Knowledge-Aware Parameter Coaching for Personalized Federated Learning

Mingjian Zhi[1], **Yuanguo Bi**[1*], **Wenchao Xu**[2], **Haozhao Wang**[3], **Tianao Xiang**[1]

[1]Northeastern University, China
[2]The Hong Kong Polytechnic University, Hong Kong, China
[3]Huazhong University of Science and Technology, China
2110657@stu.neu.edu.cn, biyuanguo@mail.neu.edu.cn, wenchao.xu@polyu.edu.hk, hz_wang@hust.edu.cn, 2010652@stu.neu.edu.cn

Abstract

Personalized Federated Learning (pFL) can effectively exploit the non-IID data from distributed clients by customizing personalized models. Existing pFL methods either simply take the local model as a whole for aggregation or require significant training overhead to induce the inter-client personalized weights, and thus clients cannot efficiently exploit the mutually relevant knowledge from each other. In this paper, we propose a knowledge-aware parameter coaching scheme where each client can swiftly and granularly refer to parameters of other clients to guide the local training, whereby accurate personalized client models can be efficiently produced without contradictory knowledge. Specifically, a novel regularizer is designed to conduct layer-wise parameters coaching via a relation cube, which is constructed based on the knowledge represented by the layered parameters among all clients. Then, we develop an optimization method to update the relation cube and the parameters of each client. It is theoretically demonstrated that the convergence of the proposed method can be guaranteed under both convex and non-convex settings. Extensive experiments are conducted over various datasets, which show that the proposed method can achieve better performance compared with the state-of-the-art baselines in terms of accuracy and convergence speed.

Introduction

Federated learning (FL) (McMahan et al. 2017) allows distributed clients to collaboratively exploit their local data to train a shared global model without delivering the local data to a centralized server, which not only avoids exposing sensitive user information but also reduces communication overhead for saving the transmissions of raw data.

Traditional FL paradigms suffer from significant performance degradation when the client data is non-identically and independently distributed (non-IID) (Kairouz et al. 2021; Tan et al. 2022a), which is quite common in the diverse FL environments and data domains. As a solution, personalized federated learning (pFL) is proposed to mitigate the non-IID issue, and many pFL methods have been studied to improve model accuracy for diverse clients including meta learning-based methods (Fallah, Mokhtari, and Ozdaglar 2020; Jiang et al. 2019; Lin, Yang, and Zhang 2020), knowledge distillation-based methods (Li and Wang 2019; Zhang et al. 2021a; Lin et al. 2020; Chen et al. 2022; Jin et al. 2023) and so on.

However, there are two critical issues in the existing pFL methods. Firstly, a local model is usually taken as a whole for global weighted aggregation, which disregards the heterogeneous knowledge represented by the layered DNN parameters and thus limits the optimization of personalized models (Yosinski et al. 2014; Zeiler and Fergus 2014). For example, shallow layers are close to the data plane, and they can reveal the low-level and regional features that are typically irrelevant to data heterogeneity, such that sharing these layers can boost the pFL training even under non-IID condition (Luo et al. 2021). Conversely, deep layers often correspond to the semantic features that may contain different knowledge from different clients, so recklessly sharing these layers can lead to degraded performance and excessive training time (Zeiler and Fergus 2014). Secondly, significant training overhead is often required to induce fine-grained personalized weights for inter-client collaboration, and thus a client cannot efficiently leverage the mutually beneficial knowledge from different clients. For example, pFedLA uses a hypernetwork to exploit the inter-similarities among non-IID clients, but it requires excessive communication rounds to converge (Ma et al. 2022).

To solve the aforementioned issues, we propose an efficient knowledge-aware parameter coaching method for personalized federated learning, where a client refers to the granular knowledge from other clients to coach the personalized training of the local parameters without incurring much additional overhead. Specifically, the server maintains a personalized model for each client, where the layer-wise parameters are a linear combination of all clients' corresponding parameters with adaptive weights. A personalized server model acts as a regularizer to guide the update of local layer-wise parameters. Moreover, the adaptive weights represented by relation cube can be derived efficiently from the clients' parameters, resulting in minimal computation burden in the server and low training time. We summarize the main contributions of our paper as follows.

- Based on the layered property of the deep neural networks (DNN) model, a relation cube is defined to explicitly demonstrate the similarity of layered knowledge among all clients. In addition, a lightweight optimization

*Corresponding author.

method is designed to learn the relation cube, which simplifies the training process and speeds up the convergence of a local model.

- We propose an efficient knowledge-aware parameter coaching method, where a personalized regularizer derived from the relation cube is designed to coach the general knowledge to be shared among all the clients and the specific knowledge to be shared among similar clients.
- We theoretically prove that the proposed parameter coaching method achieves convergence under both convex and non-convex settings. As far as we know, this is the first paper that proves the convergence of adaptive aggregation-based federated learning.
- Extensive experiments are conducted, and the results show that the proposed method is robust under various levels of heterogeneity and more accurate than the state-of-the-art personalized algorithms.

Related Works

In order to tackle the non-IID issue, many pFL methods have been developed (Mansour et al. 2020; Duan et al. 2022; Luo et al. 2021; Oh, Kim, and Yun 2021; Tang et al. 2022; Li and Zhan 2021). Among the existing methods, the regularization and layer-wise methods have been widely studied.

Regularization Methods. The regularization methods limit the local training with various regularizers to improve the personalization of local models. Several studies construct the regularizer with model parameters to provide direct guidance for local training (Li et al. 2020; Acar et al. 2021; Huang et al. 2021; Li, He, and Song 2021; Dinh, Tran, and Nguyen 2020). Some works correct the update direction for each client to reduce the data drift in non-IID setting (Karimireddy et al. 2020; Zhang et al. 2021b). Other works align the prototype of heterogeneous clients to enforce the learning for global extractor with less communication cost (Tan et al. 2022b; Xu, Tong, and Huang 2023). Recent works regularize the local training with soft label or statistic information to enhance the knowledge sharing from other clients (Jin et al. 2023; Mendieta et al. 2022).

Layer-wise Personalized Methods. Considering the distinct representations of different layers in DNN, some personalized methods propose layer-wise aggregation. Some methods keep the batch normalization layer personalized without aggregation in the server to avoid the drift of local features (Li et al. 2021b; Mills, Hu, and Min 2022). Additionally, many works focus on the aggregation of partial shallow layers of DNN with the same weights to transfer the general knowledge among clients (Oh, Kim, and Yun 2021; Arivazhagan et al. 2019; Pillutla et al. 2022; Collins et al. 2021; Li et al. 2021a). Recently, pFedLA considers the impacts of different layers and adopts hypernetwork to generate layer-wise aggregation weights for each client, such that the knowledge transfer conflicts can be avoided for some clients with large data distance (Ma et al. 2022).

In the abovementioned methods, most regularization pFL methods take the entire layered DNN model as a whole for aggregation, which may cause contrary knowledge to transfer from other clients and thus degrade the performance of local models. Some layer-wise methods aim to aggregate the partial layers with the same weights, which still explore the inter-client similarity in a coarse way. For pFedLA, a hypernetwork is used to exploit the fine-grained similarities among non-IID clients. However, the hypernetwork usually requires significant training efforts to achieve convergence which may even prevent adaptive knowledge transfer among clients. Furthermore, these layer-wise methods have no theoretical guarantee of convergence.

Knowledge-Aware Parameter Coaching Method

In this section, a knowledge-aware parameter coaching method is proposed. Firstly, the preliminary in pFL is given, and then relation cube is defined to explore the fine-grained relation among clients. Based on the relation cube, the loss function and optimization method are designed to update client models and relation cube. Finally, the detailed algorithms are outlined.

Preliminary

Assuming there are N clients, and $D_i = \{(\mathbf{x}_{ij}, y_{ij})\}_{j=1}^{m_i}$ denotes the dataset of client i ($i \in [1, N]$), and m_i is the data number of dataset D_i. In our settings, the DNN model is used, and $h(\cdot; \mathbf{w}_i)$ with the parameters $\mathbf{w}_i = [\mathbf{w}_i^1, \mathbf{w}_i^2, \cdots, \mathbf{w}_i^L]$ is denoted as the model of client i, where L is the number of DNN layers. The bias is neglected for simple representation. The local loss function for client i can be defined as

$$f_i(\mathbf{w}_i) = \frac{1}{m_i} \sum_{j=1}^{m_i} l\left(h\left(x_{ij}; \mathbf{w}_i\right), y_{ij}\right) \tag{1}$$

Then the global loss function of personalized federated learning can be denoted as

$$F(\mathbf{W}) = \sum_{i=1}^{N} \frac{m_i}{m} f_i(\mathbf{w_i}) \tag{2}$$

where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_N]$ is the set of client parameters and $m = \sum_{i=1}^{N} m_i$.

Relation Cube

To explicitly describe the similarity of layer-wise knowledge among all clients, a relation cube $\mathbf{R} \in \mathbb{R}^{N \times L \times N}$ is defined as

$$\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N] \tag{3}$$

where element $\mathbf{r}_i \in \mathbb{R}^{L \times N}$ is a matrix defined as

$$\mathbf{r}_i = \begin{bmatrix} r_{i1}^1, & r_{i2}^1 & \cdots & r_{iN}^1 \\ r_{i1}^2, & r_{i2}^2 & \cdots & r_{iN}^2 \\ \vdots & \vdots & \ddots & \vdots \\ r_{i1}^L, & r_{i2}^L & \cdots & r_{iN}^L \end{bmatrix} \tag{4}$$

The relation cube $\mathbf{R}$ is composed of the relation matrix $\mathbf{r}_i$ of client i ($i \in [1, N]$), which represents the similarity of

layered knowledge between client i and other clients. The relation matrix $\mathbf{r}_i$ is composed of relation coefficient r_{ij}^l, which represents the similarity between client i and client j on DNN layer l.

Optimization Objective

Different from the general federated learning, the proposed method maintains a specific server model for each client to aggregate the beneficial knowledge from other clients. The parameters of the server model are derived from the linear combination of the clients' parameters weighted by the relation cube. The server parameters are leveraged as a regularizer to coach the local training. Therefore, our objective is not only to minimize the empirical loss of the client as general personalized federated learning, but also minimize the distances between the local parameters and server parameters. The formulation of the local loss function of client i can be represented as

$$\begin{aligned} f(\mathbf{w}_i, \mathbf{r}_i) =& f_i(\mathbf{w}_i) + \lambda \sum_{l=1}^{L} f_s \left(\sum_{j=1}^{N} r_{ij}^l \mathbf{w}_j^l, \mathbf{w}_i^l \right) \\ &+ \frac{\beta}{2} \sum_{j=1}^{N} \sum_{l=1}^{L} \left\| r_{ij}^l - \frac{1}{N} \right\|^2 \end{aligned} \tag{5}$$

where $f_i(\mathbf{w}_i)$ defined by Eq. (1) is the general loss of client i, f_s is the coaching loss to guide the layer-wise local training and we adopt l_2-norm in this paper. The third term is used to motivate all the clients to share their layered knowledge equally. λ and β are hyperparameters that trade off the main loss and regularization term.

When the local loss function is stacked by layer, Eq. (5) can be rewritten as

$$f(\mathbf{w}_i, \mathbf{r}_i) = f_i(\mathbf{w}_i) + \lambda f_s(\mathbf{r}_i \mathbf{W}, \mathbf{w}_i) + \frac{\beta}{2} \left\| \mathbf{r}_i - \frac{\mathbf{1}}{N} \right\|^2 \tag{6}$$

where $\mathbf{1} \in \mathbb{R}^{L \times N}$ is an identity matrix.

Then, the global optimization objective can be formulated as

$$\min_{\mathbf{W}, \mathbf{R}} \left\{ F_{per}(\mathbf{W}, \mathbf{R}) := \sum_{i=1}^{N} \frac{m_i}{m} f(\mathbf{w}_i, \mathbf{r}_i) \right\} \tag{7}$$

where $\mathbf{W}$ is the parameter set of clients in Eq. (2), and $\mathbf{R}$ is the relation cube defined by Eq. (3).

Optimization Mehtod

Inspired by FedAMP (Huang et al. 2021) and FedProto (Tan et al. 2022b), we present an optimization method by alternatively updating client parameters and relation cube until the convergence of the proposed method. Consequently, there are two updating steps in the $(k+1)$-th iteration. Firstly, when client parameter set $\mathbf{W}^k$ is fixed, the relation cube $\mathbf{R}^{k+1}$ in the $(k+1)$-th iteration can be updated. Then, after relation cube $\mathbf{R}^{k+1}$ is determined and the personalized regularizer is updated, client parameter set $\mathbf{W}^{k+1}$ will be optimized.

To describe the update method in detail, the first, second and third terms of $F_{per}(\mathbf{W}, \mathbf{R})$ are denoted as $F(\mathbf{W}) := \sum_{i=1}^{N} \frac{m_i}{m} f_i(\mathbf{w}_i)$, $H(\mathbf{W}, \mathbf{R}) := \sum_{i=1}^{N} \frac{m_i}{m} f_s(\mathbf{r}_i \mathbf{W}, \mathbf{w}_i)$ and $G(\mathbf{R}) := \frac{\beta}{2} \sum_{i=1}^{N} \frac{m_i}{m} \|\mathbf{r}_i - \frac{\mathbf{1}}{N}\|^2$, and the global optimization objective (7) can be rewritten as

$$\min_{\mathbf{W}, \mathbf{R}} \{ F_{per}(\mathbf{W}, \mathbf{R}) := F(\mathbf{W}) + \lambda H(\mathbf{W}, \mathbf{R}) + G(\mathbf{R}) \} \tag{8}$$

Relation Cube Update. When the relation cube is updated in the $(k+1)$-th iteration, the parameters of all the clients in the k-th iteration have been uploaded to the server. Then the updated $\mathbf{R}^{k+1}$ is

$$\begin{aligned} \mathbf{R}^{k+1} &= \mathbf{R}^k - \eta_r \nabla_{\mathbf{R}} F_{per}(\mathbf{W}^k, \mathbf{R}) \\ &= \mathbf{R}^k - \eta_r \left(\nabla_{\mathbf{R}} G(\mathbf{R}) + \lambda \nabla_{\mathbf{R}} H(\mathbf{W}^k, \mathbf{R}) \right) \end{aligned} \tag{9}$$

where η_r is the learning rate for $\mathbf{R}$.

Specifically, for relation matrix $\mathbf{r}_i^{k+1}$ of client i, there are detailed update phases. With the fixed parameters of all the clients, the loss function of $\mathbf{r}_i^{k+1}$ can be represented as

$$f(\mathbf{r}_i) = \lambda f_s(\mathbf{r}_i \mathbf{W}^k, \mathbf{w}_i^k) + \frac{\beta}{2} \left\| \mathbf{r}_i - \frac{\mathbf{1}}{N} \right\|^2 \tag{10}$$

and then the updated $\mathbf{r}_i^{k+1}$ is

$$\begin{aligned} \mathbf{r}_i^{k+1} &= \mathbf{r}_i^k - \eta_r \nabla_{\mathbf{r}_i} f(\mathbf{r}_i) \\ &= \mathbf{r}_i^k - \eta_r \left(\lambda \nabla_{\mathbf{r}_i} f_s(\mathbf{r}_i \mathbf{W}^k, \mathbf{w}_i^k) + \beta \left(\mathbf{r}_i - \frac{\mathbf{1}}{N} \right) \right) \end{aligned} \tag{11}$$

where the gradient $\nabla_{\mathbf{r}_i} f_s$ is a matrix with the same size as $\mathbf{r}_i$ and can be computed as

$$\begin{aligned} &\nabla_{\mathbf{r}_i} f_s(\mathbf{r}_i \mathbf{W}^k, \mathbf{w}_i^k) \\ &= \left[\nabla_{r_{ij}^l} f_s \left(\sum_{j=1}^{N} r_{ij}^l \mathbf{w}_j^{l,k}, \mathbf{w}_i^{l,k} \right) \right]_{j=1,l=1}^{N,L} \end{aligned} \tag{12}$$

Clients Parameters Update. When the client parameters are updated in the $(k+1)$-th step, the relation cube in the current step is fixed. Based on the gradient decent method, the updated $\mathbf{W}^{k+1}$ is

$$\begin{aligned} \mathbf{W}^{k+1} &= \mathbf{W}^k - \eta_w \nabla_{\mathbf{W}} F_{per}(\mathbf{W}, \mathbf{R}^{k+1}) \\ &= \mathbf{W}^k - \eta_w \left(\nabla_{\mathbf{W}} F(\mathbf{W}) + \lambda \nabla_{\mathbf{W}} H(\mathbf{W}, \mathbf{R}^{k+1}) \right) \end{aligned} \tag{13}$$

where η_w is the learning rate for updating $\mathbf{W}$.

To better clarify the update process in Eq. (13), the detailed phases for individual client are provided. Firstly, the personalized regularizer, i.e., the server model, of client i is calculated in the $(k+1)$-th step to guide the local training, and can be denoted as

$$\mathbf{s}_i^{k+1} = \mathbf{r}_i^{k+1} \mathbf{W}^k, \tag{14}$$

where $\mathbf{r}_i^{k+1}$ is the fixed relation matrix at current iteration, and $\mathbf{W}^k$ is the parameters of all clients in the server. Then with the regularizer $\mathbf{s}_i^{k+1}$, the local loss Eq. (6) for updating $\mathbf{w}_i$ can be reformulated as

$$f(\mathbf{w}_i) = f_i(\mathbf{w}_i) + \lambda f_s(\mathbf{s}_i^{k+1}, \mathbf{w}_i) \tag{15}$$

Algorithm 1: Parameter Coaching Process in the Server

Input: communication rounds K, clients number N, update iterations of relation cube E_r, update iterations of client E

Output: the personalized parameters $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N]$

1: initialize and distribute $\mathbf{w}_i^1$ for client i, $i \in [1, N]$
2: initialize $\mathbf{R}^1$
3: **for** $k = 1$ to K **do**
4: **for** $i = 1$ to N in parallel **do**
5: **for** $t = 1$ to E_r **do**
6: obtain $\mathbf{R}^t$ by Eq. (9)
7: **end for**
8: $\mathbf{R}^{k+1} \leftarrow \mathbf{R}^{E_r}$
9: obtain the server parameters $\mathbf{s}_i^{k+1}$ by Eq. (14)
10: send the parameters $\mathbf{s}_i^{k+1}$ to client i
11: $\mathbf{w}_i^{k+1} \leftarrow$ Algorithm 2(E, $\mathbf{s}_i^{k+1}$)
12: **end for**
13: **end for**
14: **return** $\mathbf{W} = [\mathbf{w}_1^K, \mathbf{w}_2^K, \cdots, \mathbf{w}_N^K]$

Algorithm 2: Parameter Coaching Process in Client i

Input: update iterations of client parameters E, the corresponding server parameters at current step $\mathbf{s}_i^{k+1}$

Output: the client parameters $\mathbf{w}_i$

1: **for** $t = 1$ to E **do**
2: **for** each batch in dataset i **do**
3: update parameters $\mathbf{w}_i^{k+1}$ by Eq. (16)
4: **end for**
5: **end for**
6: **return** $\mathbf{w}_i^{k+1}$

Finally, according to Eq. (15), the updated model of client i in the $(k+1)$-th step is

$$\begin{aligned}\mathbf{w}_i^{k+1} &= \mathbf{w}_i^k - \eta_w \nabla_{\mathbf{w_i}} f(\mathbf{w}_i) \\ &= \mathbf{w}_i^k - \eta_w \left(\nabla_{\mathbf{w_i}} f_i(\mathbf{w}_i) + \lambda \nabla_{\mathbf{w_i}} f_s(\mathbf{s}_i^{k+1}, \mathbf{w}_i)\right)\end{aligned} \tag{16}$$

Furthermore, the update with Eq. (13) can be decoupled to the update with Eq. (16) of all the clients.

Parameter Coaching Procedures

Based on the above designs, the detailed procedures of the proposed knowledge-aware parameter coaching method in the server and client i are shown in Algorithm 1 and Algorithm 2, respectively.

As shown in Algorithm 1, the server firstly initializes the client parameters and relation cube randomly (lines 1-2). The relation cube is updated according to Eq. (9) with the current client parameters (lines 4-8). Then server model for each client is aggregated according to Eq. (14), and then distributed to the corresponding client (lines 9-10). The update for the client parameters is shown in Algorithm 2 (line 11). When the new models of all the clients are uploaded to the server, the new iteration begins. Finally, the personalized models of all the clients are returned (line 14). The training process of client i is shown in Algorithm 2. After the client receives the server parameters, the local model takes the server parameters as the regularizer to update the model (lines 1-5), and then the trained parameters are returned to the server (line 6).

Convergence Analysis

In this section, we analyze the convergence of the proposed method under both convex and non-convex settings. Different from the existing works in pFL (Dinh, Tran, and Nguyen 2020; Li et al. 2019), we introduce the optimization theories of Block Coordinate Descent (BCD) (Beck and Tetruashvili 2013; Bolte, Sabach, and Teboulle 2014) to pFL and give the proofs of the proposed theorems based on them. We extend the original BCD method from solving the single-variable with multi-dimension problem to the multi-variable problem by viewing the model parameters and the relation cube as different dimensions, and thus prove the convergence of the proposed method.

From the definition of $G(\mathbf{R})$, we can know $G(\mathbf{R})$ is closed, β-smooth and β-strongly convex for $\mathbf{R}$. Then the following assumptions are supposed.

Assumption 1 (closed) *$F(\mathbf{W})$ is a proper closed function.*

Assumption 2 (smooth) *(i) $F(\mathbf{W})$ is L_f-smooth; (ii) For any fixed $\mathbf{R}$, $H(\mathbf{W}, \mathbf{R})$ is L_1-smooth for $\mathbf{W}$. Likewise, for any fixed $\mathbf{W}$, $H(\mathbf{W}, \mathbf{R})$ is L_2-smooth for $\mathbf{R}$.*

Assumption 3 (convex) *(i) $F(\mathbf{W})$ is convex for $\mathbf{W}$; (ii) For any fixed $\mathbf{R}$, $H(\mathbf{W}, \mathbf{R})$ is convex for any $\mathbf{W}$. Likewise, for any fixed $\mathbf{W}$, $H(\mathbf{W}, \mathbf{R})$ is convex for any $\mathbf{R}$.*

Now, the convergence guarantee of the proposed method in the convex setting is provided.

Theorem 1 (Convergence in convex setting) *If Assumption 2 and Assumption 3 hold, the sequence $(\mathbf{W}^0, \mathbf{R}^0), \cdots, (\mathbf{W}^k, \mathbf{R}^k)$ generated by Algorithm 1 satisfies*

$$\begin{aligned}&F_{per}(\mathbf{W}^k, \mathbf{R}^k) - F_{per}(\mathbf{W}^*, \mathbf{R}^*) \\ \leq &\frac{2L_{max}}{k+4}\left(\left\|\mathbf{W}^0 - \mathbf{W}^*\right\|^2 + \left\|\mathbf{R}^0 - \mathbf{R}^*\right\|^2\right)\end{aligned} \tag{17}$$

where $(\mathbf{W}^*, \mathbf{R}^*)$ is the optimal solution of Eq. (8), $L_{max} = max\{L_w, L_r\}$, $L_w = L_f + \lambda L_1$, and $L_r = \beta + \lambda L_2$. Moreover, the learning rate of $\mathbf{W}$ is $\eta_w = \frac{1}{L_w}$ and the learning rate of $\mathbf{R}$ is $\eta_r = \frac{1}{L_r}$.

Theorem 1 implies that for any $\epsilon > 0$, the proposed method needs at least $O(\epsilon^{-1})$ iterations to find the sub-optimal solution $(\mathbf{W}_\epsilon, \mathbf{R}_\epsilon)$ of Eq. (8) such that $F_{per}(\mathbf{W}_\epsilon, \mathbf{R}_\epsilon) - F_{per}(\mathbf{W}^*, \mathbf{R}^*) \leq \epsilon$. It also establishes the global convergence for the proposed method in the convex setting. The complete proofs of Theorem 1 are given in Appendix B.

Then, with Assumptions 1 and 2, the convergence theory for the proposed method is developed under the non-convex setting.

Theorem 2 (Convergence in non-convex setting) *If Assumption 1 and Assumption 2 hold, Algorithm 1 is convergent, and a subsequence* $(\mathbf{W}^k, \mathbf{R}^k)$ *generated from Algorithm 1 with initial point* $(\mathbf{W}^0, \mathbf{R}^0)$ *can converge to the critical point of* F_{per}*, which satisfies:*

$$\lim_{k\to\infty} dist\left(\left(\mathbf{W}^k, \mathbf{R}^k\right), \omega\left(\mathbf{W}^0, \mathbf{R}^0\right)\right) = 0 \qquad (18)$$

$\omega(\mathbf{W}^0, \mathbf{R}^0) \in critF_{per}$ *is the set of limit points starting from* $(\mathbf{W}^0, \mathbf{R}^0)$ *and* $critF_{per}$ *is the critical points of* F_{per}*.* $dist(a, b)$ *is the distance between* a *and* b*.*

Theorem 2 provides theoretical guarantee of the proposed method under the non-convex setting. The complete proofs of Theorem 2 are given in Appendix C.

Experiment

In this section, we first introduce the experiment settings including the datasets, model structure, implementation details of the proposed method, and the baselines. Then we evaluate the accuracy of the proposed method on four datasets under various levels of heterogeneity, by comparisons with those of state-of-the-art personalized algorithms. We plot the curve of test accuracy and communication round to verify the convergence of the proposed method. Then, the impact of critical hyperparameters on the proposed method is evaluated as well. Finally, we visualize the relation matrix in relation cube to reveal the relevance among clients in shallow and deep layers.

All the experiments are repeated over 3 runs in Pytorch. During the training, we sample the accuracy of all the methods per round for MNIST and FMNIST, while 5 rounds for CIFAR10 and 10 rounds for CIFAR100.

Experiment Setting

Datasets. Four public benchmark datasets are used to evaluate the proposed method, MNIST, FMNIST, CIFAR10 and CIFAR100. To simulate the heterogeneous distribution of all the clients, the latent Dirichlet distribution method is adopted (Hsu, Qi, and Brown 2019), and the heterogeneity levels are set to $\alpha = \{0.1, 0.3\}$. In addition, two scenarios with and without client selection are provided. In the client selection scenario, there are 100 clients with 10% participation ratio, and all the training and test data in the dataset are used. In the scenario without client selection, there are 10 clients with 100% participation ratio, where 10% training data and test data is selected randomly from the dataset.

Model Architecture. For MNIST and FMNIST, a CNN model is used consisting of 2 convolutional layers with 5×5 filters followed by 3 fully connected layers with 512 and 128 hidden neurons. For CIFAR10 and CIFAR100, the same ResNet18 model as that in (He et al. 2016) is used.

Implementation Details. The model is trained by $K = 50$ rounds on MNIST/FMNIST, $K = 100$ rounds on CIFAR10, and $K = 200$ rounds on CIFAR100. The local epochs for $\mathbf{W}$ and $\mathbf{R}$ are set to 5 and 1 for all cases. In addition, cross-entropy loss and stochastic gradient descent method are adopted to update the client parameters and relation cube, and the learning rates for $\mathbf{W}$ and $\mathbf{R}$ are both set to 0.01.

Baselines. The proposed method is compared with various personalized methods related to regularization and layer-wise methods in addition to **local training** and **FedAvg** (McMahan et al. 2017). **FedProx** (Li et al. 2020) introduces proximal term to regularize the distance of local model and global model. **FedAMP** (Huang et al. 2021) weights personalized server models with cosine similarity to guide the local training. **MOON** (Li, He, and Song 2021) adopts contrastive learning to make local representation and global representation closer. **FedProto** (Tan et al. 2022b) utilizes prototype learning to regularize the local training of each client. **FedBABU** (Oh, Kim, and Yun 2021) shares the global extractor and trains the personalized classifier for each client. **FedBN** (Li et al. 2021b) keeps the batch normalization layer personalized and aggregates other layers by weighted averaging. **pFedLA** (Ma et al. 2022) learns the personalized model with layer-wise aggregation weights, and a hypernetwork for each client is trained to obtain the weights. In addition, for the above algorithms, the recommended hyperparameters utilized in their papers are adopted, but the communication round is same as that in our implementation for all the compared algorithms.

Performance Evaluation

We demonstrate the best mean accuracy of each baseline and the proposed method under the scenarios without and with client selection in Table 1 and Table 2, respectively.

The performance of most personalized methods is better than that of local training, which indicates the advantage of parameter sharing among clients. In addition, the performance of FedAvg method is evaluated on each client and the results show that the accuracy degrades as the heterogeneity of the clients increases. This indicates that FedAvg is not adaptive to heterogeneous clients.

Although all personalized methods demonstrate strong performance on heterogeneous datasets, the proposed method achieves the highest accuracy. Compared with regularization methods, e.g., FedProx, FedAMP, MOON and FedProto, the proposed method can capture the layer-wise similarity among different clients through the relation cube which enhances the personalized regularizer and provides fined-grained guidance for local training. On the other hand, we design a lightweight optimization method that efficiently updates client parameters and relation cube, so other layer-wise methods, e.g., FedBABU, FedBN and pFedLA, converge slower than the proposed method and lead to worse performance. Particularly, the results of pFedLA significantly underperform on all the datasets, which implies the hypernetwork employed by pFedLA to compute the relation matrix is difficult to train and converges slowly.

Convergence Evaluation

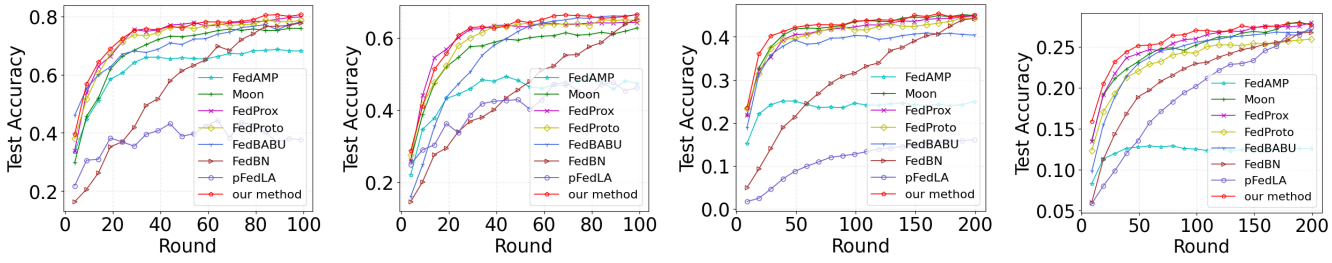
Theorems 1 and 2 present the convergence analysis of the proposed method under convex and non-convex settings, respectively. Specifically, Theorem 1 clearly demonstrates the

	MNIST(%)		FMNIST(%)		CIFAR10(%)		CIFAR100(%)	
α	0.1	0.3	0.1	0.3	0.1	0.3	0.1	0.3
local training	95.91(0.10)	93.13(0.66)	88.33(0.09)	84.77(0.23)	76.50(0.25)	54.15(0.23)	25.81(0.57)	15.56(0.62)
FedAvg	60.06(0.25)	64.78(0.04)	57.85(0.42)	76.47(0.32)	19.60(0.69)	17.92(0.50)	4.53(0.19)	4.11(0.28)
FedProx	97.93(0.66)	95.19(0.14)	96.24(0.19)	87.81(0.45)	86.83(0.76)	64.21(0.83)	37.43(0.42)	25.61(0.38)
FedAMP	97.10(0.84)	95.45(0.65)	94.43(0.12)	84.28(0.34)	78.55(0.58)	48.58(0.69)	18.04(0.24)	11.68(0.17)
MOON	97.77(0.04)	97.43(0.66)	96.61(0.35)	86.77(0.23)	88.27(0.24)	64.88(0.36)	26.33(0.07)	26.83(0.48)
FedProto	97.74(0.63)	97.32(0.58)	95.21(0.23)	86.29(0.76)	87.78(0.39)	63.99(0.66)	37.24(0.31)	26.99(0.30)
FedBABU	92.69(0.61)	90.39(0.16)	93.66(0.18)	85.54(0.37)	87.46(0.32)	65.19(0.21)	38.04(0.60)	26.96(0.57)
FedBN	83.99(0.72)	95.16(0.74)	75.14(0.13)	82.96(0.33)	71.42(0.92)	63.66(0.46)	36.10(0.08)	21.77(0.13)
pFedLA	97.80(0.31)	96.35(0.40)	91.23(0.28)	85.74(0.25)	86.78(0.84)	64.35(0.78)	29.13(0.93)	25.22(0.05)
our method	**97.97**(0.75)	**97.66**(0.38)	**96.71**(0.17)	**88.04**(0.46)	**88.32**(0.65)	**65.75**(0.84)	**38.25**(0.33)	**27.02**(0.58)

Table 1: Best mean accuracy in the scenario without client selection (10 clients) on four datasets with various heterogeneity levels. Bold fonts highlight the best accuracy.

	MNIST(%)		FMNIST(%)		CIFAR10(%)		CIFAR100(%)	
α	0.1	0.3	0.1	0.3	0.1	0.3	0.1	0.3
local training	96.11(0.26)	95.47(0.08)	89.31(0.88)	88.37(0.60)	73.41(0.24)	57.72(0.36)	37.41(0.07)	20.94(0.84)
FedAvg	41.84(0.15)	71.76(0.96)	51.38(0.38)	81.93(0.65)	18.42(0.99)	28.61(0.10)	4.25(0.68)	4.91(0.74)
FedProx	97.41(0.30)	94.94(0.24)	93.94(0.62)	90.18(0.34)	79.91(0.59)	64.76(0.24)	40.72(0.87)	27.92(0.45)
FedAMP	96.23(0.72)	91.52(0.41)	91.91(0.44)	86.11(0.21)	69.62(0.54)	49.62(0.52)	25.63(0.84)	13.08(0.89)
MOON	96.32(0.12)	95.42(0.11)	93.71(0.14)	60.70(0.56)	76.10(0.71)	62.71(0.85)	45.30(0.05)	28.00(0.71)
FedProto	97.27(0.29)	95.89(0.57)	94.42(0.26)	90.24(0.15)	78.58(0.56)	65.09(0.65)	44.49(0.74)	25.92(0.22)
FedBABU	96.10(0.07)	94.44(0.91)	92.95(0.64)	84.26(0.82)	78.75(0.11)	66.42(0.13)	41.19(0.83)	27.02(0.04)
FedBN	88.31(0.02)	95.56(0.31)	69.15(0.21)	84.24(0.37)	78.70(0.58)	65.24(0.48)	44.45(0.19)	26.88(0.14)
pFedLA	58.35(0.45)	63.49(0.55)	57.23(0.88)	77.36(0.31)	44.35(0.20)	42.74(0.20)	27.37(0.29)	16.16(0.31)
our method	**97.53**(0.61)	**96.17**(0.21)	**94.57**(0.25)	**90.34**(0.67)	**80.65**(0.78)	**66.51**(0.68)	**45.49**(0.55)	**28.05**(0.67)

Table 2: Best mean accuracy in the scenario with client selection (100 clients) on four datasets with various heterogeneity levels. Bold fonts highlight the best accuracy.



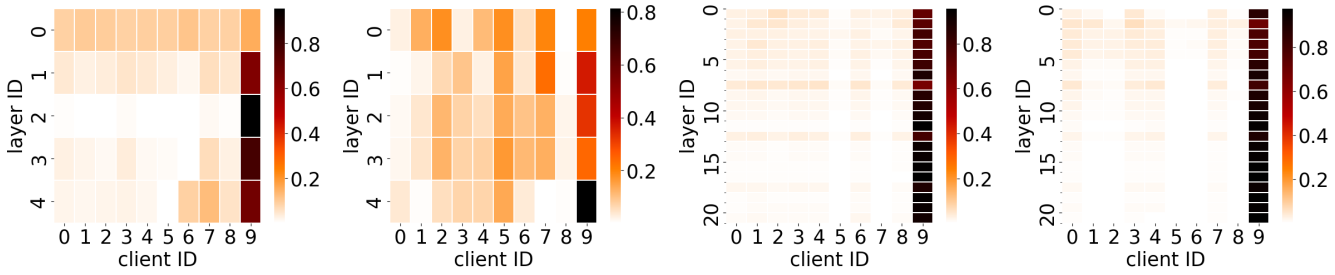
(a) CIFAR10 with $\alpha = 0.1$ (b) CIFAR10 with $\alpha = 0.3$ (c) CIFAR100 with $\alpha = 0.1$ (d) CIFAR100 with $\alpha = 0.3$

Figure 1: Comparison of convergence speed among the proposed method and baselines on CIFAR10 and CIFAR100 in the scenario with client selection.

upper bound and convergence speed under the convex setting, while Theorem 2 provides the convergence guarantee under the non-convex setting without presenting the convergence speed. Therefore, in this section, the experiments are conducted to evaluate the convergence speed of the proposed method with non-convex DNN model.

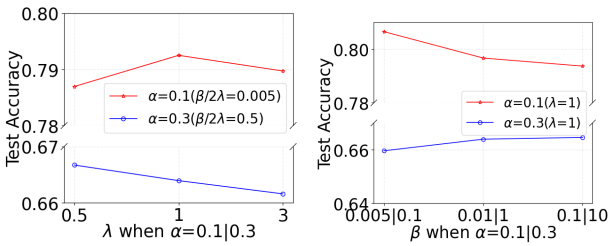
Experiments are conducted on CIFAR10 and CIFAR100 with $\alpha = \{0.1, 0.3\}$ under client selection scenario. The proposed method is compared with the above personalized baselines in the accuracy-round space. The results are shown in Figure 1.

As illustrated in Figure 1, the convergence speed of the proposed method is approximately same as partial regularization methods (e.g., FedProx and MOON), but is much faster than the layer-wise methods (e.g., FedBABU, FedBN and pFedLA), which indicates the proposed method can learn the layer-wise similarity among clients more efficiently. Moreover, the proposed method achieves higher test



(a) FMNIST with $\alpha = 0.1$ (b) FMNIST with $\alpha = 0.3$ (c) CIFAR100 with $\alpha = 0.1$ (d) CIFAR100 with $\alpha = 0.3$

Figure 2: The Relation Matrix of client 9 on FMNIST and CIFAR100 in the scenario without client selection.



(a) Various λs in $\alpha = \{0.1, 0.3\}$ (b) Various βs in $\alpha = \{0.1, 0.3\}$

Figure 3: The best mean accuracy with various hyperparameters. (a) The accuracy of the proposed method varying $\lambda = \{0.5, 1, 3\}$ with fixed $\frac{\beta}{2\lambda} = 0.005$ when $\alpha = 0.1$ and varying $\lambda = \{0.05, 1, 3\}$ with fixed $\frac{\beta}{2\lambda} = 0.5$ when $\alpha = 0.3$. (b) The accuracy of the proposed method varying $\beta = \{0.005, 0.01, 0.1\}$ and varying $\beta = \{0.1, 1, 10\}$ with both fixed $\lambda = 1$ when $\alpha = 0.1$ and $\alpha = 0.3$, respectively.

accuracy among all methods and further demonstrates the effectiveness of the proposed method.

Hyperparameter Evaluation

As shown in Eq. (6), the proposed method involves two critical hyperparameters λ and β. λ is used to trade off local general loss and personalized regularizer for updating client parameters, while β can effect regularization term for updating relation cube $\mathbf{R}$. According to Eq. (15), an appropriate larger value of λ can enhance the guidance of personalized regularizer and transfer more knowledge from other clients, but an increasing λ can also weaken the limit of equal share in Eq. (10), resulting in inadequate learning of $\mathbf{R}$.

Given the complex impacts of two hyperparameters, a comprehensive evaluation is conducted with $\alpha = \{0.1, 0.3\}$ on CIFAR10 under client selection scenario. In order to investigate the effects of λ and β independently, $\frac{\beta}{2\lambda}$ rather than β alone is fixed to quantify test accuracy when λ is varying.

Analysis of λ. The best mean accuracy varying λ in $\{0.5, 1, 3\}$ for both $\alpha = \{0.1, 0.3\}$ is shown in Figure 3 (a). Since the sharable knowledge is less with high heterogeneity level, $\frac{\beta}{2\lambda}$ is fixed with a small value (i.e., 0.005) for $\alpha = 0.1$ and the results indicate an excessively large or small λ can actually degrade test accuracy due to overfitting or underfitting the personalized regularizer when $\alpha = 0.1$. Additionally, when $\alpha = 0.3$, $\frac{\beta}{2\lambda}$ is fixed with a larger value (i.e., 0.5) that may transfer conflicted knowledge from other clients, so λ tends to be small values to reduce the guidance of the personalized regularizer.

Analysis of β. The best mean accuracy varying β in $\{0.005, 0.01, 0.1\}$ for $\alpha = 0.1$ and $\{0.1, 1, 10\}$ for $\alpha = 0.3$ is shown in Figure 3 (b). The accuracy with small β outperforms that with large β for $\alpha = 0.1$ while the tendency is contrary for $\alpha = 0.3$, which implies the fact that there is less sharable knowledge among higher-heterogeneous clients.

Vision of Relation Cube

In this subsection, we visualize the relation matrix of client 9 on FMNIST and CIFAR100 without client selection. As shown in Figure 2, the shallow layers of DNN models appear to be commonly shared by clients, such as layer 0 of the 2CNN model in FMNIST and layers $0 - 7$ of the ResNet18 model in CIFAR100, while the deeper layers seem to be more personalized. Additionally, clients with the same heterogeneity show distinct inter-layer correlations, verifying the necessity of layer-wise parameter sharing with different weights, which is the core idea of the proposed method.

Conclusion

In this paper, we have proposed a knowledge-aware parameter coaching method to leverage the granular knowledge among clients to guide local training efficiently. We have defined a relation cube to represent the similarity of heterogeneous knowledge in DNN layers among clients, which weights the personalized regularizer to share the mutually beneficial knowledge among clients. In addition, we have designed an efficient optimization method to alternately update the client parameters and relation cube, and the theoretical proofs of the proposed method under the convex and non-convex settings have been given. Finally, we have verified the proposed method on popular datasets with various levels of heterogeneity, which shows that the proposed method outperforms the state-of-the-art pFL methods in terms of accuracy and convergence speed.

Acknowledgments

This work was supported in part by National Key Research and Development Program of China under grand 2022YFE0114200, National Natural Science Foundation of China under grands U1808207 and 62302184, and a general research fund from RGC of the Hong Kong SAR, China (Project No. PolyU 15225023).

References

Acar, D. A. E.; Zhao, Y.; Matas, R.; Mattina, M.; Whatmough, P.; and Saligrama, V. 2021. Federated Learning Based on Dynamic Regularization. In *Proceedings of the 9th International Conference on Learning Representations*.

Arivazhagan, M. G.; Aggarwal, V.; Singh, A. K.; and Choudhary, S. 2019. Federated Learning with Personalization Layers. arXiv:1912.00818.

Beck, A.; and Tetruashvili, L. 2013. On the Convergence of Block Coordinate Descent Type Methods. *SIAM Journal on Optimization*, 23(4): 2037–2060.

Bolte, J.; Sabach, S.; and Teboulle, M. 2014. Proximal Alternating Linearized Minimization for Nonconvex and Nonsmooth Problems. *SIAM Journal on Optimization*, 146(1): 459–494.

Chen, Y.; Lu, W.; Qin, X.; Wang, J.; and Xie, X. 2022. MetaFed: Federated Learning among Federations with Cyclic Knowledge Distillation for Personalized Healthcare. arXiv:2206.08516.

Collins, L.; Hassani, H.; Mokhtari, A.; and Shakkottai, S. 2021. Exploiting Shared Representations for Personalized Federated Learning. In *Proceedings of the 38th International Conference on Machine Learning*, 2089–2099.

Dinh, C. T.; Tran, N. H.; and Nguyen, T. D. 2020. Personalized Federated Learning with Moreau Envelopes. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 21394–21405.

Duan, M.; Liu, D.; Ji, X.; Wu, Y.; Liang, L.; Chen, X.; Tan, Y.; and Ren, A. 2022. Flexible Clustered Federated Learning for Client-Level Data Distribution Shift. *IEEE Transactions on Parallel and Distributed Systems*, 33(11): 2661–2674.

Fallah, A.; Mokhtari, A.; and Ozdaglar, A. 2020. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 3557–3568.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. arXiv:1909.06335.

Huang, Y.; Chu, L.; Zhou, Z.; Wang, L.; Liu, J.; Pei, J.; and Zhang, Y. 2021. Personalized Cross-Silo Federated Learning on Non-IID Data. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 7865–7873.

Jiang, Y.; Konečnỳ, J.; Rush, K.; and Kannan, S. 2019. Improving Federated Learning Personalization via Model Agnostic Meta Learning. arXiv:1909.12488.

Jin, H.; Bai, D.; Yao, D.; Dai, Y.; Gu, L.; Yu, C.; and Sun, L. 2023. Personalized Edge Intelligence via Federated Self-Knowledge Distillation. *IEEE Transactions on Parallel and Distributed Systems*, 34(2): 567–580.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends in Machine Learning*, 14(1–2): 1–210.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *Proceedings of the 37th International Conference on Machine Learning*, 5132–5143.

Li, A.; Sun, J.; Li, P.; Pu, Y.; Li, H.; and Chen, Y. 2021a. Hermes: An Efficient Federated Learning Framework for Heterogeneous Mobile Clients. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 420–437.

Li, D.; and Wang, J. 2019. FedMD: Heterogenous Federated Learning via Model Distillation. arXiv:1910.03581.

Li, Q.; He, B.; and Song, D. 2021. Model-Contrastive Federated Learning. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10708–10717.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated Optimization in Heterogeneous Networks. In *Proceedings of the 3rd Machine Learning and Systems*, 429–450.

Li, X.; Huang, K.; Yang, W.; Wang, S.; and Zhang, Z. 2019. On the Convergence of FedAvg on Non-IID Data. In *Proceedings of the 7th International Conference on Learning Representations*.

Li, X.; Jiang, M.; Zhang, X.; Kamp, M.; and Dou, Q. 2021b. FedBN: Federated Learning on Non-IID Features via Local Batch Normalization. In *Proceedings of the 9th International Conference on Learning Representations*.

Li, X.; and Zhan, D. 2021. FedRS: Federated Learning with Restricted Softmax for Label Distribution Non-IID Data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Minings*, 995–1005.

Lin, S.; Yang, G.; and Zhang, J. 2020. Real-Time Edge Intelligence in the Making: A Collaborative Learning Framework via Federated Meta-Learning. arXiv:2001.03229.

Lin, T.; Kong, L.; Stich, S. U.; and Jaggi, M. 2020. Ensemble Distillation for Robust Model Fusion in Federated Learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2351–2363.

Luo, M.; Chen, F.; Hu, D.; Zhang, Y.; Liang, J.; and Feng, J. 2021. No Fear of Heterogeneity: Classifier Calibration for Federated Learning with Non-IID Data. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 5972–5984.

Ma, X.; Zhang, J.; Guo, S.; and Xu, W. 2022. Layer-wised Model Aggregation for Personalized Federated Learning. In *Proceddings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10082–10091.

Mansour, Y.; Mohri, M.; Ro, J.; and Suresh, A. T. 2020. Three Approaches for Personalization with Applications to Federated Learning. arXiv:2002.10619.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 1273–1282.

Mendieta, M.; Yang, T.; Wang, P.; Lee, M.; Ding, Z.; and Chen, C. 2022. Local Learning Matters: Rethinking Data Heterogeneity in Federated Learning. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8387–8396.

Mills, J.; Hu, J.; and Min, G. 2022. Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing. *IEEE Transactions on Parallel and Distributed Systems*, 33(3): 630–641.

Oh, J.; Kim, S.; and Yun, S.-Y. 2021. FedBABU: Towards Enhanced Representation for Federated Image Classification. arXiv:2106.06042.

Pillutla, K.; Malik, K.; Mohamed, A.; Rabbat, M. G.; Sanjabi, M.; and Xiao, L. 2022. Federated Learning with Partial Model Personalization. In *Proceedings of the 39th International Conference on Machine Learning*, 17716–17758.

Tan, A. Z.; Yu, H.; Cui, L.; and Yang, Q. 2022a. Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12): 9587–9603.

Tan, Y.; Long, G.; Liu, L.; Zhou, T.; Lu, Q.; Jiang, J.; and Zhang, C. 2022b. FedProto: Federated Prototype Learning across Heterogeneous Clients. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, 3–12.

Tang, Z.; Zhang, Y.; Shi, S.; He, X.; Han, B.; and Chu, X. 2022. Virtual Homogeneity Learning: Defending against Data Heterogeneity in Federated Learning. In *Proceedings of the 39th International Conference on Machine Learning*, 21111–21132.

Xu, J.; Tong, X.; and Huang, S.-L. 2023. Personalized Federated Learning with Feature Alignment and Classifier Collaboration. In *Proceedings of the 11th International Conference on Learning Representations*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How Transferable are Features in Deep Neural Networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 3320–3328.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and Understanding Convolutional Networks. In *Proceedings of the 13th European Conference on Computer Vision*, 818–833.

Zhang, J.; Guo, S.; Ma, X.; Wang, H.; Xu, W.; and Wu, F. 2021a. Parameterized Knowledge Transfer for Personalized Federated Learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 10092–10104.

Zhang, M.; Sapra, K.; Fidler, S.; Yeung, S.; and Alvarez, J. M. 2021b. Personalized Federated Learning with First Order Model Optimization. In *Proceedings of the 9th International Conference on Learning Representations*.