Layer-wised Model Aggregation for Personalized Federated Learning

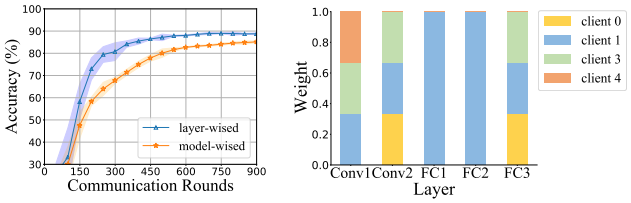
Xiaosong Ma[†,1], Jie Zhang[†,1], Song Guo[*,1,2], and Wenchao Xu[1]
[1]Department of Computing, The Hong Kong Polytechnic University
[2]The Hong Kong Polytechnic University Shenzhen Research Institute
jieaa.zhang@connect.polyu.hk, maxiaosong16@gmail.com,
{song.guo,wenchao.xu}@polyu.edu.hk

Abstract

Personalized Federated Learning (pFL) not only can capture the common priors from broad range of distributed data, but also support customized models for heterogeneous clients. Researches over the past few years have applied the weighted aggregation manner to produce personalized models, where the weights are determined by calibrating the distance of the entire model parameters or loss values, and have yet to consider the layer-level impacts to the aggregation process, leading to lagged model convergence and inadequate personalization over non-IID datasets. In this paper, we propose a novel pFL training framework dubbed Layer-wised Personalized Federated learning (pFedLA) that can discern the importance of each layer from different clients, and thus is able to optimize the personalized model aggregation for clients with heterogeneous data. Specifically, we employ a dedicated hypernetwork per client on the server side, which is trained to identify the mutual contribution factors at layer granularity. Meanwhile, a parameterized mechanism is introduced to update the layer-wised aggregation weights to progressively exploit the inter-user similarity and realize accurate model personalization. Extensive experiments are conducted over different models and learning tasks, and we show that the proposed methods achieve significantly higher performance than state-of-the-art pFL methods.

1. Introduction

Federated learning (FL) has emerged as a prominent collaborative machine learning framework to exploit inter-user similarities without sharing the private data [33, 43, 52]. When users' datasets are non-IID (independent and identically distributed), i.e., the inter-user distances are large [23, 53], sharing a global model for all clients may lead to slow convergence or poor inference performance as the model may significantly deviate from their local data [14, 56].



(a) (b)

Figure 1. A toy example: Layer-wised vs. Model-wised aggregation method. (a) Model performance of client 1. Both of two methods perform similarity-based personalized aggregation. i.e., layer-wised: perform personalized aggregation by calculating the similarity between layers; model-wised: perform personalized aggregation by calculating the similarity between models. (b) The weight of each layer for client 1 in the last communication round.

To deal with such statistical diversity, personalized federated learning (pFL) mechanisms are proposed to allow each client to train a customized model to adapt to their own data distribution [9, 12, 15, 22]. Literature status quo to achieve pFL include the data-based approaches, i.e., smoothing the statistical heterogeneity among clients' datasets [8, 16], the single-model approaches, e.g., regularization [22, 41], meta-learning [9], parameter decoupling [5, 24, 26], and the multiple-model ways, i.e., train personalized models for each client [15, 54], which can produce personalized models for each client via weighted combinations of clients' models. Existing pFL methods apply a distance metric among the whole model parameters or loss values of different clients, which is insufficient to exploit their heterogeneity since the overall distance metric cannot always reflect the importance of each local model and can lead to inaccurate combining weights or unbalance contribution from non-IID distributed datasets, and thus prevent further personalization for clients at scale. The main reason is that different layers of a neural network can have different util-

[†]Equal contribution
[*]Corresponding author

ities, e.g., the shallow layers focus more on local feature extraction, while the deeper layers are for extracting global features [6, 20, 21, 47, 49]. Measuring the model distances would ignore such layer-level differences, and cause inaccurate personalization that hinders the pFL training efficiency.

In this paper, we propose a band-new pFL framework that can realize the layer-level aggregation for FL personalization, which can accurately recognize the utility of each layer from clients' model for adequate personalization, and thus can improve the training performance over non-IID datasets. A toy example is presented to illustrate that traditional model-level aggregation based pFL method fails in reflecting the inner relationship among all local models, which motivates us to exploit an effective way to discern the layer-level impacts during the pFL training procedure.

Observation of Layer-wised Personalized Aggregation. In the toy example, we consider six clients to collaboratively learn their personalized models for a nine-class classification task. The average model accuracy is obtained via both the layer-wised and model-wised aggregation approaches, which utilize the inter-layer and inter-model similarities respectively. Figure 1 shows that higher model accuracy can be achieved by the layer-wised approach comparing with the model-wised one for a certain client. The weights of layers for this client after the last communication round are also plotted, and we show that applying different weights for different layers, e.g., the first and second fully-connected layer (i.e., FC1, FC2) on client 1 have larger weights, while the second convolution layer, i.e., Conv1 layer has smaller weights, can produce significant performance gain for the personalized model accuracy.

The toy example demonstrates the potential of the layer-wised aggregation to achieve higher performance than traditional model based pFL methods, since the layer-level similarities can reflect more accurate correlation among clients. By exploiting such layer-wised similarity and identifying the layer-level inter-user contribution, it is promising to produce efficient and effective personalized models for all clients. Motivated by such observation, we propose a novel federated training framework, namely, pFedLA, which adaptively facilitates the underlying collaboration between clients in a layer-wised manner. Specifically, at the server side, we introduce a dedicated hypernetwork for each client to learn the weights of cross-clients' layers during the pFL training procedure, which is shown to effectively boost the personalization over non-IID datasets. Extensive experiments are conducted, and we demonstrate that the proposed pFedLA can achieve higher performance than the state-of-the-art baselines over widely used models and datasets, i.e., EMNIST, FashionMNIST, CIFAR10 and CIFAR100. The contributions of the paper are summarized as follows:

- To the best of our knowledge, this paper is the first to explicitly reveal the benefits of layer-wised aggregation comparing with model-wised approaches in pFL among heterogeneous FL clients;
- We propose a layer-wised personalized federated learning (pFedLA) training framework that can effectively exploit the inter-user similarities among clients with non-IID data and produce accurate personalized models;
- We conduct extensive experiments on four typical image classification tasks, which demonstrated the superior performance of pFedLA over the state-of-the-art approaches.

2. Related Work

2.1. Personalized Federated Learning

Recently, various approaches have been proposed to realize pFL, which can be classified into the data-based and the model-based categories. Data-based approaches focus on reducing the statistical heterogeneity among clients' datasets to boost the model convergence, while model-based approaches emphasize on producing customized model structures or parameters for different clients.

The typical way of data-based pFL is to share a small amount of global data to each client [56]. Jeong et al. [8, 16] focus on data augmentation methods by generating additional data to augment its local data towards yielding an IID dataset. However, these methods usually require the FL server to know the statistical information about clients' local data distributions (e.g., class sizes, mean and standard deviation), which may potentially violate privacy policy [42]. Another line of work considers to design client selection mechanisms to approach homogeneous data distribution [30, 45, 48].

Model-based pFL methods can also be divided into two types: single-model, multiple-model approaches. Single-model based methods extended from the conventional FL algorithms like FedAvg [33] combine the optimization of the local models and global model, which consist of five different kinds of approaches: local fine-tuning [1, 36, 46], regularization [12, 13, 41], model mixture [7, 32], meta learning [9, 18] and parameter decomposition [1, 4, 5]. Considering the diversity and inherent relationship of local data, a multi-model-based approach where multiple global models are trained for heterogeneous clients is more suitable. Some researchers [10, 15, 32] propose to train multiple global models at the server, where similar clients are clustered into several groups and different models are trained for each group. Another strategy is to collaboratively train a personalized model for each individual client, e.g., FedAMP [15], FedFomo [54], MOCHA [39], KT-pFL [51] etc.

These literatures treat each client's model as a whole entity, and has yet to consider the layer-wised utility for per-

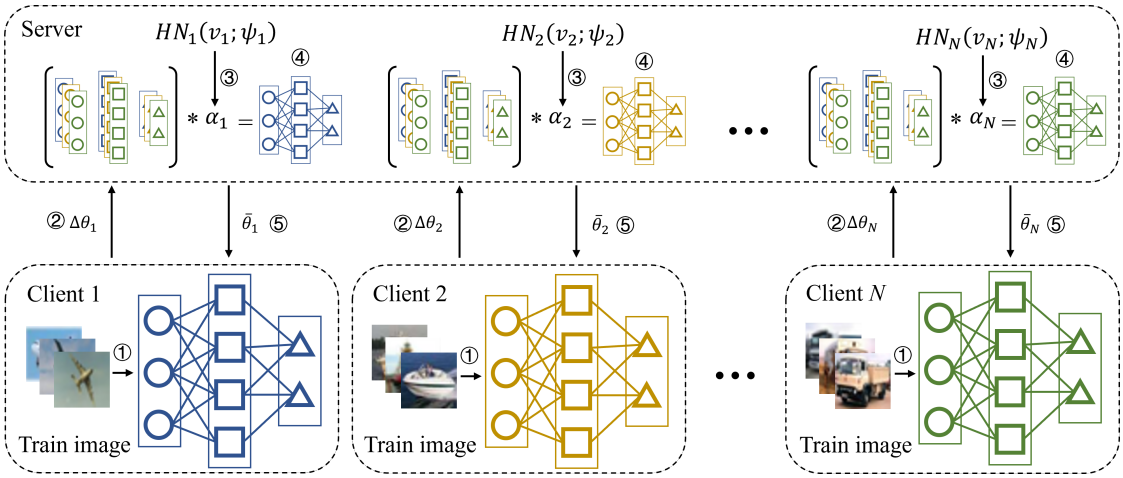


Figure 2. Framework of pFedLA. The workflow contains 5 steps: ① local training on private data; ② each client sends the update of parameters $\Delta\theta_i$ to the server; ③ the server updates the aggregation weight matrix α_i by hypernetworks $HN_i(v_i;\psi_i)$ according to $\Delta\theta_i$; ④ the server performs weighted aggregation and outputs personalized model $\bar{\theta}_i$ for the corresponding client; ⑤ each client downloads the personalized model $\bar{\theta}_i$.

sonalized aggregation. The distance metric for describing the similarity among models is inaccurate and can lead to sub-optimal performance, which motivates us to explore a fine-grained aggregation strategy to adapt to broad range of non-IID clients.

2.2. Hypernetworks

Hypernetworks [11] are used to generate parameters of other neural networks, e.g., a target network, by mapping the embeddings of the target tasks to corresponding model parameters. Hypernetworks have been widely used in various machine learning applications, such as language modeling [35,40], computer vision [17,19,27], 3D scene representation [28, 38], hyperparameter optimization [2, 25, 29, 31], neural architecture search (NAS) [3, 50], continual learning [44] and meta-learning [55]. Shamsian et al. [37] is the first to apply hypernetworks in FL, which can generate effective personalized model parameters for each client. We show that hypernetworks are capable to evaluate the importance of each model layer, and can boost the personalized aggregation in non-IID scenarios.

3. Method

In this section, we present the design of the pFedLA framework that applies the hypernetworks to conduct layer-wised personalized aggregation, which is shown in Figure 2.

3.1. Problem Formulation

In pFL, the goal is to collaboratively train personalized models among multiple clients while keeping their local data private. Considering N clients with non-IID datasets, let $\mathcal{D}_i = \{(x_j^{(i)}, y_j^{(i)})\}_{i=1}^{m_i}$ $(1 \leq i \leq N)$ be the dataset on the i-th client, where x_j is the j-th input data sample, y_j is the corresponding label. The size of the datasets on the i-th client is denoted by m_i. The size of all clients' datasets is $M = \sum_{i=1}^{N} m_i$. Let θ_i represent the model parameters of client i, the objective of pFL can be formulated as

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{N} \frac{m_i}{M} \mathcal{L}_i(\theta_i), \tag{1}$$

where

$$\mathcal{L}_i(\theta_i) = \frac{1}{m_i} \sum_{j=1}^{m_i} \mathcal{L}_{CE}(\theta_i; x_j^{(i)}, y_j^{(i)}) \tag{2}$$

where $\Theta = \{\theta_i, \ldots, \theta_N\}$ is the set of personalized parameters for all clients. $\mathcal{L}_i$ is loss function of i-th client associated with dataset $\mathcal{D}_i$. The difference between the predicted value and the true label of data samples is measured by $\mathcal{L}_{CE}$, which is the cross-entropy loss.

3.2. pFedLA Algorithm

In this section, we present our proposed pFL algorithm pFedLA, which evaluates the importance of each layer from different clients to achieve layer-wised personalized model aggregation. We apply a dedicated hypernetwork for each client on the server and train them to generate aggregation weights for each model layer of different clients. It can be seen from Figure 2 that, unlike the general FL framework that generates only one global model, pFedLA maintains a personalized model for each client at the server. Clients with similar data distribution should have high aggregation weights to reinforce the mutual contribution from each other. Our pFedLA applies a set of aggregation weight ma-

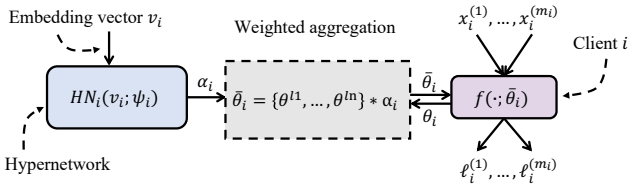


Figure 3. Illustration of one hypernetwork framework used in pFedLA. The hypernetwork HN_i takes the embedding vector v_i as input, and outputs the aggregation weight matrix α_i. After the weighted combination with intermediate parameters $\{\theta^{l1},\dots,\theta^{ln}\}$ and aggregation weight matrix α_i, client i can make local training on private data. Note that both v_i and ψ_i are updated during training.

trix α_i at the server side to progressively exploit the inter-user similarities at layer level, which is defined as

$$\alpha_i = \left[\alpha_i^{l1}, \alpha_i^{l2}, \dots, \alpha_i^{ln}\right] = \begin{bmatrix} \alpha_i^{l1,1} & \alpha_i^{l2,1} & \cdots & \alpha_i^{ln,1} \\ \alpha_i^{l1,2} & \alpha_i^{l2,2} & \cdots & \alpha_i^{ln,2} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_i^{l1,N} & \alpha_i^{l2,N} & \cdots & \alpha_i^{ln,N} \end{bmatrix} \tag{3}$$

where α_i^{ln} represents the aggregation weight vector of n-th layer in client i, while $\alpha_i^{ln,N}$ represents the aggregation weight for client N in n-th layer. For all n layers, $\sum_{j=1}^{N} \alpha_i^{ln,j} = 1$.

Different with previous pFL algorithms, instead of applying identical weight values for all layers of a client model, pFedLA considers the different utilities of neural layers, and assign a unique weight to each of them to achieve fine-grained personalized aggregation. In addition, unlike traditional methods that mathematically calculate the weights using a distance metric among the entire model parameters [15, 54], pFedLA parameterized the weights during the training phase via a set of dedicated hypernetworks. The layer-wised weights are determined by the hypernetworks, which are alternatively updated with the personalized model. Such way we can obtain effective weights as their update direction is in line with the optimization direction of the objective function. In the following, we will elaborate the updating process of the aggregation weight matrix α of pFedLA.

Each hypernetwork consists of several fully connected layers, whose input is an embedding vector that is automatically updated with the model parameters, and the output is the weight matrix α. Define the hypernetwork on client i as

$$\alpha_i = HN_i(v_i;\psi_i), \tag{4}$$

where v_i is the embedding vector and ψ_i is the parameter of client i's hypernetwork (i.e., Figure 3). Let $\{\theta^{l1},\theta^{l2},\dots,\theta^{ln}\}$ be the intermediate parameters of all clients after local training, $\theta^{ln} = \{\theta_1^{ln},\theta_2^{ln},\dots,\theta_N^{ln}\}$ is the

Algorithm 1 pFedLA Algorithm

Input: dataset $\{\mathcal{D}_1,\mathcal{D}_2,\dots,\mathcal{D}_N\}$, learning rate η. Total communication rounds T.
Output: Trained personalized models $\{\bar{\theta}_1,\bar{\theta}_2,\dots,\bar{\theta}_N\}$.

Initialize the clients' model parameters, hypernetworks parameters and embedding vectors.
procedure SERVER EXECUTES
  for each communication round $t \in \{1,\dots,T\}$ **do**
    for each client i **in parallel do**
      $\bar{\theta}_i^{(t+1)} = \{\theta^{l1},\dots,\theta^{ln}\} * HN_i(v_i^{(t)};\psi_i^{(t)})$
      $\Delta\theta_i \leftarrow ClientUpdate(\bar{\theta}_i^{(t+1)})$
      Update $\{\theta^{l1},\theta^{l2},\dots,\theta^{ln}\}$ according to $\Delta\theta_i$
      Update $v_i^{(t+1)}$ and $\psi_i^{(t+1)}$ via Eq. 10, 11
procedure CLIENTUPDATE($\bar{\theta}_i^{(t+1)}$)
  Client i receives $\bar{\theta}_i^{(t+1)}$ from the server.
  Set $\theta_i = \bar{\theta}_i^{(t+1)}$.
  for each local epoch **do**
    for mini-batch $\xi_t \subseteq \mathcal{D}_i$ **do**
      Local Training: $\theta_i = \theta_i - \eta\nabla_{\theta_i}\mathcal{L}_i(\theta_i;\xi_t)$
  return $\Delta\theta_i = \theta_i - \bar{\theta}_i^{(t+1)}$

set of n-th layer of all clients, where θ_N^{ln} are the parameters of n-th layer in client N. In pFedLA, the model parameters of client i is obtained by weighted aggregation according to α_i:

$$\bar{\theta}_i = \{\bar{\theta}_i^{l1},\bar{\theta}_i^{l2},\dots,\bar{\theta}_i^{ln}\} = \{\theta^{l1},\theta^{l2},\dots,\theta^{ln}\} * \alpha_i, \tag{5}$$

where $\bar{\theta}_i^{ln}$ can also be expressed as:

$$\bar{\theta}_i^{ln} = \sum_{j=1}^{N} \theta_j^{ln}\alpha_i^{ln,j}. \tag{6}$$

Thus the objective function of pFedLA can be derived from Eq. 1 to

$$\arg\min_{V,\Psi} \sum_{i=1}^{N} \frac{m_i}{M}\mathcal{L}_i(\{\theta^{l1},\theta^{l2},\dots,\theta^{ln}\} * HN_i(v_i;\psi_i)) \tag{7}$$

where $V = \{v_1,\dots,v_N\}$, $\Psi = \{\psi_1,\dots,\psi_N\}$. Consequently, pFedLA transforms the optimization problem for client parameters θ_i into the hypernetwork's embedding vector v_i and parameters ψ_i. In the following, we introduce the update rules of V and Ψ.

Update v_i and ψ_i. According to the chain rule, we can have the gradient of v_i and ψ_i from Eq. 7:

$$\begin{aligned} \nabla_{v_i}\mathcal{L}_i &= (\nabla_{v_i}\bar{\theta}_i)^T\nabla_{\bar{\theta}_i}\mathcal{L}_i \\ &= [\{\theta^{l1},\theta^{l2},\dots,\theta^{ln}\} * \nabla_{v_i}HN_i(v_i;\psi_i)]^T\nabla_{\bar{\theta}_i}\mathcal{L}_i, \end{aligned} \tag{8}$$

$$\begin{aligned} \nabla_{\psi_i}\mathcal{L}_i &= (\nabla_{\psi_i}\bar{\theta}_i)^T\nabla_{\bar{\theta}_i}\mathcal{L}_i \\ &= [\{\theta^{l1},\theta^{l2},\dots,\theta^{ln}\} * \nabla_{\psi_i}HN_i(v_i;\psi_i)]^T\nabla_{\bar{\theta}_i}\mathcal{L}_i. \end{aligned} \tag{9}$$

Algorithm 2 HeurpFedLA Algorithm

Input: dataset $\{D_1, D_2, \ldots, D_N\}$, learning rate η. Total communication rounds T.
Output: Trained personalized models $\{\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_N\}$.

Initialize the clients' model parameters, hypernetworks parameters and embedding vectors.
procedure SERVER EXECUTES
  for each communication round $t \in \{1, \ldots, T\}$ **do**
    for each client i **in parallel do**
      $\bar{\theta}_i^{(t+1)} = \{\theta^{l1}, \ldots, \theta^{ln}\} * HN_i(v_i^{(t)}; \psi_i^{(t)})$
      Sort $\{\alpha_i^{l1,i}, \ldots, \alpha_i^{ln,i}\}$ and obtain $\bar{\theta}_i^{retain}$
      Set $Heur\bar{\theta}_i^{(t+1)} \leftarrow \bar{\theta}_i^{(t+1)}$ not in $\bar{\theta}_i^{retain}$
      $\Delta\theta_i \leftarrow ClientUpdate(Heur\bar{\theta}_i^{(t+1)})$
      Update $\{\theta^{l1}, \theta^{l2}, \ldots, \theta^{ln}\}$ according to $\Delta\theta_i$
      Update $v_i^{(t+1)}$ and $\psi_i^{(t+1)}$ via Eq. 10, 11
procedure CLIENTUPDATE($\bar{\theta}_i^{(t+1)}$)
  Client i receives $Heur\bar{\theta}_i^{(t+1)}$ from the server.
  Set $\theta_i \leftarrow \{Heur\bar{\theta}_i^{(t+1)}, \theta_i^{retain}\}$.
  for each local epoch **do**
    for mini-batch $\xi_t \subseteq D_i$ **do**
      Local Training: $\theta_i = \theta_i - \eta\nabla_{\theta_i}\mathcal{L}_i(\theta_i; \xi_t)$
  return $\Delta\theta_i = \theta_i - \{Heur\bar{\theta}_i^{(t+1)}, \theta_i^{retain}\}$

$\nabla_{\bar{\theta}_i}\mathcal{L}_i$ can be obtained from client i's local training in each communication round and $\nabla_{v_i/\psi_i}HN_i(v_i; \psi_i)$ is the gradient of α_i in directions v_i/ψ_i. pFedLA uses a more general way to update v_i and ψ_i:

$$\begin{aligned}\Delta v_i &= (\nabla_{v_i}\bar{\theta}_i)^T\Delta\theta_i \\ &= [\{\theta^{l1}, \theta^{l2}, \ldots, \theta^{ln}\} * \nabla_{v_i}HN_i(v_i; \psi_i)]^T\Delta\theta_i,\end{aligned} \tag{10}$$

$$\begin{aligned}\Delta\psi_i &= (\nabla_{\psi_i}\bar{\theta}_i)^T\Delta\theta_i \\ &= [\{\theta^{l1}, \theta^{l2}, \ldots, \theta^{ln}\} * \nabla_{\psi_i}HN_i(v_i; \psi_i)]^T\Delta\theta_i.\end{aligned} \tag{11}$$

where $\Delta\theta_i$ is the change of model parameters in client i after local training. In accordance with Eq. 10 and 11, pFedLA updates the embedding vector and parameters of hypernetwork for client i at each communication round, and then update the aggregation weight matrix α_i.

Algorithm 1 demonstrates the pFedLA procedure. In each communication round, the clients first download the latest personalized models from the server, then use local SGD to train several epochs based on the private data. After that, the model update $\Delta\theta_i$ for each client will be uploaded to the server to update the embedding vector V and the parameter Ψ.

3.3. HeurpFedLA: Heuristic Improvement of pFedLA on Communication Efficiency

The communication overhead of pFedLA is determined by the size of $\Delta\theta_i$ sent from the clients and $\bar{\theta}_i$ sent from the server. So, there is no additional communication cost comparing with traditional FL methods, e.g., FedAvg. In this section, we propose to further reduce the communication overhead of pFedLA with negligible performance reduction, which can adapt to more general scenarios, e.g., large scale FL systems, limited communication capacities, etc.

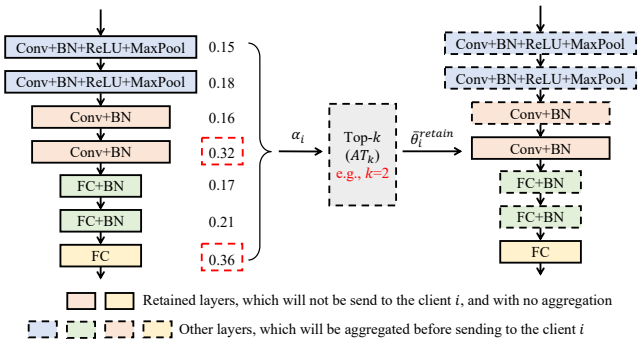


Figure 4. Illustration of top k mechanism in HeurpFedLA. The selected top k layers (i.e., retained layers) do not perform aggregation process, while the remaining layers execute the same operations as in pFedLA.

Comparing with existing works that keep some specific layers updated locally to enable communication-efficient training while retaining the performance of pFL [5, 24, 26], e.g., FedBN [24] found that local models with BN layers should exclude these parameters from the aggregating steps during training, while FedRep [5] and LG-FedAvg [26] proposed to locally learn the classifier layer and representation layers respectively, pFedLA can give an alternative guidance to determine which layers should be retained locally. To this end, we propose HeurpFedLA, a heuristic improvement of pFedLA that partial layers are retained locally, and the remaining layers are aggregated at the server side during training. The key idea of HeurpFedLA is to heuristically select the partial layers $\bar{\theta}_i^{retain}$ with top k (AT_k) aggregation weights to update locally. Specifically, by using the aggregation weights $\alpha_i^{l1,i}, \alpha_i^{l2,i}, \ldots, \alpha_i^{ln,i}$ for all layers of client i, we can sort these weights in descending order and select corresponding top k layers

$$\bar{\theta}_i^{retain} = \text{AT}_k\{\bar{\theta}_i^{l1}, \ldots, \bar{\theta}_i^{ln} | \alpha_i^{l1,i}, \ldots, \alpha_i^{ln,i}\}, \tag{12}$$

where AT_k is the top k selection function described above, and k is a hyperparameter manually denoted before training. The detailed workflow of top k selection mechanism is shown in Figure 4.

The principle behind HeurpFedLA is that layers with higher rank index should contribute more to the model personalization, which means directly using these layers in personalized model has little impact on the training performance. The retention of local layers by HeurpFedLA brings benefits in terms of communication overhead reduc-

Table 1. Average model accuracy on 10 and 100 clients over four different datasets(non-IID_1), respectively.

	EMNIST (%)		FashionMNIST (%)		CIFAR10 (%)		CIFAR100 (%)	
# Clients	10	100	10	100	10	100	10	100
Local Training	89.01±0.47	91.25±0.18	85.83±0.17	89.27±0.21	59.44±0.40	64.19±0.19	41.68±0.89	42.53±0.44
FedAvg [34]	90.45±0.76	93.71±0.38	91.24±0.98	98.36±0.26	48.57±0.63	58.43±0.29	36.64±0.67	45.19±0.33
Per-FedAvg [9]	92.58±0.28	92.38±1.14	93.63±1.83	92.35±1.55	52.54±1.79	59.54±0.39	38.79±1.89	43.72±0.25
pFedMe [41]	92.42±0.44	94.36±0.50	90.43±0.86	98.57±0.38	53.73±3.74	65.97±1.61	42.29±3.67	53.60±1.28
pFedHN [37]	93.94±0.16	**96.64**±0.91	94.83±0.33	98.80±0.92	46.98±1.91	63.71±1.26	39.67±0.52	51.36±1.77
FedBN [24]	-	-	-	-	59.36±0.92	70.88±0.36	45.18±0.42	56.16±0.38
FedRep [5]	91.82±0.15	95.23±0.12	93.17±0.26	97.15±0.09	58.01±0.56	71.94±0.22	44.33±0.63	56.47±0.41
FedFomo [54]	88.33±0.29	91.36±0.17	86.17±0.34	91.83±0.12	59.37±0.71	66.07±0.24	41.89±0.78	44.28±0.28
pFedLA (Ours)	90.65±0.41	96.34±1.35	94.34±0.29	**98.87**±0.66	**61.43**±0.56	**73.15**±0.83	**47.22**±0.77	**56.62**±0.81
HeurpFedLA (Ours)	**94.11**±0.13	95.04±0.41	**95.47**±0.47	96.95±0.44	60.02±0.74	73.05±1.02	46.47±0.83	54.43±1.37

Table 2. Average model accuracy on 10 and 100 clients over four different datasets(non-IID_2), respectively.

	EMNIST (%)		FashionMNIST (%)		CIFAR10 (%)		CIFAR100 (%)	
# Clients	10	100	10	100	10	100	10	100
Local Training	80.72±0.43	79.09±0.12	65.60±0.59	65.97±0.28	39.79±0.42	45.15±0.29	26.29±0.37	27.87±0.28
FedAvg [34]	90.43±0.58	93.91±0.32	89.09±0.57	98.25±0.38	44.89±0.21	54.03±0.37	32.24±0.74	40.89±0.46
Per-FedAvg [9]	90.86±0.78	94.09±0.18	90.78±1.12	98.53±0.95	44.48±0.82	54.40±0.44	30.86±1.11	42.56±0.28
pFedMe [41]	89.13±0.58	93.87±0.40	85.15±0.94	97.87±0.19	46.97±1.19	58.23±1.07	33.45±0.86	44.35±0.96
pFedHN [37]	91.37±0.41	94.48±0.51	93.45±0.11	**98.83**±0.82	37.49±0.94	49.90±1.66	26.35±0.93	40.27±0.82
FedBN [24]	-	-	-	-	49.79±0.33	60.62±0.42	34.94±0.50	46.42±0.54
FedRep [5]	86.81±0.29	90.32±0.08	79.13±0.56	92.04±0.23	49.16±0.73	60.36±0.57	34.19±0.74	43.51±0.34
FedFomo [54]	80.14±0.42	82.61±0.11	64.10±0.38	67.91±0.29	40.62±0.31	47.08±0.49	27.33±0.51	29.63±0.24
pFedLA (Ours)	**92.06**±0.71	**94.83**±1.04	**93.89**±0.91	98.41±0.98	**49.93**±0.96	**61.82**±1.89	35.02±0.83	**48.79**±1.60
HeurpFedLA (Ours)	91.98±0.36	93.31±0.77	92.01±0.74	98.66±0.80	49.06±0.68	60.62±1.73	**35.42**±0.49	48.72±1.75

tion from the server to the clients direction, i.e., the server can save the costs of transmitting the parameters of the retained layers.

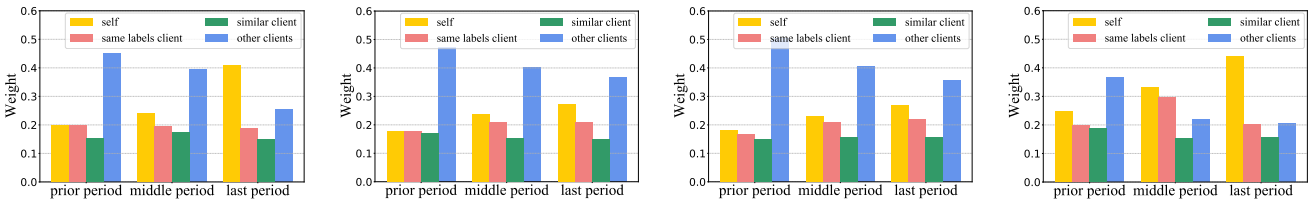
As to be demonstrated in Section 4.4, HeurpFedLA can significantly reduce the communication cost while maintaining the model performance of pFL. In large scale FL systems, it is of practical value to keep some layers from aggregation and transmission, especially for limited communication bandwidth scenarios. Furthermore, HeurpFedLA is a general training framework and can be effectively compatible with common compression schemes such as gradient quantization, sparsification, etc. The impact of retaining local layers is discussed in more detail in the next section.

4. Evaluation

4.1. Experimental Setup

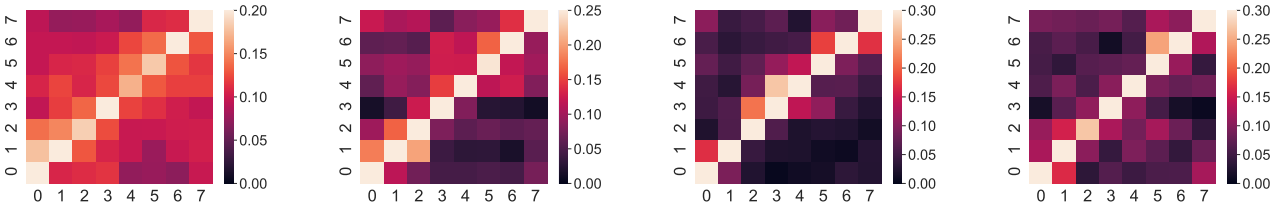
Datasets. We evaluate the pFedLA framework over four datasets, EMNIST, FashionMNIST, CIFAR10 and CIFAR100. The distribution of all data sets on the training clients is non-IID. We consider two non-IID scenarios: 1) each client is randomly assigned four classes (twelve classes per client in CIFAR100) with the same amount of data on each class; 2) each client contains all classes, while the data on each class is not uniformly distributed. Two classes in EMNIST, FashionMNIST, CIFAR10 datasets have higher number of data samples than other classes, while six classes in CIFAR100 have more data samples than the others. All data are divided into 70% training set, and 30% test set. The test set and the training set have the same data distribution for all clients.

Baselines. We compared the performance of pFedLA and HeurpFedLA with the state-of-the-art methods. In addition to **FedAvg** and **Local Training**, we also include **Per-Fedavg**, a pFL algorithm based on meta-learning; **pFedMe**, a pFL algorithm with regularization term added in the objective function; **pFedHN**, a pFL algorithm that uses hypernetworks to directly produce personalized model; **FedBN**, keeps each client's BN layer updating locally, while other layers are aggregated according to the FedAvg algorithm; **FedRep**, a pFL algorithm that keeps each client's classifier updating locally, while the other parts are aggregated at the



(a) EMNIST (b) FashionMNIST (c) CIFAR10 (d) CIFAR100

Figure 5. Change of aggregation weights during the prior, middle and last period of training phase.



(a) EMNIST (FC1) (b) FashionMNIST (FC1) (c) CIFAR10 (FC3) (d) CIFAR100 (FC3)

Figure 6. The visualization of the aggregation weights in a specific layer on EMNIST, FashionMNIST, CIFAR10 and CIFAR100. X-axis and y-axis show the IDs of clients.

server; **FedFomo**, a pFL algorithm that uses distance to calculate the aggregation weights based on the model and loss differences.

Training Details. In all experiments, we use the same CNN architectures as in FedFomo [54], FedBN [24] and pFedHN [37]. All the models have the same structure between different clients under the same setting. For CIFAR10 and CIFAR100, we add BN layers after the convolutional layers. For EMNIST and FashionMNIST, there is no BN layers in the model. The hypernetwork for computing layer-wise aggregation weights is a simple structure of several fully connected layers. The weight of each layer for a target client is calculated by a corresponding fully connected layer in the hypernetwork. For the specific structure of hypernetwork, please refer to the supplemental material. We evaluate the performance of pFedLA in two settings, i.e., 10 clients with 100% participation and 100 clients with 10% participation. The average model accuracy of all clients is obtained after 600 rounds training for 10 clients case and 2500 rounds for 100 clients.

Implementation. We simulate all clients and the server on a workstation with an RTX 2080Ti GPU, a 3.6-GHZ Intel Core i9-9900KF CPU and 64GB of RAM. All methods are implemented in PyTorch.

4.2. Performance Evaluation

For all experiments, we use cross-entropy loss and SGD optimizer with a batch size of 32. The number of local epochs is 10 for 10 clients case and 20 for 100 clients. The learning rate is 0.01 for CIFAR10 and CIFAR100, and 0.005 for EMNIST and FashionMNIST. The performance of both the baselines and the proposed pFedLA under two different non-IID cases are listed in Table 1 and Table 2, respectively. Our proposed algorithm provides superior performance than baselines over the four datasets with different data distributions in most cases. On the other hand, HeurpFedLA also outperforms the existing methods with negligible performance reduction comparing with pFedLA. The number of retained layers (k) in Table 1 and 2 is 1. The communication costs of HeurpFedLA is discussed in Section 4.4. Note that since all clients have the same amount of training data for both 10 and 100 clients cases, so the 100 clients case has much more data, and thus can provide better model accuracy.

4.3. Analysis of Weight Evolution

To demonstrate that our method can generate higher weights to those clients with similar data distribution, we conduct the experiments with 8 clients who randomly 4 data classes from the corresponding datasets. From the 8 clients, we consider a target client with 4 random data classes, one contrastive client who has the same four classes, and one similar client who has 3 same classes with the target client. We record the weight value of each layer on the target client during the training process. Figure 5 shows the evolution of the aggregation weights for the target client during the prior, middle and last periods of training phase. It can be observed that the inter-weights from other clients decrease with the training process because their data distribution is very different from the target client. Besides, for the target client, clients with more similar data distribution (e.g., the same labels client) have higher weight value than other clients (e.g., the similar client), which shows that the hypernetwork can distinguish the similarity of data distribution

Table 3. Average Model Accuracy and Communication Cost on different number of retained layers (i.e., k) over EMNIST and CIFAR10.

	EMNIST			CIFAR10			
# Number of retained layers (k)	0	1	2	0	1	2	3
Model Accuracy (%)	90.65	94.11	93.94	61.43	60.02	59.90	59.23
Communication Cost (MBytes)	491.08	488.65	312.52	693.98	418.97	382.25	379.82

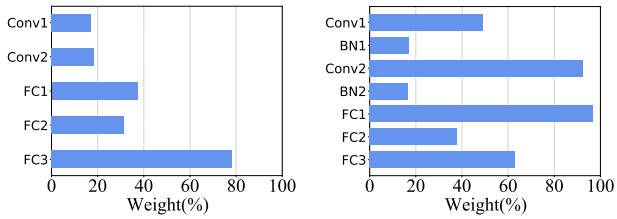
on different clients. We also conduct experiments to visualize the relationship between the aggregation weights and the data similarities among clients. We consider 8 clients assigned with ID from 0 to 7, all have four classes data. The data similarities among all clients are emulated by assigning clients of adjacent IDs with similar classes, e.g., client 1 has 4 classes data, while client 2 has three same and one different classes with client 1, and client 3 has three same and one different classes with client 2, and so on. Figure 6 shows the heatmap of the inter-weights among all 8 clients of a certain layer. It can be seen that the weights among close clients with consecutive IDs, i.e., with more overlapping classes, are larger than those of the distant clients, and the highlighted diagonal line shows that the self-weights of each client have the highest values, which further verify that pFedLA can exploit the inter-similarities among heterogeneous clients.

4.4. Analysis of Communication Efficiency

In this section, we show the performance of the proposed HeurpFedLA. Table 3 shows the average model accuracy and communication overhead when retaining different local layers that would be absent from the aggregation process. We consider 10 clients with 100% participation over the datasets EMNIST and CIFAR10. The aggregation weights of all layers for a target client are shown in Figure 7. For the CIFAR10 dataset, the weights of the first fully-connected layer have the highest values, so the model accuracy performance will be compromised if retaining some layers at local, although the communication overhead can be reduced greatly. For EMNIST dataset, what's different is that the classifier layer has the largest weights, it is observed that the average model accuracy can even increase when retaining some local layers. Such conclusion can also be found in a state-of-the-art work, FedRep [5], which indicates that removing the classifier layer from the aggregation process can improve the model performance over non-IID datasets. It can be explained intuitively that reserving some local layers can avoid the irrelevant knowledge transfer from other clients during the aggregation process.

4.5. Effect of k

Different k values are applied to show the effect of retaining local layers. Table 3 shows that if retaining different number of the top k layers, the model accuracy will not be affected significantly, which means that HeurpFedLA can apply different k values according to the available communication bandwidth for transmitting the parameters during the pFL iteration, i.e., to do a trade-off between the training efficiency and the communication costs.



(a) EMNIST (b) CIFAR10

Figure 7. The aggregation weights of all layers for the target client.

5. Conclusion

In this paper, we have proposed a novel pFL training framework called pFedLA, to achieve personalized model aggregation in a layer-wised aggregation manner. It is shown that such layer-wised aggregation can progressively reinforce the collaboration among similar clients and generate adequate personalization over non-IID datasets that outperform conventional model-wised approaches. In addition, we have provided an improved version of pFedLA that can reduce the communication overhead during the training process with negligible performance loss, and thus can be adapted to large scale FL scenarios where the communication capacity is often limited. Extensive evaluations on four different classification tasks demonstrate the feasibility and superior performance of the proposed pFedLA framework.

Acknowledgements

This research was supported by fundings from the Key-Area Research and Development Program of Guangdong Province (No. 2021B0101400003), Hong Kong RGC Research Impact Fund (No. R5060-19), General Research Fund (No. 152221/19E, 152203/20E, and 152244/21E), the National Natural Science Foundation of China (61872310), and Shenzhen Science and Technology Innovation Commission (JCYJ20200109142008673).

References

[1] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019. 2

[2] Juhan Bae and Roger Grosse. Delta-stn: Efficient bilevel optimization for neural networks using structured response jacobians. *arXiv preprint arXiv:2010.13514*, 2020. 3

[3] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017. 3

[4] Duc Bui, Kshitiz Malik, Jack Goetz, Honglei Liu, Seungwhan Moon, Anuj Kumar, and Kang G Shin. Federated user representation learning. *arXiv preprint arXiv:1909.12535*, 2019. 2

[5] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Exploiting shared representations for personalized federated learning. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139, pages 2089–2099. 1, 2, 5, 6, 8

[6] Jason Cramer, Vincent Lostanlen, Andrew Farnsworth, Justin Salamon, and Juan Pablo Bello. Chirping up the right tree: Incorporating biological taxonomies into deep bioacoustic classifiers. In *Proceedings of 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 901–905. IEEE, 2020. 2

[7] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020. 2

[8] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. Self-balancing federated learning with global imbalanced data in mobile systems. *IEEE Transactions on Parallel and Distributed Systems*, 32(1):59–71, 2020. 1, 2

[9] Alireza Fallah, Aryan Mokhtari, and Asuman E. Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *Advances in Neural Information Processing Systems, NeurIPS*, 2020. 1, 2, 6

[10] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 2020. 2

[11] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 3

[12] Filip Hanzely, Slavomír Hanzely, Samuel Horváth, and Peter Richtárik. Lower bounds and optimal algorithms for personalized federated learning. In *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 2020. 1, 2

[13] Filip Hanzely and Peter Richtárik. Federated learning of a mixture of global and local models. *arXiv preprint arXiv:2002.05516*, 2020. 2

[14] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *Proceedings of International Conference on Machine Learning, ICML*, pages 4387–4398, 2020. 1

[15] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI*, volume 35, pages 7865–7873, 2021. 1, 2, 4

[16] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018. 1, 2

[17] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Proceedings of Advances in neural information processing systems, NeurIPS*, 29:667–675, 2016. 3

[18] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019. 2

[19] Sylwester Klocek, Łukasz Maziarka, Maciej Wołczyk, Jacek Tabor, Jakub Nowak, and Marek Śmieja. Hypernetwork functional image representation. In *Proceedings of International Conference on Artificial Neural Networks, ICANN*, pages 496–510. Springer, 2019. 3

[20] Sunwoo Lee, Tuo Zhang, Chaoyang He, and Salman Avestimehr. Layer-wise adaptive model aggregation for scalable federated learning. *arXiv preprint arXiv:2110.10302*, 2021. 2

[21] Hailiang Li, Jian Weng, Yujian Shi, Wanrong Gu, Yijun Mao, Yonghua Wang, Weiwei Liu, and Jiajie Zhang. An improved deep learning approach for detection of thyroid papillary cancer in ultrasound images. *Scientific reports*, 8(1):1–12, 2018. 2

[22] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *Proceedings of International Conference on Machine Learning, ICML*, pages 6357–6368, 2021. 1

[23] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018. 1

[24] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. Fedbn: Federated learning on non-iid features via local batch normalization. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*. OpenReview.net, 2021. 1, 5, 6, 7

[25] Yawei Li, Shuhang Gu, Luc Van Gool, Radu Timofte, et al. Dhp: Differentiable meta pruning via hypernetworks. In *Proceedings of European Conference on Computer Vision and Pattern Recognition, ECCV*, 2020. 3

[26] 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. 1, 5

[27] Etai Littwin, Tomer Galanti, Lior Wolf, and Greg Yang. On infinite-width hypernetworks. *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 33, 2020. 3

[28] Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV*, pages 1824–1833, 2019. 3

[29] Jonathan Lorraine and David Duvenaud. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419*, 2018. 3

[30] Lingjuan Lyu, Jiangshan Yu, Karthik Nandakumar, Yitong Li, Xingjun Ma, Jiong Jin, Han Yu, and Kee Siong Ng. Towards fair and privacy-preserving federated deep models. *IEEE Transactions on Parallel and Distributed Systems*, 31(11):2524–2541, 2020. 2

[31] Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv preprint arXiv:1903.03088*, 2019. 3

[32] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020. 2

[33] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2017. 1, 2

[34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, pages 1273–1282. PMLR, 2017. 6

[35] Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 4061–4070, 2021. 3

[36] Johannes Schneider and Michail Vlachos. Personalization of deep learning. *arXiv preprint arXiv:1909.02803*, 2019. 2

[37] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 3, 6, 7

[38] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 2020. 3

[39] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S. Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2017. 2

[40] Joseph Suarez. Language modeling with recurrent highway hypernetworks. In *Proceedings of Advances in neural information processing systems, NeurIPS*, pages 3267–3276, 2017. 3

[41] Canh T Dinh, Nguyen Tran, and Tuan Dung Nguyen. Personalized federated learning with moreau envelopes. *Proceedings of Advances in Neural Information Processing Systems, NeurIPS*, 33, 2020. 1, 2, 6

[42] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. Towards personalized federated learning. *arXiv preprint arXiv:2103.00710*, 2021. 2

[43] Nguyen H Tran, Wei Bao, Albert Zomaya, Minh NH Nguyen, and Choong Seon Hong. Federated learning over wireless networks: Optimization model design and analysis. In *Proceedings of IEEE Conference on Computer Communications, INFOCOM*, pages 1387–1395. IEEE, 2019. 1

[44] Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F Grewe. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019. 3

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

[46] Kangkang Wang, Rajiv Mathews, Chloé Kiddon, Hubert Eichner, Françoise Beaufays, and Daniel Ramage. Federated evaluation of on-device personalization. *arXiv preprint arXiv:1910.10252*, 2019. 2

[47] Rudong Xu, Yiting Tao, Zhongyuan Lu, and Yanfei Zhong. Attention-mechanism-containing neural networks for high-resolution remote sensing image classification. *Remote Sensing*, 10(10):1602, 2018. 2

[48] Miao Yang, Akitanoshou Wong, Hongbin Zhu, Haifeng Wang, and Hua Qian. Federated learning with class imbalance reduction. *arXiv preprint arXiv:2011.11266*, 2020. 2

[49] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition, CVPR*, pages 2403–2412, 2018. 2

[50] Chris Zhang, Mengye Ren, and Raquel Urtasun. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749*, 2018. 3

[51] Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, and Feijie Wu. Parameterized knowledge transfer for personalized federated learning. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[52] Jie Zhang, Song Guo, Zhihao Qu, Deze Zeng, Yufeng Zhan, Qifeng Liu, and Rajendra A Akerkar. Adaptive federated learning on non-iid data with resource constraint. *IEEE Transactions on Computers*, 2021. 1

[53] Jie Zhang, Zhihao Qu, Chenxi Chen, Haozhao Wang, Yufeng Zhan, Baoliu Ye, and Song Guo. Edge learning: The enabling technology for distributed big data analytics in the edge. *ACM Computing Surveys (CSUR)*, 54(7):1–36, 2021. 1

[54] Michael Zhang, Karan Sapra, Sanja Fidler, Serena Yeung, and Jose M. Alvarez. Personalized federated learning with first order model optimization. In *Proceedings of the 9th International Conference on Learning Representations, ICLR 2021*, 2021. 1, 2, 4, 6, 7

[55] Dominic Zhao, Johannes von Oswald, Seijin Kobayashi, João Sacramento, and Benjamin F Grewe. Meta-learning via hypernetworks. 2020. 3

[56] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018. 1, 2