Hot Pluggable Federated Learning

Lei Shen[†,⋆] **Zhenheng Tang**[†,⋆] **Lijun Wu**[♮] **Yonggang Zhang**[†]
Xiaowen Chu[♯] **Tao Qin**[♮] **Bo Han**[†]
[†] Department of Computer Science, Hong Kong Baptist University
[♮] Microsoft Research
[♯] DSA Thrust, The Hong Kong University of Science and Technology (Guangzhou)
{cslshen, zhtang, ygzhang, bhanml}@comp.hkbu.edu.hk
apeterswu@gmail.com xwchu@ust.hk {taoqin}@microsoft.com

Abstract

Personalized federated learning (PFL) achieves high performance by assuming clients only meet test data locally, which is not held in many generic federated learning (GFL) scenarios. In this work, we show that Personalized models (PMs) can be used to enhance GFL. However, storing and selecting whole models requires impractical computation and communication costs. 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) [7] suffers from data heterogeneity [7, 28, 40, 30, 67, 9], where clients have different data distributions. Personalized federated learning [63, 10, 9] (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 [40, 30, 29, 9].

However, in real-world scenarios, FL users may encounter test data different from local training data [44, 47, 25, 67], but which may appear in other training data. For example, when one traveling abroad, the personal map app might recommend entirely different restaurants from

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

[⋆] Equal Contribution. Correspondence to Bo Han (bhanml@comp.hkbu.edu.hk).

International Workshop on Federated Foundation Models In Conjunction with NeurIPS 2024 (FL@FM-NeurIPS 2024).

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 [63, 10, 9]. 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 [2], FedRep [10] and FedRoD [9] 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?*

To explore this problem, our core idea is to select suitable PMs for inference on clients according to incoming test data. However, this naive solution leads to privacy concerns, large system overheads and poor scalability. To this end, inspired by model components [58, 52], we propose a general and effective framework named *Hot-Pluggable Federated Learning* (HPFL) to practically solve the problems applying PMs in GFL settings.

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 performance advantage of PFL over GFL, and leverage it to boost the GFL performance. 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 a single model.
- We propose a general, efficient and effective framework HPFL to utilize PMs in GFL (Section 4). And we add noise on communicated markers to provide differential privacy protection with theoretical guarantee (Section 4.4).
- 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 Related Works

Generic Federated Learning. To address the data heterogeneity problem, FedProx [40] and MOON [38] 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 [70, 30, 68]. Some works share intermediate features [26, 20, 68] or extra data [67] to reduce client drift. 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 [15], knowledge distillation [83, 37], adaptive regularization and model mixtures [19, 12, 11] are used to enhance personal knowledge learning of models. Some works [41, 36] allow clients to learn different PM structures. KNN-per [49] constructs PMs by replacing classifiers with non-parametric methods based on local datasets. FedRep [10] and FedRoD [9] 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

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

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 [53, 43] focus on generalizing a federated model trained on multiple source domains to unseen target domains. FedTHE [27] 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. 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.

3 Preliminary

3.1 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 [30, 72, 67]:

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

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.

3.2 Personalized FL

Different from the object function of GFL, the PFL aims to learn multiple personalized models which fit well on different datasets individually: [37, 9, 39]:

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

where $\mathcal{R}$ is a regularizer [9] that varies with different algorithms, Ω is used to collaborate clients. We call each obtained locally personalized model θ_m as *PM*.

4 HPFL: Hot-Pluggable Federated Learning

In this section, we will first introduce the design of HPFL in Section 4.1 with the Algorithm 1. Then, we illustrate that directly selecting PM faces some fatal obstacles, including the large system overheads and privacy concerns in Section 4.2. Lastly, the selection method is introduced in Section 4.3.

4.1 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 [50] 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 [81, 4], 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:

$$\min_{\theta_m^\rho} \mathcal{L}_P(\theta_m) = \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \ell(\rho_m \circ g(\xi_m), \xi_m). \tag{3}$$

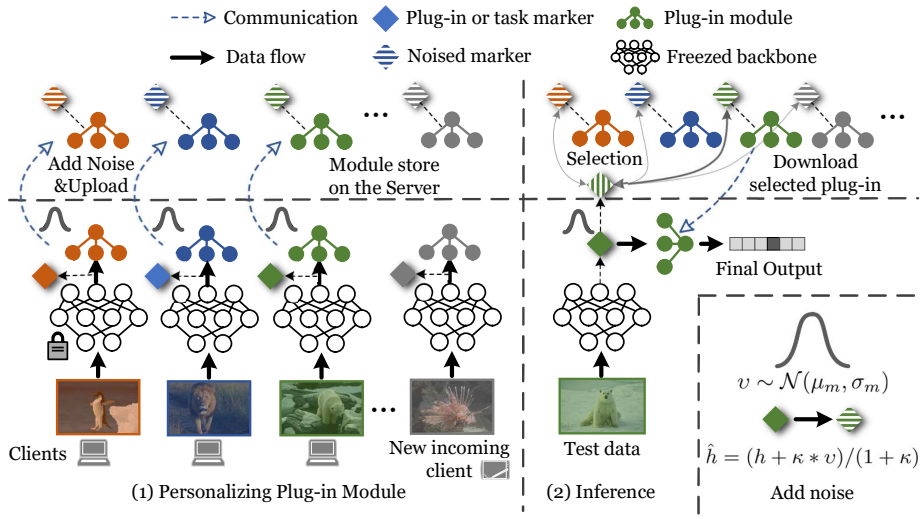

Figure 1: The framework of HPFL.

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 PMs θ_m^{pfl} can be constructed by θ^g and θ_m^ρ, inference becomes as $f(\theta_m^{pfl}, \xi_m) = \rho_m \circ g(\xi_m)$.

Inference and selecting plug-in module. In HPFL, we define some plug-in marker $\mathcal{H}_m$ 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 4.3.

4.2 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 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.

4.3 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 [21, 42, 48, 41].

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 into:

$$S_{dist}(d, h_{test}, \hat{h}_1, ..., \hat{h}_M) = \arg\min_{m \in \mathcal{M}} d(\hat{h_m}, \hat{h}_{test}), \tag{4}$$

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 [46] to measure the distance between plug-in markers and noised features of test data (task marker). We also exploit other distance measures like SVCCA [55], CKA [32] and out-of-distribution confidence based selection methods and provide results in Appendix D.

4.4 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) [1, 5, 74, 78], 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 4.1 is shown in Appendix 3.

Theorem 4.1. *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 [85]. The failed reconstruction (in Appendix E.2) of raw data demonstrate that HPFL can defend model inversion attacks successfully.

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 3 (PFL).
  Obtaining plug-in marker $\mathcal{H}_m$ (e.g. noised features explained in Section 4.3 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 $\leftarrow \rho_i \circ g(\xi | \xi \sim \mathcal{D}_{test})$.

5 Experiments

5.1 Experimental Setup

Federated Datasets and Models. We conduct experiments on four commonly used image classification datasets in FL, including CIFAR-10 [33], CIFAR-100 [33], Fashion-MNIST [75], and Tiny-ImageNet [35], with Latent Dirichlet Sampling (Dir) partition method (α = 0.1, 0.05) to simulate data heterogeneity following [22, 38, 48, 67]. 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 [22, 64]. We train ResNet-18 [23], MobileNet [24] 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 [50], FedSAM [54]; advanced PFL algorithms including FedPer [3], FedRep [10], PerFedMask [57]; FedRoD [9] both for GFL and PFL; and a test-time adaption method FedTHE [27]. 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,

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.

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**

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 [3]. 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: Accuracy of different κ

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

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 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 3. 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.

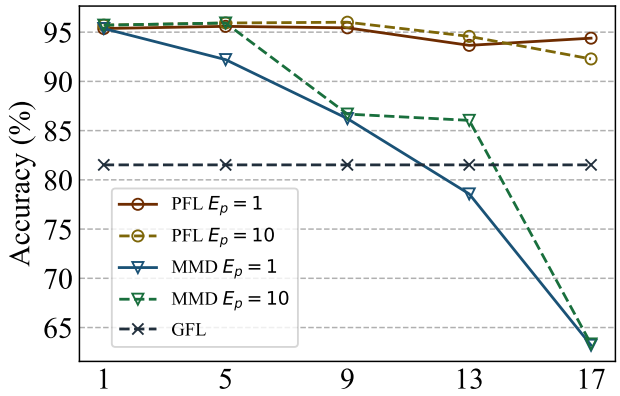

Figure 2: Accuracy with different numbers of plug-in layers. X-axis represents the number of layers in ResNet-18 for plug-ins.

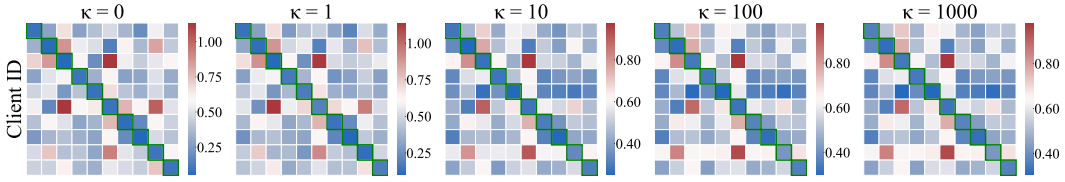

Figure 3: Selection score maps with different noise coefficients. Blocks with green anchor mean the corresponding client selects the plug-in and download it. Blocks with green anchor lying in diagonal indicate that clients choose plug-ins of themselves when met their own test data, which conforms to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

5.3 Selection Accuracy

The more flexible the models are, the better? As shown in Figure 2 and Figure 5, 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

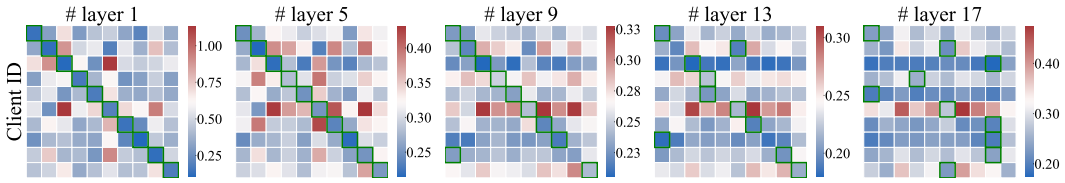

Figure 4: Selection score maps with different numbers of layers in ResNet-18 plug-ins own. X-axis represents Plug-in ID.

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.

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.

Table 4: # 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)

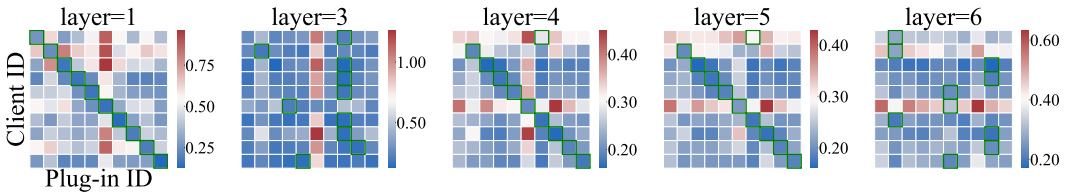

Figure 5: Selection score maps with different numbers of plug-in layers on CIFAR-10 ($\alpha = 0.05$)

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 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 3.

Table 5: Results of FCL

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

5.4 Federated Continual Learning

Federated continual learning (FCL) [79] 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 conduct an experiment to display the potential of HPFL to solve catastrophic forgetting met in FCL. We first displayed the catastrophic forgetting 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 6, 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 6, our experiment shows HPFL can significantly outperform naive FCL in GFL and mitigate the catastrophic forgetting issue in FCL.

Table 6: 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)

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 [77], 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 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.

References

[1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019.

[3] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary. Federated learning with personalization layers. *CoRR*, abs/1912.00818, 2019.

[4] Y. M. Asano, C. Rupprecht, and A. Vedaldi. A critical analysis of self-supervision, or what we can learn from a single image. In *ICLR*, 2020.

[5] B. Balle and Y.-X. Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *International Conference on Machine Learning*, pages 394–403. PMLR, 2018.

[6] W. Bao, T. Wei, H. Wang, and J. He. Adaptive test-time personalization for federated learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[7] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv e-prints*, page arXiv:1602.05629, Feb. 2016.

[8] H. Chang, V. Shejwalkar, R. Shokri, and A. Houmansadr. Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. *arXiv preprint arXiv:1912.11279*, 2019.

[9] H.-Y. Chen and W.-L. Chao. On bridging generic and personalized federated learning for image classification. In *International Conference on Learning Representations*, 2021.

[10] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai. Exploiting shared representations for personalized federated learning. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 2089–2099. PMLR, 18–24 Jul 2021.

[11] Y. Deng, M. M. Kamani, and M. Mahdavi. Adaptive personalized federated learning, 2020.

[12] C. T. Dinh, N. H. Tran, and T. D. Nguyen. Personalized federated learning with moreau envelopes, 2020.

[13] C. Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

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

[15] A. Fallah, A. Mokhtari, and A. Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[16] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[17] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16937–16947. Curran Associates, Inc., 2020.

[18] A. Ghorbani and J. Zou. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pages 2242–2251. PMLR, 2019.

[19] F. Hanzely and P. Richtárik. Federated learning of a mixture of global and local models, 2020.

[20] W. Hao, M. El-Khamy, J. Lee, J. Zhang, K. J. Liang, C. Chen, and L. C. Duke. Towards fair federated learning with zero-shot data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3310–3319, 2021.

[21] C. He, M. Annavaram, and S. Avestimehr. Group knowledge transfer: Federated learning of large cnns at the edge. In *Advances in Neural Information Processing Systems 34*, 2020.

[22] C. He, S. Li, J. So, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[25] T.-M. H. Hsu, H. Qi, and M. 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*, pages 76–92. Springer, 2020.

[26] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *NeurIPS*, 2018.

[27] L. Jiang and T. Lin. Test-time robust personalization for federated learning. In *International Conference on Learning Representations (ICLR)*, 2023.

[28] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[29] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[30] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. *arXiv preprint arXiv:1910.06378*, 2019.

[31] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

[32] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529. PMLR, 09–15 Jun 2019.

[33] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[34] J. Langford, A. J. Smola, and M. Zinkevich. Slow learners are fast. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS'09, page 2331–2339, Red Hook, NY, USA, 2009. Curran Associates Inc.

[35] Y. Le and X. S. Yang. Tiny imagenet visual recognition challenge. 2015.

[36] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li. Lotteryfl: Empower edge intelligence with personalized and communication-efficient federated learning. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 68–79, 2021.

[37] D. Li and J. Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[38] Q. Li, B. He, and D. Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10713–10722, 2021.

[39] T. Li, S. Hu, A. Beirami, and V. Smith. Ditto: Fair and robust federated learning through personalization. In *ICML*, 2021.

[40] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, volume 2, pages 429–450, 2020.

[41] P. P. Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[42] T. Lin, L. Kong, S. U. Stich, and M. Jaggi. Ensemble distillation for robust model fusion in federated learning. In *NeurIPS*, 2020.

[43] Q. Liu, C. Chen, J. Qin, Q. Dou, and P.-A. Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space, 2021.

[44] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang. Fedvision: An online visual object detection platform powered by federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13172–13179, 2020.

[45] Z. Liu, Y. Chen, H. Yu, Y. Liu, and L. Cui. Gtg-shapley: Efficient and accurate participant contribution evaluation in federated learning. *ACM Trans. Intell. Syst. Technol.*, 13(4), may 2022.

[46] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *International conference on machine learning*, pages 2208–2217. PMLR, 2017.

[47] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang. Real-world image datasets for federated learning. *arXiv preprint arXiv:1910.11089*, 2019.

[48] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021.

[49] O. Marfoq, G. Neglia, R. Vidal, and L. Kameni. Personalized federated learning through local memorization. In *International Conference on Machine Learning*, pages 15070–15092. PMLR, 2022.

[50] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

[51] K. L. Ng, Z. Chen, Z. Liu, H. Yu, Y. Liu, and Q. Yang. A multi-player game for studying federated learning incentive schemes. In *IJCAI International Joint Conference on Artificial Intelligence*, page 5279, 2020.

[52] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

[53] X. Peng, Z. Huang, Y. Zhu, and K. Saenko. Federated adversarial domain adaptation. In *International Conference on Learning Representations*, 2019.

[54] Z. Qu, X. Li, R. Duan, Y. Liu, B. Tang, and Z. Lu. Generalized federated learning via sharpness aware minimization. In *International Conference on Machine Learning*, pages 18250–18280. PMLR, 2022.

[55] M. Raghu, J. Gilmer, J. Yosinski, and J. Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[56] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive Neural Networks. *arXiv e-prints*, page arXiv:1606.04671, June 2016.

[57] M. Setayesh, X. Li, and V. W.S. Wong. Perfedmask: Personalized federated learning with optimized masking vectors. In *Proc. of International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, May 2023.

[58] H. Shah, A. Ilyas, and A. Madry. Decomposing and editing predictions by modeling model computation. *arXiv preprint arXiv:2404.11534*, 2024.

[59] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik. Personalized federated learning using hypernetworks. In *ICML*, 2021.

[60] M. Shin, C. Hwang, J. Kim, J. Park, M. Bennis, and S.-L. Kim. Xor mixup: Privacy-preserving data augmentation for one-shot federated learning. *arXiv preprint arXiv:2006.05148*, 2020.

[61] R. Shokri and V. Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1310–1321, 2015.

[62] R. H. L. Sim, Y. Zhang, M. C. Chan, and B. K. H. Low. Collaborative machine learning with incentive-aware model rewards. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[63] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

[64] Z. Tang, X. Chu, R. Y. Ran, S. Lee, S. Shi, Y. Zhang, Y. Wang, A. Q. Liang, S. Avestimehr, and C. He. Fedml parrot: A scalable federated learning system via heterogeneity-aware scheduling on sequential and hierarchical training. *arXiv preprint arXiv:2303.01778*, 2023.

[65] Z. Tang, S. Shi, B. Li, and X. Chu. Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication. *IEEE Transactions on Parallel and Distributed Systems*, pages 1–13, 2022.

[66] Z. Tang, Y. Wang, X. He, L. Zhang, X. Pan, Q. Wang, R. Zeng, K. Zhao, S. Shi, B. He, et al. Fusionai: Decentralized training and deploying llms with massive consumer-level gpus. *arXiv preprint arXiv:2309.01172*, 2023.

[67] Z. Tang, Y. Zhang, S. Shi, X. He, B. Han, and X. Chu. Virtual homogeneity learning: Defending against data heterogeneity in federated learning. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 21111–21132. PMLR, 17–23 Jul 2022.

[68] Z. Tang, Y. Zhang, S. Shi, X. Tian, T. Liu, B. Han, and X. Chu. Fedimpro: Measuring and improving client update in federated learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[69] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei. Ldp-fed: Federated learning with local differential privacy. In *Proceedings of the third ACM international workshop on edge systems, analytics and networking*, pages 61–66, 2020.

[70] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 7611–7623, 2020.

[71] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.

[72] B. E. Woodworth, K. K. Patel, and N. Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *Advances in Neural Information Processing Systems*, 33:6281–6292, 2020.

[73] 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.

[74] Z. Wu, Q. Li, and B. He. A coupled design of exploiting record similarity for practical vertical federated learning. *Advances in Neural Information Processing Systems*, 35:21087–21100, 2022.

[75] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[76] C. Xie, S. Koyejo, and I. Gupta. Asynchronous Federated Optimization. *arXiv e-prints*, page arXiv:1903.03934, Mar. 2019.

[77] H. Yang, X. Zhang, P. Khanduri, and J. Liu. Anarchic federated learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25331–25363. PMLR, 17–23 Jul 2022.

[78] Z. Yang, Y. Zhang, Y. Zheng, X. Tian, H. Peng, T. Liu, and B. Han. Fedfed: Feature distillation against data heterogeneity in federated learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[79] J. Yoon, W. Jeong, G. Lee, E. Yang, and S. J. 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*, pages 12073–12086. PMLR, 18–24 Jul 2021.

[80] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[81] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014.

[82] H. Yu, Z. Liu, Y. Liu, T. Chen, M. Cong, X. Weng, D. Niyato, and Q. Yang. A sustainable incentive scheme for federated learning. *IEEE Intelligent Systems*, 35(4):58–69, 2020.

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

[84] R. Zeng, C. Zeng, X. Wang, B. Li, and X. Chu. Incentive mechanisms in federated learning and game-theoretical approach. *IEEE Network*, pages 1–7, 2022.

[85] N. Zhao, Z. Wu, R. W. Lau, and S. Lin. What makes instance discrimination good for transfer learning? In *International Conference on Learning Representations*, 2020.

[86] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu. Asynchronous stochastic gradient descent with delay compensation. In *ICML*. JMLR, 2017.

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 [40] and MOON [38] 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 [70, 30, 65, 66]. With a level of privacy protection, some works propose to share intermediate features [26, 20] or extra data [67, 60, 42] 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 [16] to Federated Learning for faster adaptation on local training data in clients. The Model-Agnostic Meta-Learning [16] (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 [15] 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 [83]. 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 [19, 12, 11] utilize some regularization and adaptive model mixture to learn personalized models. FedMD [37] 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 [39] 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 [73, 36] propose to allow clients to learn different personalized model structures. LotteryFL [36] 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 [59] 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 [41] 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 [10] proposes to learn a global feature extractor and personalized

classifiers. FedRoD [9] 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 [18, 45, 62]. Furthermore, there should be a well-designed incentive mechanism [51, 82, 84] 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) [31] 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 [31] finds the model parameters that are good for both previous and subsequent tasks using the Fisher Information Matrix. Progressive Neural Network approach [56] 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 [80] dynamically decides the model capacity to learn a compact overlapping knowledge sharing among tasks.

Federated Continual Learning (FCL) [79] 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 **??**.

A.5 Asynchronous FL

Asynchronous FL (Async-FL) [76] means to ease the constraint of the synchronous communication mechanism of classic federated optimization schemes [50]. 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 [34, 86]. Stale updates are usually controlled by some staleness coefficients [76] or compensated by [86] other newer gradients. Anarchic FL [77] 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 [53] 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 [43] 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. FedTHE & FedTHE+ [27] 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. TTPFL [6] 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 [6] can also be utilized in `HPFL` to further boost its performance.

B Proof

B.1 Differential Privacy

In this section, we prove that our protection scheme in Section 4.3 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 [14]:

Theorem B.1. *$\forall(\epsilon, \delta) \in (0, 1)$, 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.2. *$\forall \epsilon > 0$ 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.1, 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 4.3, we have:

Theorem B.3. *$\forall(\epsilon, \delta) \in (0, 1)$, 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 4.3.

Assumption B.4. 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 [1, 61], then

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

Recalled from Section 4.3, 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,

$$\Delta_2 g = max_{x,x'} \, ||f(x) - f(x')|| < 4C^2. \tag{5}$$

Derived from Theorem B.1, Lemma B.5 is obtained:

Lemma B.5. $\forall(\epsilon, \delta) \in (0,1)$*, 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.5, we have Lemma B.6:

Lemma B.6. $\forall(\epsilon, \delta) \in (0,1)$*, 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.7. $\forall(\epsilon, \delta) \in (0,1)$*, 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.2, 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 4.3, we have:

Theorem B.8. $\forall \epsilon > 0$ *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 5, it holds

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

Then, According to Theorem B.2,

Lemma B.9. *$\forall \epsilon > 0$ 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.9, we have

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

$$\epsilon \geq \frac{16C^4}{2\sigma^2} = \frac{16C^4}{2(\kappa * \sigma_m)^2} \tag{6}$$

the procedure g is (ϵ, δ)-DP,

which completes the proof. □

Combining Theorem B.3 and Theorem B.8, we get

Theorem 4.1. *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:

$$\text{Accuracy}\left(\Omega, \theta_1, \ldots, \theta_M\right) = \frac{1}{M} \sum_{i=1}^{M} \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \mathcal{T}\left(f\left(\theta_i, \xi_m\right), \xi_m\right) \tag{7}$$

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

$$\mathcal{T}(\text{prediction}, \text{sample}) = \mathbb{1}(\text{predciton} = y_{sample}) \tag{8}$$

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 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

$$\theta_i = \theta^{pfl}_{C_i(\xi_m, i)}, m \in 1, 2, ..., M \tag{9}$$

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.

$$C_i(\xi_m, i) = i \tag{10}$$

substitute Equation 10 into Equation 9, we have

$$\theta_i = \theta^{pfl}_{C_i(\xi_m,i)} = \theta^{pfl}_i \tag{11}$$

then substitute Equation 11 into Equation 7, we have

$$\begin{aligned}
\text{Accuracy}\left(\Omega, \theta_1, \ldots, \theta_M\right) &= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \mathcal{T}\left(f\left(\theta_i, \xi_m\right), \xi_m\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_m\right), \xi_m\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{i=1}^{M}\sum_{m=1}^{M} p_m \frac{1}{n_m}\sum_{j=1}^{n_m} \mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_j\right), \xi_j\right) \\
&= \frac{1}{M}\sum_{i=1}^{M}\sum_{m=1}^{M} \frac{n_m}{N}\frac{1}{n_m}\sum_{j=1}^{n_m} \mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_j\right), \xi_j\right) \\
&= \frac{1}{MN}\sum_{i=1}^{M}\sum_{m=1}^{M}\sum_{j=1}^{n_m} \mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_j\right), \xi_j\right) \\
&= \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} \mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_j\right), \xi_j\right) \\
&= \frac{1}{M}\sum_{i=1}^{M} \underbrace{\mathbb{E}_{\xi_D \sim \mathcal{D}}[\mathcal{T}\left(f\left(\theta^{pfl}_i, \xi_D\right), \xi_D\right)]}_{\text{accuracy of PM in client } i \text{ on global data}}
\end{aligned} \tag{12}$$

Equation 12 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 7; while for our proposed methods `HPFL`, because all clients have same selection method C

$$C_i(\xi_m, i) = \underset{n}{argmax}\, g\left(\xi_n, \xi_m\right) = C(\xi_m) \tag{13}$$

the origin metric turns into

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

$$
\begin{aligned}
&= \sum_{m=1}^{M} p_m \frac{1}{n_m} \sum_{j=1}^{n_m} \mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)}, \xi_j\right), \xi_j\right) \\
&= \sum_{m=1}^{M} p_m \mathbb{E}_{\xi_m \sim \mathcal{D}_m} \mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_m)}, \xi_j\right), \xi_j\right) \\
&= \sum_{i=1}^{M} p_i \underbrace{\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \mathcal{T}\left(f\left(\theta^{pfl}_{C(\xi_i)}, \xi_j\right), \xi_j\right)}_{\text{accuracy of PM selected by client i on its own data}},
\end{aligned} \tag{14}
$$

Equation 14 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 [10], 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 6 and Figure 7 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.

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 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

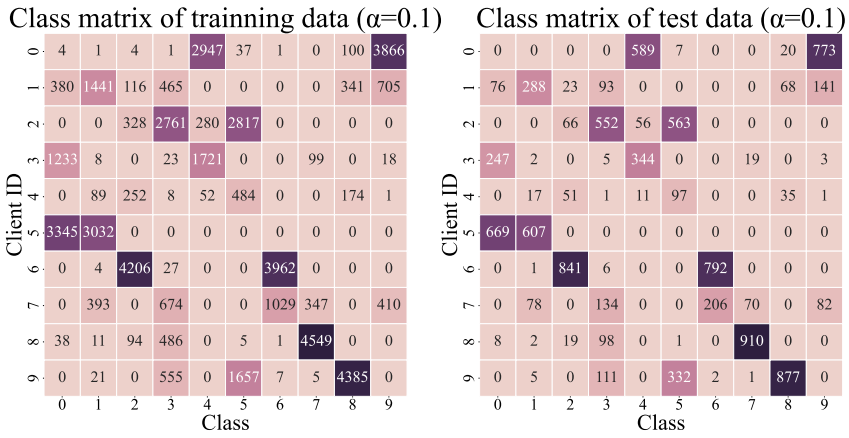

Figure 6: Data partitioning on CIFAR-10 (α=0.1)

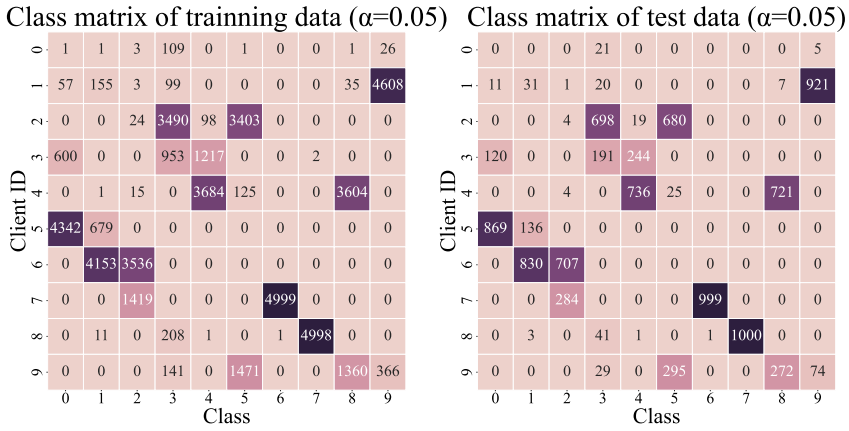

Figure 7: Data partitioning on CIFAR-10 (α=0.05)

experiments using FedRoD to testify HPFL's compatibility with other GFL methods. The results are given in Table 7. 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.

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

Table 7: 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)

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 [17], 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, 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 [85] 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. 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 [48, 37, 20, 8] 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) [13, 71, 69], 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.

Using the averaged feature to select the plug-ins is a practical way of protecting privacy as practiced in [48], 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 maintain a good performance as shown in Table 8.

Table 8: 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

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 [50].

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 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.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [No]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide proof of our theoretical results for lower bound of PM with GFL in Appendix and differential privacy in Appendix

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We detailedly describe every component of our method and experiment setup in Section 4 and Section 5.1, respectively. And we have uploaded our experiment codes for checking reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Please refers to the uploaded supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Please refer to Section 5.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Appendix C.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the potential privacy risk of our work, propose a protection scheme against the risk and provide both theoretical guarantee 4.4 and empirical verification E.2 for our privacy protection scheme.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper does not pose any risks for misusing.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: This paper has cites each dataset, baseline, model properly. Our citations for all existing assets used in this work obey CC-BY 4.0.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.