Knowledge-Aware Parameter Coaching for Communication-Efficient Personalized Federated Learning in Mobile Edge Computing

Mingjian Zhi, Yuanguo Bi, *Member, IEEE*, Lin Cai, *Fellow, IEEE*, Wenchao Xu, Haozhao Wang, Tianao Xiang, and Qiang He

Abstract—Personalized Federated Learning (pFL) can improve the accuracy of local models and provide enhanced edge intelligence without exposing the raw data in Mobile Edge Computing (MEC). However, in the MEC environment with constrained communication resources, transmitting the entire model between the server and the clients in traditional pFL methods imposes substantial communication overhead, which can lead to inaccurate personalization and degraded performance of mobile clients. In response, we propose a Communication-Efficient pFL architecture to enhance the performance of personalized models while minimizing communication overhead in MEC. First, a Knowledge-Aware Parameter Coaching method (KAPC) is presented to produce a more accurate personalized model by utilizing the layer-wise parameters of other clients with adaptive aggregation weights. Then, convergence analysis of the proposed KAPC is developed in both the convex and non-convex settings. Second, a Bidirectional Layer Selection algorithm (BLS) based on self-relationship and generalization error is proposed to select the most informative layers for transmission, which reduces communication costs. Extensive experiments are conducted, and the results demonstrate that the proposed KAPC achieves superior accuracy compared to the state-of-the-art baselines, while the proposed BLS substantially improves resource utilization without sacrificing performance.

Index Terms—Federated Learning, Personalization, Communication Optimization, Mobile Edge Computing.

1 Introduction

In Mobile Edge Computing (MEC), an edge server instead of a cloud server can centrally explore valuable data residing on mobile clients for model training, which can reduce latency and bandwidth usage, and enhance overall system performance [1], [2]. However, transmitting raw data from mobile clients to edge servers not only consumes substantial communication resources but also introduces the risk of data leakage. Federated Learning (FL) has been emerging as a secure and efficient distributed machine learning architecture, which can train a shared global model by aggregating the local model of each client [3]. By avoiding raw data exchange, FL can prevent the exposure of sensitive client information and reduce communication overhead. Consequently, FL has gained widespread adoption in MEC, providing edge intelligence in healthcare, intelligent traffic systems, and industrial engineering [4].

Due to the complexity of the physical environment in MEC, data maintained on each mobile client exhibits heterogeneity in both classes and quantity, implying that client data is non identically and independently distributed (non-IID). Thus, the traditional FL in MEC leads to a significant degradation in inference accuracy and convergence rate, which is attributed to the suboptimal solutions derived from non-IID clients being integrated into the global model [5]. Consequently, Personalized Federated Learning (pFL) is proposed to mitigate the non-IID issue, which allows distributed clients to customize their models individually instead of adopting a single global model [6].

Although the integration of pFL and MEC holds potential benefits, pFL methods encounter several challenges in MEC. First, insufficient exploration of heterogeneous knowledge within layered deep neural networks (DNN) adversely affects the performance of mobile clients. The parameters across different layers in a DNN model encapsulate diverse knowledge, varying impacts on distinct clients during the aggregation process [7], [8]. Consequently, employing uniform weights for all layers (e.g., in FedAvg [3]) can lead to the transfer of conflicting knowledge and the omission of similar knowledge, resulting in inaccurate personalized models. Particularly in the MEC setting, where unstable connectivity links prevail, inadequate exploration of layer-wise knowledge diminishes the utilization of client models and increases communication rounds to achieve convergence, thereby impeding training efficiency.

Secondly, the substantial computational overhead associated with fine-grained collaboration among many clients makes pFL in MEC impractical. The overhead is primarily attributed to utilizing model information from all clients to

- *M. Zhi, Y. Bi and T. Xiang are with the School of Computer Science and Engineering, Northeastern University, Shenyang 110169, China, e-mail: 2110657@stu.neu.edu.cn, biyuanguo@mail.neu.edu.cn, 2010652@stu.neu.edu.cn*
- *L. Cai is with the Department of Electrical & Computer Engineering, University of Victoria, Victoria, BC, Canada V8W3P6, e-mail: cai@ece.uvic.ca*
- *W. Xu is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China, e-mail: wenchao.xu@polyu.edu.hk*
- *H. Wang is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, e-mail: hz_wang@hust.edu.cn*
- *Q. He is with the School of Medicine and Biological Information Engineering, Northeastern University, Shenyang 110169, China, e-mail: heqiang@bmie.neu.edu.cn*

Corresponding author: Yuanguo Bi.

explore the layer-wise similarity, incurring a considerable burden on the server compared to pFL methods [9]. Given the prolonged training time due to excessive computational overhead, mobile clients may leave the system, which disrupts the training process. Therefore, the development of an efficient method to explore fine-grained relationships among clients becomes imperative.

Finally, the transmission of the entire model between mobile clients and the edge server results in a substantial consumption of communication resources, posing an intolerable burden on mobile clients with limited resources. Such high communication costs can lead to increased latency and packet loss rate, which cause low training efficiency.

To solve the above issues, by extending our previous work [10], we propose a communication-efficient pFL architecture, which is composed of the Knowledge-Aware Parameter Coaching method (KAPC) to enhance the local model of each client and Bidirectional Layer Selection method (BLS) to efficiently utilize the communication resources. Firstly, the server initializes a personalized model for each client, where the layer-wise parameters are derived from a linear combination of all clients' parameters with adaptive weights. Then the personalized server model is distributed to each client as a regularizer to guide the update of local layer-wise parameters. Moreover, the adaptive weights represented by a relationship cube can be updated efficiently based on the clients' parameters with less computation burden on the server side. In order to optimize the communication costs, we employ the generalization theory to identify the most valuable layers and upload the selected parameters instead of the entire parameters. The main contributions of this paper are summarized as follows.

- Based on the layered property of DNN models, we propose a Knowledge-Aware Parameter Coaching method (KAPC) to swiftly and granularly explore the similarity of layered knowledge across all clients, aimed at enhancing the personalization of the local model. Subsequently, a lightweight optimization method is designed to update the client parameters and fine-grained similarities.
- We prove that the proposed KAPC 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.
- We design a Bidirectional Layer Selection method (BLS) based on self-relationship and generalization bound to optimize the proposed KAPC in communication overhead. This method effectively reduces communication costs without sacrificing performance.
- Extensive experiments validate the robustness of the proposed KAPC across various levels of heterogeneity, showcasing its superior accuracy compared to the state-of-the-art personalized methods. Furthermore, the proposed BLS successfully reduces communication costs of upstream and downstream without compromising performance.

The rest of the paper is organized as follows. The related works are discussed in Section 2. Both the system architecture and the preliminary are presented in Section 3. The knowledge-aware parameter coaching method and its theoretical guarantees are introduced in Section 4. The bidirectional layer selection method is presented in Section 5. The communication-efficient pFL method combining the knowledge-aware parameter coaching method and the layer selection method is illustrated in Section 6. The experimental results on various datasets are showcased in Section 7. Finally, the paper is concluded in Section 8.

2 Related Works

2.1 Federated Learning in Mobile Edge Computing

The application of FL in MEC has recently gained attention for its potential to provide edge intelligence while keeping privacy preservation in resource-constrained environments. Researchers have explored innovative methods to seamlessly employ FL into MEC architectures, providing augmented privacy, minimized communication overhead, and heightened model accuracy. He *et al* [11] propose a novel scheme aimed at accelerating the training process in 6G-enabled MEC by jointly optimizing straggler issues and resource allocation strategies. Subsequently, Abdellatif *et al* [12] concentrate on the reliability of the global model when local data is dynamic and computational resources are limited within MEC. Additionally, Li *et al* [13] introduce an energy-aware FL training system, enhancing global performance while considering bandwidth capacity on an edge server and energy capacity on individual devices. From these studies, it can be deduced that the most challenging problem in FL training within MEC pertains to the limited resources available on clients.

2.2 Personalized Federated Learning

In order to solve the non-IID issue, many pFL methods have been developed by customizing the personalized model for each client to avoid the draft of the global model [14], [15], [16]. Among the existing methods, the regularization and layer-wise methods have been widely studied.

The regularization methods limit the local training with various regularizers to improve the personalization of the local model. Several studies construct the regularizer with model parameters to provide direct guidance for local training [17], [18]. Some works correct the update direction for each client to reduce the data drift in non-IID settings [19], [20]. Other works align the prototype of heterogeneous clients to enforce the learning for global extractor with less communication cost [21], [22]. Recent works regularize the local training with the soft label or statistic information to enhance the knowledge sharing from other clients [23], [24].

Considering the distinct representations of different layers in DNN, some personalized methods have been presented by developing layer-wise aggregation. Some methods keep the batch normalization layer personalized without aggregation in the server to avoid the drift of local features [25], [26]. Additionally, many works focus on the aggregation of partial shallow layers of DNN with the same weights to transfer the general knowledge among clients [27], [28]. Recently, pFedLA has considered the impacts of different layers and adopted hypernetwork to

generate layer-wise aggregation weights for each client, where the knowledge transfer conflicts can be avoided for heterogeneous clients [9].

In the abovementioned methods, most regularized pFL methods take the entire layered DNN model as a whole for aggregation, which may cause conflict knowledge transfer from other clients and thus degrade the accuracy of local models. Some layer-wise methods aim to aggregate the partial layers with the same weights, so they still explore the inter-client similarity in a coarse way and cause the inadequate personalization of local models. In pFedLA [9], 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.

2.3 Communication-efficiency Federated Learning

The traditional FL methods upload all the model parameters to the server, which burdens the limited communication resources in MEC. Therefore, communication-efficiency methods have attracted much attention [29]. Several works adopt communication compressing methods to reduce the communication cost, commonly including quantization and sparsification. Specifically, quantization-based methods [30], [31] represent the model parameters with a lower bit precision (e.g., 8-bit integers) instead of full precision (e.g., 32-bit floating-point numbers), which can reduce the amount of transmission data. Sparsification-based methods [32], [33] set a significant portion of model parameters to zero to avoid being transmitted between the server and clients. In addition, other works focus on the partial layers of the DNN model to be uploaded to enhance the global model and reduce the communication overhead [27], [28].

Although compression-based methods can be adopted in the server and clients to save both upstream and downstream communication resources, they may degrade the performance of aggregated models due to the compressed information, such as lower-precision parameters in quantization-based methods and fewer parameters in sparsification-based methods. On the other hand, many partial-layer-based methods send the feature extractor to the server for aggregation and keep the classifier personalized, but few works can provide a theoretical performance guarantee for the layer selection of the feature extractor.

3 System Architecture and Preliminary

3.1 System Architecture

The proposed system architecture is shown in Fig. 1, where the edge server plays a pivotal role in exploring fine-grained relationships and updating personalized server model for each client, and the end device trains the local model. Specifically, the server initiates and sends a personalized server model to each client. Subsequently, the client utilizes the server model as a regularizer to guide local training. Upon completion of local training, the client sends the parameters back to the server. Then, the server utilizes these parameters to update layer-wise weights and generates a new personalized regularizer, which is shown as the process on the edge server in Fig. 1. A comprehensive discussion of this personalized method is provided in Section 4. Furthermore, to address the constraints of limited communication resources in MEC, the proposed architecture incorporates an additional communication-efficient component. This component minimizes uploading and downloading parameters between the server and clients by selecting the most informative layers, which is illustrated in the clients in Fig. 1. The detailed communication-efficient method is proposed in Section 5.

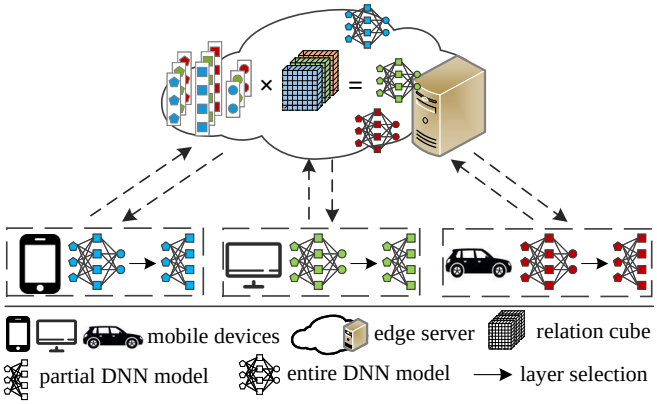


Fig. 1: The illustration of the system architecture.

3.2 The Preliminary

Assuming there are N clients in the MEC. Client i is equipped with a dataset $D_i = \{(\mathbf{x}_{ij}, y_{ij})\}_{j=1}^{m_i}$ independently and identically (IID) sampled from a distribution P_i, where $i \in [1, N]$, $x_{ij} \in \mathcal{X}$, $y_{ij} \in \mathcal{Y}$ and m_i is the data number of dataset D_i. In general, suppose $P_i \neq P_j$ if $i \neq j$ to simulate heterogeneous clients. Moreover, the DNN model is used in our settings, and $h(\cdot; \mathbf{w}_i) \in \mathcal{H}$ 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. An activation function with 1-Lipschitz continuity $\phi(\cdot)$ is adopted in DNN model. Consequently, the output of l-th layer DNN can be represented as $h(h(\cdot, \mathbf{w}_i^{l-1}), \mathbf{w}_i^l) = \phi(\mathbf{w}_i^l \cdot h(\cdot, \mathbf{w}_i^{l-1}))$ which can be written as h_i^l for short. The bias is neglected for sample representation.

The local loss function for client i can be defined as

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

where $g(.)$ is the loss function. Then, the global loss function of personalized federated learning is given by

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

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

Moreover, since the layer selection method on the client side is based on the generalization theory, the related definition is presented as follow. For client i, the empirical loss

defined as Eq. (1) is denoted as $f_{D_i}(\mathbf{w}_i)$ in the generalization theory, while the expected loss can be denoted as

$$f_{P_i}(\mathbf{w}_i) = \mathop{\mathbb{E}}_{(\mathbf{x}_{ij},y_{ij})\sim P_i} g\left(h\left(x_{ij};\mathbf{w}_i\right),y_{ij}\right). \tag{3}$$

4 Knowledge-aware Parameter Coaching Method

Based on the above architecture, a Knowledge-Aware Parameter Coaching method, termed as KAPC, is proposed, which leverages the layer-wise similarities among clients and facilitates knowledge sharing in pFL. Specifically, the server models are updated with the parameters of the clients weighted by a relationship cube as defined in subsection 4.1. Subsequently, the updated server model acts as a regularizer to guide the local training for the corresponding client as illustrated in subsection 4.2. Once the clients have completed their training, the relationship cube is updated with the new parameters of clients in the server as explained in subsection 4.3.

4.1 Relationship Cube

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

$$\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N], \tag{4}$$

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

$$\mathbf{r}_i = \begin{bmatrix} r_{i1}^1, & r_{i2}^1 & \cdots & r_{iN}^1 \\ r_{i1}^2, & r_{i2}^2 & \cdots & r_{iN}^2 \\ \vdots & \vdots & \ddots & \vdots \\ r_{i1}^L, & r_{i2}^L & \cdots & r_{iN}^L \end{bmatrix}. \tag{5}$$

The relationship cube $\mathbf{R}$ is composed of the relationship matrix $\mathbf{r}_i$ of each client $i, i \in [1, N]$, which represents the similarity of layered knowledge between client i and other clients. The relationship matrix $\mathbf{r}_i$ is composed of relationship coefficient $r_{ij}^l, j \in [1, N]$, which represents the similarity between client i and client j on DNN layer l. The relationship matrix also can be represented as a vector of relationship vector, that is, $\mathbf{r}_i = [\mathbf{r}_i^1, \mathbf{r}_i^2, \cdots, \mathbf{r}_i^L]$, where relationship vector $\mathbf{r}_i^l = [r_{i1}^l, r_{i2}^l, \cdots, r_{iN}^l]$ is the relationship between client i and other clients on layer l. A schematic structure of $\mathbf{R}$ is shown in Figure 2.

4.2 Optimization Objective

Unlike general federated learning, the proposed KAPC maintains a specific server model for each client to aggregate the beneficial knowledge from other clients. The parameters of the personalized server model are derived by the linear combination of the clients' parameters weighted by the relationship cube. The server parameters are leveraged as a regularizer to coach the local training. Therefore, the optimization objective not only minimizes the empirical loss function of the client as general personalized federated learning, but also minimizes the distance between the local

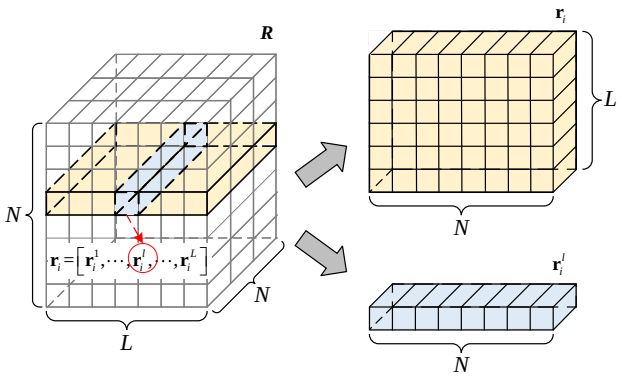


Fig. 2: The schematic structure of relationship cube $\mathbf{R}$. Relationship matrix $\mathbf{r}_i$ is shown as a yellow block, which represents the relationship between client i and other clients on layered knowledge. Relationship vector $\mathbf{r}_i^l$ is shown as a blue block, which represents the relationship of client i with other clients on layers l.

parameters and server parameters. The formulation of the local loss function for client i can be represented as

$$\begin{aligned} f(\mathbf{w}_i, \mathbf{r}_i) =& f_i(\mathbf{w}_i) + \lambda \sum_{l=1}^{L} f_s\left(\sum_{j=1}^{N} r_{ij}^l \mathbf{w}_j^l, \mathbf{w}_i^l\right) \\ &+ \frac{\beta}{2} \sum_{j=1}^{N}\sum_{l=1}^{L} \left\| r_{ij}^l - \frac{1}{N} \right\|^2, \end{aligned} \tag{6}$$

where $f_i(\mathbf{w}_i)$ is the general loss function of client i defined by Eq. (1), f_s is the coaching loss function to guide the layer-wise local training, and we adopt $L2$-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 function and regularization term.

When the loss function is stacked by layer, Eq. (6) can be rewritten as

$$f(\mathbf{w}_i, \mathbf{r}_i) = f_i(\mathbf{w}_i) + \lambda f_s(\mathbf{r}_i\mathbf{W}, \mathbf{w}_i) + \frac{\beta}{2} \left\| \mathbf{r}_i - \frac{\mathbf{1}}{N} \right\|^2, \tag{7}$$

where $\mathbf{1}\in\mathbb{R}^{L\times N}$ is an identity matrix with the same size as $\mathbf{r}_i$.

Then, the global optimization objective can be formulated as

$$\min_{\mathbf{W},\mathbf{R}} \left\{ F_{per}(\mathbf{W}, \mathbf{R}) := \sum_{i=1}^{N} \frac{m_i}{m} f(\mathbf{w}_i, \mathbf{r}_i) \right\}, \tag{8}$$

where $\mathbf{W}$ is the parameters set of clients in Eq. (2), and $\mathbf{R}$ is the relationship cube defined by Eq. (4).

4.3 Optimization Method

In order to update client parameters and the relationship cube, we develop an alternative optimization method, which includes two updating steps in the $(k+1)$-th iteration. Firstly, when client parameter set $\mathbf{W}^k$ is fixed, the relationship cube $\mathbf{R}^{k+1}$ in the $(k+1)$-th iteration can be updated. Then, after the relationship cube is updated and the personalized regularizer is determined, the client parameter set will be optimized.

To describe the update method in detail, we denote the first, second and third term of $F_{per}(\mathbf{W},\mathbf{R})$ 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 (8) can be rewritten as

$$\min_{\mathbf{W},\mathbf{R}}\left\{F_{per}(\mathbf{W},\mathbf{R}) := F(\mathbf{W}) + \lambda H(\mathbf{W},\mathbf{R}) + G(\mathbf{R})\right\}. \quad (9)$$

Relationship Cube Update. When the relationship 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

$$\begin{aligned}\mathbf{R}^{k+1} &= \mathbf{R}^k - \eta_r\nabla_{\mathbf{R}}F_{per}(\mathbf{W}^k,\mathbf{R})\\ &= \mathbf{R}^k - \eta_r\left(\nabla_{\mathbf{R}}G(\mathbf{R}) + \lambda\nabla_{\mathbf{R}}H(\mathbf{W}^k,\mathbf{R})\right),\end{aligned} \quad (10)$$

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

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

$$f(\mathbf{r}_i) = \lambda f_s(\mathbf{r}_i\mathbf{W}^k,\mathbf{w}_i^k) + \frac{\beta}{2}\left\|\mathbf{r}_i - \frac{\mathbf{1}}{N}\right\|^2. \quad (11)$$

Secondly, $\mathbf{r}_i^{k+1}$ can be updated with gradient descent method as

$$\begin{aligned}\mathbf{r}_i^{k+1} &= \mathbf{r}_i^k - \eta_r\nabla_{\mathbf{r}_i}f(\mathbf{r}_i)\\ &= \mathbf{r}_i^k - \eta_r\left(\lambda\nabla_{\mathbf{r}_i}f_s(\mathbf{r}_i\mathbf{W}^k,\mathbf{w}_i^k) + \beta\left(\mathbf{r}_i - \frac{\mathbf{1}}{N}\right)\right),\end{aligned} \quad (12)$$

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

$$\begin{aligned}&\nabla_{\mathbf{r}_i}f_s(\mathbf{r}_i\mathbf{W}^k,\mathbf{w}_i^k)\\ &= \left[\nabla_{r_{ij}^l}f_s\left(\sum_{j=1}^{N}r_{ij}^l\mathbf{w}_j^{l,k},\mathbf{w}_i^{l,k}\right)\right]_{j=1,l=1}^{N,L}.\end{aligned} \quad (13)$$

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

$$\begin{aligned}\mathbf{W}^{k+1} &= \mathbf{W}^k - \eta_w\nabla_{\mathbf{W}}F_{per}(\mathbf{W},\mathbf{R}^{k+1})\\ &= \mathbf{W}^k - \eta_w\left(\nabla_{\mathbf{W}}F(\mathbf{W}) + \lambda\nabla_{\mathbf{W}}H(\mathbf{W},\mathbf{R}^{k+1})\right),\end{aligned} \quad (14)$$

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

To better clarify the update process in Eq. (14), the detailed phases for individual client are provided. Firstly, the relationship matrix $\mathbf{r}_i^{k+1}$ is normalized, and then the personalized regularizer, i.e., the server model, of client i is calculated in the $(k+1)$-th step to guide the local training, which is denoted as

$$\mathbf{s}_i^{k+1} = \mathbf{r}_i^{k+1}\mathbf{W}^k. \quad (15)$$

In Eq.(15), $\mathbf{r}_i^{k+1}$ is the fixed relationship matrix in the current iteration, and $\mathbf{W}^k$ is the parameters of all clients in the server. Next, with the regularizer $\mathbf{s}_i^{k+1}$, the local loss for updating $\mathbf{w}_i$ can be reformulated as

$$f(\mathbf{w}_i) = f_i(\mathbf{w}_i) + \lambda f_s(\mathbf{s}_i^{k+1},\mathbf{w}_i). \quad (16)$$

Finally, according to Eq. (16), the updated model of client i in the $(k+1)$-th step is

$$\begin{aligned}\mathbf{w}_i^{k+1} &= \mathbf{w}_i^k - \eta_w\nabla_{\mathbf{w_i}}f(\mathbf{w}_i)\\ &= \mathbf{w}_i^k - \eta_w\left(\nabla_{\mathbf{w}_i}f_i(\mathbf{w}_i) + \lambda\nabla_{\mathbf{w}_i}f_s(\mathbf{s}_i^{k+1},\mathbf{w}_i)\right).\end{aligned} \quad (17)$$

Furthermore, the update with Eq. (14) can be divided into the update with Eq. (17) of all the clients.

Algorithm 1 The procedures of KAPC in the server

Input: the number of communication K, the number of clients N, the number of update iterations for relationship cube E_r, the number of update iterations for all the clients 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** $t=1$ to E_r **do**
5: obtain $\mathbf{R}^t$ by Eq. (10)
6: **end for**
7: $\mathbf{R}^{k+1}\leftarrow\mathbf{R}^{E_r}$
8: **for** $i=1$ to N in parallel **do**
9: obtain the server parameters $\mathbf{s}_i^{k+1}$ by Eq. (15)
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 The procedures of KAPC in client i

Input: the number of update iterations 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. (17)
4: **end for**
5: **end for**
6: **return** $\mathbf{w}_i^{k+1}$

4.4 Parameter Coaching Procedures

Based on the above designs, the detailed procedures of the proposed KAPC in the server and client i are shown in Algorithm 1 and Algorithm 2, respectively.

As depicted in Algorithm 1, the server initializes the client parameters and relationship cube randomly, where the round index starts from 1 (lines 1-2). The relationship cube is then updated using Eq. (10) with the current client parameters (lines 4-7). Next, the server aggregates the personalized model for each client according to Eq. (15) (lines 8-9), distributes them to the respective clients (line 10), and updates the client parameters with Algorithm 2 (line 11).

Once all the clients have uploaded their new models to the server, a new iteration begins. Finally, the server returns the personalized models of all the clients (line 14).

The training process for client i is outlined in Algorithm 2. Upon receiving the server parameters, the client i utilizes them as a regularizer to update th local model (lines 1-5), and subsequently, the trained parameters are returned to the server (line 6).

4.5 Convergence Analysis

In this subsection, the convergence of the proposed KAPC is analyzed under both the convex and non-convex settings. Different from the existing works in pFL [18], [34], we introduce the optimization theories of Block Coordinate Descent (BCD) [35], [36] to pFL and give the proof of the following 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 relationship cube as different dimensions, and thus, prove the convergence of the proposed method. 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, we provide the convergence guarantee for the proposed method under the convex setting.

Theorem 1 (Convergence under the 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

$$\begin{aligned} &F_{per}(\mathbf{W}^k,\mathbf{R}^k)-F_{per}(\mathbf{W}^*,\mathbf{R}^*)\\ \leq&\frac{2L_{\max}}{k+4}\left(\left\|\mathbf{W}^0-\mathbf{W}^*\right\|^2+\left\|\mathbf{R}^0-\mathbf{R}^*\right\|^2\right), \end{aligned} \tag{18}$$

where $(\mathbf{W}^*,\mathbf{R}^*)$ is the optimal solution of Eq. (9), $L_{\max}=\max\{L_w,L_r\}$, $L_w=L_f+\lambda L_1$, 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 $\eta_2=\frac{1}{L_r}$.

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. (9) 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 KAPC in a convex setting. The full proofs of Theorem 1 are given in Appendix B.

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

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

$$\lim_{k\to\infty} dist\left(\left(\mathbf{W}^k,\mathbf{R}^k\right),\omega\left(\mathbf{W}^0,\mathbf{R}^0\right)\right)=0. \tag{19}$$

$\omega(\mathbf{W}^0,\mathbf{R}^0)\in critF_{per}$ is the set of limit points of the sequence 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 the theoretical guarantee of the proposed method under the non-convex setting. The complete proofs of Theorem 2 are given in Appendix C.

The above analysis is based on $G(\mathbf{R})$ with $L2$-norm, which is closed, β-smooth and convex for $\mathbf{R}$. Thus, if adopting other coaching loss functions with the β-smoothness and convexity, the above theorems still hold.

5 Bidirectional Layer Selection Method

In this section, a Bidirectional Layer Selection method (BLS) is proposed, which aims to optimize communication overhead for KAPC. As illustrated in Fig. 3, BLS involves transmitting partial layers of the DNN model between the server and clients without compromising performance. The BLS process begins with updating the relationship cube in KAPC using current client models. Next, the layered regularizer is updated with the relationship vector, excluding vectors with high self-relationships. Subsequently, the server distributes the partial-layered regularizer, which guides the local training of clients as KAPC designs. After local training, a layer selection method based on generalization bound is executed, and the parameters of the selected layers are uploaded to the server for the update in the next round. The detailed methods in the server and clients are introduced in Section 5.1 and Section 5.2, respectively.

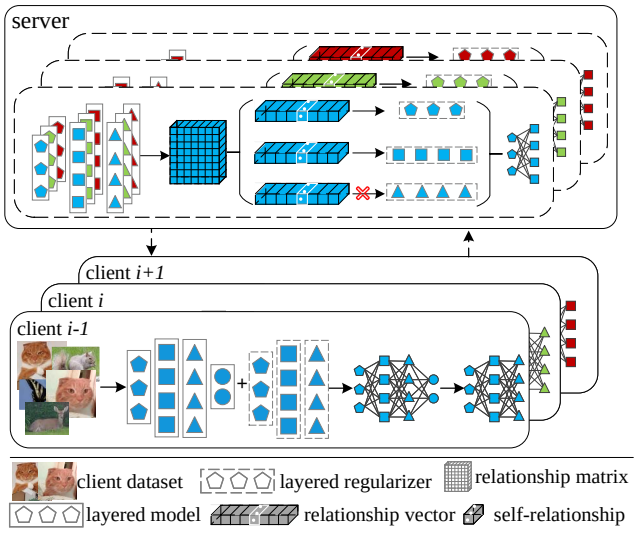


Fig. 3: The illustration of BLS method in the server and clients.

Algorithm 3 The procedures of layer selection with self-relationship in the server for client i at k-th iteration

Input: the threshold of self-relationship μ, the current training round k, the regularizer parameters of client i at k-th iteration $\mathbf{s}_i^k$, the relationship matrix of client i at k-th iteration $\mathbf{r}_i^k$

Output: the partial-layer regularizer at current iteration after layer selection for client i, $\hat{\mathbf{s}}_i^{k+\frac{1}{2}}$

1: **for** $l = 1$ to L **do**
2: **if** $r_{ii}^{l,k} \geq \mu$ **then**
3: $\mathbf{s}_i^{l,k} \leftarrow \phi$
4: **end if**
5: **end for**
6: $\hat{\mathbf{s}}_i^{k+\frac{1}{2}} \leftarrow \mathbf{s}_i^k$
7: **return** $\hat{\mathbf{s}}_i^{k+\frac{1}{2}}$

5.1 The Layer Selection with Self-Relationship in the Server

According to the definition of $\mathbf{s}_i$, the l-th layered regularizer parameters of client i (i.e., $\mathbf{s}_i^l$) can be expressed as a linear combination of corresponding layer parameters from other client models, which is illustrated in Fig. 4.

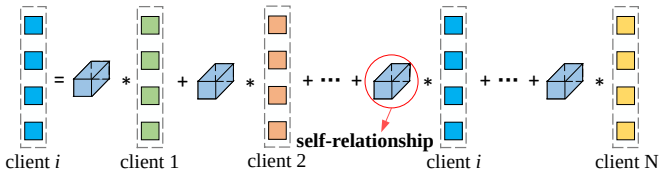


Fig. 4: The illustration of $\mathbf{s}_i^l$ ($\mathbf{s}_i^l$ can be derived as the linear combination of corresponding layer parameters from other clients).

However, when self-relationship r_{ii}^l is excessively high, the value of $\mathbf{s}_i^l$ becomes nearly identical to $\mathbf{w}_i^l$, which indicates it does not acquire valuable knowledge from other clients. Consequently, a layered regularizer with a high self-relationship cannot offer constructive guidance to client i when acting as a personalized regularizer. In order to reduce the communications, we avoid distributing $\mathbf{s}_i^l$ when its self-relationship r_{ii}^l exceeds a predetermined threshold. Although the relationship vector with a high self-relationship is not used to update the new layered regularizer, it reveals the accurate inter-layer relationships among clients and is preserved in the relationship cube in the server.

The detailed layer selection method in the server is shown as Algorithm 3. Firstly, the regularizer parameters and relationship matrix of client i are given, then the selection criterion for undownloaded layers is evaluated for each layer of client i (lines 1-2). The selected layers of personalized regularizer are set to empty (lines 3-4). Finally, the regularizer with partial layers is returned (lines 6-7).

5.2 The Layer Selection with Generalization in the Client

5.2.1 Layer Selection Criterion

The regularization method can typically reduce generalization error and further enhance the performance of model. However, the reduction of generalization error varies across different layered for the DNN model. To reduce communication costs, we select the layers that can reduce generalization error most effectively when utilized as a regularizer. Specifically, the average generalization error with the selected-layered regularizer is less than that with a full-layered regularizer.

In KAPC, shallow layers are responsible for capturing general features, while deeper layers capture more specific features. Uploading shallow layers allows the model to absorb general knowledge from other clients, thereby enhancing its ability to recognize basic features. In contrast, deeper layers, which represent more personalized features, require less information from other clients. Consequently, sharing shallow layers more efficiently improves the performance of the local model and reduces generalization error, whereas sharing deeper layers results in a less pronounced decrease. Moreover, the generalization error can be estimated by the gap between expected loss and empirical loss, and its derivation requires a continuous loss function [37]. Therefore, the continuous layers from the beginning should be selected for sharing, and the problem of layer selection in the client is transformed to determining how many layers should be uploaded.

For client i, the full-layered regularizer is denoted as s_i and $f_{D_i}(\mathbf{w}_i, \mathbf{s}_i) = f_{D_i}(\mathbf{w}_i) + \lambda\|\mathbf{w}_i - \mathbf{s}_i\|^2$ as the empirical loss with full-layered regularizer. Likewise, the partial-layered regularizer is denoted as $\hat{\mathbf{s}}_i$ and $f_{D_i}(\hat{\mathbf{w}}_i, \hat{\mathbf{s}}_i) = f_{D_i}(\hat{\mathbf{w}}_i) + \lambda\|\hat{\mathbf{w}}_i - \hat{\mathbf{s}}_i\|^2$ as the empirical loss with partial-layered regularizer. We define $\hat{\mathbf{s}}_i$ as

$$\hat{\mathbf{s}}_i^l = \begin{cases} \mathbf{r}_i^l \mathbf{W}^l, & \text{if } l \leq \Omega_i, \\ \hat{\mathbf{w}}_i^l, & l > \Omega_i. \end{cases} \tag{20}$$

where $\Omega_i \in [1, L]$ is the number of uploaded layers for client i, and $\hat{\mathbf{w}}_i^l$ is the l-th layered parameters of local model in the last round. Then, the criterion to select uploaded layers can be defined as

Definition 1 (The Layer-Selection Criterion of Client i).

$$\frac{f_{P_i}(\hat{\mathbf{w}}_i, \hat{\mathbf{s}}_i) - f_{D_i}(\hat{\mathbf{w}}_i, \hat{\mathbf{s}}_i)}{\Omega_i} \leq \frac{f_{P_i}(\mathbf{w}_i, \mathbf{s}_i) - f_{D_i}(\mathbf{w}_i, \mathbf{s}_i)}{L}, \tag{21}$$

where $f_{P_i}(\hat{\mathbf{w}}_i, \hat{\mathbf{s}}_i)$ is expected loss with partial-layered regularizer and $f_{P_i}(\mathbf{w}_i, \mathbf{s}_i)$ is expected loss with full-layered regularizer.

5.2.2 The Quantization of the Criterion

In order to quantify the criterion, Theorem 2 in [37] is introduced to bound generalization error of the loss function with full-layered regularizer (i.e., $f_{P_i}(\mathbf{w}_i, \mathbf{s}_i) - f_{D_i}(\mathbf{w}_i, \mathbf{s}_i)$ in Definition 1). For client i, the problem to minimize empirical loss with a full-layered regularizer (i.e., Eq. (16)) can be reformulated and slacked as

$$\begin{aligned} \arg\min_{\mathbf{w}_i} f_{D_i}(\mathbf{w}_i) &:= \frac{1}{m_i} \sum_{j=1}^{m_i} g\left(h\left(x_{ij}; \mathbf{w}_i\right), y_{ij}\right) \\ s.t. \|\mathbf{w}_i^l - \mathbf{s}_i^l\|^2 &\leq (\tau_i^l)^2, \end{aligned} \tag{22}$$

where $l \in [1, L]$ is the l-th layer of the DNN model and τ_i^l is slack variable of layer l. $f_{D_i}(\mathbf{w}_i)$ is the empirical loss of client i defined as Eq. (1), and $g(\cdot)$ is a ρ-Lipschitz function.

The function family of the solution for problem (22) is denoted as $\mathcal{F}_{s_i}$ with bound $[a,b]$. In addition, we denote the $\max\|\mathbf{w}_i^l\|^2=(B_i^l)^2$, then $\max\|\mathbf{s}_i^l\|^2=(B_i^l)^2$ can be inferred according to Eq. (15).

Theorem 2 in [37] can be rewritten as follows to demonstrate the generalization bound of $\mathcal{F}_{s_i}$.

Theorem 3 (The Generalization Bound of $\mathcal{F}_{s_i}$). For any $\delta\in[0,1]$, with the probability at least $1-\delta$, the generalization of $f\in\mathcal{F}_{s_i}$ can be bounded by

$$\begin{aligned}f_{P_i}(\mathbf{w}_i,\mathbf{s}_i)\leq& f_{D_i}(\mathbf{w}_i,\mathbf{s}_i)+\frac{2\sqrt{2}c_i\rho A_i\cdot 2^L}{\sqrt{m_i}}\cdot\prod_{j=1}^{L}{B_i}^j\\&\cdot\left(\frac{\tau^L}{2B_i^L}+\sum_{k=1}^{L-1}\frac{\tau^{L-k}\prod_{j=1}^{k}\sqrt{n^{L+1-j}}}{2{B_i}^{L-k}}\right)\\&+3(b-a)\sqrt{\frac{log\frac{2}{\delta}}{2m_i}},\end{aligned}\tag{23}$$

where c_i is the class number of dataset D_i, $\|\mathbf{x}_{ij}\|\leq A_i$, and n^l is the number of columns in $\mathbf{w}_i^l$.

Correspondingly, to bound the generalization error of the loss function with paritial-layered regularizer (i.e., $f_{P_i}(\hat{\mathbf{w}}_i,\hat{\mathbf{s}}_i)-f_{D_i}(\hat{\mathbf{w}}_i,\hat{\mathbf{s}}_i)$ in Definition 1), the problem to minimize empirical loss with partial-layer regularizer can be formulated as

$$\begin{aligned}&\arg\min_{\hat{\mathbf{w}}_i} f_{D_i}(\hat{\mathbf{w}}_i):=\frac{1}{m_i}\sum_{j=1}^{m_i}g\left(h\left(x_{ij};\hat{\mathbf{w}}_i\right),y_{ij}\right)\\&s.t.\|\hat{\mathbf{w}}_i^l-\hat{\mathbf{s}}_i^l\|^2\leq(\hat{\tau}_i^l)^2,\end{aligned}\tag{24}$$

where $\hat{\tau}_i^l$ is slack variable of layer l. Simultaneously, we denote $\max\|\hat{\mathbf{w}}_i^l\|^2=(\hat{B}_i^l)^2$ and infer $\max\|\hat{\mathbf{s}}_i^l\|^2=(\hat{B}_i^l)^2$ due to the normalized weights for aggregation.

According to the definition of $\hat{\mathbf{s}}_i^l$ (i.e., Eq. (20)), we can bound $\hat{\tau}_i^l$ more precisely, that is,

$$\hat{\tau}_i^l=\begin{cases}\hat{\tau}_i^l, & \text{if } l\leq\Omega_i,\\ 0, & \text{if } l>\Omega_i.\end{cases}\tag{25}$$

Likewise, the function family of solution for problem (24) is denoted as $\mathcal{F}_{\hat{s}_i}$, and the bound of $\mathcal{F}_{\hat{s}_i}$ can be included in $[a,b]$. Then, with the same assumption and proof architecture of Theorem 3, the generalization bound of $\mathcal{F}_{\hat{s}_i}$ is provided as follows.

Theorem 4 (The Generalization Bound of $\mathcal{F}_{\hat{s}_i}$). For any $\delta\in[0,1]$, with the probability at least $1-\delta$, the generalization of $f\in\mathcal{F}_{\hat{s}_i}$ can be bounded by

$$\begin{aligned}f_{P_i}(\hat{\mathbf{w}}_i,\hat{\mathbf{s}}_i)\leq& f_{D_i}(\hat{\mathbf{w}}_i,\hat{\mathbf{s}}_i)+\frac{2\sqrt{2}d_{\Omega_i}\rho A_i\cdot 2^{\Omega_i}}{\sqrt{m_i}}\cdot\prod_{j=1}^{L}\hat{B}_i^j\\&\cdot\left(\frac{\hat{\tau}_i^{\Omega_i}}{2\hat{B}_i^{\Omega_i}}+\sum_{k=1}^{\Omega_i-1}\frac{\hat{\tau}_i^{\Omega_i-k}\prod_{j=1}^{k}\sqrt{n^{\Omega_i+1-j}}}{2\hat{B}_i^{\Omega_i-k}}\right)\\&+3(b-a)\sqrt{\frac{log\frac{2}{\delta}}{2m_i}},\end{aligned}\tag{26}$$

where d_{Ω_i} is the output dimension of the Ω_i-th layered DNN $h_i^{\Omega_i}$.

The complete proofs of Theorem 4 are provided in Appendix D.

Therefore, the layer-selection criterion of client i (i.e., Definition 1) can be quantified as

$$\begin{aligned}&\frac{2\sqrt{2}d_{\Omega_i}\rho A_i\cdot 2^{\Omega_i}\cdot\prod_{j=1}^{L}\hat{B}_i^j\cdot\hat{M}(\Omega_i)}{\Omega_i\cdot\sqrt{m_i}}\\&\leq\frac{2\sqrt{2}c_i\rho A_i\cdot 2^L\cdot\prod_{j=1}^{L}B_i^j\cdot M}{L\cdot\sqrt{m_i}}+\Delta\left(\frac{1}{L}-\frac{1}{\Omega_i}\right),\end{aligned}\tag{27}$$

where $\hat{M}(\Omega_i)=\frac{\hat{\tau}_i^{\Omega_i}}{2\hat{B}_i^{\Omega_i}}+\sum_{k=1}^{\Omega_i-1}\frac{\hat{\tau}_i^{\Omega_i-k}\prod_{j=1}^{k}\sqrt{n^{\Omega_i+1-j}}}{2\hat{B}_i^{\Omega_i-k}}$, $M=\frac{\tau_i^L}{2B_i^L}+\sum_{k=1}^{L-1}\frac{{\tau_i}^{L-k}\prod_{j=1}^{k}\sqrt{n^{L+1-j}}}{2{B_i}^{L-k}}$ and $\Delta=3(b-a)\sqrt{\frac{log\frac{2}{\delta}}{2m_i}}$.

Then inequality (27) is rearranged to obtain

$$\frac{Ld_{\Omega_i}\cdot\hat{M}(\Omega_i)}{\Omega_i\cdot c_i\cdot 2^{L-\Omega_i}}\cdot\prod_{j=1}^{L}\frac{\hat{B}_i^j}{B_i^j}+\frac{\Delta\sqrt{m_i}(L-\Omega_i)}{2\sqrt{2}\cdot\rho A\cdot 2^L\cdot\Omega_i c_i\prod_{j=1}^{L}B_i^j}\leq M.\tag{28}$$

Since the second term in left of inequality (28) is almost zero due to exponent and continuous multiplication terms, we neglect it to obtain the final quantified criterion

$$\frac{Ld_{\Omega_i}\cdot\hat{M}(\Omega_i)}{\Omega_i\cdot c_i\cdot 2^{L-\Omega_i}}\cdot\prod_{j=1}^{L}\frac{\hat{B}_i^j}{B_i^j}\leq M.\tag{29}$$

5.2.3 Layer Selection Method with Generalization

In the criterion shown as inequality (29), the bounds of full-layered regularizer and trained parameters are required (i.e., $\{\tau_i^l\}_{l=1}^L$ and $\{B_i^l\}_{l=1}^L$). However, as the proposed method designed, only the selected layers are able to acquire the regularizer $\hat{\mathbf{s}}_i^l$ and further their corresponding bound $\hat{\tau}_i^l$. Therefore, a straightforward approach is proposed to estimate both bounds with the staled information.

When client i is selected for the first time, the full-layered regularizer is distributed to client i from the server, which acts as a regularizer to train the local model using Algorithm 2. After training, client i calculates the Frobenius norm of the l-th layered parameters and the l-th layered regularizer to estimate B_i^l and τ_i^l, which can be calculated as

$$B_i^l=\|\mathbf{w}_i^l\|, l\in[1,L],\tag{30}$$

$$\tau_i^l=\|\mathbf{w}_i^l-\mathbf{s}_i^l\|, l\in[1,L].\tag{31}$$

In addition, when $\Omega_i=L$, the regularizer is also full-layered, allowing us to estimate $\{B_i^l\}_{l=1}^L$ and $\{\tau_i^l\}_{l=1}^L$ with the maximum of original and current values. Similarly, $\hat{B}_i^l$ can also estimated with Eq. (30) when $\hat{\mathbf{w}}_i^l$ is inputted, and $\hat{\tau}_i^l$ can also estimated with Eq. (31) when partial-layered regularizer is inputted.

Once $\{\tau_i^l\}_{l=1}^L$ and $\{B_i^l\}_{l=1}^L$ have been obtained, the right term in inequality (29) can be determined. Then a heuristic-based method is proposed to search for the optimal Ω_i. Furthermore, in the k-th step, the neighboring layer to Ω_i^{k-1} can be searched firstly to reduce the computation costs.

In this end, the layer selection method of client i is summarized in Algorithm 4. Firstly, the class number c_i, the data size m_i and the column number of $\mathbf{w}_i^l$, n^l are

Algorithm 4 The procedures of layer selection with generalization in client i at k-th iteration

Input: the number of update iterations for client E, client parameters at the k-step $\hat{\mathbf{w}}_i^k$, current communication round k, the selected time of client i, q

Output: the max index of uploaded layers Ω_i^k at k-th iteration

1: determinate c_i, m_i, $n^l, l{\in}[1,L]$, according to dataset D_i and model $h(\cdot,\mathbf{w}_i)$
2: initialize $B_i^l=0$ and $\tau_i^l=0$ for $l{\in}[1,L]$
3: **if** $q_i=0$ or $\Omega_i^{k-1}=L$ **then**
4: **for** $l=1$ to L **do**
5: $B_i^{l,k} \leftarrow Eq.\ (30)$, $\tau_i^{l,k} \leftarrow Eq.\ (31)$
6: $B_i^l \leftarrow \max\{B_i^{l,k}, B_i^l\}$, $\tau_i^l \leftarrow \max\{\tau_i^{l,k}, \tau_i^l\}$
7: **end for**
8: $\Omega_i^k \leftarrow L-1$
9: **return** Ω_i^k
10: **else**
11: **for** $l=1$ to L **do**
12: $\hat{B}_i^{l,k} \leftarrow Eq.\ (30)$, $\hat{\tau}_i^{l,k} \leftarrow Eq.\ (31)$
13: $B_i^{l,k} \leftarrow B_i^l$, $\tau_i^{l,k} \leftarrow \tau_i^l$
14: **end for**
15: $\Omega_i \leftarrow \Omega_i^{k-1}$
16: **if** inequality (29) is satisfied **then**
17: **repeat**
18: $\Omega_i \leftarrow \Omega_i - 1$
19: **until** inequality (29) is not satisfied
20: **else**
21: **repeat**
22: $\Omega_i \leftarrow \Omega_i + 1$
23: **until** inequality (29) is satisfied
24: **end if**
25: **end if**
26: $\Omega_i^k \leftarrow \Omega_i$
27: **return** Ω_i^k

determined for client i based on the dataset and model (line 1). Then, the bounds of parameters and full-layered regularizer of all layers are initialized as 0 (line 2). Next, if client i is first selected (i.e., $q_i=0$) or the number of uploaded layers in the last round is L, which indicates that the regularizer is full-layered, the values for full-layered parameters and regularizer (i.e., $B_i^{l,k}$ and $\tau_i^{l,k}$) at current round can be calculated according to Eq. (30) and Eq. (31), respectively (lines 3-5). Then, these bounds are estimated by the maximum of original and current values (lines 6-7). In this case, the number of uploaded layers can be reset to $L-1$ and returned (lines 8-9). Otherwise, if the regularizer is partial-layered, $\hat{B}_i^{l,k}$ and $\hat{\tau}_i^{l,k}$ can be estimated, while the previous values can be used for $B_i^{l,k}$ and $\tau_i^{l,k}$ (lines 10-14). To save the computation cost, we search for the optimal layer number starting from the last one (line 15). If the criterion for layer selection is satisfied with the current layer, the layers will be decreased until the condition is no longer met. Otherwise, the layers will be increased until the condition is satisfied (lines 16-25). Finally, the number of selected layers is returned (lines 26-27).

6 Communication-efficient Parameter Coaching Method

In this section, the communication-efficient parameter coaching method with the combination of KAPC in Section 4 and BLS in Section 5 is introduced.

However, the layer selection method used by clients prevents the server from receiving the parameters of the same layers from all clients simultaneously. As a result, updating $\hat{\mathbf{s}}$ with Eq. (15) is impossible due to the absence of the latest layered parameters. However, the relationship cube has the ability to adaptively learn the similarity of clients on the corresponding layer, which can be beneficial even with outdated parameters. Therefore, we fill in the missing layered parameters with stale values to update $\hat{\mathbf{r}}_i$ and $\hat{\mathbf{s}}_i$, which are detailed as follows. The update process of $\hat{\mathbf{r}}_i$ remains the same as Eq. (12) but with the combined $\hat{\mathbf{W}}$ as defined in Eq. (33).

$$\hat{\mathbf{r}}_i^{k+1} = \hat{\mathbf{r}}_i^k - \eta_r\left(\lambda\nabla_{\mathbf{r}_i} f_s(\mathbf{r}_i\hat{\mathbf{W}}^{k+1}, \hat{\mathbf{w}}_i^{k+1}) + \beta\Big(\hat{\mathbf{r}}_i - \frac{\mathbf{1}}{N}\Big)\right), \tag{32}$$

where

$$\begin{aligned}\hat{\mathbf{W}}^{k+1} &= \left\{\hat{\mathbf{w}}_i^{k+1}\right\}_{i=1}^N \\ &= \begin{cases}\hat{\mathbf{w}}_i^{l,k+1}, & \text{if } l{\leq}\Omega_i^{k+1},\\ \hat{\mathbf{w}}_i^r, r=\max\limits_t\{\hat{\mathbf{w}}_i^{l,t}\}, & \text{else.}\end{cases}\end{aligned} \tag{33}$$

Correspondingly, the partial-layered regularizer of client i (i.e., $\hat{\mathbf{s}}_i$) can be updated as

$$\hat{\mathbf{s}}_i^k = \hat{\mathbf{r}}_i^k\hat{\mathbf{W}}^k. \tag{34}$$

The KAPC with layer selection in the server is summarized in Algorithm 5. Different from Algorithm 1, except for initializing the relationship cube and client models, the selected time for each client is set to 0 (lines 1-3). Next, the clients are randomly selected to participant federated learning, and the corresponding relationship cube $\mathbf{R}'$ is sliced based on participants (lines 5-6). Then, the parameters of client models are stacked using Eq. (33), and the relationship cube $\mathbf{R}'$ is updated with the original method (lines 7-11). After that, the relationship in $\hat{\mathbf{R}}$ among participants is updated to the full relationship cube $\mathbf{R}$ (line 12). The partial-layered regularizer is updated with Eq. (34), and the layer selection method is executed based on Algorithm 3 (lines 13-15). Then, the partial-layered regularizer with selected layers is distributed to the corresponding client, and the updated model with selected layers is sent back from the local client based on Algorithm 6 (lines 16-18). Finally, the personalized client models are returned (lines 19-21).

Furthermore, the training procedures of KAPC with layer selection in client i is described in Algorithm 6. The training process follows the same steps as Algorithm 2 (lines 1-5), with the distinction that the layer selection method is performed after the model training and then the selected time is counted (lines 6-7). Finally, the model with partial-layered parameters is returned to the server (line 8).

7 Experiment

In this section, the experiment settings are introduced including the datasets, model structure, implementation de-

Algorithm 5 The procedures of KAPC with layer selection in the server

Input: the number of communication rounds K, the number of clients N, selected ratio β, the number of update iterations for relationship cube E_r, the number of update iterations for all the client E, threshold to filter the self-relationship μ

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

initialize $\mathbf{R}^1$
initialize the client model $\mathbf{w}_i^1$ for $i \in [0, N]$
initialize the selected time $q_i = 0$ for $i \in [0, N]$
for $k = 1$ to K **do**
  randomly select a set $\mathcal{C}^k$ to participant federated training, where $|\mathcal{C}^k| = N * \beta$
  slice $\mathbf{R}^k$ according to $\mathcal{C}^k$ to obtain $\mathbf{R}'^k$, and normalize $\mathbf{R}'^k$ by row
  $\hat{\mathbf{W}}^k \leftarrow$ Eq. (33)
  for $t = 1$ to E_r **do**
    $\hat{\mathbf{R}}^{k,t} \leftarrow$ Eq. (32) with $\mathbf{R}'^{k,t}$
  end for
  $\hat{\mathbf{R}}^k \leftarrow \hat{\mathbf{R}}^{k,E_r}$
  update the corresponding relationship in $\mathbf{R}^k$ with $\hat{\mathbf{R}}^k$, and normalize $\mathbf{R}^k$ by row
  for client $i \in \mathcal{C}^k$ in parallel **do**
    $\hat{\mathbf{s}}_i^k \leftarrow$ Eq. (34)
    $\hat{\mathbf{s}}_i^{k+\frac{1}{2}} \leftarrow$ Algorithm 3(β, k, $\hat{\mathbf{s}}_i^k$, $\hat{r}_i^k$)
    distribute the parameters $\hat{\mathbf{s}}_i^{k+\frac{1}{2}}$ to client i
    $\hat{\mathbf{w}}_i^{k+1} \leftarrow$ Algorithm 6(E, $\hat{\mathbf{s}}_i^{k+\frac{1}{2}}$, q_i)
  end for
end for
$\hat{\mathbf{W}}^{k+1} = \{\hat{\mathbf{w}}_1^{k+1}, \hat{\mathbf{w}}_2^{k+1}, \cdots, \hat{\mathbf{w}}_N^{k+1}\}$
return $\hat{\mathbf{W}}^K$

Algorithm 6 The procedures of KAPC with layer selection in client i

Input: the number of update iterations for client parameters E, the corresponding regularizer parameters at the k-step $\hat{\mathbf{s}}_i^k$, the selected time q

Output: the partial-layered client parameters $\hat{\mathbf{w}}_i^l, l \leq \Omega_i$

for $t = 1$ to E **do**
  for each batch in dataset i **do**
    $\hat{\mathbf{w}}_i^{k+1} \leftarrow$ Eq. (14)
  end for
end for
$\Omega_i^{k+1} \leftarrow$ Algorithm 4(E, $\hat{\mathbf{w}}_i$, k, q)
$q \leftarrow q + 1$
return $\hat{\mathbf{w}}_i^l, l \leq \Omega_i^{k+1}$

tails of the proposed method, and the baselines to be compared. Then, the performance overview of the proposed methods on the four datasets with various levels of heterogeneity is presented, which includes the comparisons with the state-of-the-art personalized algorithms in accuracy and convergence speed. Next, the impact of critical hyperparameters on the proposed methods is evaluated. Finally, the proposed KAPC and BLS are implemented in the simulated federated learning system and further the communication and computation costs are evaluated.

7.1 Experiment Setting

Dataset. Four public benchmark datasets are used to evaluate the proposed method, MNIST, FashionMNIST (FMNIST), CIFAR10 and CIFAR100. To simulate the heterogeneous distribution of all the clients, the latent Dirichlet distribution method is adopted [38], and the heterogeneity levels are set to $\alpha = \{0.1, 0.3\}$, where a smaller value indicates more significant heterogeneity. In addition, two scenarios with and without client selection are considered. 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 from the dataset are randomly selected.

Model Architecture. For MNIST and FMNIST, a CNN model is used, which consists 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 [39] 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 $E = 5$ and $E_r = 1$ for all cases. In addition, cross-entropy loss and stochastic gradient descent method are adopted to update the client parameters and relationship cube [40], and the learning rates for $\mathbf{W}$ and $\mathbf{R}$ are both set to 0.01.

Baselines. The proposed method are compared with various personalized methods as follows.

- **Local training** is conducted for each client on its training dataset. For MNIST and FMNIST, the training epoch is set to 5, while 10 epochs for CIFAR10 and CIFAR100. Then the model is tested on the local dataset and then the best test accuracy of all the clients are averaged as the baseline.
- **FedAvg** [3] aggregates trained models of all the selected clients to obtain a global model. The global model is tested on the local dataset of each client, and the averaged best accuracy of all the clients is adopted as the baseline.
- **FedProx** [41] introduces proximal term to regularize the distance between a local model and the global model.
- **FedAMP** [42] weights personalized server models with cosine similarity to guide the local training.
- **MOON** [40] adopts contrastive learning to make local and global representations close.
- **FedBABU** [27] shares the global extractor and trains the personalized classifier for each client.
- **FedBN** [25] keeps the batch normalization layer of each client personalized and aggregates other layers by weighted averaging to enhance the personalized model.
- **pFedLA** [9] learns the personalized model with layer-wise aggregation weights, where a hypernetwork for each client is trained to obtain the weights.

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	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 [3]	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 [41]	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 [42]	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 [40]	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)
FedBABU [27]	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 [25]	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 [9]	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)
KAPC	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)
KAPC+BLS	**98.00**(0.03)	96.83(0.18)	96.57(0.08)	**88.78**(0.24)	87.54(0.56)	65.20(0.38)	**38.50**(0.18)	26.24(0.37)

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.

	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 [3]	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 [41]	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 [42]	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 [40]	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)
FedBABU [27]	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 [25]	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 [9]	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)
KAPC	**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)
KAPC+BLS	97.05(0.80)	**96.54**(0.49)	**95.07**(0.01)	90.23(0.53)	79.96(0.64)	65.50(0.21)	**46.17**(0.36)	27.63(0.94)

In addition, for all the compared algorithms, the recommended hyperparameters presented in their papers are adopted, but the number of communication is the same as that in the implementation details for all the compared algorithms.

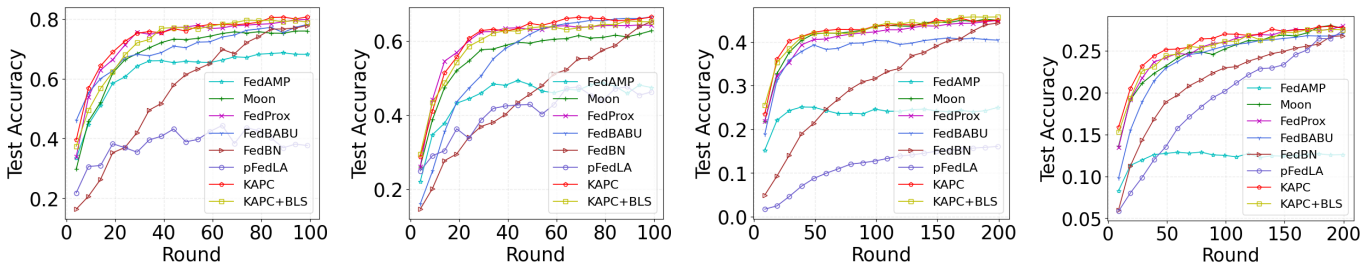
7.2 Performance Overview

Accuracy Comparison. The accuracy for each round is tested, and then the average accuracy of all the selected clients is calculated. The best average accuracy from all the training rounds (i.e., best mean accuracy) is selected as the criterion to reflect the performance of each baseline and the proposed methods. The results under the scenarios without and with client selection are shown in Table 1 and Table 2, respectively.

Most personalized methods outperform local training, indicating the benefit of parameter sharing among clients. Additionally, the performance of global model trained with FedAvg is evaluated on each client and the results show that the accuracy declines with increasing client heterogeneity. This suggests that FedAvg is not adaptive to heterogeneous clients. Furthermore, FedAvg exhibits significantly poorer performance compared to the majority of personalized methods, specifically on realistic datasets like CIFAR10 and CIFAR100. The degraded performance indicates that a single global model without any personalized approaches fails to adapt to diverse client data, particularly when the data exhibit complexity and heterogeneity.

Despite the strong performance of all personalized methods on heterogeneous datasets, the proposed KAPC achieves the highest accuracy. In comparison to regularization methods such as FedProx, FedAMP, and MOON, KAPC captures the layer-wise similarities among diverse clients via the relationship cube, which enhances the personalized regularizer and provides fine-grained guidance for local training. Additionally, a lightweight optimization method have been designed, which can efficiently update both client parameters and the relationship cube. Consequently, other layer-wise methods such as FedBABU, FedBN, and pFedLA converge slower and yield inferior performance in a few communication rounds. Particularly, the results of pFedLA demonstrate significant underperformance across all datasets, indicating that the hypernetwork utilized by pFedLA for computing the relationship matrix is challenging to train with fast convergence speed. Furthermore, KAPC with the BLS achieves comparable or even better accuracy than the single KAPC, indicating that the proposed BLS does not degrade the performance with less client information transmissions.

Convergence Comparison. Theorems 1 and 2 provide the convergence analysis of KAPC under both the convex and non-convex settings. Specifically, Theorem 1 establishes the upper bound and convergence speed under the con-



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

Fig. 5: Comparison of convergence speed among the proposed method and personalized baselines on CIFAR10 and CIFAR100 in the scenario with client selection.

vex setting, whereas Theorem 2 offers the convergence guarantee under the non-convex setting without explicitly presenting the convergence speed. Consequently, we conduct experiments to evaluate the convergence speed of the proposed KAPC with a non-convex DNN model.

The performance of KAPC is evaluated on CIFAR10 and CIFAR100 datasets with $\alpha = \{0.1, 0.3\}$, and a ResNet18 model is used. The client selection scenario is adopted, which follows the same settings as those described in Section 7.1. KAPC and KAPC with BLS are compared against all the personalized baselines mentioned above in accuracy-round space, and the results are shown in Figure 5.

As illustrated in Figure 5, the convergence speed of KAPC is comparable to that of partial regularization methods such as FedProx and MOON. However, it outperforms the layer-wise approaches (e.g., FedBABU, FedBN, and pFedLA) by a significant margin, which indicates that KAPC efficiently captures the layer-wise similarities among clients. Moreover, KAPC achieves superior test accuracy compared to all other methods, providing further evidence of its effectiveness. Furthermore, the convergence speed of KAPC with BLS is similar with that of the single KAPC, which implies the partial-layered regularizer derived by BLS does not slow the convergence speed.

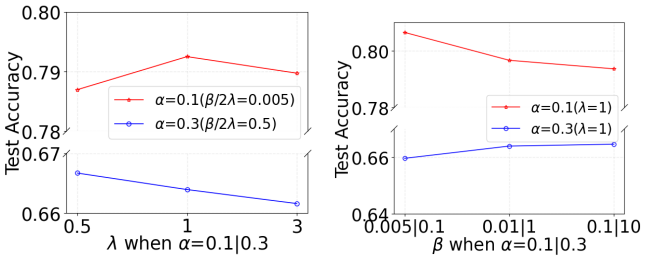
7.3 Ablation Studies

In this subsection, the effects of hyperparameters are evaluated, including regularization parameters λ, β, and the threshold of self-relationship μ.

7.3.1 Regularization Hyperparameters

As shown in Eq. (7), λ is used to balance the local general loss and the personalized regularizer when updating client parameters, whereas β can influence the regularization term for $\mathbf{R}$. Referring to Eq. (16), a suitably larger λ amplifies the influence of the personalized regularizer, facilitating the acquisition of beneficial knowledge from other clients and consequently enhancing the generalization capabilities of the local model. Nevertheless, as λ increases, it has the potential to weaken the constraint of equal sharing presented in Eq. (11), leading to insufficient learning of $\mathbf{R}$.

Considering the intricate influences of the two hyperparameters, a comprehensive evaluation on KAPC is performed by exploring various combinations of (λ, β). The



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

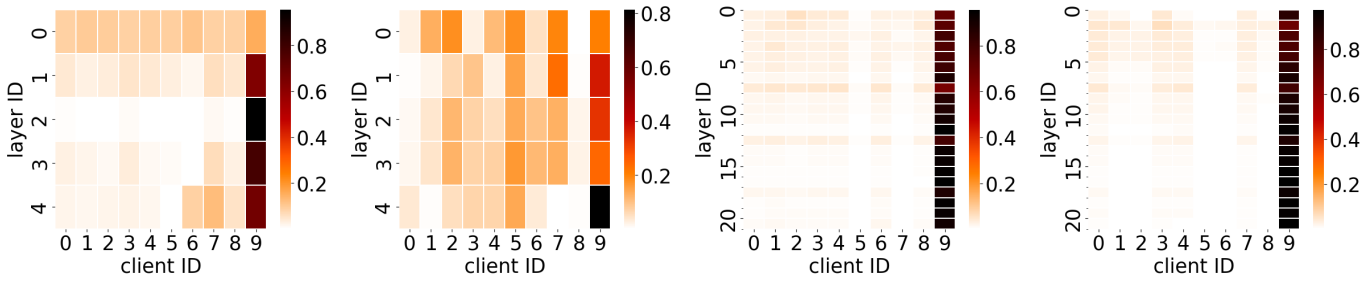
Fig. 6: The best mean accuracy with various hyperparameters. (a) The accuracy of the proposed method varying $\lambda = \{0.5, 1, 3\}$ with fixed $\frac{\beta}{2\lambda} = 0.005$ when $\alpha = 0.1$ and varying $\lambda = \{0.05, 1, 3\}$ with fixed $\frac{\beta}{2\lambda} = 0.5$ when $\alpha = 0.3$. (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.

test accuracy of KAPC is measured on CIFAR10 with $\alpha = \{0.1, 0.3\}$ in the scenario without client selection.

To study the individual impacts of λ and β, we set $\frac{\beta}{2\lambda} = 0.005$ rather than fixing β alone, to measure the test accuracy with different λ values varying from the set $\{0.5, 1, 3\}$. Likewise, the influence of β is evaluated by varying its value among $\{0.005, 0.01, 0.1\}$ while keeping λ fixed at 1.

Analysis of λ. As illustrated in Figure 6 (a), the impacts of λ on the test accuracy of KAPC are analyzed. Since the sharable knowledge is less with high heterogeneity level, $\frac{\beta}{2\lambda}$ is fixed with a small value (e.g., 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 (e.g., 0.5) that may transfer conflicted knowledge from other clients, so λ tends to a small value 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 6 (b). The accuracy with small β outperforms that with large β for $\alpha = 0.1$ while the tendency



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

Fig. 7: The relationship matrix of client 9 on FMNIST and CIFAR100 in the scenario without client selection.

is contrary for $\alpha = 0.3$, which implies that there is less sharable knowledge among higher-heterogeneous clients.

7.3.2 Self-Relationship Hyperparameters

As shown in Algorithm 3, μ is used to filter the layers with a high self-relationship to avoid invalid downstream. However, a smaller μ can save more communication costs but less knowledge sharing with other clients, which may lead to degraded accuracy. A large μ cannot reduce the communication costs efficiently.

To investigate the effect of μ, several experiments are conducted by varying $\mu = \{0.3, 0.5, 0.7, 0.9\}$ on CIFAR10 with $\alpha = 0.1$ in client selection scenario, and the results are shown in Table 3.

TABLE 3: The accuracy and communication cost of KAPC with BLS in different μ. U-Comm and D-Comm denote the uploaded and downloaded communication cost, respectively

μ	Accuracy(%)	Comm(MB)	U-Comm(MB)	D-Comm(MB)
0.3	77.79	550.93	387.29	163.64
0.5	80.01	631.80	380.70	251.10
0.7	80.02	708.36	380.88	327.48
0.9	79.70	799.52	373.94	425.58

Analysis of μ. When μ is a small value (e.g., $\mu = 0.3$), the communication cost is significantly reduced, particularly the downloaded cost. However, the accuracy suffers a significant drop because a smaller μ makes it much easier not to distribute the aggregated parameters, resulting in invalid local training without any knowledge sharing from other clients. Additionally, when μ has a larger value (e.g., $\mu = 0.9$), the communication cost cannot be reduced as the condition to avoid being downloaded from the server becomes challenging to achieve, and thus, almost all parameters are sent to clients. Meanwhile, the accuracy remains almost the same with large μ, suggesting that knowledge sharing with other clients has reached saturation, and there is no additional knowledge to improve local training.

7.4 Vision of Relationship Cube

In order to verify the effectiveness of the relationship cube, experiments are performed to capture the inter-layer correlations among clients. Figure 7 presents the relationship matrix of client 9 on FMNIST and CIFAR100 without any client selection. The figure displays the client ID on the horizontal axis and the model layer ID on the vertical axis. The color shading represents the level of similarity between corresponding layers across clients.

As shown in Figure 7, clients tend to commonly share the shallow layers of DNN models, such as layer 0 of the 2CNN model for FMNIST and layers $0 - 7$ of the ResNet18 model for CIFAR100. Conversely, the deeper layers exhibit a more personalized pattern. Furthermore, clients with the same heterogeneity (i.e., the same α) exhibit distinctive inter-layer correlations, providing further evidence to support the necessary for layer-wise parameter sharing, which constitutes the core principle of the proposed KAPC.

7.5 System Performance

In this subsection, the server and clients in federated learning are simulated with a multiprocessing approach. CIFAR10 with $\alpha = 0.1$ under the client selection is adopted, which keeps the settings consistent with those described in subsection 7.1. To evaluate communication time in real-world scenarios, we simulate the wireless communication environment with the model in [43], and the communication parameters are detailed in Table 4.

TABLE 4: The communication parameter configuration

Bandwidth of client i	randomly sampled from $[10, 30]MHz$
Transmission power of client i	randomly sampled from $[23, 30]dBm$
Path loss constant	-40dB
Small-scale fading channel power gain	1
reference distance	1m
Large-scale path loss factor	4
Distance between client i and the server	randomly sampled from $[100, 500]$m
Noise power spectral density	-174dBm/Hz
Uplink co-channel interference	sampled from Gaussian distribution $N(1, 4)$
Bandwidth of the server	50MHz
Transmission power of the server	30dBm
Downlink co-channel interference	sampled from Gaussian distribution $N(1, 9)$

In order to present the performance of the proposed KAPC and BLS in the system, the efficiency of communication, computation and memory is considered. Communication efficiency is influenced by many factors, such as the number of communication rounds, the size of the communication stream, and the quality of the communication channel. Therefore, the average execution time per round and the size of the communication stream are evaluated. Furthermore, the number of floating point operations (FLOPs) is evaluated to reflect the computation efficiency. Finally, the memory usage in the server is used to represent the memory efficiency.

TABLE 5: Average execution time per round among the proposed methods and baselines on CIFAR10 with $\alpha = 0.1$ in the client selection scenario, where U-comm, D-comm, Client and Server denote the time of uploading model, downloading model, local training and model aggregating, respectively. The number in parentheses is the percentage.

methods	Time(s)	U-Comm(s/%)	D-Comm(s/%)	Client(s/%)	Server(s/%)
FedAvg	32.70	19.28(58.96)	7.16(21.89)	5.81(17.77)	0.45(1.38)
FedProx	37.39	19.34(51.73)	7.16(19.15)	10.37(27.73)	0.52(1.39)
FedAMP	46.51	19.31(41.52)	7.16(15.39)	8.89(19.12)	11.15(23.97)
MOON	39.63	19.21(48.47)	7.16(18.07)	12.87(32.48)	0.39(0.98)
FedBABU	36.04	19.27(53.47)	7.15(19.84)	9.20(25.53)	0.42(1.16)
FedBN	34.37	19.26(56.04)	7.15(20.80)	7.46(21.71)	0.50(1.45)
pFedLA	40.96	19.27(47.05)	7.16(17.48)	10.90(26.61)	3.63(8.86)
KAPC	37.94	19.29(50.84)	7.15(18.85)	8.65(22.80)	2.85(7.51)
KAPC+BLS	36.04	15.93(44.20)	5.49(15.23)	13.18(36.57)	1.44(4.00)

TABLE 6: Communication cost and computation cost per round of the proposed methods and baselines on CIFAR10 with $\alpha = 0.1$ in the client selection scenario. U-Comm and D-Comm denote the upload and download communication cost, respectively. C-Comp and S-Comp denote computation cost of local training on the clients and model aggregation on the server, respectively.

methods	U-Comm(MB)	D-Comm(MB)	C-Comp(GFLOPs)	S-Comp(GFLOPs)
FedAvg [3]	426.54	426.54	424.13	0.22
FedProx [41]	426.54(→)	426.54(→)	524.77	0.22
FedAMP [42]	426.54(→)	426.54(→)	524.77	5.59
MOON [40]	426.54(→)	426.54(→)	848.25	0.22
FedBABU [27]	426.35(↓0.19)	426.35(↓0.19)	424.12	0.22
FedBN [25]	426.17(↓0.37)	426.17(↓0.37)	424.13	0.22
pFedLA [9]	426.54(→)	426.54(→)	424.13	1.12
KAPC	426.54(→)	426.17(↓0.37)	524.77	0.56
KAPC+BLS	380.88(↓45.66)	327.48(↓99.06)	524.80	0.56

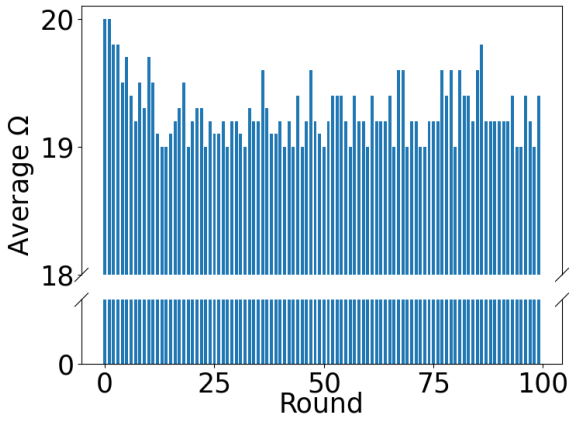


Fig. 8: The average selected layers to be uploaded (i.e., Ω_i) per round on CIFAR10 with $\alpha = 0.1$ in client selection scenario.

Execution Time. The average execution time of the proposed methods and baselines is shown in Table 5. FedAvg takes the least time since it does not involve any additional operations for personalization. FedAMP has the longest execution time due to its complex aggregation rules, as indicated by the significant time consumed for aggregation in the server compared with other methods. pFedLA has longer execution time compared to other methods except for FedAMP because it trains the hypernetwork to obtain the aggregation weights, consuming more server time. MOON exhibits the third long execution time due to the additional computation required by a client to obtain representations of both the global model and the last local model.

The proposed KAPC has a comparable execution time compared to other regularization-based methods, which is longer than FedProx but less than FedAMP and MOON. Although it requires more time to train the relationship cube and aggregate the personalized model in the server, the regularizer based on the parameters is easy to train, which results in a balanced execution time. On the other hand, KAPC requires less time for aggregation compared to pFedLA due to the design of the relationship cube and the optimization method. Furthermore, KAPC with BLS takes more time in the client as it needs to select the uploaded layers. However, it results in less time for uploading and downloading model due to transmitting partial layers, which is more significant in the real-world scenarios with limited communication resources, such as the Internet of Things and the Internet of Vehicles.

Communication Cost. As shown in Table 6, the communication costs of all selected clients per round are evaluated, including the upstream and downstream. The majority of the compared methods transmit all model parameters (e.g., FedProx, FedAMP, MOON, pFedLA and the proposed KAPC), resulting in the same communication costs as FedAvg. Additionally, FedBABU only shares the feature extractor, which includes the model except for the last full connected layer. Consequently, its communication cost is slightly lower than that of FedAvg. FedBN keeps the Batch Norm layer personalized, resulting in the decrease of the corresponding communication cost.

The proposed KAPC with BLS demonstrates a significant reduction of 23.22% in download communication cost (approximately 99.06MB) with $\mu = 0.7$ due to the layer selection methods based on self-relationship in the server. Moreover, by utilizing a smaller value of μ, the download communication costs can be further decreased. For detailed

results, Table 3 can be referred. Additionally, KAPC with BLS also achieves a remarkable 10.70% decrease of upload communication cost (approximately 45.66MB), which can be attributed to the adoption of the proposed layer selection method based on generalization bound in the client. To further demonstrate the effectiveness of layer selection method in the client, the average value of Ω_i per round is recorded, which is presented in Figure 8. The results reveal that during the majority of communication rounds, the parameters of the first 19 layers are transmitted to the server for personalized aggregation, while the remaining two layers (i.e., a convolution layer and a fully connected layer) are deemed unnecessary to be uploaded. Additionally, the average number of uploaded layers varies as the training progresses, suggesting that the optimal aggregation part of DNN model may not be invariant.

Computation Cost. The computation costs of local training and server aggregating are evaluated with the FLOPs counting tool, pytorch-OpCounter, and the results are presented in Table 6. The regularization-based methods require higher computation costs for local training compared to FedAvg, while other methods except FedBABU keep similar costs as FedAvg. Because FedBABU only updates the extractor in the training phase, it incurs slightly lower computation costs than others. Additionally, FedAMP and pFedLA require complex operations for aggregation, resulting in higher computation costs in the server compared with other methods.

The proposed KAPC has the same computation cost for local training as other regularization-based methods (e.g., FedProx and FedAMP), and the proposed BLS in the client burdens less training cost. However, KAPC requires additional aggregation operations, which increases the server computation costs. This can be supported by powerful server and is justified by the improvement in accuracy.

Memory Cost. Since KAPC maintains a server model for each client, the experiments are conducted to demonstrate the memory cost of KAPC in the server. Since our experiments are simulated with multiprocessing approach, we use *psutil* tool in Python to collect the memory usage of the server process. For comparison, FedAvg is tested in the same settings as a baseline.

The memory usage of KAPC in the server is 1101.62MB, compared to 753.02MB for FedAvg, indicating an increase of 348.6MB. This increase is attributed to the additional consumption of personalized server models and the relationship cube, which is manageable for powerful servers. Additionally, in case the server's memory is in shortage, the update process for the relationship cube in KAPC can be conducted in a layer-wise manner. Specifically, the l-th layered parameters of the selected clients and the l-th layered relationship vector are loaded to the memory sequentially to further reduce the memory cost.

8 Conclusion

In the paper, we have proposed a communication-efficient pFL architecture with knowledge-aware parameter coaching, which leverages granular knowledge among clients to guide local training efficiently without inducing excessive communication overhead. A relationship cube has been defined to represent the similarity of heterogeneous knowledge in DNN layers among clients, enabling the fine-grained aggregation of personalized regularizers to share mutually beneficial knowledge. Additionally, an efficient update method has been designed to learn the client parameters and relationship cube alternately, accompanied by theoretical analysis for the proposed method under the convex and non-convex settings. To reduce communication costs, a bidirectional layer selection method based on the generalization theory and self-relationship has been proposed, offering performance guarantee for efficient communication. Finally, the proposed methods have been evaluated on popular datasets with various levels of heterogeneity, demonstrating superior performance in accuracy and convergence speed compared to the state-of-the-art pFL methods, and the bidirectional layer selection method can reduce at least 10% and 23% uploaded and downloaded communication costs without sacrificing performance.

References

[1] W. Wu, L. He, W. Lin, and R. Mao, "Accelerating Federated Learning over Reliability-Agnostic Clients in Mobile Edge Computing Systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1539–1551, 2021.

[2] Y. Wu, Y. Song, T. Wang, L. Qian, and T. Q. S. Quek, "Non-orthogonal multiple access assisted federated learning via wireless power transfer: A cost-efficient approach," *IEEE Transactions on Communications*, vol. 70, no. 4, pp. 2853–2869, 2022.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[4] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, 2020.

[5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[6] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 9587–9603, 2022.

[7] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2014, pp. 3320–3328.

[8] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 818–833.

[9] X. Ma, J. Zhang, S. Guo, and W. Xu, "Layer-wised model aggregation for personalized federated learning," in *Proceddings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 10 082–10 091.

[10] M. Zhi, Y. Bi, W. Xu, H. Wang, and T. Xiang, "Knowledge-aware parameter coaching for personalized federated learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[11] J. He, S. Guo, M. Li, and Y. Zhu, "Acefl: Federated learning accelerating in 6g-enabled mobile edge computing networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1364–1375, 2023.

[12] A. A. Abdellatif, M. S. Allahham, N. Khial, A. Mohamed, A. Erbad, and K. Shaban, "Reliable federated learning for age sensitive mobile edge computing systems," in *Proceddings of the IEEE International Conference on Communications (ICC)*, 2023, pp. 1622–1627.

[13] Y. Li, W. Liang, J. Li, X. Cheng, D. Yu, A. Y. Zomaya, and S. Guo, "Energy-aware, device-to-device assisted federated learning in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 7, pp. 2138–2154, 2023.

[14] M. Duan, D. Liu, X. Ji, Y. Wu, L. Liang, X. Chen, Y. Tan, and A. Ren, "Flexible clustered federated learning for client-level data distribution shift," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2661–2674, 2022.

[15] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2021, pp. 5972–5984.

[16] Z. Tang, Y. Zhang, S. Shi, X. He, B. Han, and X. Chu, "Virtual homogeneity learning: Defending against data heterogeneity in federated learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2022, pp. 21 111–21 132.

[17] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[18] C. T. Dinh, N. H. Tran, and T. D. Nguyen, "Personalized federated learning with moreau envelopes," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2020, pp. 21 394–21 405.

[19] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020, pp. 5132–5143.

[20] M. Zhang, K. Sapra, S. Fidler, S. Yeung, and J. M. Alvarez, "Personalized federated learning with first order model optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[21] Y. Tan, G. Long, L. Liu, T. Zhou, Q. Lu, J. Jiang, and C. Zhang, "Fedproto: Federated prototype learning across heterogeneous clients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, pp. 8432–8440.

[22] J. Xu, X. Tong, and S.-L. Huang, "Personalized federated learning with feature alignment and classifier collaboration," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023.

[23] H. Jin, D. Bai, D. Yao, Y. Dai, L. Gu, C. Yu, and L. Sun, "Personalized edge intelligence via federated self-knowledge distillation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 567–580, 2023.

[24] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, "Local learning matters: Rethinking data heterogeneity in federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8387–8396.

[25] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "Fedbn: Federated learning on non-iid features via local batch normalization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[26] J. Mills, J. Hu, and G. Min, "Multi-task federated learning for personalised deep neural networks in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 630–641, 2022.

[27] J. Oh, S. Kim, and S.-Y. Yun, "Fedbabu: Towards enhanced representation for federated image classification," *arXiv preprint arXiv:2106.06042*, 2021.

[28] A. Li, J. Sun, P. Li, Y. Pu, H. Li, and Y. Chen, "Hermes: An efficient federated learning framework for heterogeneous mobile clients," in *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2021, p. 420–437.

[29] X. Cao, T. Başar, S. Diggavi, Y. C. Eldar, K. B. Letaief, H. V. Poor, and J. Zhang, "Communication-efficient distributed learning: An overview," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 4, pp. 851–873, 2023.

[30] F. Sattler, A. Marban, R. Rischke, and W. Samek, "Cfd: Communication-efficient federated distillation via soft-label quantization and delta coding," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 4, pp. 2025–2038, 2022.

[31] Y. Ji and L. Chen, "Fedqnn: A computation–communication-efficient federated learning framework for iot with low-bitwidth neural network quantization," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2494–2507, 2023.

[32] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020.

[33] Y. Mao, Z. Zhao, M. Yang, L. Liang, Y. Liu, W. Ding, T. Lan, and X. Zhang, "Safari: Sparsity-enabled federated learning with limited and unreliable communications," *IEEE Transactions on Mobile Computing*, pp. 1–12, 2023.

[34] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[35] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[36] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *SIAM Journal on Optimization*, vol. 146, no. 1, pp. 459–494, 2014.

[37] H. Gouk, T. Hospedales, and massimiliano pontil, "Distance-based regularisation of deep networks for fine-tuning," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[38] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[40] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10 708–10 717.

[41] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of the Machine Learning and Systems*, 2020, pp. 429–450.

[42] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-iid data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 7865–7873.

[43] X. Deng, J. Li, C. Ma, K. Wei, L. Shi, M. Ding, and W. Chen, "Low-latency federated learning with dnn partition in distributed industrial iot networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 3, pp. 755–775, 2023.

[44] N. Golowich, A. Rakhlin, and O. Shamir, "Size-independent sample complexity of neural networks," *Information and Inference: A Journal of the IMA*, vol. 9, no. 2, pp. 473–504, 2020.

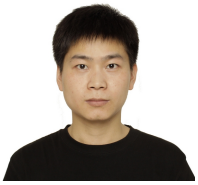
[45] A. Maurer, "A vector-contraction inequality for rademacher complexities," in *Proceedings of Algorithmic Learning Theory*, 2016, pp. 3–17.

Mingjian Zhi received the B.S. degree in network engineering from the Hebei University of Technology, Tianjin, China, in 2018, and the M.S. degree in computer application technology in 2021 from Northeastern University, Shenyang, China, where she is currently working toward the Ph.D. degree in computer science and technology with the School of Computer Science and Engineering. Her current research interests include personalized federated learning and applications in mobile edge computing.

Yuanguo Bi (M'11) received the Ph.D. degree in computer science from Northeastern University, Shenyang, China, in 2010. He was a Visiting Ph.D. Student with the BroadBand Communications Research (BBCR) lab, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada from 2007 to 2009. He is currently a Professor with the School of Computer Science and Engineering, Northeastern University. He has authored/coauthored more than 50 journal/conference papers, including high quality journal papers, such as IEEE JSAC, IEEE TWC, IEEE TITS, IEEE TVT, IEEE IoT Journal, IEEE Communications Magazine, IEEE Wireless Communications, IEEE Network, and mainstream conferences, such as IEEE Global Communications Conference, IEEE International Conference on Communications. His research interests include medium access control, QoS routing, multihop broadcast, and mobility management in vehicular networks, software-defined networking, and mobile edge computing. Dr. Bi has served as an Editor/Guest Editor for IEEE Communications Magazine, IEEE Wireless Communications, IEEE ACCESS. He has also served as the Technical Program Committee member for many IEEE conferences.

Lin Cai (S'00-M'06-SM'10-F'20) has been with the Department of Electrical & Computer Engineering at the University of Victoria since 2005 and is currently a Professor. She is an NSERC E.W.R. Steacie Memorial Fellow, a Canadian Academy of Engineering (CAE) Fellow, an Engineering Institute of Canada (EIC) Fellow, and an IEEE Fellow. In 2020, she was elected as a Member of the Royal Society of Canada's College of New Scholars, Artists and Scientists, and a 2020 "Star in Computer Networking and Communications" by N2Women. Her research interests span several areas in communications and networking, with a focus on network protocol and architecture design supporting ubiquitous intelligence. She received the NSERC Discovery Accelerator Supplement (DAS) Grants in 2010 and 2015, respectively. She co-founded and chaired the IEEE Victoria Section Vehicular Technology and Communications Joint Societies Chapter. She has been elected to serve the IEEE Vehicular Technology Society (VTS) Board of Governors, 2019 - 2024, and served as its VP Mobile Radio from 2023 to 2024. She served as a Board Member of IEEE Women in Engineering from 2022 to 2024, and a Board Member of IEEE Communications Society (ComSoc) from 2024 - 2026.

Wenchao Xu is a research assistant professor at The Hong Kong Polytechnic University. He received his Ph.D. degree from University of Waterloo, Canada, in 2018. Before that he received the B.E. and M.E. degrees from Zhejiang University, Hangzhou, China, in 2008 and 2011, respectively. In 2011, he joined Alcatel Lucent Shanghai Bell Co. Ltd., where he was a Software Engineer for telecom virtualization. He has also been an Assistant Professor at School of Computing and Information Sciences in Caritas Institute of Higher Education, Hong Kong. His research interests includes wireless communication, Internet of things, distributed computing and AI enabled networking.



Haozhao Wang is working as a PostDoc in the School of Computer Science and Technology, Huazhong University of Science and Technology. In the same department, he received his Ph.D. in Dec. 2021. He worked as a research assistant in the Department of Computing at The Hong Kong Polytechnic University from Jun. 2018 to Jun. 2021. His research interests include Distributed Machine Learning, Federated Learning, and AI security.

Tianao Xiang received the B.E. degree in software engineering from Northeastern University, Shenyang, China, in 2018. He is currently pursuing the Ph.D. degree in computer science at Northeastern University, Shenyang, China. His research interests include mobile edge computing, Internet of vehicles and federated learning, etc.

Qiang He received the Ph.D. degree in computer application technology from the Northeastern University, Shenyang, China in 2020. He also worked with School of Computer Science and Technology, Nanyang Technical University, Singapore as a visiting PhD researcher from 2018 to 2019. He is currently an Associated Professor at the College of Medicine and Biological Information Engineering, Northeastern University, Shenyang, China. His research interests include machine learning, social network analytic, data mining, health care, infectious diseases informatics, etc. He has published more than 70 journal articles and conference papers, including IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Cybernetics, IEEE Transactions on Cloud Computing, IEEE Transactions on Computational Social Systems, IEEE Transactions on Cognitive and Developmental Systems.