian noise needed to be apply with certain function as $0 < \epsilon < 1$. Then, we found that for our protection scheme g in section 3.3, we have:

Theorem B.5. $\forall(\epsilon, \delta) \in (0, 1)$*, if*

$$\epsilon = O\left(\frac{\sqrt{2ln(1.25/\kappa * \sigma_m)}}{\delta}\right)$$

, the procedure g is (ϵ, δ)*-DP.*

Proof. First, we calculate L2 sensitivity Δ_2 in our protection scheme g in Section 3.3.

Assumption B.6. Let $f(\cdot)$ be the backbone (also called as feature extractor) , x is local data extracted from local training data D, i.e. $x \in D$, then $f(x)$ is the raw features, we have

$$\forall x \in D, -C < f(x) < C, \text{where } C \text{ is a constant.}$$

Assumption 1 is often assumed in protecting gradient with DP as in (Abadi et al., 2016; Shokri & Shmatikov, 2015), then

$$\Delta_2 f = max_{x,x'} ||f(x) - f(x')||_2 < 4C^2$$

Recalled from Section 3.3, with μ_m and σ_m^2 separately denote the mean and variance of the raw features, and κ denotes the noisy coefficient, our protection scheme is formed as

$$M(x) = \big(f(x) + \kappa * \mathcal{N}(\mu_m, \sigma_m^2)\big) / (1+\kappa) = 1/(1+\kappa) * f(x) + \kappa/(1+\kappa) * \mu_m + \mathcal{N}(0, \kappa^2 * \sigma_m^2)$$

The bound of μ_m can be obtained with the definition of mean:

$$-C < \mu_m = \underset{x \in D_m}{E}(f(x)) < C$$

Denote $g(x) = 1/(1+\kappa) * f(x) + \kappa/(1+\kappa) * \mu_m$, we have

$$\forall x \in D, -C < g(x) = 1/(1+\kappa) * f(x) + \kappa/(1+\kappa) * \mu_m < C$$

Therefore,

$$\Delta_2 g = max_{x,x'} ||f(x) - f(x')|| < 4C^2. \tag{13}$$

Derived from Theorem B.3, Lemma B.7 is obtained:

Lemma B.7. $\forall(\epsilon, \delta) \in (0, 1)$, *the procedure g is* $(\epsilon, \delta) - DP$ *if*

$$\sigma > \Delta_2 g \frac{\sqrt{2ln(1.25/\epsilon)}}{\delta}$$

By rearranging variables in Lemma B.7, we have Lemma B.8:

Lemma B.8. $\forall(\epsilon, \delta) \in (0, 1)$, *for*

$$\epsilon > \Delta_2 g \frac{\sqrt{2ln(1.25/\sigma)}}{\delta}$$

, the procedure g is (ϵ, δ)*-DP with* $M(x) = f(x) + N(0, \sigma^2)$

Here our Gaussian mechanism's $\sigma = \kappa * \sigma_m$, therefore we have

Theorem B.9. $\forall(\epsilon, \delta) \in (0, 1)$, *for*

$$\epsilon > \Delta_2 g \frac{\sqrt{2ln(1.25/\delta)}}{\sigma} = 4C^2 \frac{\sqrt{2ln(1.25/\delta)}}{\kappa * \sigma_m},$$

the procedure g is (ϵ, δ)*-DP,*

which completes the proof. □

With Theorem B.4, we can estimate the magnitude of Gaussian noise needed to be apply with certain function as $\epsilon > 1$. Then, we found that for our protection scheme g in section 3.3, we have:

Theorem B.10. *$\forall \epsilon > 0$ and $0 < \delta < 1/2 - e^{-3\epsilon}/\sqrt{2\pi\epsilon}$, if*

$$\epsilon \geq O(\frac{1}{2(\kappa * \sigma_m)^2})$$

, the procedure g is (ϵ, δ)-DP.

Proof. According to Equation 13, it holds

$$\Delta_2 g < 4C^2.$$

Then, According to Theorem B.4,

Lemma B.11. *$\forall \epsilon > 0$ and $0 < \delta < 1/2 - e^{-3\epsilon}/\sqrt{2\pi\epsilon}$, the procedure g is $(\epsilon, \delta) - DP$ if*

$$\sigma \geq \Delta_2/\sqrt{2\epsilon} = 4C^2/\sqrt{2\epsilon}.$$

By rearranging variables in Lemma B.11, we have

Theorem B.12. *$\forall \epsilon > 0$ and $0 < \delta < 1/2 - e^{-3\epsilon}/\sqrt{2\pi\epsilon}$, for*

$$\epsilon \geq \frac{16C^4}{2\sigma^2} = \frac{16C^4}{2(\kappa * \sigma_m)^2} \tag{14}$$

the procedure g is (ϵ, δ)-DP,

which completes the proof. □

Combining Theorem B.5 and Theorem B.10, we get

Theorem 3.1. *For the procedure of obtaining and sharing markers $H(x_m) = (g(x_m) + \kappa * \upsilon)/(1 + \kappa)$, (ϵ, δ)-DP holds if the ϵ, δ conforms to any of the two conditions: (1) $\forall \epsilon, \delta \in (0, 1), \epsilon \geq O\left(\frac{\sqrt{2ln(1.25/(\kappa * \sigma_m))}}{\delta}\right)$; (2) $\forall 0 < \delta < 1/2 - e^{-3\epsilon}/\sqrt{2\pi\epsilon}, \epsilon \geq O(\frac{1}{2(\kappa * \sigma_m)^2})$.*

C Experiment Configuration

C.1 Hardware and Software Configuration

We conduct experiments using NVIDIA A100 40GB GPU, AMD EPYC 7742 64-Core Processor Units. The operating system is Ubuntu 20.04.1 LTS. The pytorch version is 1.12.1. The numpy version is 1.23.2. The cuda version is 12.0.

C.2 Implement of Simplified Metrics and Proof

The original metric under the GFL-PM setting in classification tasks should be:

$$\text{Accuracy}\left(\Omega, \theta_1, \ldots, \theta_M\right) = \frac{1}{M} \sum_{i=1}^{M} \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \mathcal{T}\left(f\left(\theta_i, \xi_m\right), \xi_m\right) \tag{15}$$

where $\mathcal{T}(\cdot, \cdot)$ is the function judging whether the prediction of the model is the same with the real label, specifically

$$\mathcal{T}(\text{prediction}, \text{sample}) = \mathbb{1}(\text{predciton} = y_{sample}) \tag{16}$$

where y_{sample} is the label of sample. $f\left(\theta_i, \xi_m\right)$ is the prediction of the model used in final inference on client i for the sample ξ_m, the model is parameterized with θ_i. And our way of determining personalized model using when inferencing on client i is to select from all the plug-ins, i.e. the PMs obtained from optimizing Equation 2 on local data: $\Omega^{pfl}, \theta_1^{pfl}, ..., \theta_M^{pfl} = \arg\min_{\Omega, \theta_1, ..., \theta_M} \mathcal{L}_P(\Omega, \theta_1, ..., \theta_M)$, so we have

$$\theta_i = \theta^{pfl}_{C_i(\xi_m, i)}, m \in 1, 2, ..., M \tag{17}$$

where $C_i(\xi_m, i)$ is the selection made for client i based on test data ξ_m and client i. C_i is the selection algorithm of the client i.

For traditional personalized methods, the clients will only use personalized models trained locally, i.e.

$$C_i(\xi_m, i) = i \tag{18}$$

substitute Equation 18 into Equation 17, we have

$$\theta_i = \theta^{pfl}_{C_i(\xi_m,i)} = \theta^{pfl}_i \tag{19}$$

then substitute Equation 19 into Equation 15, we have

$$\begin{aligned}
\text{Accuracy}\left(\Omega, \theta_1, \ldots, \theta_M\right) &= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\mathcal{T}\left(f\left(\theta_i, \xi_m\right), \xi_m\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_m\right), \xi_m\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \frac{1}{n_m}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_j\right), \xi_j\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} \frac{n_m}{N}\frac{1}{n_m}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_j\right), \xi_j\right) \\
&= \frac{1}{MN}\sum_{i=1}^{M}\sum_{m=1}^{M}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_j\right), \xi_j\right) \\
&= \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_j\right), \xi_j\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\underbrace{\mathbb{E}_{\xi_D\sim\mathcal{D}}[\mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_D\right), \xi_D\right)]}_{\text{accuracy of PM in client } i \text{ on global data}}
\end{aligned} \tag{20}$$

Equation 20 represents **the averaged accuracy of all personalized models on the global dataset**, so we can calculate the averaged accuracy of all personalized models on the global dataset as the metrics of simplified metrics instead of original complicated metrics 15; while for our proposed methods `HPFL`, because all clients have same selection method C

$$C_i(\xi_m, i) = \underset{n}{argmax}\, g\left(\xi_n, \xi_m\right) = C(\xi_m) \tag{21}$$

the origin metric turns into

$$\begin{aligned}
\text{Accuracy}\left(\Omega, \theta_1, \ldots, \theta_M\right) &= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\mathcal{T}\left(f\left(\theta_{argmax_n\, g(\xi_n,\xi_m)}, \xi_m\right), \xi_m\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m\sim\mathcal{D}_m}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)}, \xi_m\right), \xi_m\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{i=1}^{M} p_m \frac{1}{n_m}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)}, \xi_j\right), \xi_j\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} \frac{n_m}{N}\frac{1}{n_m}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)}, \xi_j\right), \xi_j\right) \\
&= \frac{1}{MN}\sum_{i=1}^{M}\sum_{m=1}^{M}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)}, \xi_j\right), \xi_j\right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{N}\sum_{m=1}^{M}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)},\xi_j\right),\xi_j\right)\\
&= \sum_{m=1}^{M}\frac{n_m}{N}\frac{1}{n_m}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)},\xi_j\right),\xi_j\right)\\
&= \sum_{m=1}^{M}p_m\frac{1}{n_m}\sum_{j=1}^{n_m}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)},\xi_j\right),\xi_j\right)\\
&= \sum_{m=1}^{M}p_m\mathbb{E}_{\xi_m\sim\mathcal{D}_m}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)},\xi_j\right),\xi_j\right)\\
&= \sum_{i=1}^{M}p_i\underbrace{\mathbb{E}_{\xi_i\sim\mathcal{D}_i}\mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_i)},\xi_j\right),\xi_j\right)}_{\text{accuracy of PM selected by client i on its own data}},
\end{aligned}
\tag{22}
$$

Equation 22 represents the averaged accuracy of clients testing on their own personalized dataset with models equipped with their selected plug-ins based on their own data, weighted with number of samples in data on clients. With these simplification of metrics, we can more efficiently test GFL-PM performance of both the traditional personalized methods (FedPer, FedRoD, FedRep) and `HPFL`.

C.3 Hyper-parameters

We use SGD without momentum as the optimizer for all experiments, with a batch size of 128 and weight decay of 0.0001. The learning rate is set as 0.1 for both the training of the global model and the fine-tuning on local datasets. The main results shown in Tabel 2 are conducted with 1-layer plug-ins (i.e. only classifier).

Special hyperparameters of some baseline methods are :

FedRep: Local personalize epoch is set as 1.

PerFedMask: The partition percent of validation is 0.1, personalized fine-tuning epoch E_p after calculating mask is set as 5 as the official implementation did.

FedTHE: We follow the official implementation: the smoothing factor of test history descriptor maintained by the Exponential Moving Average (EMA) α equals 0.1; the smoothing factor interpolating the test feature and the test history descriptor β equals 0.3.

FedSAM: We follow the official implementation: the parameters for SAM minimizers $\rho = 0.1$, $\eta = 0$.

C.4 Extra Explanation on Experiments

Due to the limited space of the main text, we show a more detailed explanation of the experiment in this section.

For the construction of the personalized test dataset, to make the training data and test data of a client have the same distribution following the settings of most PFL methods (Collins et al., 2021), we count the number of samples $S_{train}(c,m)$ in each class c of training data of client m and split test data of that clients in that distribution (which means client m have $\frac{\mathcal{S}_{train}(c,m)}{\sum_{m=1}^{\mathcal{N}}\sum_{c=1}^{\mathcal{C}}\mathcal{S}_{train}(c,m)}\times\sum_{m=1}^{M}\sum_{c=1}^{\mathcal{C}}\mathcal{S}_{test}(c,m)$ test samples in class c, here $\mathcal{C}$ denotes the number of classes in overall dataset, M denotes the number of clients in the FL system), Figure 5 and Figure 6 shows that the data partition of training data and test data are almost identical as expected in PFL.

To report the best result of all baseline methods, we report the accuracy of their best inference global model on global data during the whole training process. For our method, we also use the best inference model as the backbone of `HPFL` for fair comparison.

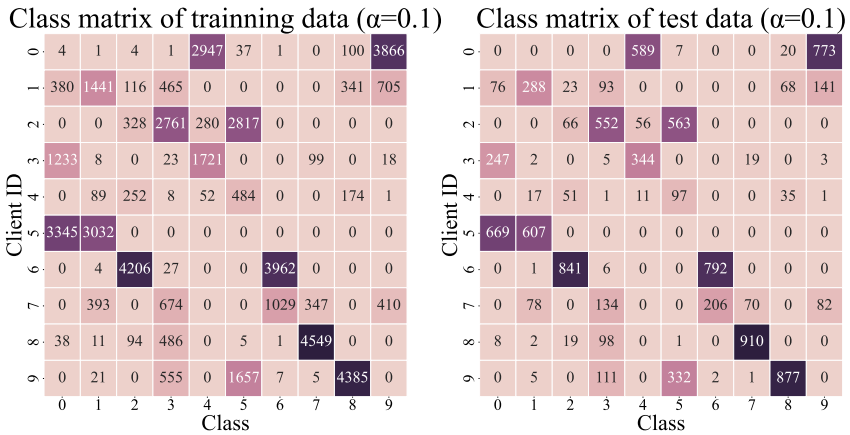


Figure 5: Data partitioning on CIFAR-10 (α=0.1)

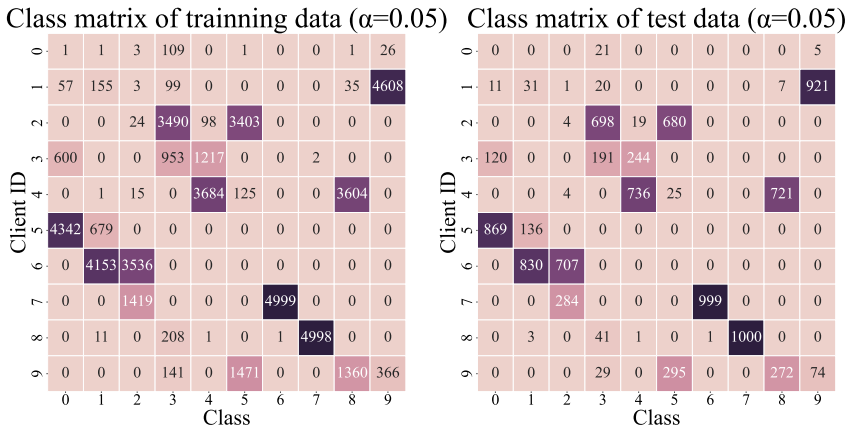


Figure 6: Data partitioning on CIFAR-10 (α=0.05)

D EXTRA EXPERIMENT RESULTS

Due to the limited space of the main text, we show more experiment results in this section.

D.1 MORE RESULTS ABOUT DIFFERENT NUMBERS OF LAYERS OF PLUG-IN

In this part, we are trying to explain why the selection method degrades when the number of plug-in layers increases in an intuitive way. We propose two possible ways leading to the degradation: (1) selection methods can't handle the large dimension of markers that vastly increases when the number of plug-in layers increases, as shown in Table 6. (2) A larger number of plug-in layers means we are using markers extracted with more shallow layers, and these markers tend to be more local, which may not be so helpful for the selection method to assess the similarity of the distributions they are sampled from. For further study, we may conduct experiments to testify these two conjectures. Once the conjectures are testified, we will try to find ways to solve these two problems. However, despite the difficulty of choosing, large plug-ins also multiply the computation time and resources needed in training them, the network bandwidth required to transmit them, and so on. As a result, large plug-ins are generally not good options in `HPFL` from our perspective.

We also explore selection with plug-ins composed of different numbers of layers in different settings, like α=0.05, M=10 on CIFAR-10. Figure 4 and Figure 7 show that with the number of plug-in layers going up, the selection becomes more difficult and unstable as we claimed before.

Table 6: # marker dimensions versus # plug-ins layers on CIFAR-10.

# plug-ins layers	1	3	4	5	6
# marker dimensions	512	512×4×4 (8,192)	256×8×8 (16,384)	128×16×16 (32,768)	64×32×32 (131,072)

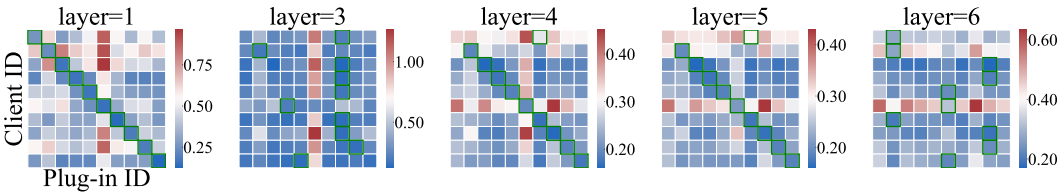


Figure 7: Selection score maps with different numbers of plug-in layers on CIFAR-10 ($\alpha = 0.05$)

D.2 More Results about Selection Methods (MMD, SVCCA, CKA, OOD)

MMD, SVCCA and CKA. MMD (Long et al., 2013) is a kernel-based statistical metric used to determine whether given two distributions are the same. SVCCA (Raghu et al., 2017) exploits singular value decomposition and canonical correlation analysis to compare the markers learned by different DNNs. CKA (Kornblith et al., 2019) utilizes the normalized HSIC (Gretton et al., 2005) to measure the similarity. CKA is invariant to the invertible linear transformation. Thus, it can measure meaningful similarities between representations of high dimension (Kornblith et al., 2019).

Here we summary the formulas of MMD, SVCCA and CKA to better explain our method:

MMD

Given observations $X := \{x_1, \ldots, x_m\}$ and $Y := \{y_1, \ldots, y_n\}$,

$$MMD_u^2[X,Y] = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} k\left(x_i, x_j\right) + \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k\left(y_i, y_j\right) - \frac{2}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} k\left(x_i, y_j\right).$$

, where $k(\cdot)$ is the kernel function.

CKA

Let K and K' be two kernel functions defined over $\mathcal{X} \times \mathcal{X}$ such that $0 < \mathrm{E}\left[K_c^2\right] < +\infty$ and $0 < \mathrm{E}\left[K_c'^2\right] < +\infty$. Then, the alignment between K and K' is defined by

$$\rho\left(K, K'\right) = \frac{\mathrm{E}\left[K_c K_c'\right]}{\sqrt{\mathrm{E}\left[K_c^2\right] \mathrm{E}\left[K_c'^2\right]}}.$$

SVCCA

1. Input: X, Y.

2. Perform: SVD(X), SVD(Y). Output: $X^{'} = UX, Y^{'} = VY$.

3. Perform CCA($X^{'}, Y^{'}$). Output: $X^{\tilde{}} = W_X X^{'}, Y^{\tilde{}} = W_Y Y^{'}$ and $corrs = \{\rho_1, ..., \rho_{min(m_1,m_2)}\}$, where m_1 and m_2 is the number of samples of X and Y.

OOD detection based methods. Out-of-distribution (OOD) detection aims to find out whether the test data is OOD or not. Current OOD detection methods includes the norm of gradients (Huang et al., 2021), distance-based methods (Sun et al., 2022), reconstruction based methods (Zhou, 2022), classifier based methods (Katz-Samuels et al., 2022; Du et al., 2022). Intuitively, if the test data is OOD to one plug-in, we can discard this plug-in. Thus, based on this insight, we train an OOD classifier τ_m for each plug-in ρ_m. Each OOD classifier will output an OOD confidence $\tau_m(h_{test})$. Lower $\tau_m(h_{test})$, less possible that h_{test} is OOD to the plug-in ρ_m. Then, the select problem turns

from Equation 6 into:

$$S_{OOD}(h_{test}, \tau_1, ..., \tau_M) = \arg\min_{i \in \mathcal{M}} \tau_i(h_{test}). \tag{23}$$

The τ_m is trained during the process of optimizing ρ_m. On each client m, we generate random noise $\epsilon \sim \mathcal{N}(0, \mu_\epsilon)$ as the OOD data, the h_m is seen as the in-distribution (ID) data. OOD data has label 1, and ID data has label 0. We use a linear classifier as the OOD classifier τ_m whose input dimension is same with h_m and the output dimension is 1. For OOD data, we hope the τ_m outputs 1. We use the cross-entropy loss to train the OOD classifier. Different from the distance based methods, the OOD detection based method does not need to communication the markers $\hat{h}_m$, which significantly increases the privacy security.

We show the results of our previous attempts using SVCCA, CKA, and OOD as the selection methods in this section. The overall training and inference process is the same as mentioned in Section 3. When implementing SVCCA in $\mathcal{M}$=100, we encountered the problem that the number of samples in a client was not enough for SVCCA, which required at least $N_{component}$ components. To deal with it, we change the number of components used to calculate SVCCA similarity between the plug-in markers and task markers to min(min(num_sample), $N_{component}$). From Table 7 and Figure 8, it is easy to conclude that SVCCA, CKA (with 4 kinds of kernel), OOD all failed to select correct plug-ins. Take `HPFL` based on OOD detection as an example, OOD classifiers failed to give good predictions on whether test data is OOD or not for plug-ins because in federated learning, it is difficult to get OOD data as negative samples when training locally, which will cause OOD classifiers only see in-distribution(ID) data. We tried to solve this problem by generating random images as OOD data to train the OOD classifiers. It is easy to see this method did not work as shown in Figure 8. Therefore, it is difficult to tell whether there is an appropriate way to train OOD classifiers to determine whether OOD detection can be utilized in `HPFL`. There we propose a possible way to train OOD classifiers: after clients upload the plug-in markers to the server, it is possible to utilize these markers to train a good OOD classifier and select plug-ins using these classifiers. We take this as a future direction in exploring more selection methods that can be used in `HPFL`.

D.3 More results about selection accuracy

In this part, we take a closer look at the selection accuracy using MMD by visualizing all the selection situation on CIFAR-10, FMNIST, CIFAR-100, with three settings {α=0.1, $\mathcal{M}$=10}, {α=0.05, $\mathcal{M}$=10}, {α=0.1, $\mathcal{M}$=100} as arrange from left to right in Figure 9, 10, 11. The colors of the blocks denote the MMD scores, where red represents a relatively high score, and blue represents a relatively low score. The green box on the block(i,j) implies that client i is choosing plug-in j with minimal MMD score for local test data. MMD always helps clients select plug-in trained on the client itself (the green boxes denoting final choice of plug-ins all lie on the diagonal of score heatmap) when α=0.1, $\mathcal{M}$=10 and α=0.05, $\mathcal{M}$=10 in CIFAR-10 and FMNIST as shown in Figure 9, Figure 10. When it comes to a FL system with more clients, like $\mathcal{M}$=100, even though there is some clients choosing inappropriate plug-ins, most of them can still choose plug-ins trained on their own data. However, When conducted on a more heterogeneous dataset CIFAR-100, judging which plug-ins to choose becomes a more difficult task, which can be easily observed in Figure 11, as the increasing number of green boxes not located on the diagonal is indicating worse plug-ins selection. Additionally, the score map of CIFAR-100 is overall much whiter than the score map of CIFAR-10 and FMNIST, denoting that the scores of different plug-ins are close to each other and is more challenging to choose from when MMD encountered with CIFAR-100. Thus, how to improve selection methods used in `HPFL` is a crucial problem when met more heterogeneous data together with fewer samples on each client, and will be important for future work.

D.4 More Results about Noise

When using the `HPFL` methods based on MMD, it is required that the distribution of local features, or local features are transmitted together with the plug-in trained in the client, so that the other clients are able to select appropriate plug-ins based on these information. However, transmitting raw features is faced with the risk of data leakage when met inversion attacks. In order to better protect privacy safety, we tried to add Gaussian noise generated with the distribution of local features to the origin features, surprisingly found that adding noise according to the distribution of the features not damage

Table 7: Experiment results of HPFL using SVCCA, CKA and OOD. Noisy coefficient κ=1, FedAvg is fine-tuned with the whole model instead of only part of model as in HPFL.

Clients	10						100		
Non-IID	Dir(0.1)			Dir(0.05)			Dir(0.1)		
Test Set	GFL		PFL	GFL		PFL	GFL		PFL
Method/Model	GM	PM	PM	GM	PM	PM	GM	PM	PM
CIFAR-10									
HPFL(SVCCA) $E_p = 1$	81.5	62.8	95.4	62.4	32.7	96.0	73.6	61.0	95.0
HPFL(SVCCA) $E_p = 10$	81.5	62.5	95.8	62.4	34.7	96.3	73.6	47.0	95.7
HPFL(Linear-CKA) $E_p = 1$	81.5	56.1	95.4	62.4	55.2	96.0	73.6	70.5	95.0
HPFL(Linear-CKA) $E_p = 10$	81.5	55.1	95.8	62.4	44.6	96.3	73.6	60.3	95.7
HPFL(RBF-CKA) $E_p = 1$	81.5	61.0	95.4	62.4	55.2	96.0	73.6	70.7	95.0
HPFL(RBF-CKA) $E_p = 10$	81.5	55.9	95.8	62.4	44.6	96.3	73.6	59.9	95.7
HPFL($Linear - CKA_{debias}$) $E_p = 1$	81.5	63.9	95.4	62.4	47.2	96.0	73.6	66.3	95.0
HPFL($Linear - CKA_{debias}$) $E_p = 10$	81.5	59.0	95.8	62.4	37.7	96.3	73.6	53.0	95.7
HPFL($RBF - CKA_{debias}$) $E_p = 1$	81.5	61.0	95.4	62.4	35.0	96.0	73.6	68.4	95.0
HPFL($RBF - CKA_{debias}$) $E_p = 10$	81.5	56.7	95.8	62.4	40.2	96.3	73.6	55.7	95.7
HPFL(OOD) $E_p = 1$	81.5	66.3	95.4	62.4	54.4	96.0	73.6	64.0	95.0
HPFL(OOD) $E_p = 10$	81.5	16.0	95.8	62.4	3.9	96.3	73.6	27.9	95.7
FMNIST									
HPFL(SVCCA) $E_p = 1$	86.0	61.8	98.3	76.1	41.7	99.0	90.2	90.0	97.9
HPFL(SVCCA) $E_p = 10$	86.0	49.7	98.4	76.1	49.3	99.2	90.2	87.7	98.8
HPFL(Linear-CKA) $E_p = 1$	86.0	67.6	98.3	76.1	73.2	99.0	90.2	89.1	97.9
HPFL(Linear-CKA) $E_p = 10$	86.0	62.0	98.4	76.1	65.5	99.2	90.2	88.6	98.8
HPFL(RBF-CKA) $E_p = 1$	86.0	67.6	98.3	76.1	73.0	99.0	90.2	89.4	97.9
HPFL(RBF-CKA) $E_p = 10$	86.0	63.3	98.4	76.1	65.5	99.2	90.2	88.7	98.8
HPFL($Linear - CKA_{debias}$) $E_p = 1$	86.0	66.6	98.3	76.1	44.7	99.0	90.2	89.9	97.9
HPFL($Linear - CKA_{debias}$) $E_p = 10$	86.0	51.8	98.4	76.1	37.7	99.2	90.2	88.3	98.8
HPFL($RBF - CKA_{debias}$) $E_p = 1$	86.0	61.2	98.3	76.1	50.7	99.0	90.2	89.9	97.9
HPFL($RBF - CKA_{debias}$) $E_p = 10$	86.0	51.8	98.4	76.1	27.9	99.2	90.2	87.6	98.8
HPFL(OOD) $E_p = 1$	86.0	83.9	98.3	76.1	73.6	99.0	90.2	87.2	97.9
HPFL(OOD) $E_p = 10$	86.0	42.7	98.4	76.1	37.5	99.2	90.2	88.4	98.8
CIFAR-100									
HPFL(SVCCA) $E_p = 1$	68.6	68.2	83.3	65.3	68.2	87.4	59.7	51.8	81.2
HPFL(SVCCA) $E_p = 10$	68.6	55.2	85.7	65.3	55.2	88.8	59.7	39.9	84.1
HPFL(Linear-CKA) $E_p = 1$	68.6	63.4	83.3	65.3	63.4	87.4	59.7	51.0	81.2
HPFL(Linear-CKA) $E_p = 10$	68.6	55.0	85.7	65.3	55.0	88.8	59.7	40.0	84.1
HPFL(RBF-CKA) $E_p = 1$	68.6	64.1	83.3	65.3	64.1	87.4	59.7	51.4	81.2
HPFL(RBF-CKA) $E_p = 10$	68.6	50.9	85.7	65.3	50.9	88.8	59.7	38.8	84.1
HPFL($Linear - CKA_{debias}$) $E_p = 1$	68.6	62.8	83.3	65.3	62.8	87.4	59.7	50.7	81.2
HPFL($Linear - CKA_{debias}$) $E_p = 10$	68.6	52.6	85.7	65.3	52.6	88.8	59.7	38.8	84.1
HPFL($RBF - CKA_{debias}$) $E_p = 1$	68.6	62.9	83.3	65.3	62.9	87.4	59.7	50.8	81.2
HPFL($RBF - CKA_{debias}$) $E_p = 10$	68.6	55.9	85.7	65.3	55.9	88.8	59.7	38.4	84.1
HPFL(OOD) $E_p = 1$	68.6	63.5	83.3	65.3	63.5	87.4	59.7	49.5	81.2
HPFL(OOD) $E_p = 10$	68.6	66.7	85.7	65.3	66.7	88.8	59.7	46.5	84.1

the performance, according to Table 8. Further study may transmit Gaussian noise generated with the distribution of the local features instead of the noised features. In fact, when κ reaches a high value like 1000 in Figure 3, the noised features can be approximately considered to degenerate into the Gaussian noise. From Table 8, Figure 3 and Figure 12, we can observe increasing κ to a large value doesn't hurt much performance of HPFL. Therefore, we will explore using pure Gaussian noise generated with the distribution of the local features to replace the noised features to better protect privacy in the future.

D.5 Experiment about Federated Continual Learning

With the increasing real applications of Federated Learning, Federated Continual Learning (FCL) has attracted the attention of researchers. In this part, we conduct an experiment to display the potential of HPFL to solve catastrophic forgetting met in FCL. We first displayed the catastrophic forgetting

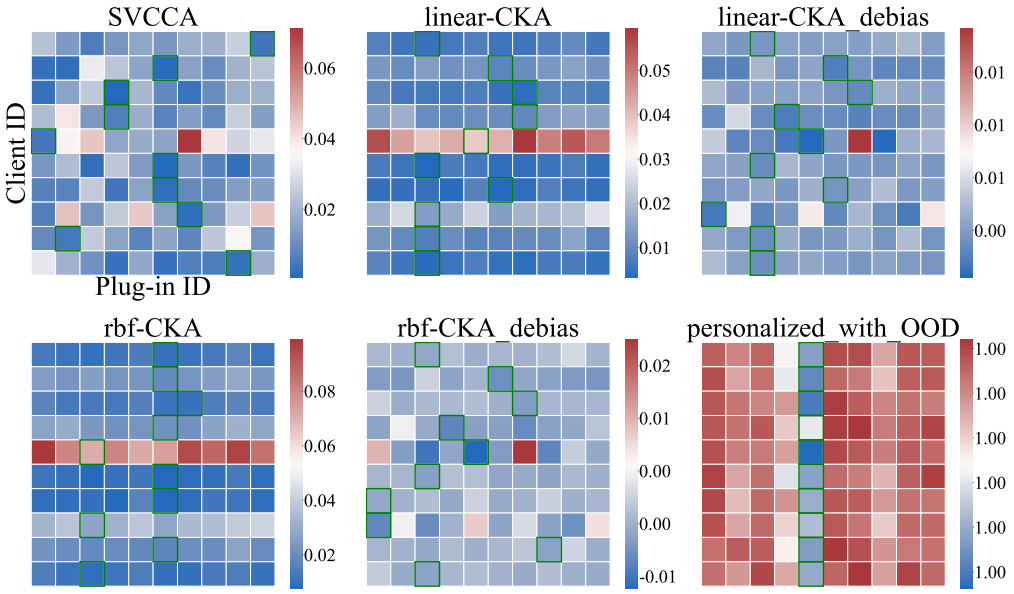


Figure 8: Selection score maps of SVCCA, CKA, OOD on CIFAR-10 ($\alpha = 0.1$)

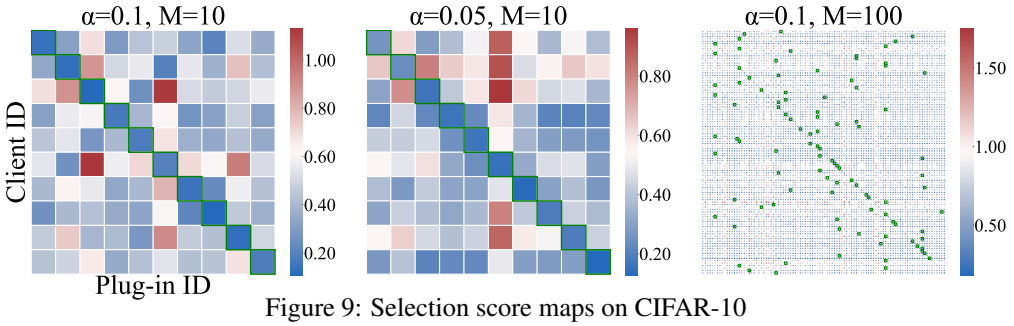


Figure 9: Selection score maps on CIFAR-10

issue in naive FCL. Then we utilized HPFL to solve this problem. Suppose we had 10 clients in the FL system. We first trained 500 epochs on client 0-4 with FedAvg, and then we trained another 500 epochs on client 5-9. We trained the backbone of HPFL and the global model of naive FCL in Nvidia V100 GPU and the rest of the experiment on Nvidia A100. For naive FCL, We had to adjust the learning rate to 0.05 when training on client 5-9 during 500-1000 epoch in case of training divergence. For FCL under HPFL, we froze the backbone of the model after training 500 epochs on client 0-4 and training 5 plug-ins on client 0-4 for 1 epoch, respectively. Then we kept training on client 5-9 with the invariant backbone, after another 500 epochs, we trained 5 plug-ins on client 5-9, respectively. From Table 9, we observe that the accuracy of naive FCL significantly drops from 78.6 to 52.8, showing that training on clients 5-9 during 500-1000 rounds makes the global model severely forget the knowledge about clients 0-4. We show a promising way of using HPFL to mitigate this problem. When met a new task, HPFL allows clients to quickly adapt to their local data by fine-tuning only a few epochs and uploading the plug-in to the server, like what happened at the 500 round in our experiment. After training in some new tasks, it is about time to conduct inference on all clients, we train plug-ins on new tasks, as we do on clients 5-9 in our experiment, and select plug-ins for every client. In that case, we are able to select and download the plug-ins better suited for test data with similar distribution, instead of having no choice but to use a global model having forgotten the knowledge of previous tasks. As is shown in Table 9, our experiment shows HPFL can significantly outperform naive FCL in GFL and mitigate the catastrophic forgetting issue in FCL.

D.6 More Results about Backbone Training Methods

As long as the used GFL methods are able to train a strong general feature extractor, HPFL is able to utilize the feature extractor to train the personalized plug-ins and extract features. We conduct

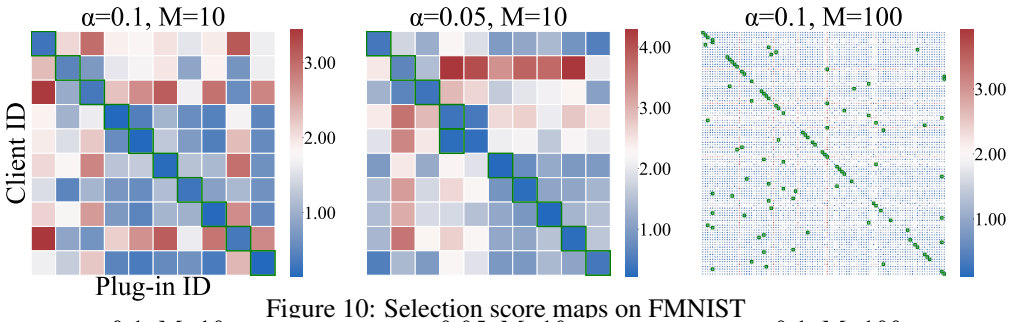


Figure 10: Selection score maps on FMNIST

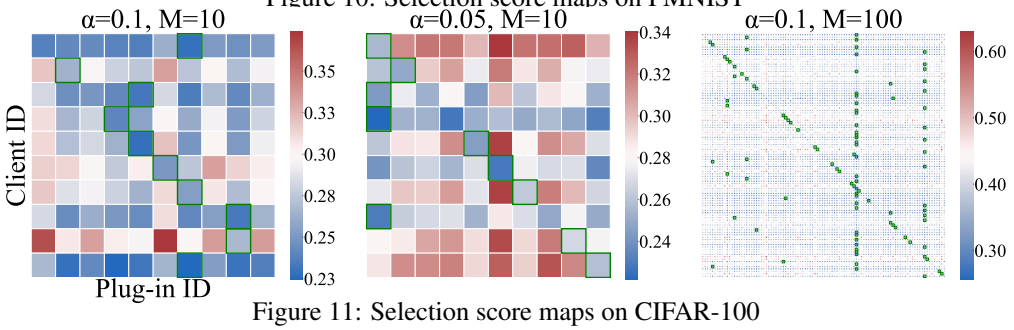


Figure 11: Selection score maps on CIFAR-100

experiments using FedRoD to testify HPFL's compatibility with other GFL methods. The results are given in Table 10. Number of clients equal $\mathcal{M} = 10$, local fine-tuning epoch $E_p = 10$, local datasets are partitioned in $Dir(0.1)$. Other settings remain the same as the main experiments in Table 2.

From the overall performance of HPFL(FedRoD), we can see that HPFL using FedRoD as its backbone training method is comparable to that using FedAvg, which confirms HPFL is compatible with the GFL methods other than FedAvg. We also observe an interesting fact that even if FedRoD shows excellent performance in GFL-GM (surpasses FedAvg in many datasets and settings), fine-tuning the backbone trained with it is not advantageous as shown in PFL-PM (only comparable with fine-tuning on the backbone trained with FedAvg). From this phenomenon, we presume the advantage of FedRoD in GFL-GM should mainly be attributed to its global head trained with a class-balanced loss instead of its backbone.

D.7 Error Bars

Due to time and resource limits, we can only provide error bars of part of our numerous experiments, the results are shown in Table 11. All experiments are conducted with Cifar-10 dataset, partition with $Dir(0.1)$, involved 10 clients. Other settings follows that of Table 2. We calculate the error bars with the standard error of the mean.

From the table above, we can observe that HPFL's error significantly lower than FedSAM, which shows the robustness of our method HPFL when encountering different random simulations.

E Discussion on Privacy Problem

As HPFL requires local clients to share auxiliary information on local data and plug-ins to help inference, it may raise concern about data privacy of HPFL. We attempt to analyze the risk of privacy leakage in HPFL respectively from sharing auxiliary information and plug-ins.

E.1 Privacy Risks of Sharing Plug-ins

In HPFL, we ask local clients to upload part of their personalized models to the server, which means every personalized model is possibly accessible to all clients. This potential sharing with other clients will raise concerns about the risk of privacy leakage. However, in classic Federated Learning algorithms like FedAvg, there also exists similar behavior of sharing global model, and it is difficult to recover training samples from the final model shared over the whole FL system. Instead, research shows that it is possible to recover training data of clients from gradients transmitted to the server

Table 8: Accuracy of Different noise coefficient κ on CIFAR-10.

Noise coefficient κ	0		1		10		100		1000	
Fine-tune epoch E_{tune}	1	10	1	10	1	10	1	10	1	10
$\alpha = 0.1, \mathcal{M} = 10$										
Accuracy	95.4	95.7	95.4	95.7	95.4	95.7	95.4	95.7	95.4	95.7
$\alpha = 0.05, \mathcal{M} = 10$										
Accuracy	96.0	96.3	96.0	96.3	73.7	74.0	71.1	70.5	71.1	70.5
$\alpha = 0.1, \mathcal{M} = 100$										
Accuracy	95.4	92.0	95.4	95.7	95.4	95.7	95.4	95.7	95.4	95.7

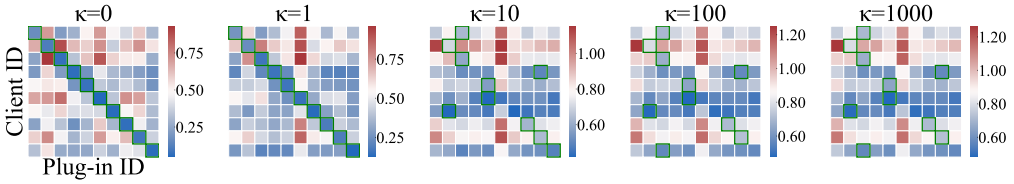


Figure 12: Selection score maps with different noise coefficient on CIFAR-10 (α=0.05, $\mathcal{M}$=10)

(Geiping et al., 2020), which will not happen in HPFL except for the training period of the backbone model, which is able to be solved with regular privacy protect techniques like differential privacy (DP) which is widely used to protect potential privacy risks of GFL algorithms, and not a special problem of HPFL. Even extreme concern on potential privacy risk of storing plug-ins in the server can be solved by only requesting plug-ins after selection as described in Appendix F.2, clients providing the plug-ins can ask the server to delete the plug-ins after sending the plug-ins to the clients in need.

E.2 Privacy Risks of Sharing Auxiliary Information

Since HPFL asks clients to share auxiliary information with the server, once data breach happens in the communication period between clients and the server or the information is not properly kept in the server, the leaked information may lead to attacks, such as feature inversion attacks. Here we resorted image reconstruction by feature inversion method in (Zhao et al., 2020) to check whether the raw image can be reconstructed by inverting the representation through the pretrained global backbone model parameters, exploring whether data privacy will be threatened if both auxiliary information and backbone model is leaked. Experiments are conducted on CIFAR-10 with a ResNet-18 trained on CIFAR-10 under the Federated Learning setting used in our main experiments. As we can see in Figure 14, if we use raw features as the auxiliary information, real pictures can be easily recovered by feature inversion methods. To handle this risk of privacy leakage, here we propose three ways to prevent the problem: (1) add noise to the features; (2) use the averaged feature to select the plug-ins; (3) use model-based selection methods like OOD.

Adding noise to transmitted information is often practiced in the Federated Learning called Differential Privacy(DP), which is utilized to protect gradient against Differential attacks. Inspired by DP, we attempt to add noise to the transmitted auxiliary information, and below we use the same recovery method to recover the original image from the markers. More specifically, we add Gaussian noise to protect the features from privacy risk, which is a commonly-adopted method for (ϵ, δ)-DP. Plenty of previous works (Luo et al., 2021; Li & Wang, 2019; Hao et al., 2021; Chang et al., 2019) transmit noised features to exchange auxiliary information without privacy leakage. As DP used in Federated Learning is mainly for protecting FL system from differential attacks (also called membership inference attacks) (Dwork, 2006; Wei et al., 2020; Truex et al., 2020), which attempt to get information on membership based on differences between models in different rounds. HPFL doesn't involve multiple rounds communication except for traditional GFL backbone training phase. Therefore, differential attack raise no additional privacy risks to protect from in HPFL, and there is not need for HPFL to protect privacy with differential privacy. To this end, we leave out detailed discussion of differential privacy in our paper. Instead, we focus more on model inversion

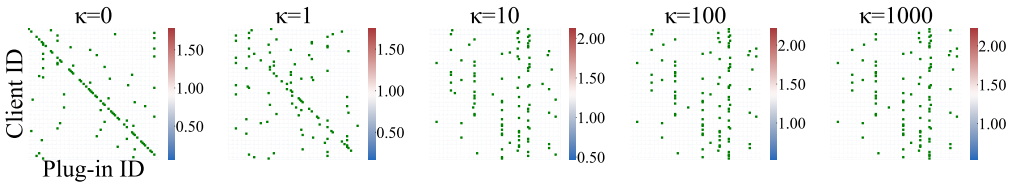


Figure 13: Selection score maps with different noise coefficient on CIFAR-10 (α=0.1, $\mathcal{M}$=100)

Table 9: catastrophic forgetting issue in Naive FCL.

Algorithm	Naive FCL (500R)	Naive FCL (1000R)
Test data	data from Client 0-4	
Method/Model	GM	
Accuracy	78.6	52.8 (↓ 25.8)

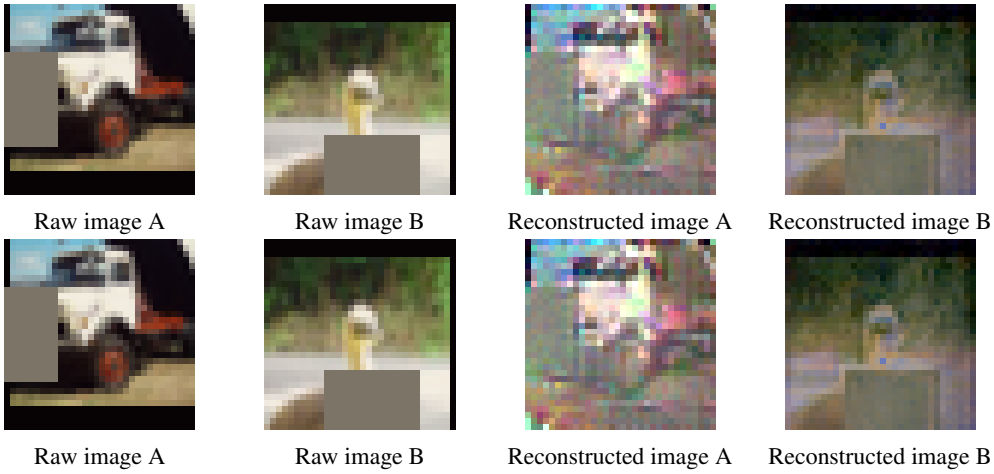


Figure 14: Image reconstructed from raw features

attack and show recovery results with the markers in Figure 15. As noise coefficient κ increases, the reconstructed image is less similar to the raw images. It is worth noting that when $\kappa = 1$ as in our main experiment, the reconstructed images are already hard to tell any information about the raw images. If there is still concern in this situation, the clients can increase the noise coefficient κ to a higher level with the risk of hurting performance.

Using the averaged feature to select the plug-ins is a practical way of protecting privacy as practiced in (Luo et al., 2021), inspired by their work, we attempted to select plug-ins with the average of all features on local clients. However, we assumed that simply averaging all features leads to the lack of information to select plug-ins properly, thus degrades the performance of `HPFL`. Therefore, we tried to divide the features into groups and take the average in every group. With enough samples in every group, we can prevent privacy leakage as presented in Figure 16 and maintain a good performance as shown in Table 12.

Utilizing model-based selection methods like OOD to select the plug-ins, due to these methods avoid sharing direct information about raw data or features, they are exposed to less risk of data leakage. It is more difficult for the attacker to attack the clients with the model parameters than with the data information due to less information contained in it, which can be proved by the data processing inequality (McMahan et al., 2017).

Table 10: Ablation study of backbone training methods.

Clients	10 (sample 50% each round)		
Non-IID	Dir(0.1)		
Test Setting	GFL-PM		
Method		HPFL(FedAvg)	HPFL(FedRoD)
CIFAR-10			
CIFAR-10	GFL-GM	81.5	85.3 (↑ 3.8)
	GFL-PM	95.7	96.0 (↑ 0.3)
	PFL-PM	95.7	96.0 (↑ 0.3)
FMNIST			
FMNIST	GFL-GM	86.0	87.9 (↑ 1.9)
	GFL-PM	98.4	98.4 (↑ 0)
	PFL-PM	98.4	98.4 (↑ 0)
CIFAR-100			
CIFAR-100	GFL-GM	68.6	69.9 (↑ 1.3)
	GFL-PM	72.2	68.5 (↓ 3.7)
	PFL-PM	85.7	85.5 (↓ 0.2)
Tiny-ImageNet-200			
Tiny-ImageNet-200	GFL-GM	56.5	57.4 (↑ 0.9)
	GFL-PM	50.9	56.0 (↑ 5.1)
	PFL-PM	73.7	74.7 (↑ 1.0)

Table 11: Error bars of partial experiments.

Fine-tuning Iteration	Seed=0	Seed=1	Seed=2	Error Bar
FedSAM				
1	47.8	47.4	51.4	± 2.20
10	42.3	42.3	45.2	± 1.67
HPFL				
1	95.4	95.3	95.4	± 0.06
10	95.7	95.6	95.8	± 0.07

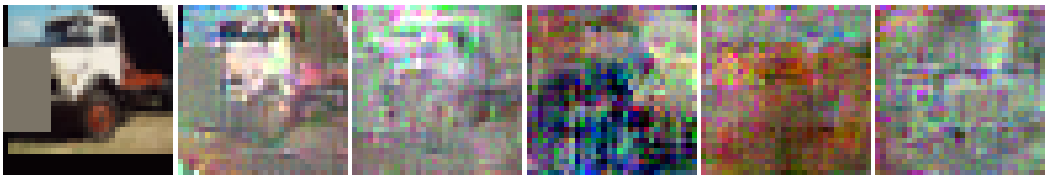
F Real-world Application

F.1 Real-world GM-PFL

To better illustrate the GFL-PM setting we propose and demonstrate its importance, we give some examples exhibiting the significance of our proposed set-up below:

Case 1: Some clients may have insufficient computing resources or local training data to fine-tune a deep learning model in a cross-device setting. In these situations, training distribution can be regarded as an empty set $\emptyset$. In this way, the client cannot get a personalized model by locally fine-tuning the global model. In traditional GFL and PFL setting, the client has no choice but to adopt the global model and endure the lack of personalization. This problem is caused by the mismatch of training data distribution and test data distribution, as assumed in our proposed set-up, and is solvable with our proposed method HPFL by exploiting personalized plug-ins from other clients.

Case 2: A car with a personalized automated driving system (ADS) has driven out of the previous city it used to be. It requires to personalize on geometric data from the present city it is now in for improving the performance of the ADS in this new city. Classic GFL and PFL in this situation leave the ADS no option but to collect the geometric data and personalize on it after the collection completes, and accept the temporary performance loss using the previous personalized model before



Raw image κ=0 κ=1 κ=10 κ=100 κ=1000

Figure 15: Image reconstructed from the markers, κ is noise coefficient denoting the scale of noise added on raw features, the bigger κ is, the larger noise is added on the markers transmitted to select plug-ins.

Raw images Reconstructed image

Figure 16: Image reconstructed from the averaged feature, every group is composed by raw features of 10 raw images

finishing the new personalization, since the distribution of test data has greatly changed. It's another example where the discrepancy between training data (geometric data from the previous city) and test data (geometric data from the present city) threatens the availability of FL systems. While with our proposed method designed to solve the problem, the ADS can attempt to access the plug-ins from car owners living in the present city.

Case 3: Imagine a person is traveling from a high latitude area to an equatorial region, and the recommender system on their phone is supported by federated learning. If the recommender system uses the personalized model trained when in the high latitude area, it will continue to prompt thick down jackets for the person, which is clearly an unexpected and unreasonable recommendation. With our method, one can get the same recommendation as the local people with plug-ins on their phones without time to fine-tune the model again.

F.2 Scalability Issues of HPFL and Their Solutions

F.2.1 Potential Scalability Problem of Sharing Plug-ins

For plug-ins: In fact, `HPFL` can be applied to both cross-device and cross-silo setting, with a slight modification in cross-device setting where the number of clients is overwhelmingly large. We introduce two methods to enhance the scalability problem of `HPFL` as follows: 1) To handle the massive plug-ins needed to be stored in the server, the server can cluster the plug-ins with client-cluster methods in a similar way as done in IFCA (Ghosh et al., 2020), CFL (Sattler et al., 2020), FL+HC (Briggs et al., 2020), and so on. Then the server aggregates the plug-ins in the same clusters to keep a controllable number of plug-ins, like in $O(1)$ or $O(log\mathcal{M})$, where $\mathcal{M}$ denotes the number of clients. 2) Actually a simpler solution can naturally originate from our selection method: we can use selection score to measure the similarity between two plug-ins and abandon some similar plug-ins to reduce the storage space taken up by plug-ins. We carried out an experiment to validate such initial solution and found even with a such naive method, we can significantly reduce the number of plug-ins and maintain higher performance than best baseline GFL methods. As the results shown in Figure 17: (1) `HPFL` are able to maintain 1/3 number of plug-ins and still surpass the best GFL algorithm FedTHE under the experiment setting (see threshold = 0.2); (2) `HPFL` are able to achieve similar accuracy with a half number of plug-ins as that in original `HPFL` (see threshold = 0.1). The server can significantly reduce the number of plug-ins in both ways, thus increase the scalability of our method without much performance loss.

Table 12: Accuracy of Different average group on CIFAR-10.

# of raw features in every group	3		10	
E_p	1	10	1	10
$\alpha = 0.1$, $\mathcal{M} = 10$				
Accuracy	81.5	80.1	76.8	79.1
$\alpha = 0.05$, $\mathcal{M} = 10$				
Accuracy	96.0	96.1	87.6	86.1
$\alpha = 0.1$, $\mathcal{M} = 100$				
Accuracy	81.4	83.8	76.8	75.3

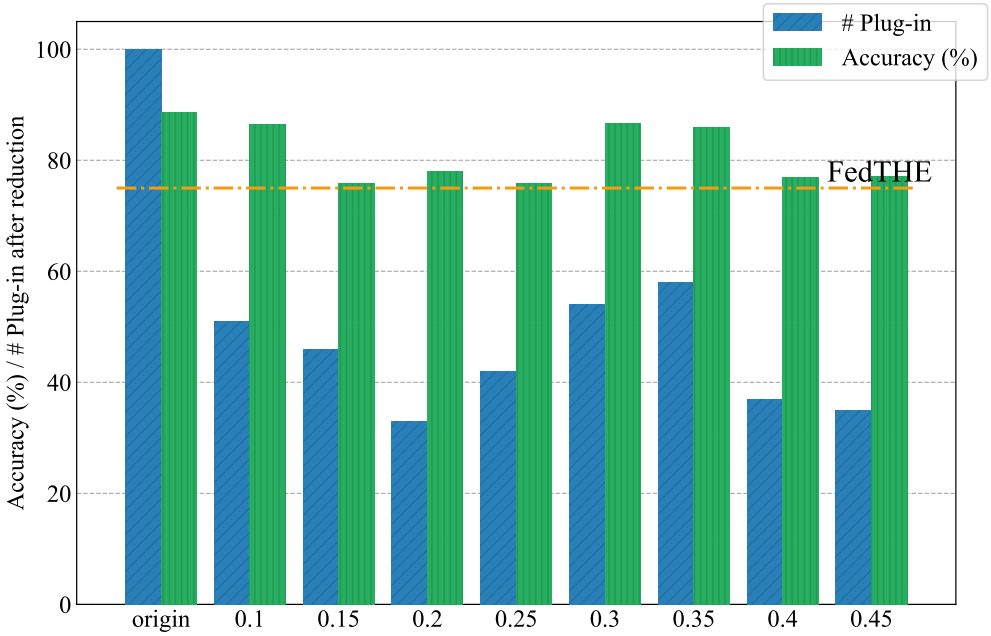


Figure 17: Accuracy and # Plug-in after picking plug-ins with selection scores. X-axis represents the threshold to eliminate plug-ins. If the normalized MMD score between new-coming plug-ins and present plug-ins lower than the threshold, the new-coming plug-ins will be abandon. Experiments are done with $\alpha = 0.1$, $\mathcal{M} = 100$, $E_p = 1$ in CIFAR-10 dataset.

A simpler method is enough for solving the issue of the massive plug-ins in clients: as our selection method doesn't require the presence of plug-ins, clients may not upload the plug-ins after training. Instead, the server can request the appropriate plug-in from the corresponding client after calculating the selection scores. Considering the common issue in cross-client setting, FL systems may encounter client dropout (Li et al., 2020a; Kairouz et al., 2021; Tan et al., 2022b). In a situation where the client with the most appropriate plug-in is out of connection, the server may attempt to request plug-ins one by one with the selection score. To avoid downloading all plug-ins and plug-in markers to clients, if the number of clients grows to a large number, clients can choose to add noise to their local test features and send the task markers to the server to select the plug-ins. In this way, each client can get the exact plug-in they need without the need to download all the plug-ins and the plug-in markers, which will cause a great communication cost with a great number of clients in the FL. We conduct experiments to test the feasibility of this method against the communication and storage burden of `HPFL` in FL systems with plenty of clients. The result are shown in Table 13.

Table 13: Experiment results of sharing task markers. §: we focus more on GFL setting. Numbers in **ForestGreen** highlight highest values in GFL setting. E_p denotes the epoch of fine-tuning. Other hyper-paramters follows the experiments in Table 2.

Clients	10 (sample 50% each round)						100 (5% each round)					
Non-IID	Dir(0.1)			Dir(0.05)			Dir(0.1)			Dir(0.05)		
Test Set	GFL§		PFL	GFL§		PFL	GFL§		PFL	GFL§		PFL
Method/Model	GM	PM	PM	GM	PM	PM	GM	PM	PM	GM	PM	PM
CIFAR-10												
HPFL($\hat{h}_{test}$) $E_p = 1$	81.5	95.4	95.4	62.4	96.0	96.0	73.6	**91.7**	94.9	47.9	**85.2**	93.9
HPFL($\hat{h}_{test}$) $E_p = 10$	81.5	**95.7**	95.7	62.4	**96.3**	96.3	73.6	90.3	95.7	47.9	**85.2**	95.3
FMNIST												
HPFL($\hat{h}_{test}$) $E_p = 1$	86.0	98.3	98.3	76.1	99.1	99.1	90.2	97.9	97.9	86.1	**95.3**	98.1
HPFL($\hat{h}_{test}$) $E_p = 10$	86.0	**98.4**	98.4	76.1	**99.2**	99.2	90.2	**98.6**	98.8	86.1	94.0	98.7
CIFAR-100												
HPFL($\hat{h}_{test}$) $E_p = 1$	68.6	**75.7**	83.3	65.3	**78.8**	87.4	59.7	**67.5**	81.2	47.9	72.7	84.1
HPFL($\hat{h}_{test}$) $E_p = 10$	68.6	69.5	85.7	65.3	78.0	88.9	59.7	63.8	84.1	47.9	**75.5**	86.4
Tiny-ImageNet-200												
HPFL($\hat{h}_{test}$) $E_p = 1$	**56.5**	51.8	70.8	54.9	**55.5**	74.7	47.2	**58.6**	71.3	42.1	**59.2**	74.7
HPFL($\hat{h}_{test}$) $E_p = 10$	**56.5**	47.4	73.7	54.9	50.3	77.0	47.2	57.7	73.2	42.1	57.8	76.5

F.2.2 Online Inference of HPFL

Our method is designed to infer in batches, online test-time adaptation where test samples arrive one by one is not our main application scenario (Hoi et al., 2021; Jiang & Lin, 2022; Tan et al., 2023). Selecting plug-ins for every sample may incur expensive computation costs. However, when met a similar situation and the computation costs are unavoidable, clients can give up downloading plug-ins. Similar to method stated in Appendix F.2.1, following instructions free clients from the storage burden brought by downloading plug-ins and corresponding plug-in markers for every sample: simply send the task marker to the server; let the server select the appropriate plug-in; then infer at the server side; and finally return the inference result back to the client. In this way, communication and latency issues with online inference can be solved with slight modifications in our proposed method.