HyperFLoRA: Federated Learning with Instantaneous Personalization

Qikai Lu* Di Niu* Mohammadamin Samadi Khoshkho* Baochun Li†

Abstract

Federated learning is a decentralized approach to training machine learning models while preserving data privacy. To accommodate data heterogeneity among clients, a longstanding issue in Federated Learning, many Personalized Federated Learning (PFL) strategies decompose each client model into global modules, which are collaboratively learned by all clients and the server, and local modules, which are only trained locally on private data. While these strategies require every client to participate in training, in reality, many client devices lack sufficient data or computing resources to perform meaningful local training, making it difficult to achieve personalization for every client. In this paper, we present HyperFLoRA, a PFL framework that leverages knowledge learned from training-capable clients to enable the immediate creation of personalized models for training-incapable or new clients. HyperFLoRA uses adapters for personalization to minimize communication costs and client training workload while employing a trainable hypernetwork to generate personalized adapter weights for each client using minimal client statistical information. From experiments conducted on both convolutional and Transformer neural networks, HyperFLoRA can achieve superior model personalization performance for new clients that did not participate in training than conventional PFL methods, while significantly reducing training-related communication costs and client workload.

Keywords — Federated Learning, Model Personalization, Hypernetworks, Adapters

1 Introduction

Federated Learning (FL) can leverage massive data distributed over a multitude of clients for distributed machine learning while promoting data privacy. Conventional FL frameworks [1] delegate each client to update a copy of the global model with its private dataset; these trained client models are then aggregated into a new global model. Ideally, iteratively repeating this process amalgamates the knowledge of respective clients without directly revealing their data to the server. However, real-world FL systems are often hampered by significant data heterogeneity, where data distributions vary significantly across clients, leading to a poorly performing global model when directly deployed on every client [2].

To alleviate the impact of client data heterogeneity, Personalized Federated Learning (PFL) has been a highly studied area in recent years. A popular PFL strategy [3–5] is to decompose every client model into global and local modules. The global modules are aggregated at the server to amalgamate knowledge from all clients, while the local modules are privately trained to adapt to the respective client data distribution. However, these PFL frameworks still necessitate that all clients be training-capable (i.e., can perform local training). This constraint makes most PFL frameworks infeasible for realistic scenarios involving many client devices which are training-incapable, e.g., cellphones with insufficient computing resources, data samples, battery power, etc.

In this paper, we introduce the problem of *Instantaneously Personalized Federated Learning* (IPFL), with the goal of acquiring personalized models for training-incapable clients. For brevity, we henceforth refer to all training-capable clients as *participants*, and training-incapable clients as *bystanders*. We propose the HyperFLoRA (**Hyper**network **F**ederated Learning with **Lo**w-**R**ank **A**daptation) framework to address the IPFL problem. HyperFLoRA uses low-rank adapters [6] to adapt client models to their respective data distribution and learns a global hypernetwork to directly generate adapter weights for each client model. In designing HyperFLoRA, we make the following contributions:

- We introduce Low-Rank Adaptation (LoRA) adapters [6] into federated learning to achieve client model personalization, which significantly reduces the communication cost and participant workload induced by training in PFL.
- To enable instantaneous personalization on bystanders, we propose a learnable global hypernetwork to directly generate the low-rank adapter weights required by each client for personalization, based on its representation vector. The client representation vector is composed of minimal infor-

*University of Alberta, 116 St & 85 Ave, Edmonton, AB, Canada (qikai@ualberta.ca, dniu@ualberta.ca, msamadik@ualberta.ca).

†University of Toronto, 27 King's College Cir, Toronto, ON, Canada (bli@ece.toronto.edu).

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

mation computed from its data distribution to preserve data privacy.

- We further propose a novel client-pairing scheme as an augmentation strategy for hypernetwork training. This is critical for facilitating reliable adapter weight generation for personalized client models when the number of training-capable participants is limited.

We evaluate the effectiveness of HyperFLoRA on both convolutional and Transformer models. Additionally, we compare our framework with various PFL methods on bystander performance, participant performance, communication cost, and workload (the number of participant-trained parameters). Based on experiment results, HyperFLoRA achieves up to 11% improvement in bystander personalization over pFedHN [7] while incurring magnitudely lower communication and participant workload during training.

The rest of this paper is organized as follows. Section 2 reviews related work on PFL. Section 3 formulates the IPFL problem. Section 4 presents the HyperFLoRA design. Section 5 discusses the experiment results. Section 6 concludes the paper.

2 Related Work

In this section, we review recent literature relevant to PFL and Adapter Finetuning, as these two topics are highly related to the IPFL.

2.1 Personalized Federated Learning. Two common PFL strategies are global model personalization and model decoupling [8]. Global model personalization trains a single global model that accommodates the data distribution of all clients. This can be achieved through loss regularization. For example, FedProx [9] penalizes client models for deviating from the global model through L2 regularization. FedCL [10] uses Elastic Weight Consolidation (EWC) to account for weight importance during regularization. SCAFFOLD [11] corrects client training through the use of control variates to influence client updates. Similarly, FedDC [12] tracks and corrects client deviations by using local drift variables. FedDA [13] employs momentum-based updates to ensure better convergence stability over heterogeneous clients. MOON [14], during client model training, simultaneously minimizes its agreement with the previous client model and maximizes its agreement with the global model. FedDF [15] distills client knowledge into the global model by training with unlabelled samples annotated by client ensembling. Works such as [16–19] also explored PFL under the context of meta-learning.

Model decoupling methods separate the client model into public and private modules, where public modules undergo standard FL, and private modules are used for personalization. For example, LG-FedAvg [5] proposed to learning a compact encoder globally while allowing the model head to personalize. This is further expanded by FedBABU [4], which first pretrains the model encoder on randomly initialized heads and then finetune the heads for personalization. Work by [20], investigated the effects of simultaneous and alternated updates of public and private modules within client models. Work by [21] proposed personalization of CNN models by decoupling on convolution channels. FedRoD [22] employs two head components, one public and one private, to simultaneously train a generic and a personalized predictor.

Some model decoupling methods also permit the dynamic selection of public and private modules. PartialFed [23] introduced an adaptive partial loading scheme, where a strategy is learned to load effective layers from the global model to the client model. FedMN [24] builds client models by learning to choose sub-modules from a collection of module blocks. The pFedHN framework [7] generates the weights for entire client models. Specifically, pFedHN federatedly trains a hypernetwork to produce personalized model weights based on the client representation received. This design is highly flexible, as it theoretically allows for the extrapolation of personalized model weights for clients without necessitating client training for personalization.

2.2 Adapter Finetuning. Transformer adapters, as presented by [25] and [26], allow for efficient knowledge transfer when applying pre-trained large models to downstream tasks. LoRA, proposed by [6], improved on previous methods by using low-rank adapters to reduce computation overhead during finetuning and feedforward. Adapters have also been applied for FL, as exemplified by the works of [27] and [28].

3 Problem Formulation

IPFL extends on conventional PFL by possessing two types of clients: training-capable participants, and training-incapable bystanders. Each participant i has its own private training set $D_i = (x_n, y_n)_{n=1}^{N_i}$ and a client representation vector r_i computed from D_i. On the other hand, each bystander i only has its client representation r_i. (In actual deployment, the bystander r_i can be predicted from the expected data distribution of the client-specific task.) Collectively, we define the set of all participants as $\mathbb{C}_{part} = \{i\}_{i=1}^{M_{part}}$ and the set of all bystanders as $\mathbb{C}_{byst} = \{i\}_{i=1}^{M_{byst}}$, where M_{part} and M_{byst} are, respectively, the number of participants and

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

bystanders.

The IPFL problem is further separated into two phases: a training phase followed by a testing phase. During training, the objective is the joint reduction of training loss over all clients. Since only participants are involved in training, the learning objective is defined as

$$\min \frac{1}{M_{part}} \sum_{i=1}^{M_{part}} \frac{1}{N_i} \sum_{n=1}^{N_i} L(x_n, y_n; \theta_i = h(r_i; \psi)), \tag{3.1}$$

Here the θ_i refers to the weights of the client model. For subsequent explanations, we further denote it as the output of a weight-generation function $h(\cdot; \psi)$ when given the input r_i.

The testing phase evaluates the effectiveness of the proposed IPFL strategy. It is assessed through mean accuracy over a collection of clients. Assuming that all clients are equally important, this is defined as

$$\text{Score} = \frac{1}{M} \sum_{i=1}^{M} \text{Acc}(D_i; \theta_i = h(r_i; \psi)). \tag{3.2}$$

This evaluation can be done on either the set of all bystanders ($M = M_{byst}$) or on the set of all participants ($M = M_{part}$). Further information about the evaluation metrics is presented in Section 5.

4 Method

HyperFLoRA introduces two novelties. First, it integrates a hypernetwork function h predicated on weights ψ, and client-specific adapters ρ_i, into a PFL framework. The adapters facilitate efficient personalization when combined with a pretrained global model θ_{pret}, while the hypernetwork learns to generate personalized adapter weights for each client. Second, HyperFLoRA introduces a client-pairing augmentation mechanism to boost the personalization performance of the hypernetwork. This is pertinent in FL systems where the participants are too few to adequately train the hypernetwork. As an overview, the HyperFLoRA framework consists of repeated execution of Steps 2-4:

1. The server pretrains a global model θ_{pret} through conventional FedAvg [1] and broadcasts it to all participants.

2. The server receives r_i from some participants and sends their respective $\rho_i = h(r_i, \psi)$.

3. Each participant builds its client model θ_i from ρ_i and θ_{pret}. Afterward, only ρ_i is trained with the private dataset. This can either be done through *standard training* or *client-paired training.*

4. The server collects the trained ρ_i from all participants and updates the hypernetwork weights ψ.

The component design and details for standard and client-paired training in HyperFLoRA are presented below. The overall framework is also illustrated in Figure 1.

4.1 Component Designs.

This section describes the design and intent for implementing LoRA adapters, the hypernetwork, and the client-specific representation vector.

4.1.1 Hypernetwork.

The hypernetwork ψ used in HyperFLoRA is designed based on pFedHN [7]. Architecturally, the hypernetwork is composed entirely of linear layers (aside from activation layers) and consists of an encoder trunk and parallel out-branching single-layer regressor heads. The encoder trunk converts a received client representation r_i into a latent encoding $h_i \in \mathbb{R}^{d_\psi}$. This encoding is then passed to each model head to generate the weights for a corresponding module in the target client model θ_i. Note that, the weight count of each head scales to the number of weights in its corresponding module, Thus, even for a moderately sized model, the size of ψ can be quite large. This complicates hypernetwork training and may result in the generation of subpar θ_i. Moreover, if the hypernetwork must generate the entire θ_i, then it cannot leverage knowledge gained from model pretraining. To circumvent these problems, HyperFLoRA restricts the hypernetwork to only generate the LoRA adapter weights ρ_i.

4.1.2 LoRA Adapter.

A LoRA adapter [6] consists of two linear layers, $W_a \in \mathbb{R}^{d_a \times d_{low}}$ and $W_b \in \mathbb{R}^{d_{low} \times d_b}$, where $d_{low} \ll d_a, d_b$. To build a client model θ_i, multiple adapters are attached to selected linear layers within the global model θ_{pret}. Given an arbitrary linear weight $W \in \mathbb{R}^{d_a \times d_b}$, this is defined as

$$z = Wx + W_a W_b x. \tag{4.3}$$

By setting a small d_{low}, the weight count of the LoRA adapter, $(d_a + d_b) \times d_{low}$, should be significantly smaller than the weight count of the original linear layer, $d_a \times d_b$. This property reduces the communication cost and participant training workload in HyperFLoRA.

While its original application is for Transformer models, the LoRA adapter can theoretically be attached to any linear layer. Since client personalization is effectively a downstream task adaptation problem, client model training should emphasize adjusting the head weights. Thus, an additional LoRA adapter is attached to the model head linear layer. For later discussions, we

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

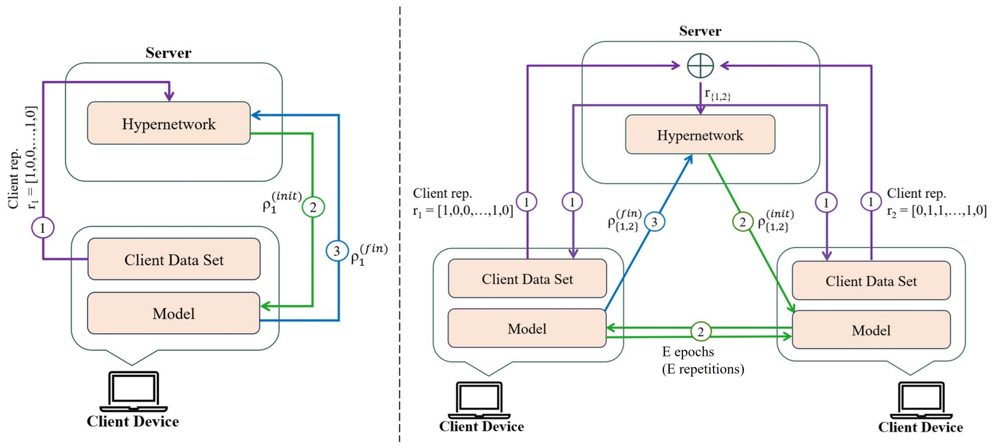

Figure 1: Diagram of HyperFLoRA. **Left** shows standard training. **Right** shows client-paired training. In both cases, the representation vector is first fed into the hypernetwork. The adapter is then sent to the relevant participant(s) for personalization. The finalized adapter is returned to the server for hypernetwork update.

further separate ρ into $\rho^{(enc)}$ for adapters in the model encoder and $\rho^{(hd)}$ for adapters in the model head.

4.1.3 Client Representation. The client representation vector is an indicator vector defined by $r_i \in \{0,1\}^K$, where K is the total number of classes. If a client i possesses data samples from class k, then the k-th element of the vector is 1. Otherwise, the element is 0. This simple design offers two advantages. First, information about the client dataset D_i provided by the indicator vector is superficial, as neither the individual data samples nor class proportions are known by the server, thereby ensuring data privacy. Second, for real-world applications, bystanders may have no or few data samples to determine its data distribution. In this case, the bystander can build its r_i using a set of classes based on the expected task.

4.2 Standard Training.

The hypernetwork is federatedly trained over multiple communication rounds. Given that all participants already possess the pre-trained model weights θ_{pret}, each round starts with the server gathering the representation vectors r_i from a randomly sampled set of participants $\mathbb{C}_{sel}$. For each r_i, a corresponding set of adapter weights $\rho_i^{(init)} = h(r_i;\psi)$ is generated and sent to participant i. Participant i then constructs its client model θ_i and trains the model using its private dataset D_i. During client training, θ_{pret} weights are frozen and only ρ_i weights are updated. Afterward, the set of finalized adapter weights $\rho_i^{(fin)}$ is sent to the server to update ψ.

The loss function for hypernetwork training is defined as a mean-squared error (MSE). Based on server interactions with a single participant i, this loss is formulated as

$$L_i = \frac{1}{2}(\rho_i^{(fin)} - \rho_i^{(init)})^2. \tag{4.4}$$

Recall that $\rho_i^{(init)} = h(r_i;\psi)$. Thus, $\rho_i^{(init)}$ is differentiable by ψ. Also, consider $\rho_i^{(fin)}$ as the target. The loss gradient computed from client i is therefore

$$g_i = \nabla_\psi h(r_i;\psi)(\rho_i^{(init)} - \rho_i^{(fin)}). \tag{4.5}$$

At any communication round, multiple participants collaborate in hypernetwork weight update. Given that hypernetwork training is facilitated through SGD, the per-round update is expressed as

$$\psi \leftarrow \psi - \gamma \frac{1}{M_{sel}} \sum_{i \in \mathbb{C}_{sel}} g_i. \tag{4.6}$$

The γ refers to the learning rate. Over several communication rounds, the hypernetwork will learn to generate better $\rho_i^{(init)}$. This also entails that the corresponding $\rho_i^{(fin)}$ should also become increasingly better personalized for participant i. Ignoring limitations due to overtraining or errors in function approximation for both ψ and θ_i, the hypernetwork training process would ideally bootstrap toward optimal personalization.

4.3 Client-Paired Training.

Ignoring the data non-IID property due to $\rho_i^{(fin)}$ changing overtime, hypernetwork training in HyperFLoRA can be viewed as a conventional supervised machine learning problem. Each client can be perceived as a single data sample $(r_i, \rho_i^{(fin)})$. The participant and bystander clients can therefore be considered as, respectively, training

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

and testing samples. This reveals an issue of data scarcity for hypernetwork training. Specifically, if the number of participants is far fewer than the number of possible variations of vector representations r, or $M_{part} \ll 2^K$, then the trained hypernetwork will likely generate poorly personalized weights for a bystander. In other words, there may be insufficient real participants for the hypernetwork to generalize from. To address this issue, we pair up participants in $\mathbb{C}_{sel}$ to form $\mathbb{C}_{pair}$. Each pair of participants i and j then creates a synthetic pseudo-client {i, j} for training.

To synthesize client representation $r_{\{i,j\}}$ of a pseudo-client, the set of all classes present in either local datasets D_i or D_j are tallied. Thereafter, half of the classes are randomly selected and preserved, with the rest discarded. Note that this is conducted using only the r_i and r_j in the server. Given that $\mathbb{K}_i$ refers to all classes present in D_i, and $\mathbb{K}_j$ for all classes in D_j, this is shown as:

$$r_{\{i,j\}} = \mathbb{1}_{k \in \text{Sample-Half}(\mathbb{K}_i \cup \mathbb{K}_j)}. \tag{4.7}$$

The representation $r_{\{i,j\}}$ is sent to both participants i and j. Each participants then respectively prepare a separate training set, $(D_i)_{\{i,j\}}$ and $(D_j)_{\{i,j\}}$, by removing samples belonging to classes not recorded under $r_{\{i,j\}}$.

Model training with the pseudo-client involves repeated exchanges of the adapters between client i and j. First, the adapter weights $\rho^{(init)}_{\{i,j\}} = h(r_{\{i,j\}}, \psi)$ is generated. The adapter weights are then sent to client i and trained using samples from $(D_i)_{\{i,j\}}$. Afterward, the adapter weights $\rho_{\{i,j\}}$ is communicated from participant i to j, and trained with $(D_j)_{\{i,j\}}$. The inter-client exchange of adapters would repeat several times, with the adapter weights alternately training on the two clients. Finally, $\rho^{(fin)}_{\{i,j\}}$ is sent to the server for hypernetwork update through Equation 4.5. With the definition that $|\mathbb{C}_{sel}| = M_{sel}$ and $|\mathbb{C}_{pair}| = M_{pair}$, the update combining both training methods can be formulated as

$$\psi \leftarrow \psi - \gamma \frac{1}{M_{sel} + M_{pair}} \Big(\sum_{i \in \mathbb{C}_{sel}} g_i + \sum_{\{i,j\} \in \mathbb{C}_{pair}} g_{\{i,j\}} \Big). \tag{4.8}$$

By exchanging the adapter weights instead of training samples, data privacy is ensured. Although repeated adapter weights exchange does incur additional communication costs, so long as the adapter has a relatively small weight count, the overall communication cost per round would still be lower than the cost of even a single exchange of the pretrained model weight.

5 Experiments

In this section, we present the experiment setup, results, and discussions.

5.1 Experiment Setup. The HyperFLoRA framework is assessed on two datasets, CIFAR10 and CIFAR100. For each dataset, we partition the samples into disjoint training, validation, and test sets. We then follow the data partition scheme introduced from the work by [1] to further divide the datasets among the $C = 100$ clients. In this scheme, the degree of data heterogeneity is inversely proportional to the partitioning hyperparameter s, which entails the number of data shards each client receives. By default, we set $s = 10$. Summarily, for both CIFAR10 and CIFAR100, each client has 450 training, 50 validation, and 100 test samples. Finally, to realize the IPFL scenario, we designate 80 clients as participants and 20 as bystanders. Note that since bystanders cannot conduct training, their training sets are discarded.

The training phase is separated into a pretraining process and a personalization process. During pretraining, the θ_{pret} is trained through conventional FedAvg with the participants. The optimal model checkpoint is identified through participant validation, where all participants assess the checkpoint θ_{pret} using their local validation set. The model with the highest averaged accuracy, as defined by Equation 3.2, is selected for the subsequent personalization phase. During personalization, only hypernetwork ψ and the client-specific adaptors ρ undergo training, with θ_{pret} fixed. The optimal hypernetwork is again determined through participant validation. Specifically, each participant receives adapter weights ρ generated by the checkpoint hypernetwork ψ, which are then used for assessment on their respective validation set. In the final testing phase, the reported scores are then computed by Equation 3.2 over the test set of the respective clients.

Additional information about the model architectures and the experiment hyperparameters are presented in the Supplementary Materials.

5.2 Assessment Metrics. The assessment metrics measured include the following. **Bystander Accuracy:** This metric is reported as the *mean* ± *std* accuracy over all bystanders. It is the main assessment metric for the IPFL problem. Note that this metric is omitted for baselines that rely on client training for personalization, as the bystanders are training-incapable. **Participant Accuracy:** This metric is also reported as *mean* ± *std*, but over all participants. It is the conventional assessment metric for PFL. The metric is used to further explain the performance of tested frame-

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

works. **Communication Cost:** The communication loss is computed as the sum of all weights communicated between the server and clients within one training round. If inter-client communication is required, all communications are assumed to be bridged through the server. For clarity, the formula for computing the costs are also provided. **Client Trainable Weight Count (CTWC):** The CTWC measures the amount of model weights that are updated during client training. Generally, a significantly smaller CTWC value entails less training workload on the participant device. **Hypernetwork Size (HS):** This metric records the number of weights in the hypernetwork. It is used to justify explanations about the performance of hypernetwork-based frameworks.

5.3 Baselines. The following baselines are implemented for this experiment.

Participant Finetuning: these baselines focus on model personalization for participants by decoupling the client model into different modules. Each module is then assigned to one of three training schemes: public, private, or frozen. Public (**Pub**) weights are both client-trained and server-aggregated. Private (**Prv**) weights are only client-trained and not server-aggregated. Frozen (**Frz**) weights are pretrained and never trained for personalization. Note that these frameworks are not applicable for bystander personalization, as they necessitate client training. However, they are useful for identifying the performance upper bound of HyperFLoRA for analysis purposes.

FedProx [9]: This baseline is used to assess the performance of a regularization-based framework for both participant and bystander personalization.

pFedHN [7]: Since bystanders are training-incapable, most PFL strategies cannot be adequately applied to the IPFL problem. Thus, we mainly compare HyperFLoRA against pFedHN. The original implementation only uses one participant per round for hypernetwork training, which differs from conventional FL schemes that sample a group of participants per round. Thus we also assess a modified **pFedHN-Parallel** design that, in each round, aggregates gradients from a sampled group of participants for hypernetwork update.

HyperFLoRA Variants: We derive variations of the HyperFLoRA frameworks to assess the validity of certain design decisions and to check for potential improvements. These include the following. **HyperFLoRA w/o CP** where the client-paired training is removed. **HyperFLoRA Seq**, where the hypernetwork follows the same sequential gradient update scheme used by pFedHN. **HyperFLoRA+$\theta^{(hd)}$**, where the hypernetwork learns to generate personalized weights for both $\rho^{(enc)}$ and $\theta^{(hd)}$; the $\rho^{(hd)}$ is consequently unused.

5.4 Preliminary Experiments. We first deployed a LeNet-5 [29] model as θ to verify the feasibility of the HyperFLoRA design. Results from Table 1 confirm that HyperFLoRA is indeed effective for IPFL. The results from ρ-Priv+θ-Frz demonstrate that training the LoRA adapters alone while freezing the global model can achieve good personalization for the participants. The participant accuracy of pFedHN, while notably worse than ρ-Priv+θ-Frz is nevertheless better than that of Pretrain. This suggests that pFedHN can generate moderately personalized weights for each participant. However, the pFedHN bystander accuracy is worse than that of Pretrain, showing that this method alone is ineffective for IPFL. On the other hand, HyperFLoRA simultaneously achieved superior bystander accuracy over pFedHN and approaches the optimal participant accuracy of ρ-Priv+θ-Frz. This shows that HyperFLoRA effectively combines the hypernetwork and LoRA components, and performs well for both PFL and IPFL. Finally, we note that FedProx does not show any notable improvement over Pretrain, and is thus unsuitable for either participant or bystander personalization. Based on the success achieved in the preliminary experiment, additional experiments, which replace LeNet-5 with ViT [30], are then conducted.

5.5 General Discussion. Summarily, it is observed that HyperFLoRA, or one of its derivative designs, can achieve superior bystander (IPFL) score while approaching the optimal participant (PFL) score. We discuss the results for CIFAR10 and CIFAR100 separately due to differences between the datasets.

5.5.1 CIFAR10. HyperFLoRA significantly outperforms pFedHN and pFedHN-Parallel, with an 11% improvement in bystander accuracy and an 8% improvement in participant accuracy. This observation can be explained in two ways. First, since the hypernetwork in HyperFLoRA only generates the adapter weights, it can directly leverage the knowledge contained in the pretrained global weights θ_{pret}. In contrast, since pFedHN must repeatedly generate new weights for the entire θ_{pret} for hypernetwork training, it must learn to approximate this knowledge. Compared to HyperFLoRA, pFedHN thus possesses more sources (model weights) where approximation errors within the hypernetwork could be introduced to the client model, thereby leading to inferior personalization. Second, the weight count of ψ in pFedHN far exceeds that of HyperFLoRA to accommodate for the larger output space. This introduces greater complexity during hypernetwork training, espe-

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

Table 1: Results with LeNet-CIFAR10

Framework	**Bystander Acc.**	**Participant Acc.**	**Communication Cost**	**CTWC**	**HS**
Pretrain	70.75 ± 6.96	68.94 ± 7.57	1.03E6 $= 2M_{sel}\|\theta\|$	6.41E4	-
ρ-Prv+θ-Frz	-	78.00 ± 7.21	0	8.50E2	-
FedProx	70.75 ± 6.74	69.42 ± 6.76	1.03E6 $= 2M_{sel}\|\theta\|$	6.41E3	-
pFedHN	69.45 ± 6.95	72.05 ± 6.83	1.03E6 $= M_{sel}\|r\| + 2M_{sel}\|\theta\|$	6.41E4	8.30E6
HyperFLoRA	**78.30 ± 5.47**	**77.86 ± 6.40**	**5.46E4** $= 2M_{sel}\|r\| + (2M_{sel} + 4EM_{pair})\|\rho\|$	**8.50E2**	**1.44E5**

Table 2: Results with ViT-CIFAR10

Framework	**Bystander Acc.**	**Participant Acc.**	**Communication Cost**	**CTWC**	**HS**
Pretrain	69.25 ± 5.99	68.10 ± 6.07	3.33E6 $= 2M_{sel}\|\theta\|$	2.08E5	-
$\theta^{(hd)}$-Prv+$\theta^{(enc)}$-Pub	-	75.96 ± 5.90	3.32E6 $= 2M_{sel}\|\theta^{(enc)}\|$	2.08E5	-
$\theta^{(enc)}$-Prv+$\theta^{(hd)}$-Pub	-	70.99 ± 6.53	1.04E4 $= 2M_{sel}\|\theta^{(hd)}\|$	2.08E5	-
θ-Prv	-	71.33 ± 6.67	0	2.08E5	-
ρ-Prv+θ-Frz	-	75.16 ± 5.70	0	2.12E3	-
FedProx	70.85 ± 5.94	67.94 ± 5.77	3.33E6 $= 2M_{sel}\|\theta\|$	2.08E5	-
pFedHN	61.00 ± 6.34	64.97 ± 8.09	3.33E6 $= M_{sel}\|r\| + 2M_{sel}\|\theta\|$	2.08E5	2.69E7
pFedHN-Parallel	64.55 ± 6.55	68.09 ± 7.34	3.33E6 $= M_{sel}\|r\| + 2M_{sel}\|\theta\|$	2.08E5	2.69E7
HyperFLoRA-noCP	75.15 ± 6.64	76.33 ± 6.05	**3.40E4** $= M_{sel}\|r\| + 2M_{sel}\|\rho\|$	**2.12E3**	**3.08E5**
HyperFLoRA-Seq	74.30 ± 6.42	76.11 ± 5.92	1.36E5 $= 2M_{sel}\|r\| + (2M_{sel} + 4EM_{pair})\|\rho\|$	**2.12E3**	**3.08E5**
HyperFLoRA+$\theta^{(hd)}$	**76.20 ± 5.95**	**76.53 ± 6.32**	1.73E5 $= 2M_{sel}\|r\| + (2M_{sel} + 4EM_{pair})(\|\rho^{(enc)}\| + \|\theta^{(hd)}\|)$	2.70E3	3.82E5
HyperFLoRA	75.35 ± 6.48	75.92 ± 6.15	1.36E5 $= 2M_{sel}\|r\| + (2M_{sel} + 4EM_{pair})\|\rho\|$	**2.12E3**	**3.08E5**

cially given that the number of client representation r_i is scarce due to the limited number of clients. This results in poorly trained ψ weights, which results in inferior personalization by pFedHN. Finally, we remark that the communication cost per round of HyperFLoRA is smaller than pFedHN by magnitudes, thus further demonstrating our design as an improvement.

We further compared the derivative HyperFLoRA designs with respect to the original HyperFLoRA. First, we note that the HyperFLoRA-Seq yields slightly worse bystander accuracy, despite its participant accuracy being comparable to that of HyperFLoRA. This suggests that the client-sequential training scheme, as was employed in the original pFedHN, is inferior at extrapolating personalized weights for new bystanders. Second, at least for CIFAR10, removing the client-paired training process in HyperFLoRA did not result in notable performance degradation. This is explained by the relative simplicity of the CIFAR10 dataset due to only having 10 classes. Given the shards per client s is set to 10, the data distribution between different clients would be relatively similar (but still sufficiently different to necessitate personalization), thus rendering the improvement from client-paired training relatively marginal. Finally, the leading performance of HyperFLoRA+$\theta^{(hd)}$ indicates that better performance could be achieved by training the head and adapter weights jointly, with the tradeoff being an increase in the communication cost. Note that this design is more feasible for models with fewer $\theta^{(hd)}$ weights (a smaller head). Otherwise, the communication of a large $\theta^{(hd)}$ would incur significantly more communication cost than only finetuning the head adapter $\rho^{(hd)}$.

5.5.2 CIFAR100.

Since CIFAR100 is more complex due to possessing 100 classes, the overall accuracy scores of all frameworks are noticeably lower. Furthermore, a stronger discrepancy between bystander and participant accuracy is observed. This is because the greater diversity between client data distributions, and consequently variations in representation vectors, renders the learning and extrapolation of personalized model weights significantly harder for the hypernetwork. Nevertheless, HyperFLoRA achieved superior results over pFedHN and pFedHN-Parallel, with at least an 8% difference in bystander accuracy and a 3% difference in participant accuracy. Additionally, HyperFLoRA communication cost is at least one magnitude smaller than pFedHN. Interestingly, pFedHN performed better than pFedHN-Parallel in CIFAR100 for bystanders, which is opposite to the observation made in CIFAR10.

Analyzing the performance of the derivative designs with respect to the original HyperFLoRA, we make the following remarks. First, the inclusion of client-paired training in this scenario results in a notable gain of 2% in bystander accuracy. Combined with observations on CIFAR10, this suggests that client-paired training is best applied in scenarios where the diversity of client representations (per client data distributions) far exceeds the number of actual participants. Second, the slightly inferior performance by HyperFLoRA-Seq suggests that, overall, sequential hypernetwork update does not result in any notable benefits on HyperFLoRA performance, despite such observations being made for

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

Table 3: Results with ViT-CIFAR100

Framework	**Bystander Acc.**	**Participant Acc.**	**Communication Cost**	**CTWC**	**HS**
Pretrain	30.10 ± 6.07	31.11 ± 7.29	$3.42\text{E}6 = 2M_{sel}\lvert\theta\rvert$	2.14E5	-
$\theta^{(hd)}$-Prv+$\theta^{(enc)}$-Pub	-	65.97 ± 6.52	$3.32\text{E}6 = 2M_{sel}\lvert\theta^{(enc)}\rvert$	2.14E5	-
$\theta^{(enc)}$-Prv+$\theta^{(hd)}$-Pub	-	65.75 ± 6.33	$1.04\text{E}5 = 2M_{sel}\lvert\theta^{(hd)}\rvert$	2.14E5	-
θ-Prv	-	66.17 ± 5.78	0	2.14E5	-
ρ-Prv+θ-Frz	-	64.10 ± 6.66	0	2.21E3	-
FedProx	30.60 ± 6.07	31.01 ± 6.40	$3.42\text{E}6 = 2M_{sel}\lvert\theta\rvert$	2.14E5	-
pFedHN	36.35 ± 8.31	60.00 ± 6.86	$3.42\text{E}6 = M_{sel}\lvert r\rvert + 2M_{sel}\lvert\theta\rvert$	2.14E5	2.76E7
pFedHN-Parallel	30.20 ± 6.31	60.54 ± 6.88	$3.42\text{E}6 = M_{sel}\lvert r\rvert + 2M_{sel}\lvert\theta\rvert$	2.14E5	2.76E7
HyperFLoRA-noCP	42.80 ± 9.01	62.83 ± 5.99	**3.62E4** $= M_{sel}\lvert r\rvert + 2M_{sel}\lvert\rho\rvert$	**2.21E3**	**3.31E5**
HyperFLoRA-Seq	44.30 ± 8.20	63.08 ± 6.29	$1.43\text{E}5 = 2M_{sel}\lvert r\rvert + (2M_{sel} + 4EM_{pair})\lvert\rho\rvert$	**2.21E3**	**3.31E5**
HyperFLoRA+$\theta^{(hd)}$	44.50 ± 8.26	62.91 ± 6.06	$5.49\text{E}5 = 2M_{sel}\lvert r\rvert + (2M_{sel} + 4EM_{pair})(\lvert\rho^{(enc)}\rvert + \lvert\theta^{(hd)}\rvert)$	8.55E3	1.15E6
HyperFLoRA	**44.95 ± 8.50**	**63.38 ± 5.90**	$1.43\text{E}5 = 2M_{sel}\lvert r\rvert + (2M_{sel} + 4EM_{pair})\lvert\rho\rvert$	**2.21E3**	**3.31E5**

the pFedHN results. Finally, the inclusion of model head for hypernetwork weight generation is not effective in more complex scenarios, contrary to the leading performance achieved in CIFAR10.

Comparing the participant accuracy of HyperFLoRA with those of the Participant Finetuning frameworks, HyperFLoRA performed only slightly worse. Furthermore, we note that ρ-Prv+θ-Frz is also worse than θ-Prv by about 2%. Along with the pFedHN results, these results reveal two constraints limiting the personalization performance. The first constraint is the use of only adapters for personalization. Intuitively, using well-trained adapters with a frozen model is inferior to well-trained weights for an entire model, as the former have far fewer weights available to accommodate the client data distribution. The second constraint lies in the use of hypernetwork to approximate the adapter weights in the client model. Given the limited number of participant clients and the relatively large output space, the hypernetwork-generated ρ weights are inferior to optimally trained ρ weights regardless of the integration of client-paired training or other client augmentation strategies. These two constraints must be addressed for future works to achieve further improvements in both bystander and participant personalization.

6 Conclusion

In this paper, we introduce and investigate the problem of Instantaneous Personalized Federated Learning, with the objective of leveraging training-capable *participant* clients to create personalized models for training-incapable and resource-constrained *bystander* clients. To solve this problem, we present the HyperFLoRA framework, which relies on participants to train a hypernetwork to directly generate personalized models for any given client. To minimize the communication cost and computation workload incurred during the training process, the hypernetwork learns to only generate weights for the LoRA adapter used in model personalization. Furthermore, to address the problem of limited number of participants, additional pseudo-clients are created through client-pairing to serve as an augmentation strategy for hypernetwork training. From experiments on both convolutional and Transformer models, HyperFLoRA is shown to be superior to baseline methods on bystander personalization (IPFL) performance, approaches optimal participant personalization when compared to conventional PFL, and minimizes the overall communication cost as well as the number of model parameters trained by each participant.

References

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

[2] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 1698–1707. IEEE, 2020.

[3] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *International conference on machine learning*, pages 2089–2099. PMLR, 2021.

[4] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. Fedbabu: Towards enhanced representation for federated image classification. *arXiv preprint arXiv:2106.06042*, 2021.

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

[6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited

Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[7] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, pages 9489–9502. PMLR, 2021.

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

[9] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[10] Xin Yao and Lifeng Sun. Continual local training for better initialization of federated models. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1736–1740. IEEE, 2020.

[11] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

[12] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10112–10121, 2022.

[13] Jiayin Jin, Jiaxiang Ren, Yang Zhou, Lingjuan Lyu, Ji Liu, and Dejing Dou. Accelerated federated learning with decoupled adaptive optimization. In *International Conference on Machine Learning*, pages 10298–10322. PMLR, 2022.

[14] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.

[15] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

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

[17] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

[18] Canh T Dinh, Nguyen Tran, and Josh Nguyen. Personalized federated learning with moreau envelopes. *Advances in Neural Information Processing Systems*, 33:21394–21405, 2020.

[19] Shiyu Liu, Shaogao Lv, Dun Zeng, Zenglin Xu, Hui Wang, and Yue Yu. Personalized federated learning via amortized bayesian meta-learning. *arXiv preprint arXiv:2307.02222*, 2023.

[20] Krishna Pillutla, Kshitiz Malik, Abdel-Rahman Mohamed, Mike Rabbat, Maziar Sanjabi, and Lin Xiao. Federated learning with partial model personalization. In *International Conference on Machine Learning*, pages 17716–17758. PMLR, 2022.

[21] Yiqing Shen, Yuyin Zhou, and Lequan Yu. Cd2-pfed: Cyclic distillation-guided channel decoupling for model personalization in federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10041–10050, 2022.

[22] Hong-You Chen and Wei-Lun Chao. On bridging generic and personalized federated learning for image classification. *arXiv preprint arXiv:2107.00778*, 2021.

[23] Benyuan Sun, Hongxing Huo, Yi Yang, and Bo Bai. Partialfed: Cross-domain personalized federated learning via partial initialization. *Advances in Neural Information Processing Systems*, 34:23309–23320, 2021.

[24] Tianchun Wang, Wei Cheng, Dongsheng Luo, Wenchao Yu, Jingchao Ni, Liang Tong, Haifeng Chen, and Xiang Zhang. Personalized federated learning via heterogeneous modular networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1197–1202. IEEE, 2022.

[25] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[26] Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. *arXiv preprint arXiv:2004.03829*, 2020.

[27] Peyman Passban, Tanya Roosta, Rahul Gupta, Ankit Chadha, and Clement Chung. Training mixed-domain translation models via federated learning. *arXiv preprint arXiv:2205.01557*, 2022.

[28] Dongqi Cai, Yaozong Wu, Shangguang Wang, Felix Xiaozhu Lin, and Mengwei Xu. Fedadapter: Efficient federated learning for modern nlp. *arXiv preprint arXiv:2205.10162*, 2022.

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

[30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Copyright © 2024 by SIAM
Unauthorized reproduction of this article is prohibited