Adaptive Block-Wise Regularization and Knowledge Distillation for Enhancing Federated Learning

Jianchun Liu, *Member, IEEE, ACM*, Qingmin Zeng, Hongli Xu, *Member, IEEE*, Yang Xu, *Member, IEEE*, Zhiyuan Wang, and He Huang, *Senior Member, IEEE, Member, ACM*

***Abstract*— Federated Learning (FL) is a distributed model training framework that allows multiple clients to collaborate on training a global model without disclosing their local data in edge computing (EC) environments. However, FL usually faces statistical heterogeneity (*e.g.*, non-IID data) and system heterogeneity (*e.g.*, computing and communication capabilities), resulting in poor model training performance. To deal with the above two challenges, we propose an efficient FL framework, named *FedBR*, which integrates the idea of block-wise regularization and knowledge distillation (KD) into the pioneering FL algorithm *FedAvg*, for resource-constrained edge computing. Specifically, we first divide the model into multiple blocks according to the layer order of deep neural network (DNN). The server only sends some consecutive model blocks instead of an entire model to clients for communication efficiency. Then, the clients make use of knowledge distillation to absorb the knowledge of global model blocks to alleviate statistical heterogeneity during local training. We provide a theoretical convergence guarantee for *FedBR* and show that the convergence bound will decrease as the increasing number of model blocks sent by the server. Besides, since the increasing number of model blocks brings more computing and communication costs, we design a heuristic algorithm (GMBS) to determine the appropriate number of model blocks for clients according to their varied data distributions, computing, and communication capabilities. Extensive experimental results show that *FedBR* can reduce the bandwidth consumption by about 31%, and achieve an average accuracy improvement of around 5.6% compared with the baselines under heterogeneous settings.**

***Index Terms*— Federated learning, edge computing, heterogeneity, regularization, knowledge distillation.**

I. Introduction

As THE Internet of Things (IoT) continues to grow rapidly, the amount of data generated on IoT devices such as mobile phones, wearable devices, and monitors is increasing [2]. Since data needs to be transferred to cloud servers in traditional cloud computing, dealing with these increasing amounts of data will lead to higher latency and a worse user experience [3]. Hence, a new computing paradigm has emerged, edge computing (EC) [4], which puts the processing and storage of data at the network edge, to reduce the bandwidth and processing pressure on cloud servers.

As the computing power of edge devices (or clients) becomes more and more powerful, a distributed model training framework, federated learning (FL) [5], [6], is applied to handle more complex tasks for edge computing. In the pioneering FL algorithm *FedAvg* [5], multiple clients use their local data to train a model and then only upload their local models to a parameter server for global aggregation. The server then distributes the aggregated model back to clients for further local training. The model transmission process will continue until the model converges. In this way, FL will reduce latency since model transmission consumes less time compared to data transmission. Moreover, clients' privacy is well protected because the server will not directly access clients' local data.

Despite the above benefits, performing efficient federated learning still faces three main challenges in EC. (1) **Statistical Heterogeneity**. The local data of the clients are usually generated according to the preference and the location of clients [7]. For example, most of the images that are taken by the monitors located in the community are of residents, while monitors located at the crossroad will mostly take images of vehicles. Therefore, the data distribution among different clients is significantly varied. Data samples from different clients are usually not independent and identically distributed (non-IID) and cannot represent the samples of the overall data distribution (*i.e.*, the distribution of total data from all clients). This characteristic of non-IID data will seriously hurt the model training performance and reduce the convergence rate [8]. (2) **System Heterogeneity**. The clients participating in the model training have different computing and communication capabilities [9], [10], which are closely related to the time of model updating and transmission, respectively. Typically, the completion time increases as their computing and communication capabilities weaken. The slowest client takes the longest completion time due to its weakest computing or communication capabilities. In synchronous FL [11], the most commonly used federated learning architecture, the training time of each global round always depends on the slowest client, resulting in a longer completion time.

Manuscript received 18 April 2023; revised 9 June 2023; accepted 29 July 2023; approved by IEEE/ACM Transactions on Networking Editor G. Fanti. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB3301501; in part by the National Science Foundation of China (NSFC) under Grant 61936015, Grant 62132019, and Grant 62102391; in part by the Jiangsu Province Science Foundation for Youths under Grant BK20210122; and in part by the Xiaomi Young Talents Program. Some preliminary results of this paper have been accepted by IEEE/ACM IWQoS 2023 [DOI: 10.1109/IWQoS57198.2023.10188770]. *(Corresponding author: Hongli Xu.)*

Jianchun Liu, Qingmin Zeng, Hongli Xu, Yang Xu, and Zhiyuan Wang are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China, and also with the Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu 215123, China (e-mail: jcliu17@ustc.edu.cn; zqmin@mail.ustc.edu.cn; xuhongli@ustc.edu.cn; xuyangcs@ustc.edu.cn; cswangzy@mail.ustc.edu.cn).

He Huang is with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China (e-mail: huangh@suda.edu.cn).

Digital Object Identifier 10.1109/TNET.2023.3301972

(3) **Communication Limitation**. It takes a lot of bandwidth to deliver models between the server and clients during the training. Therefore, the limited communication bandwidth on the server is also a bottleneck in FL [12], [13]. For example, in FL with hundreds of clients jointly training one VGG16 [14] (size is about 500MB), each global round needs to consume more than 50GB of bandwidth, which may cause network congestion on the server.

To mitigate the impact of statistical heterogeneity, some technologies such as data augmentation [15], client sampling [16], and regularization [17] have been applied to *FedAvg*. The data augmentation is a technique that can enhance the diversity of training data by employing random transformations or knowledge transfer [18]. However, the server needs to collect the label distribution information of clients (*e.g.*, the number of samples in each class), thus leaking the privacy of clients. The client sampling combined with *FedAvg* alleviates the statistical heterogeneity by selecting clients with independent and identically distributed (IID) data [19]. However, some clients are always not selected resulting in reduced amounts of data for training such that the performance of model training will degrade. Applying regularization to *FedAvg* effectively alleviates the impact of statistical heterogeneity by making the local model more similar to the global model [20]. Compared to data augmentation and client sampling, regularization obtains more global information and thus improves the generalization performance of the global model. *FedMLB* [21] is a new regularization method combined with knowledge distillation (KD) technology [22], which blocks the model hierarchically based on *FedAvg* and constructs multiple auxiliary branches. It puts the output obtained by each auxiliary branch and the output obtained by the local model into the loss function as a regularization item through knowledge distillation to obtain more global information. Although this method can effectively alleviate the non-IID data issue, the introduction of auxiliary branches greatly increases the computing cost [23]. Besides, since all clients use the same number of auxiliary branches and receive the entire global model, *FedMLB* performs poorly under system heterogeneity and limited communication resource according to the results of experiments in Section V.

Motivated by *FedMLB*, we design an efficient federated learning framework *FedBR*, which integrates the idea of block-wise regularization and knowledge distillation into *FedAvg*, to solve the above three challenges simultaneously. Specifically, we first divide the model into multiple blocks according to the layer order of DNN, where each block consists of multiple consecutive layers in the neural network. Then, we utilize the global model blocks sent by the server to construct multiple paths (*i.e.*, from input to output) for the local models of clients. To alleviate the impact of non-IID data, we introduce a new regularization technique to absorb the knowledge of the global model blocks via KD, where KD is adopted to reduce the discrepancy between the outputs of different paths. Since the different number of global model blocks can construct a varied number of paths for regularization, we call it block-wise regularization. The later layers of the neural network are more important to the model training performance than the previous layers [24]. As a result, the server only distributes the last several consecutive blocks of the global model in *FedBR*, thus significantly reducing the communication cost compared with *FedAvg*. Although more number of global model blocks allow clients to learn more global information, it will lead to more computing and communication cost. Therefore, *how to dynamically determine the proper number of global model blocks sent by the server for each client* is a key challenge. In summary, the main contributions of our work are as follows:

- We propose an efficient federated learning framework, *FedBR*, which reduces the communication cost by only distributing the partial global model and alleviates the heterogeneous challenges by introducing block-wise regularization and knowledge distillation to absorb knowledge of the global model blocks.
- We provide a theoretical convergence guarantee for *FedBR* and show that the convergence bound will decrease as the number of model blocks sent by the server increases.
- We propose a heuristic algorithm (termed as GMBS) that adaptively determines the number of global model blocks for clients according to their varied data distributions, computing, and communication capabilities.
- We build a simulation environment and evaluate the performance of *FedBR* through extensive experiments. The experimental results show that *FedBR* can reduce the bandwidth consumption by about 31%, and achieve an average accuracy improvement of around 5.6% compared with the baselines under heterogeneous settings.

To present our research on the efficient federated learning framework *FedBR*, this paper is organized as follows: Section II firstly gives some preliminaries of FL, then systematically introduces our proposed *FedBR* framework, illustrates the block-wise regularization, and finally gives the formal definition of the problem. The convergence analysis is discussed in Section III, while the designed algorithm for the problem is presented in Section IV. We also show our experimental results in Section V. Finally, Sections VI and VII provide an overview of related work and a conclusion, respectively.

II. Preliminaries and Problem Formulation

A. Federated Learning

In FL, multiple clients can collaboratively train an efficient global model under the arrangement of the server. The goal of federated learning with N clients is to minimize the average loss function:

$$\min_{x} f(x) = \frac{1}{N}\sum_{n=1}^{N} f_n(x), \tag{1}$$

where x is the model parameters and $f_n(x)$ is the local loss function of client $n \in \{1, \cdots, N\}$. The local loss function of client n is defined as $f_n(x) \triangleq \mathbb{E}_{\varphi_n \sim D_n}[F_n(x; \varphi_n)]$, where φ_n is a sample of the local dataset D_n and $F_n(x; \varphi_n)$ is the corresponding loss to the sample for client n.

In FL, enabling the model to reach target accuracy requires several rounds of global training. To decrease the computing

TABLE I
SUMMARY OF IMPORTANT SYMBOLS

Symbol	Definition
x	the model parameters
$f(x)$	the global loss function
$f_n(x)$	the local loss of client n
N	the total number of clients
K	the number of clients involved in the training
T	the total number of global rounds
I	the total number of local iterations
e	the number of local iterations per global round
x^t	the global model at global round t
$x_k(i)$	the local model of client k after i local iterations
M	the total number of model blocks
α_k	the number of model blocks received by client k
τ	the intervals to send the entire global model
G_*	the global model block
L_*	the local model block
γ	the learning rate
B_c	the total budget of computing resource
B_b	the total budget of communication resource
H_t	the completion time of global round t
$\mathcal{F}$	the convergence threshold of the loss value
ϵ	the non-IID level
δ	the temperature of knowledge distillation
λ_1	the weight of the overall loss of the hybrid paths
λ_2	the weight of knowledge distillation term

and communication overhead, FL usually performs multiple local model updating (*i.e.*, local iteration) between two adjacent global rounds and only selects a part of clients to perform model training. That is, each global round consists of several local iterations. Note that we use round or epoch to represent global updates and iteration to denote local updates in this paper. We adopt e to represent the number of local iterations between two adjacent global rounds. Besides, K ($K \leq N$) represents the number of clients participating in the model training. At each global round, FL (*e.g.*, *FedAvg*) usually is divided into the following three steps [20]:

(1) **Model Broadcast**: At the beginning of global round $t \in \{0, 1, \cdots, T\}$, the server distributes the current global model to K clients, where T is the total number of global rounds.

(2) **Local Updating**: Each client $k \in \{1, 2, \cdots, K\}$ performs e local iterations on its own local dataset D_k using the received global model. Then, client k sends its updated local model to the server.

(3) **Model Aggregation**: After the server collected all local models from K clients, the server uses specific aggregation rule [25] (*e.g.*, model parameter averaging) to update the global model.

Note, some important symbols used in this paper are listed in Table I.

B. Overview of FedBR

The model training performance of traditional FL methods (*e.g.*, *FedAvg*) will decrease dramatically when data distribution among clients is non-IID [20]. To this end, we propose

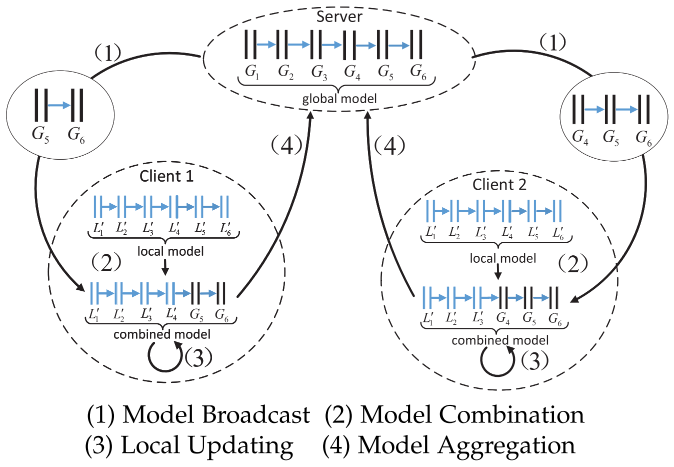

Server
G_1 G_2 G_3 G_4 G_5 G_6
global model
(1)
(1)
G_5 G_6
G_4 G_5 G_6
(4)
(4)
Client 1
L'_1 L'_2 L'_3 L'_4 L'_5 L'_6
local model
(2)
L'_1 L'_2 L'_3 L'_4 G_5 G_6
combined model
(3)
Client 2
L'_1 L'_2 L'_3 L'_4 L'_5 L'_6
local model
(2)
L'_1 L'_2 L'_3 G_4 G_5 G_6
combined model
(3)
(1) Model Broadcast (2) Model Combination
(3) Local Updating (4) Model Aggregation

Fig. 1. The workflow of *FedBR*. Multiple consecutive layers in the neural network constitute a model block (*e.g.*, the two layers form a model block in the figure). G_* and L'_* represent the global model blocks (black blocks) and the local model blocks (blue blocks) saved by clients at the previous global round, respectively, where $*$ denote the indexes of the model block.

an efficient federated learning framework with block-wise regularization (*FedBR*) in this section. By adopting block-wise regularization and knowledge distillation techniques, *FedBR* can enable the clients to absorb more global knowledge during the Local Updating, thereby mitigating the impact of non-IID data on training performance. Besides, *FedBR* makes full use of network resources by distributing different numbers of global model blocks according to the varied computing and communication capabilities of the clients.

Specifically, we divide the model into M blocks, where M is determined by the server according to the layer structure of the model. For example, the model with the twelve layers is divided into six blocks and each block consists of two consecutive layers in Fig. 1. Since current deep learning models tend to have a hierarchical structure, the method of partitioning model blocks can be easily extended to most deep learning models. The main difference between *FedBR* and *FedAvg* is reflected in the Model Broadcast and the Local Updating. Besides, *FedBR* adds the Model Combination between the Model Broadcast and the Local Updating to describe the process of forming the combined model. Concretely, the workflow of *FedBR* is divided into the following four steps as shown in Fig. 1:

(1) **Model Broadcast**: At the beginning of global round t, the server sends different numbers of consecutive global model blocks to the clients according to the varied data distributions, computing, and communication capabilities of clients. We use x^t and α_k^t to denote the global model and the number of global model blocks which are sent to client k at global round t, respectively. To prevent client k from forgetting the information belonging to the first several blocks of the global model ($e.g.$, G_1, G_2, G_3 in Fig. 1), the server will distribute an entire global model to all clients every $\tau+1$ rounds, where $\tau \geq 1$. That is, when global round t is a multiple of $\tau+1$, the server sends the entire global model x^t to client k; otherwise, the server sends the last α_k^t consecutive global model blocks to client k.

(2) **Model Combination**: If client k receives the last α_k^t blocks of the global model at global round t, it will combine

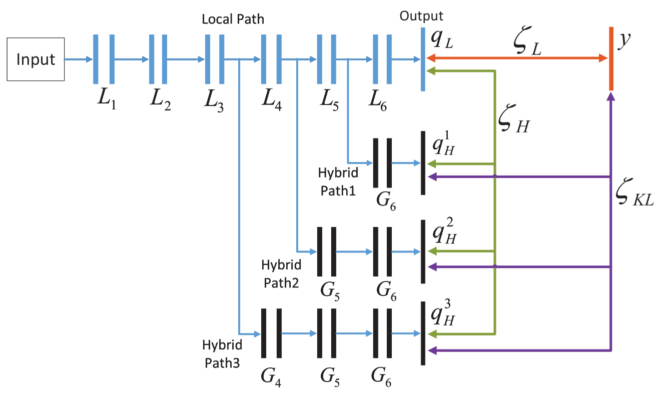


Fig. 2. The block-wise regularization in *FedBR*. Given global round t, the local network of client k consists of α_k^t hybrid paths and one local path. The blue arrow indicates the forward propagation path of the input and q denotes the output of each path. y and ζ represent the truth label of the input and loss term, respectively. Double-headed arrows indicate different loss terms in the loss function.

the first $M - \alpha_k^t$ blocks of the local model $x_k^t(te)$ with the last α_k^t blocks of the global model to form the combined model. We adopt $x_k^t(i)$ to represent the local model of client k after i local iterations, where i is equal to the total number of local iterations from the beginning of model training. If client k receives an entire global model at global round t, it will directly take the received model as the combined model. For example, as shown in Fig 1, Client 1 receives the global model blocks $\{G_5, G_6\}$ and combines them with the local model blocks $\{L_1', L_2', L_3', L_4'\}$ of the previous global round to form the combined model.

(3) **Local Updating**: At global round t, client k uses the combined model as the new local model $x_k^t(te)$. Then, client k performs e local iterations based on its own local dataset D_k using the block-wise regularization in Fig. 2 to obtain updated local model $x_k^{t+1}((t+1)e)$. When e local iterations are completed, client k returns the entire local model $x_k^{t+1}((t+1)e)$ to the server. Meanwhile, client k saves the local model $x_k^{t+1}((t+1)e)$ to form the new combined model at the next global round $t+1$.

(4) **Model Aggregation**: After collecting all local models from K clients, the server aggregates them to derive an up-to-date global model x^{t+1} according to the following equation:

$$x^{t+1} = \frac{1}{K}\sum_{k=1}^{K} x_k^{t+1}((t+1)e). \tag{2}$$

Then, *FedBR* continues to perform the Model Broadcast at the next global round $t+1$ until the global model reaches convergence or network resources are exhausted.

C. Illustration of Block-Wise Regularization

To better illustrate how the local models of clients are updated by using block-wise regularization and knowledge distillation in *FedBR*, we give an example in Fig. 2. We assume that the model is divided into six blocks and the server distributes the last three model blocks to client k at a certain global round t (*i.e.*, $\alpha_k^t = 3$). After client k receives these three blocks (*i.e.*, $\{G_4, G_5, G_6\}$), it will combine them with the first three blocks of the local model stored by client k at global round $t-1$ to form the combined model (*e.g.*, $\{L_1', L_2', L_3', G_4, G_5, G_6\}$ for Client 2 in Fig. 1). Client k uses this combined model to cover the local model ($\{L_1, L_2, L_3, L_4, L_5, L_6\}$) and performs e local iterations on it. In the forward propagation, the path containing only local model blocks is called local path, and the path consisting of local model blocks and global model blocks is called hybrid path. Each input of client k goes through α_k^t hybrid paths (*e.g.*, $\{L_1, L_2, L_3, L_4, G_5, G_6\}$) and one local path (*e.g.*, $\{L_1, L_2, L_3, L_4, L_5, L_6\}$). Consequently, each input will obtain four outputs (*i.e.*, $\{q_L, q_H^1, q_H^2, q_H^3\}$) in Fig. 2. Then, client k puts the four outputs and the truth label y into the loss function defined in Section IV-A, where we adopt knowledge distillation based on the output of local path and the outputs of hybrid paths to obtain knowledge of global model blocks. Note that we apply the standard cross-entropy loss to the outputs of all paths for model training. The KL-divergence (*i.e.*, knowledge distillation function) between the output of the local path and the outputs of hybrid paths are employed for regularization. During the Local Updating, we only update the parameters of local model blocks while keeping the parameters of global model blocks unchanged, *i.e.*, we only update blocks $\{L_1, L_2, L_3, L_4, L_5, L_6\}$ and keep blocks $\{G_4, G_5, G_6\}$ unchanged.

The primary goal of our work is to enhance the training performance of the global model in a resource-efficient manner, particularly in heterogeneous settings, through the use of *FedBR*. The adoption of block-wise regularization as illustrated in Fig. 2 will improve the performance as more global information is learned via KD when the clients perform the Local Updating. Meanwhile, the server takes heterogeneous data distributions, computing, and communication capacities among clients into consideration when it selects proper α_k for client k. Intuitively, when the data distribution of client k deviates from the overall data distribution, more global model blocks are needed to acquire more global information. However, the computing and communication costs become larger as the number of transmitted model blocks increases. Therefore, *how to determine the appropriate α_k for client k while ensuring the generalization performance of the global model is a key challenge for FedBR.*

D. Problem Definition

In this section, we provide the formalized definition of federated learning with the model blocks transmission (FLMBT) problem. Let there be N clients in the network, with only K clients participating in model training ($K \leq N$). This work takes into account two major types of resources, namely computing and communication resources. The total budgets for computing and communication resources in the network are denoted as B_c and B_b, respectively. Besides, the computing cost of one batch and the consumption of transferring the entire model are denoted as c and b, respectively. In *FedBR*, we ignore the computing cost caused by the Model Aggregation because it is trivial compared to the computing cost of the Local Updating [26]. Let c_k^t represent the computing cost of client k at each local iteration of global round t. Then, the computing cost of client k at global round t can be expressed as $e \cdot c_k^t$. Because there are α_k^t hybrid paths in the local network of client k, the computing cost of one local iteration at global round t can be expressed as:

$$c_k^t = (1 + \frac{\alpha_k^t(\alpha_k^t + 1)}{2M}) \cdot \frac{n_k}{B} \cdot c, \tag{3}$$

where n_k denotes the number of local samples in client k, B represents the size of batch, and the coefficient (*i.e.*, $1 + \alpha_k^t(\alpha_k^t + 1)/2M$) is related to the multiplexing of intermediate results between the local path and hybrid paths. We use b_k^t to denote the average communication cost of client k at global round t. While t is a multiple of τ, client k uploads and downloads the whole model. Thus, the communication cost of client k can be expressed as $2 \cdot b$. Otherwise, client k needs to upload the entire local model and download several global model blocks. Consequently, the communication cost of client k can be expressed as $(1 + \alpha_k^t/M) \cdot b$. Combining the above two cases, the average communication cost of client k in each global ground can be expressed as:

$$b_k^t = (1 + \frac{1}{\tau} + (1 - \frac{1}{\tau})\frac{\alpha_k^t}{M}) \cdot b, \tag{4}$$

We aim to minimize the completion time while finding the proper α_k^t for each client k at round t in *FedBR*. Accordingly, the FLMBT problem can be formulated as follows:

$$\min \sum\nolimits_{t=1}^{T} H_t$$
$$s.t. \begin{cases} f(x^T) \le \mathcal{F}, & \\ \sum_{t=1}^{T}\sum_{k=1}^{K} e \cdot c_k^t \le B_c, & \forall k, t \\ \sum_{t=1}^{T}\sum_{k=1}^{K} b_k^t \le B_b, & \forall k, t \\ \alpha_k^t \in \{1, \cdots, M-1\}, & \forall k, t, \end{cases} \tag{5}$$

where H_t represents the completion time of global round t, *i.e.*, the time required by K clients to complete their local training after the last global round. The first inequality reflects the convergence condition, where $\mathcal{F}$ indicates the convergence threshold of the loss value after T global rounds. The second set of inequalities specifies that the total computing cost during T global rounds should not exceed the computing budget of the network. The third set of inequalities ensures that the communication cost during T global rounds should not exceed the communication budget of the network. The fourth group of formulas denotes that the number of blocks downloaded by client k should be an integer and not exceed the number of total model blocks.

In reality, directly solving the FLMBT problem in Eq. (5) is a challenging task. This is due to the fact that the decision variable of FLMBT is an integer, which is a typical characteristic of integer programming problems. Finding an optimal solution for an integer programming problem is usually an NP-hard task [27]. Consequently, solving problem in Eq. (5) at each global round can be time-consuming. Additionally, the dynamic network conditions make the problem even more complex. However, we believe that with feedback on the computing and communication time during the training process, a greedy-based approach can be an effective way to efficiently solve the FLMBT problem while meeting the resource constraints at each global round. Thus, we propose to develop a greedy-based algorithm to solve the FLMBT problem.

III. Convergence Analysis

In this section, we propose a reliable convergence guarantee for *FedBR* based on the non-convex optimization assumption that is solved using distributed SGD [28]. To analyze the convergence, we begin by making some common assumptions that have been widely used in previous works [29], [30].

Assumption 1: (*Smoothness:*) Each function $f_n(x)$ is smooth with modulus L, *i.e.*,

$$f_n(y) - f_n(x) \le \nabla f_n(y - x) + \frac{L}{2}||y - x||^2, \quad \forall n,\ \forall x,\ \forall y,$$
$$||\nabla f_n(x) - \nabla f_n(y)|| \le L^2||x - y||^2, \quad \forall n,\ \forall x,\ \forall y.$$

Assumption 2: (*Bounded variances and second moments:*) There exist constants $\sigma > 0$ and $G > 0$ such that

$$\mathbb{E}_{\varphi_n \sim \mathcal{D}_n}||\nabla F_n(x; \varphi_n) - \nabla f_n(x)||^2 \le \sigma^2, \quad \forall x, \forall n,$$
$$\mathbb{E}_{\varphi_n \sim \mathcal{D}_n}||\nabla F_n(x; \varphi_n)||^2 \le G^2, \quad \forall x, \forall n.$$

For the convenience of analysis, we assume that all clients participate in the training process (*i.e.*, $K = N$). Meanwhile, suppose that there are I local iterations totally during the model training (*i.e.*, $I = T \cdot e$). Fix the index of local iteration i, we define the average of the model as

$$x^t(i) = \frac{1}{N}\sum_{n=1}^{N} x_n^t(i), \tag{6}$$

where $x_n^t(i)$ represents the local model of client n with $i \in [te, (t+1)e)$. When i is a multiple of e (*i.e.*, $i = t \cdot e$), the server performs the Model Aggregation to update the global model (*i.e.*, $x^t \triangleq x^{t-1}(i-1)$), where the $(i-1)$-th local iteration is the last iteration of the global round $t-1$ and the i-th local iteration is the first iteration of global round t. After client n receives the global model blocks or the whole global model, it will form a combined model according to the previous rule in Section II-B and use this combined model to cover its local model at $i-1$ iteration (*i.e.*, $x_n^{t-1}(i-1) \triangleq x_n^t$, where x_n^t represents the combined model of client n after the Model Combination). Then, client n continues for the i-th local iteration based on SGD to obtain $x_n^t(i)$. To express the relationship between the combined model of client n and the global model, we adopt an upper bound β^2, where $||x^t - x_n^t||^2 \le \beta^2$. Moreover, every client can locally observe independent and unbiased stochastic gradients denoted by $G_n^i = \nabla F_n(x_n^{t'}(i-1); \varphi_n^i)$ with $\mathbb{E}_{\varphi_n^i \sim \mathcal{D}_n}[G_n^i|\varphi^{[i-1]}] = \nabla f_n(x_n^{t'}(i-1))$, where $\varphi^{[i-1]} \triangleq [\varphi_n^\tau]_{n \in \{1,2,\cdots,N\}, \tau \in \{1,2,\cdots,i-1\}}$ represents all the randomness up to iteration $i-1$ and $t' = \lfloor \frac{i-1}{e} \rfloor$. As a result, we have

$$x_n^t(i) = x_n^{t'}(i-1) - \gamma G_n^i, i \in [1, I], \tag{7}$$

where γ denotes the learning rate. The following lemma provides a bound on the difference between $x^t(i)$ and $x_n^t(i)$.

Lemma 1: Under Assumption 1, the proposed framework can ensure

$$\mathbb{E}[||x^t(i) - x_n^t(i)||^2] \leq 2\beta^2 + 8\gamma^2 e^2 G^2, \quad \forall n,\ \forall i,$$

where G is a constant defined in Assumption 2.

Proof of Lemma 1. Fix $i \geq 1$ and $n \in \{1, 2, \cdots, N\}$. Considering the largest $i_0 \leq i$ such that $x^t = x^{t-1}(i_0)$ (Note that such i_0 always exists and $i - i_0 \leq e$.), we have

$$x_n^t(i) = x_n^t - \gamma \sum_{\tau=i_0+1}^{i} G_n^\tau \tag{8}$$

where γ is the learning rate. By (6) and (8), we have

$$x^t(i) = x^t - \gamma \sum_{\tau=i_0+1}^{i} \frac{1}{N} \sum_{n=1}^{N} G_n^\tau \tag{9}$$

Thus, we have

$$\begin{aligned}
&\mathbb{E}[||x^t(i) - x_n^t(i)||^2] \\
&= \mathbb{E}[||x^t - \gamma \sum_{\tau=i_0+1}^{i} \frac{1}{N} \sum_{n=1}^{N} G_n^\tau - (x_n^t - \gamma \sum_{\tau=i_0+1}^{i} G_n^\tau)||^2] \\
&= \mathbb{E}[||(x^t - x_n^t) + \gamma(\sum_{\tau=i_0+1}^{i} G_n^\tau - \sum_{\tau=i_0+1}^{i} \frac{1}{N} \sum_{n=1}^{N} G_n^\tau)||^2] \\
&\overset{(a)}{\leq} 2\mathbb{E}[||x^t - x_n^t||^2 + ||\gamma(\sum_{\tau=i_0+1}^{i} G_n^\tau - \sum_{\tau=i_0+1}^{i} \frac{1}{N} \sum_{n=1}^{N} G_n^\tau)||^2] \\
&= 2\mathbb{E}[||x^t - x_n^t||^2] + 2\gamma^2 \mathbb{E}[|| \sum_{\tau=i_0+1}^{i} G_n^\tau - \sum_{\tau=i_0+1}^{i} \frac{1}{N} \sum_{n=1}^{N} G_n^\tau||^2] \\
&\overset{(b)}{\leq} 2\beta^2 + 2\gamma^2 (i - i_0) \mathbb{E}[\sum_{\tau=i_0+1}^{i} ||G_n^\tau - \frac{1}{N} \sum_{n=1}^{N} G_n^\tau||^2] \\
&\overset{(c)}{\leq} 2\beta^2 + 4\gamma^2 (i - i_0) \mathbb{E}[\sum_{\tau=i_0+1}^{i} ||G_n^\tau||^2 + \sum_{\tau=i_0+1}^{i} ||\frac{1}{N} \sum_{n=1}^{N} G_n^\tau||^2] \\
&\overset{(d)}{\leq} 2\beta^2 + 8\gamma^2 (i - i_0)^2 G^2 \\
&\leq 2\beta^2 + 8\gamma^2 e^2 G^2
\end{aligned}$$

where (a)-(c) are obtained by utilizing the inequality $|| \sum_{i=1}^{n} z_i||^2 \leq n \sum_{i=1}^{n} ||z_i||^2$ for any vectors z_i and any positive integer n (using $n = 2$ in (a), $n = i - i_0$ in (b), $n = 2$ in (c)); and (d) follows from Assumption 2.

Theorem 1: If $0 < \gamma \leq \frac{1}{L}$, the convergence bound for our proposed framework is as follows:

$$\begin{aligned}
\frac{1}{I} \sum_{i=1}^{I} \mathbb{E}[||\nabla f(x^{t'}(i-1))||^2] \leq \frac{2}{\gamma I} (f(x^0) - f(x^*)) + \frac{L}{N} \gamma \sigma^2 \\
+ 2\beta^2 L^2 + 8\gamma^2 e^2 G^2 L^2,
\end{aligned}$$

where $t' = \lfloor \frac{i-1}{e} \rfloor$ and x^* represents the optimal model which minimizes the global loss function f.

Proof of Theorem 1. Fix $i \geq 1$. By (6) and (7), we have

$$x^t(i) = x^{t'}(i-1) - \gamma \frac{1}{N} \sum_{n=1}^{N} G_n^i \tag{10}$$

By the smoothness of f, we have

$$\begin{aligned}
\mathbb{E}[f(x^t(i))] \leq \mathbb{E}[f(x^{t'}(i-1))] \\
+ \mathbb{E}[\langle \nabla f(x^{t'}(i-1)), x^t(i) - x^{t'}(i-1) \rangle] \\
+ \frac{L}{2} \mathbb{E}[||x^t(i) - x^{t'}(i-1)||^2]
\end{aligned} \tag{11}$$

Not that

$$\begin{aligned}
&\mathbb{E}[||x^t(i) - x^{t'}(i-1)||^2] \\
&\overset{(a)}{=} \gamma^2 \mathbb{E}[|| \frac{1}{N} \sum_{n=1}^{N} G_n^i||^2] \\
&\overset{(b)}{=} \gamma^2 \mathbb{E}[|| \frac{1}{N} \sum_{n=1}^{N} (G_n^i - \nabla f_n(x_n^{t'}(i-1))||^2] \\
&\quad + \gamma^2 \mathbb{E}[|| \frac{1}{N} \sum_{n=1}^{N} \nabla f_n(x_n^{t'}(i-1))||^2] \\
&\overset{(c)}{=} \gamma^2 \frac{1}{N^2} \sum_{n=1}^{N} \mathbb{E}[||G_n^i - \nabla f_n(x_n^{t'}(i-1))||^2] \\
&\quad + \gamma^2 \mathbb{E}[|| \frac{1}{N} \sum_{n=1}^{N} \nabla f_n(x_n^{t'}(i-1))||^2] \\
&\overset{(d)}{\leq} \frac{1}{N} \gamma^2 \sigma^2 + \gamma^2 \mathbb{E}[|| \frac{1}{N} \sum_{n=1}^{N} \nabla f_n(x_n^{t'}(i-1))||^2]
\end{aligned} \tag{12}$$

where (a) follows from (10); (b) holds by the fact that $\mathbb{E}[||Z||^2] = \mathbb{E}[||Z - \mathbb{E}[Z]||^2] + ||\mathbb{E}[Z]||^2$ for any random vector Z and using $\mathbb{E}[G_n^i] = \nabla f_n(x_n^{t'}(i-1))$; (c) is tenable because each $G_n^i - \nabla f_n(x_n^{i-1})$ has mean 0 and is independent for all clients; and (d) follows from Assumption 2.

In addition, note that

$$\begin{aligned}
&\mathbb{E}[\langle \nabla f(x^{t'}(i-1)), x^t(i) - x^{t'}(i-1) \rangle] \\
&\overset{(a)}{=} -\gamma \mathbb{E}[\langle \nabla f(x^{t'}(i-1)), \frac{1}{N} \sum_{n=1}^{N} G_n^i \rangle] \\
&\overset{(b)}{=} -\gamma \mathbb{E}[\langle \nabla f(x^{t'}(i-1)), \frac{1}{N} \sum_{n=1}^{N} \nabla f_n(x_n^{t'}(i-1)) \rangle] \\
&\overset{(c)}{=} -\frac{\gamma}{2} \mathbb{E}[||\nabla f(x^{t'}(i-1))||^2 + || \frac{1}{N} \sum_{n=1}^{N} \nabla f_n(x_n^{t'}(i-1))||^2 \\
&\quad - ||\nabla f(x^{t'}(i-1)) - \frac{1}{N} \sum_{n=1}^{N} \nabla f_n(x_n^{t'}(i-1))||^2]
\end{aligned} \tag{13}$$

where (a) follows from (10); (c) follows from the fact that $\langle z_1, z_2 \rangle = \frac{1}{2}(||z_1||^2 + ||z_2||^2 - ||z_1 - z_2||^2)$ for any two vector z_1, z_2 of the same dimension; (b) follows because

$$\begin{aligned}
&\mathbb{E}[\langle \nabla f(x^{t'}(i-1)), \frac{1}{N} \sum_{n=1}^{N} G_n^i \rangle] \\
&= \mathbb{E}[\mathbb{E}[\langle \nabla f(x^{t'}(i-1)), \frac{1}{N} \sum_{n=1}^{N} G_n^i \rangle | \varphi^{[i-1]}]] \\
&= \mathbb{E}[\langle \nabla f(x^{t'}(i-1)), \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}[G_n^i | \varphi^{[i-1]}] \rangle]
\end{aligned}$$

$$= \mathbb{E}[\langle \nabla f(x^{t'}(i-1)), \frac{1}{N}\sum_{n=1}^{N} \nabla f_n(x_n^{t'}(i-1))\rangle]$$

where the first equation follows by the iterated law of expectations, the second equation follows because $x^{t'}(i-1)$ is determined by $\varphi^{[i-1]}$.

Substituting (12) and (13) into (11), we have

$$\begin{aligned}
&\mathbb{E}[f(x^t(i))] \\
&\leq \mathbb{E}[f(x^{t'}(i-1))] - \frac{\gamma-\gamma^2 L}{2}\mathbb{E}[||\frac{1}{N}\sum_{n=1}^{N}\nabla f_n(x_n^{t'}(i-1))||^2] \\
&\quad - \frac{\gamma}{2}\mathbb{E}[||\nabla f(x^{t'}(i-1))||^2] + \frac{L}{2N}\gamma^2\sigma^2 \\
&\quad + \frac{\gamma}{2}\mathbb{E}[||\nabla f(x^{t'}(i-1)) - \frac{1}{N}\sum_{n=1}^{N}\nabla f_n(x_n^{t'}(i-1))||^2] \\
&\overset{(a)}{\leq} \mathbb{E}[f(x^{t'}(i-1))] - \frac{\gamma-\gamma^2 L}{2}\mathbb{E}[||\frac{1}{N}\sum_{n=1}^{N}\nabla f_n(x_n^{t'}(i-1))||^2] \\
&\quad - \frac{\gamma}{2}\mathbb{E}[||\nabla f(x^{t'}(i-1))||^2] + \frac{L}{2N}\gamma^2\sigma^2 \\
&\quad + \gamma\beta^2L^2 + 4\gamma^3e^2G^2L^2 \\
&\overset{(b)}{\leq} \mathbb{E}[f(x^{t'}(i-1))] - \frac{\gamma}{2}\mathbb{E}[||\nabla f(x^{t'}(i-1))||^2] \\
&\quad + \frac{L}{2N}\gamma^2\sigma^2 + \gamma\beta^2L^2 + 4\gamma^3e^2G^2L^2
\end{aligned} \tag{14}$$

where (b) follows $0 < \gamma \leq \frac{1}{L}$; (a) follows because

$$\begin{aligned}
&\mathbb{E}[||\nabla f(x^{t'}(i-1)) - \frac{1}{N}\sum_{n=1}^{N}\nabla f_n(x_n^{t'}(i-1))||^2] \\
&= \frac{1}{N^2}\mathbb{E}[||\sum_{n=1}^{N}\nabla f_n(x^{t'}(i-1)) - \sum_{n=1}^{N}\nabla f_n(x_n^{t'}(i-1))||^2] \\
&\leq \frac{1}{N}\mathbb{E}[\sum_{n=1}^{N}||\nabla f_n(x^{t'}(i-1)) - \nabla f_n(x_n^{t'}(i-1))||^2] \\
&\leq \frac{L^2}{2}\mathbb{E}[\sum_{n=1}^{N}||x^{t'}(i-1) - x_n^{t'}(i-1)||^2] \\
&\leq 2\beta^2L^2 + 8\gamma^2e^2G^2L^2
\end{aligned}$$

where the first equation follows $f(x) \triangleq \frac{1}{N}\sum_{n=1}^{N} f_n(x)$, the first inequality follows by using the inequality $||\sum_{i=1}^{n} z_i||^2 \leq n\sum_{i=1}^{n}||z_i||^2$ for any vectors z_i and any positive integer n (using $n = N$), the second inequality can be obtained by applying the smoothness property of Assumption 1, and the third inequality is obtained by utilizing Lemma 1.

Dividing (14) both sides by $\frac{\gamma}{2}$ and rearranging terms yields

$$\begin{aligned}
&\mathbb{E}[||\nabla f(x^{t'}(i-1))||^2] \\
&\quad \leq \frac{2}{\gamma}(\mathbb{E}[f(x^{t'}(i-1))] - \mathbb{E}[f(x^t(i))]) \\
&\quad\quad + \frac{L}{N}\gamma\sigma^2 + 2\beta^2L^2 + 8\gamma^2e^2G^2L^2
\end{aligned}$$

Summing over $i \in \{1, 2, \cdots, I\}$ and dividing both sides by I yields

$$\begin{aligned}
&\frac{1}{I}\sum_{i=1}^{I}\mathbb{E}[||\nabla f(x^{t'}(i-1))||^2] \\
&\quad \leq \frac{2}{\gamma I}(\mathbb{E}[f(x^0(0)) - f(x^t(i))]) \\
&\quad\quad + \frac{L}{N}\gamma\sigma^2 + 2\beta^2L^2 + 8\gamma^2e^2G^2L^2 \\
&\quad \leq \frac{2}{\gamma I}(f(x^0) - f(x^*)) \\
&\quad\quad + \frac{L}{N}\gamma\sigma^2 + 2\beta^2L^2 + 8\gamma^2e^2G^2L^2
\end{aligned}$$

where $t' = \lfloor\frac{i-1}{e}\rfloor$ and x^* denotes the optimal model which minimizes the global loss function f.

We utilize the expected squared gradient norm in Theorem 1 to analyze the convergence in non-convex settings, as in previous work [31]. It is observed that the convergence bound is positively correlated with the upper bound of the distance between the global model and the combined model of client n. As more global model blocks are sent by the server, the upper bound of the distance becomes smaller, which in turn results in a tighter convergence bound.

IV. Algorithm Design

In this section, we propose a heuristic algorithm to solve the FLMBT problem in Eq. (5). First, we give the local loss function for each client according to the block-wise regularization shown in Fig. 2 and design a feedback variable to determine the number of blocks for each client according to the returned information of clients during the training process. Then, we design a decision variable and introduce the proposed algorithm in detail.

A. Preliminaries for Algorithm Design

(1) Local Loss Function: Because we adopt knowledge distillation to enable clients to learn more global information from global model blocks, the knowledge distillation term needs to be included in the loss function. As a result, the local loss function of client k mainly consists of two kinds of forms, where one is the cross-entropy loss (a common loss function in deep learning) and the other is the knowledge distillation term. We use CrossEntropy(a, b) to represent the cross-entropy loss between two same size vectors a and b. The cross-entropy loss of the local path can be expressed:

$$\zeta_L = \text{CrossEntropy}(q_L, y), \tag{15}$$

where y denotes the label of the sample. The overall cross-entropy loss of the hybrid paths is defined as:

$$\zeta_H = \frac{1}{m}\sum_{j=1}^{m}\text{CrossEntropy}(q_H^j, y), \tag{16}$$

where m represents the number of hybrid paths for client k at global round t (*i.e.*, $m = \alpha_k^t$). Besides, we use the knowledge distillation term for regularization to enable the local path to absorb knowledge on hybrid paths. The following equation gives the overall knowledge distillation term between the local path and hybrid paths:

$$\zeta_{KL} = \frac{1}{m}\sum_{j=1}^{m} \mathrm{KL}(\hat{q}_H^j, \hat{q}_L), \tag{17}$$

where $\mathrm{KL}(a, b)$ denotes the Kullback-Leibler (KL) divergence between two same size vectors a and b, and $\hat{q}$ indicates the output using hyperparameter temperature δ [32]. Temperature δ affects the smooth level of softmax output. Then, the total loss function of client k is defined as:

$$\zeta = \zeta_L + \lambda_1 \cdot \zeta_H + \lambda_2 \cdot \zeta_{KL}, \tag{18}$$

where λ_1 ($\lambda_1 \geq 0$) and λ_2 ($\lambda_2 \geq 0$) are hyperparameters that determine the weights of ζ_H and ζ_{KL} in the loss function. The local model of client k is updated based on the loss function in Eq. (18) and its own dataset D_k. Note, we only update the parameters of local model blocks and keep the parameters of global model blocks unchanged.

(2) Feedback Variable: The key challenge to solve problem in Eq. (5) is to find the appropriate α_k^t for each client k at the start of any global round t while minimizing the completion time. The number of blocks α_k^t should depend on the data distribution as well as the communication and computing time of client k. For data distribution, we use the bias between local models sent by clients and the global model after the Model Aggregation to simulate the bias between local data distributions on clients and the overall data distribution. The difference between the local model and the global model is proportional to the bias between the local data distribution and the overall data distribution [33]. We use d_k to denote the deviation between the data distribution of client k and the overall data distribution. Therefore, d_k is quantified by the following equation:

$$d_k = \frac{||x^t - x_k^t(te)||^2}{\sum\limits_{k=1}^{K} ||x^t - x_k^t(te)||^2}. \tag{19}$$

Obviously, when the data distribution of client k deviates from the overall data distribution seriously, client k needs to receive more global model blocks to learn more global knowledge. Therefore, α_k^t should be positively correlated with d_k. Let $t_{k,b}$ and $t_{k,c}$ represent the communication time and the computing time of client k, respectively. To facilitate comparison, we employ the following normalizations for the communication and computing time of client k:

$$t'_{k,b} = \frac{t_{k,b}}{\sum\limits_{k'=1}^{K} t_{k',b}},$$
$$t'_{k,c} = \frac{t_{k,c}}{\sum\limits_{k'=1}^{K} t_{k',c}}. \tag{20}$$

Apparently, the communication time of client k will increase as the increasing number of global model blocks ($i.e., \alpha_k$) is sent by the server. Moreover, client k totally has ($\alpha_k^t + 1$) computing paths at global round t according to Fig. 2 during the Local Updating. Thus, the computing time of client k is positively correlated with α_k^t. In summary, when $t'_{k,b}$ and $t'_{k,c}$ are relatively small, it means that client k takes shorter communication and computing time than other clients, so the server needs to send more global model blocks to client k in order to avoid resource idleness on client k. That is, α_k^t is inversely proportional to $t'_{k,b}$ and $t'_{k,c}$ for client k.

Based on the above analysis, we design a feedback variable $r_{k,m}^t$:

$$r_{k,m}^t \triangleq \frac{d_k \cdot \Delta\zeta_L}{e^{t'_{k,c}+t'_{k,b}}}, \tag{21}$$

where the exponential function is applied to enhance the effect of time cost on the feedback variable, ζ_L is defined in Section IV-A, and $\Delta\zeta_L$ indicates the accuracy improvement of client k. $\Delta\zeta_L$ is the ζ_L at global round $t-1$ minus the ζ_L at global round t. The server will calculate $r_{k,m}^t$ for each client k according to the feedback information returned by client k at global round $t-1$. Then, the server makes use of $r_{k,m}^t$ to decide the value of α_k^t for client k, which will be described in detail in Section IV-B.

B. Detailed Description of the Proposed Algorithm

To determine the proper α_k^t for client k, we design a greedy-based model block selection (GMBS) algorithm. The server holds one vector p_k for each client k to remember the feedback information of the previous global round. We define p_k as follows:

$$p_k = \{p_{k,1}, p_{k,2}, \cdots, p_{k,M-1}\}, \tag{22}$$

where $p_{k,m}$ corresponds to the feedback information of sending m global model blocks to client k. We need to remember previous feedback information to decrease the interference of error feedback information caused by a certain round. Therefore, $p_{k,m}$ is updated by the following equation:

$$p_{k,m} = \lambda r_{k,m}^t + (1-\lambda)p_{k,m}, \tag{23}$$

where λ ($0 \leq \lambda \leq 1$) is a hyperparameter that reflects the weight of real-time feedback information. Since the number of global model blocks is randomly selected at the beginning of training and the states of clients are variable, we cannot always adopt m corresponding to the highest value of feedback information as α_k for the decision of client k. Therefore, we add a penalty item for feedback information $p_{k,m}$. Finally, we use the following decision variable $V_{k,m}$ to select the number of global model blocks:

$$V_{k,m} = p_{k,m} + \frac{\sqrt{\ln{(t+1)}}}{n_{k,m}+1}, \tag{24}$$

where $n_{k,m}$ represents the frequency that the server selects m global model blocks to send to client k. We choose m corresponding to the largest decision variable as the number of global model blocks received by client k at round t ($i.e., m = \arg\max\limits_{m} V_{k,m}$). In the second term of Eq. (24), the denominator indicates the priority of m that is frequently selected by the server will decrease and the numerator indicates the

Algorithm 1 Greedy-Based Model Blocks Selection (GMBS)

```
Require: The number of clients participating in training $K$; the local batch size $B$; the number of local iterations $e$; hyperparameters $\lambda$, $\lambda_1$, $\lambda_2$, $\delta$.
Ensure: The global model $x^T$
    Initialize $x^0, x_k^0(0), p_k, n_{k,m}, \alpha_k^1$.
 1: for each global round $t = \{1, 2, \cdots, T\}$ do
 2:   for each client $k \in \{1, 2, \cdots, K\}$, the server do
 3:     Update $p_{k,m}$ by Eqs. (19), (20), (21), and (23).
 4:     for each $p_{k,m} \in p_k$ do
 5:       Calculate $V_{k,m}$ according to Eq. (24).
 6:     end for
 7:     Make decision $\alpha_k^t, m = \arg\max_m V_{k,m}$.
 8:     Update frequency $n_{k,m} = n_{k,m} + 1$.
 9:   end for
10:   The server distributes $\alpha_k^t$ global model blocks to each client $k \in \{1, \cdots, K\}$.
11:   for each client $k \in \{1, \cdots, K\}$ do
12:     Perform $e$ local iterations according to Eq. (18).
13:     Record time $t_{k,b}$ and $t_{k,c}$.
14:     Return $t_{k,b}$, $t_{k,c}$ and $x_k^t(t \cdot e)$ to the server.
15:   end for
16:   The server aggregates models to obtain $x^t$ by Eq. (2).
17: end for
```

influence of the second term will weaken as the training goes on.

The GMBS algorithm is introduced in detail in Alg. 1. At the beginning of training, we initialize $p_k = \mathbf{0}$, $n_{k,m} = 0$ for $\forall k \in \{1, \cdots, K\}, m \in \{1, \cdots, M-1\}$ and randomly initialize global model x^0, local model $x_k^0(0)$, α_k^1 for each client k (Line 1). At the start of global round t, the server selects proper α_k^t for each client k (Lines 3-8). First, the server updates feedback information $p_{k,m}$ according to the return value sent by client k at global round $t-1$ (Line 4). Then, the server makes use of p_k to calculate the decision variable V_k for each m (Lines 5-6). Finally, the server chooses m corresponding to the largest $V_{k,m}$ as the number of global model blocks sent to client k and updates frequency $n_{k,m}$ corresponding to m (Lines 7-8). Note, since there is no feedback information when $t = 1$, the decision process (Lines 4-7) is not needed and α_k^1 is randomly initialized. After the server selects all α_k^t for each client k, the server will distribute α_k^t global model blocks to each client according to the Model Broadcast rule described in Section II-B (Line 9). Client k will perform e local iterations according to the loss function defined in Section IV-A after client k receives the global model blocks (Line 11). At the last local iteration of client k, the client k needs to record the communication and computing time $t_{k,b}$ and $t_{k,c}$ to calculate $p_{k,m}$ used to make the decision at the next global round (Line 12). Then, client k returns $t_{k,b}$, $t_{k,c}$, and the updated local model $x_k^t(te)$ to the server (Line 13). The server will perform the Model Aggregation so as to form an up-to-state global model after it receives all local models sent by clients (Line 14). Finally, we obtain the global model x^T after T global rounds of training.

V. Performance Evaluation

In this section, we conduct nine sets of experiments to prove the effectiveness of *FedBR*. We first introduce the experimental setup in detail and then show the experimental results.

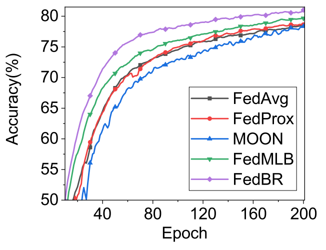
A. Experimental Setup

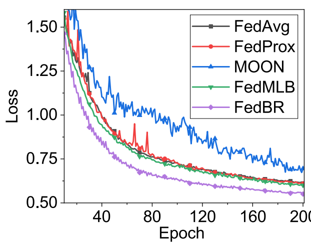
Environment Setup: All the experiments are conducted on an AMAX deep learning workstation with an Intel(R) Xeon(R) Gold 5218R CPU, 4 NVIDIA GeForce RTX 3090 GPUs, and 256 GB RAM. We build an FL simulation environment and implement all methods under the PyTorch framework [34]. As represented in [35], we generate a total of 100 clients in the simulation and randomly activate 10 of them to participate in the model training so that we efficiently simulate the training process of *FedBR* and the baselines. It is worth noting that our method can be easily scaled up to include more edge nodes.

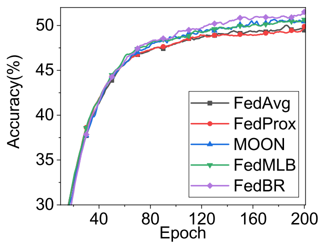
Datasets and Model: We evaluate the performance of *FedBR* and the baselines on two commonly used real-world datasets in federated learning, namely CIFAR10 and CIFAR100 [36]. Both CIFAR10 and CIFAR100 contain 60,000 RGB color images with the size of 32 $\times$ 32 , where the training set and the testing set contain 50,000 and 10,000 images, respectively. CIFAR10 and CIFAR100 are composed of 10 classes and 100 classes, respectively. During the experiments, we evenly divide the number of training samples to each client, that is to say, each client has 5,000 training images. Besides, we train a ResNet18 [37] model, which is often adopted for image recognition tasks, with a size of 42.65MB on both CIFAR10 and CIFAR100. ResNet18 consists of seventeen convolutional layers and one fully connected layer. Due to the existence of residual blocks in ResNet18, we divide the model into six model blocks during the experiments. The first convolutional layer forms the first block, every four subsequent convolutional layers form a block, and the last fully connected layer forms the sixth block. Since current deep learning models tend to have a hierarchical structure, our approach to partitioning model blocks can also be easily extended to other deep learning models.

Data Division: Different data distributions, *i.e.*, IID and non-IID data, among clients have a significant influence on the model training performance. We use ϵ to represent the non-IID level, which ranges from $\{0, 1, \cdots, 9\}$. When $\epsilon = 0$, it represents the data distribution is IID. For CIFAR10, when $\epsilon \in \{1, \cdots, 9\}$, it denotes that the $\epsilon \times 10\%$ local data of the client belongs to the same class and the rest of the local data is evenly divided into the remaining nine classes. For CIFAR100, when $\epsilon \in \{1, \cdots, 9\}$, it denotes that the local dataset of each client lacks $\epsilon \times 10$ kinds of images and the local data is evenly distributed among the remaining $100 - \epsilon \times 10$ classes. In our experiments, we mainly consider four data distributions (*i.e.*, $\epsilon \in \{0, 5, 7, 9\}$) to verify the impact of data distribution on model training performance, including IID data and the three different non-IID levels.

Baselines and Metrics: We compare the performance of *FedBR* together with the four baselines, *i.e.*, *FedAvg* [5], *FedProx* [20], *MOON* [38], and *FedMLB* [21]. *FedAvg* is the first proposed method of federated learning, using model


(a) Testing accuracy on CIFAR10


(b) Testing loss on CIFAR10


(c) Testing accuracy on CIFAR100

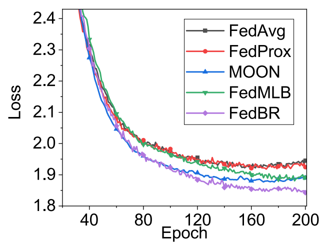

(d) Testing loss on CIFAR100

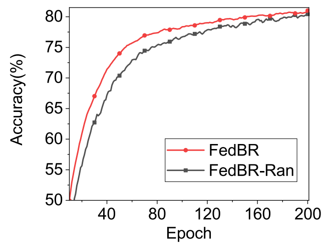
Fig. 3. Testing accuracy and loss of different methods under the non-IID level $\epsilon = 7$.

averaging to aggregate models. Compared with *FedAvg*, *FedProx* adds a proximal term between the global model and the local model to the loss function, making the local training more stable. *MOON* combines comparative learning at the model level with *FedAvg* to correct the local model training of the clients by exploiting the similarity between model representations. In *FedMLB*, multiple hybrid pathways are introduced to absorb more knowledge of the global model to mitigate the negative impact of non-IID data. In this paper, we evaluate the performance of our proposed framework using the following three metrics: (1) Testing accuracy. In each global round, we evaluate the training performance (*i.e.*, accuracy) of the global model on the testing set. (2) Time cost. We record the completion time (*i.e.*, computing and communication time) when the model training achieves the given target accuracy. In each global round, the completion time of the system is equal to the longest completion time among the clients participating in the training. (3) Communication cost. We record the total bandwidth consumption of all clients for uploading and downloading models during the model training.

Heterogeneity Settings: We adopt heterogeneous communication and computing capabilities among clients to simulate system heterogeneity. For the heterogeneous communication capabilities among clients, we set the bandwidth between the server and clients to range from 40Mbps to 280Mbps and randomly change the bandwidth every 10 global rounds. Note that according to [39], current mobile phones operate in frequencies between 0.8 to 2.5 GHz and are capable of average download speeds of 230 Mbps in the 5G network. Besides, we assign different computing capabilities to each client. Specifically, for ResNet18 on CIFAR100, the measured computing cost on 100 devices varies between 1.3s to 6.7s, 2.0s to 10.0s, 2.3s to 11.2 s, 2.4s to 12.1s and 2.3 to 11.7s for *FedAvg*, *FedPorx*, *MOON*, *FedMLB* and *FedBR* in every global round, respectively. We randomly set the computing time with such ranges for varied clients in the training process, and randomly change the computing time every 10 global rounds. It is noted that we set the bandwidth and computing time as the average value of the above range when we are not conducting system heterogeneity experiments.

B. Experimental Results

We conduct nine sets of simulations to evaluate the effectiveness of our proposed method. The simulation results are presented as follows:


(a) Testing accuracy on CIFAR10.

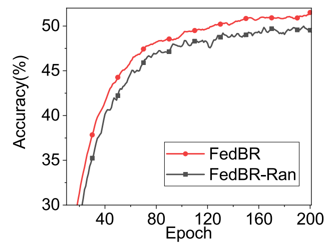

(b) Testing accuracy on CIFAR100.

Fig. 4. Comparison between *FedRan* and *FedBR*.

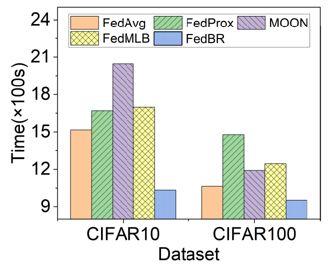
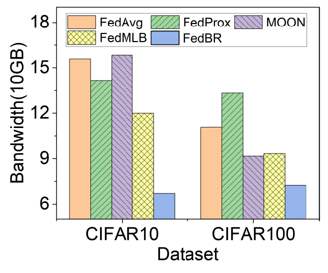
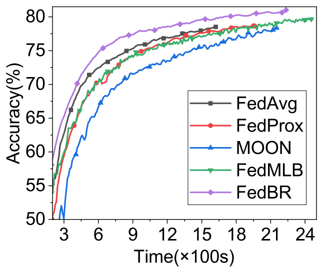
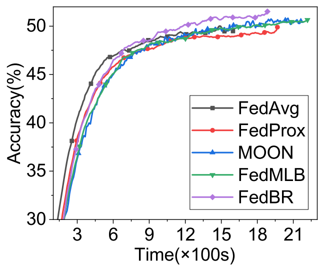



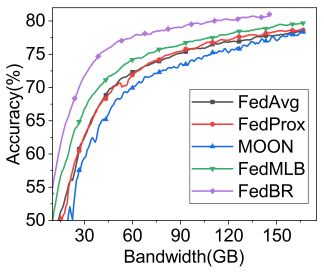
Fig. 5. Time and bandwidth consumption after the model achieves the target accuracy on CIFAR10 (78%) and CIFAR100 (49%).

Convergence Performance: We first show the testing performance (*e.g.*, accuracy and loss) of the global model trained on CIFAR10 and CIFAR100 when data distribution across the clients is non-IID (*e.g.*, $\epsilon = 7$). The detailed experimental results are shown in Fig. 3. Whether on CIFAR10 or CIFAR100, *FedBR* achieves the highest accuracy and the smallest loss, which indicates that *FedBR* is more effective than the baselines within non-IID data. For example, after 70 rounds of global training on CIFAR10 with non-IID level $\epsilon = 7$, the testing accuracy of *FedBR* is 77.0%, while the testing accuracy of *FedAvg*, *FedProx*, *MOON*, and *FedMLB* is 71.4%, 72.0%, 69.8%, and 74.0% respectively. According to the experimental results, we can see that compared with *FedMLB* using the same number of global model blocks to obtain global knowledge, determining the appropriate number of global model blocks for different clients in *FedBR* can better improve the generalization performance of the global model under the resource constraints.

Effectiveness of GMBS: To verify the effectiveness of the decision algorithm GMBS, we conduct comparative experiments between random decision-making and GMBS


(a) Testing accuracy with completion time constraint on CIFAR10


(b) Testing accuracy with completion time constraint on CIFAR100


(c) Testing accuracy with bandwidth constraint on CIFAR10

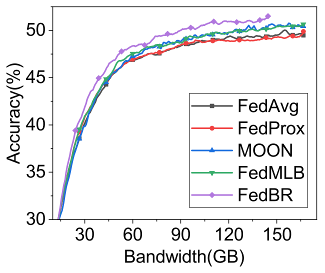

(d) Testing accuracy with bandwidth constraint on CIFAR100

Fig. 6. Testing accuracy with completion time and bandwidth constraints under the non-IID level $\epsilon = 7$.

decision-making, which are termed as *FedBR-Ran* and *FedBR* respectively, on CIFAR10 and CIFAR100. The experimental results are shown in Fig. 4, where the horizontal axis represents the number of global epochs and the vertical axis denotes the testing accuracy of the global model. The experimental results show that the testing accuracy of the model trained by *FedBR* is better than that of the model trained by *FedBR-Ran* on both CIFAR10 and CIFAR100. For example, given 200 training of global epochs on CIFAR100, the testing accuracy of *FedBR* is about 52.4%, while that of *FedBR-Ran* is about 49.5%. In other words, *FedBR* improves the accuracy performance by about 2.9% compares to *FedBR-Ran*. In addition, Fig. 4 reveals that *FedBR* converges faster than *FedBR-Ran*. Therefore, our proposed algorithm GMBS significantly shows the effectiveness compared with the random decision-making.

Resource Consumption: We evaluate the resource consumption (*e.g.*, time and bandwidth) of *FedBR* and the baselines to achieve a target testing accuracy (*e.g.*, 78% on CIFAR10 and 49% on CIFAR100), as shown in Fig. 5. For time cost, *FedBR* consumes the least time among all methods to train the model either on CIFAR10 or CIFAR100 according to the left plot of Fig. 5. This is well explained, although our *FedBR* will increase the time cost because of the calculation with the hybrid paths, it still accelerates model convergence by using block-wise regularization and knowledge distillation to obtain more global information of the global model. Besides, *FedBR* only requires the server to send a few consecutive global model blocks to the clients thus reducing the communication time compared to the three baselines. For example, when CIFAR100 is trained on ResNet18 under $\epsilon = 7$, *FedBR* takes 954 seconds, which is 10.4% less than *FedAvg* (1,065 seconds), 35.4% less than *FedProx* (1,479 seconds), 19.9% less than *MOON* (1,191 seconds), and 23.4% less than *FedMLB* (1,245 seconds). For communication cost, as shown in the right subplot of Fig. 5, itFedBR consumes the least communication bandwidth to train the model either on CIFAR10 or CIFAR100 when the global model achieves a target accuracy. This is because the four baselines transmit the entire model in one global round, while the server only sends partial model blocks to clients in *FedBR*. For instance, when CIFAR100 is trained on ResNet18 under $\epsilon = 7$, we obtain that the communication bandwidth consumed by *FedBR* is 72.5GB, which saves 34.6%, 47.6%, 20.9%, and 22.3% of bandwidth when *FedAvg* (110.8GB), *FedProx* (138.3GB), MOON (91.6GB), and *FedMLB* (93.3GB) achieve the same testing accuracy, respectively.

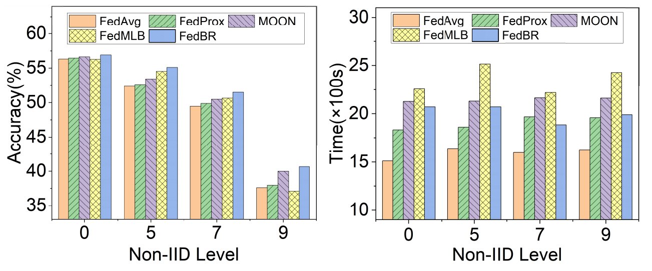
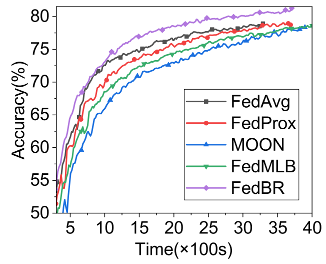


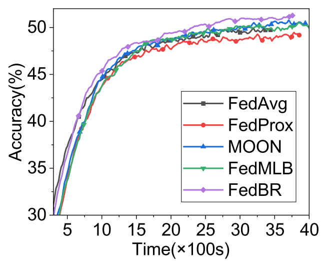
Fig. 7. Testing accuracy and time consumption on CIFAR100 under different non-IID levels.

Effect of Different non-IID Levels: Given a fixed number of global rounds (*e.g.*, 200), the testing accuracy and time consumption of the model training are shown in Fig. 7. According to the left plot of Fig. 7, under the IID setting, *FedBR* achieves nearly the same testing accuracy as the other three methods. However, *FedBR* can achieve the highest testing accuracy under the non-IID setting by introducing block-wise regularization and knowledge distillation techniques. The improvement of testing accuracy for *FedBR* over three baselines will increase with the increasing non-IID level. For example, under extremely non-IID settings (*i.e.*, $\epsilon = 9$), the testing accuracy of *FedBR* on CIFAR100 is 40.7%, while the testing accuracy of *FedAvg*, *FedProx*, *MOON*, and *FedMLB* is 37.6%, 37.9%, 39.9%, and 37.1%, respectively. Thus, *FedBR* effectively alleviates the problem of model performance degradation caused by the non-IID issue. For time consumption, as shown in the right subplot of Fig. 7, we can see that as the non-IID level increases, the time cost of *FedBR* relative to other methods will become smaller. The reason for this phenomenon may be that the server will send fewer global model blocks to clients under a higher non-IID level.

Impact of Resource Constraints: We conduct experiments to test the performance (*i.e.*, accuracy) of *FedBR* and three baselines with the resource (*e.g.*, completion time and bandwidth) constraints. For the completion time constraint, Fig. 6(a) and Fig. 6(b) show that the testing accuracy will increase with the increasing completion time budget on both CRFAR10 and CIFAR100. Note that *FedBR* can achieve the best training performance in four methods with the same completion time budget. For example, the testing accuracy of



(a) Testing accuracy on CIFAR10 under system heterogeneity.



(b) Testing accuracy on CIFAR100 under system heterogeneity.

Fig. 8. Testing accuracy on different datasets under system heterogeneity.

TABLE II

THE IMPACT OF THE TEMPERATURE δ WITH RESPECT TO THE TESTING ACCURACY OF *FedBR* ON CIFAR10 AND CIFAR100 AFTER 200 ROUNDS OF GLOBAL TRAINING (THE NON-IID LEVEL $\epsilon = 7$)

δ	0.75	1	1.25	1.5	2
CIFAR10	0.8056	**0.8101**	0.8126	0.8107	0.8101
CIFAR100	0.5087	**0.5151**	0.5112	0.5009	0.4982

FedBR is about 77.4% on CIFAR10 when given the completion time is 900s, while that of *FedAvg*, *FedProx*, *MOON*, and *FedMLB* is about 75.3%, 74.1%, 71.5%, and 74.0%, respectively. For bandwidth constraint, more communication bandwidth budget will significantly improve the performance of all methods according to Fig. 6(c) and Fig. 6(d). However, *FedBR* obtains better performance than *FedAvg*, *FedProx*, and *FedMLB* while consuming the same network bandwidth. For instance, when the total network bandwidth budget is set to 50GB, the testing accuracy of *FedBR* on CIFAR10 is approximately 76.8%, while that of *FedAvg*, *FedProx*, *MOON*, and *FedMLB* is approximately 70.8%, 70.3%, 68.3%, and 72.3%, respectively, as illustrated in the results.

Effect of System Heterogeneity: To demonstrate that our method is still effective under system heterogeneity, we conduct experiments on both CIFAR10 and CIFAR100 under the same non-IID level $\epsilon = 7$ as the homogeneous systems. Compared with the previous situation without system heterogeneity shown in Fig. 6(a) and Fig. 6(b), no matter whether on CIFAR10 or CIFAR100, all methods will consume more completion time when achieving a target test accuracy as shown in Fig. 8. For instance, given the target accuracy of 78%, *FedAvg*, *FedProx*, *MOON*, *FedMLB*, and *FedBR* needs to consume 1,516s, 1,668s, 2,120s, 1,698s, and 1,034s in the case of homogeneous system, respectively; while *FedAvg*, *FedProx*, *MOON*, *FedMLB*, and *FedBR* needs to consume 2,534s, 3,091s, 3,653s, 3,552s, and 1,782s in the case of heterogeneous systems, respectively. As a result, the existence of system heterogeneity will significantly degrade the model training performance. Besides, according to Fig. 8, our *FedBR* achieves the highest testing accuracy on different datasets while consuming the same completion time compared to the baselines. Therefore, *FedBR* still can obtain the best training performance compared to the baselines even under heterogeneous settings.

TABLE III

THE IMPACT OF THE WEIGHTS OF THE REGULARIZATION TERM IN THE LOSS FUNCTION EQ. (18) WITH RESPECT TO THE TESTING ACCURACY OF *FedBR* ON CIFAR100 AFTER 200 ROUNDS OF GLOBAL TRAINING (THE NON-IID LEVEL $\epsilon = 7$)

λ_1 \ λ_2	0	1	1.5	2
0	49.58%	50.76%	51.04%	50.93%
1	50.18%	**51.51%**	50.21%	50.37%
1.5	50.21%	50.53%	50.46%	51.43%
2	49.95%	51.22%	50.71%	50.61%

TABLE IV

THE IMPACT OF THE INTERVAL τ WITH RESPECT TO THE TESTING ACCURACY OF *FedBR* ON CIFAR10 AND CIFAR100 AFTER 200 ROUNDS OF GLOBAL TRAINING (THE NON-IID LEVEL $\epsilon = 7$)

τ	1	2	3	4	5
CIFAR10	**0.8101**	0.7935	0.8001	0.7966	0.7928
CIFAR100	**0.5151**	0.4942	0.4865	0.4817	0.4844

TABLE V

THE IMPACT OF THE INTERVAL τ WITH RESPECT TO THE BANDWIDTH COST OF *FedBR* ON CIFAR100 AFTER 200 ROUNDS OF GLOBAL TRAINING (THE NON-IID LEVEL $\epsilon = 7$)

τ	1	2	3	4	5
bandwidth (GB)	**145.1**	139.3	135.2	137.9	131.8

Effect of Temperature δ: The hyperparameter temperature δ in the KL divergence (*i.e.*, knowledge distillation function) in Eq. (17) will affect the KL term in the loss function Eq. (18). We test the performance of *FedBR* by using different temperature values. Table II shows the impact of different temperature values on the testing accuracy of the *FedBR* on two different datasets, *i.e.*, CIFAR10 and CIFAR100. We can see that the performance of the model will be improved if we raise the temperature a little. However, if we increase the temperature too large, the performance of the model will be decreased. Therefore, we set the default temperature value δ to 1 according to the results of the experiments.

Impact of Weights λ_1 **and** λ_2: To prove the effectiveness of the two regularization terms that we add to the loss function in Eq. (18), we conduct experiments to record the testing accuracy under different weight combinations of λ_1 and λ_2. According to Table III, it can be illustrated that both loss function terms ζ_H and ζ_{KL} in Eq. (18) improve the performance of the global model compared to directly using only the cross entropy term ζ_L in the loss function. For example, if we do not add ζ_H and ζ_{KL} to the loss function (*i.e.*, $\lambda_1 = 0$ and $\lambda_2 = 0$), the testing accuracy of the model is 49.6%. When we set the weights $\lambda_1 = 1$ and $\lambda_2 = 1$, the testing accuracy of the model is 51.5%. Note that we set the two default regularization term weights to 1 in the experiments (*i.e.*, $\lambda_1 = 1$ and $\lambda_2 = 1$).

Effect of Interval τ: The interval τ indicates that the server distributes the entire global model to the clients every τ global rounds. We explore the effect of different interval values τ with respect to the training performance of model on CIFAR100 as shown in Table IV. It can be seen that if we set a relatively large τ, the performance of *FedBR* will degrade. Besides, according to Table V, when we set a relatively large τ, the bandwidth consumption of *FedBR* will be reduced after 200 rounds of global training. For example, when we set τ to 1 on CIFAR100 under the non-IID level $\epsilon = 7$, the testing accuracy of the model is 0.515 and the bandwidth consumption is about 145.1 GB. When we set τ to 5, the testing accuracy of the model is 0.484 and the bandwidth consumption is about 131.8 GB. Therefore, we can adjust τ according to whether the performance of the model is strict or the bandwidth requirement is strict in the network. It is noted that we set τ to 1 in the rest of the experiments.

To summarize, *FedBR* substantially outperforms the three baselines. Firstly, our *FedBR* significantly alleviates the non-IID issue under resource constraints and system heterogeneity, especially in extreme non-IID cases. Secondly, *FedBR* outperforms the baselines in terms of the generalization performance of the global model, such as testing accuracy, while requiring less resource consumption, including network bandwidth and completion time. According to the experimental results, *FedBR* achieves the highest accuracy under the same network bandwidth consumption and completion time compared to the three baselines.

VI. Related Works

Federated learning (FL) [5] has gained significant attention and has been extensively studied as a promising approach to train machine learning models in edge computing.

One main area of research is the most common non-IID data issue in federated learning. Shen et al. [40] propose an agnostic constrained learning formulation to deal with the class imbalance problem in FL. Collins et al. [41] propose that the server only aggregates the representation layer of the model and the clients learn their own heads for personalization to better fit their local data. Oh et al. [42] advocate that clients only update the parameters of the model body during the model training with a fixed random model head. The head will be fine-tuned for personalization after the model training. Chen et al. [43] enable the global model to obtain good generalization ability and fit the non-IID local data of clients well by adjusting the local loss function of the clients. Shamsian et al. [44] design a hypernetwork for the server, which generates a personalized subnetwork (local model) for the client to better fit its local data distribution. Li et al. [20] apply regularization to *FedAvg* effectively alleviates the impact of non-IID data by making the local model close to the global model. Karimireddy et al. [45] propose *SCAFFOLD*, which continuously corrects the model update direction by adding a correction item in each local iteration to prevent the update direction from being biased. Acar et al. [46] dynamically change the local objective in each global epoch, called *FedDyn*, to ensure that the local optimum is consistent with the stagnation point of the global objective. Li et al. [38] propose *MOON*, which exploits the similarity between model representations to correct individual local training. Although these methods can alleviate the impact of non-IID data, most of them will introduce huge computing costs when performing Local Updating. Meanwhile, the entire model is still transmitted between the server and the clients in these methods, leading to massive communication costs. As a result, they perform poorly in resource-constrained (*e.g.*, computing and communication resources) networks.

To alleviate the impact of different computing and communication capabilities among the clients (*i.e.* system heterogeneity) on the model training performance, some related works has been proposed by researchers. Ma et al. [47] design an adaptive method to select different learning rates and batch sizes for clients according to their varied computing and communication capabilities. Xu et al. [48] propose to support running different numbers of local iterations according to clients' heterogeneous capabilities. Nishio et al. [49] apply the client selection mechanism to heterogeneous federated learning scenarios based on clients' properties. Although these methods can alleviate the negative impact of system heterogeneity to some extent, the whole model is still communicated between the server and the clients. As a result, these methods cannot effectively improve communication efficiency and cannot be applied to scenarios with limited resources.

Asynchronous federated learning [50], [51], [52] aims to reduce communication costs by allowing the server to perform global aggregation only after receiving a local model uploaded by a specified client. However, this asynchronous method prevents the server from aggregating local models of all clients, so the global model cannot fit the overall data distribution well [10]. Furthermore, another commonly used technique in distributed machine learning or FL is model compression, which can further reduce communication cost during model training [53]. Model or gradient compression techniques reduces the communication data volume during training by transmitting a compact model update instead of a whole model between the server and the clients, which can be achieved through the use of sparsification schema [54] or quantization operator [55]. For the sparsification schema, Stich et al. [54] introduce error compensation to improve training performance with model compression by keeping track of accumulated errors in memory. For the quantization technique, Basu et al. [55] proposed to use a quantizer (either stochastic or deterministic 1-bit sign) in combination with sparsification and error compensation to achieve further model compression and reduce communication resources. However, model compression will damage the model accuracy and cause more computing overhead to achieve the target testing accuracy.

Applying knowledge distillation technology to federated learning can effectively reduce network bandwidth and deal with system heterogeneity by switching logit outputs instead of models between server and clients [56], [57]. Besides, providing a public unlabeled dataset as a proxy dataset, knowledge distillation alleviates the negative effect under the non-IID setting by enriching the global model with the integrated knowledge of local models, which is more effective than simple parameter averaging. However, the clients need to access public unlabeled proxy data, which is often difficult to achieve in real-world scenarios [58]. Data-free knowledge distillation is proposed to avoid the use of public unlabeled proxy datasets via a small data generator [59], [60]. Unfortunately, the generation of the data generator requires the label information of the clients, which maybe leak the privacy of the clients. Although *FedBR* we proposed also adopts knowledge distillation, it avoids the use of additional data since the block-wise regularization method enables one input to generate multiple outputs. *FedBR* combines knowledge distillation and regularization so that it deals with the heterogeneous issues effectively while significantly saving network resources. Therefore, the model trained by *FedBR* also obtains good performance even under heterogeneous settings and resource constraints.

VII. Conclusion

In this work, we have proposed the *FedBR* framework, which integrates the idea of block-wise regularization and knowledge distillation into *FedAvg*, to deal with statistical heterogeneity and system heterogeneity for resource-constraint edge computing. We have designed a heuristic algorithm (GMBS) to adaptively determines the number of global model blocks for each client. We have built a simulation environment and evaluated the performance of *FedBR*. The experimental results indicate that *FedBR* is effective in enhancing model accuracy while reducing resource consumption (*e.g.*, network bandwidth and completion time).

References

[1] Q. Zeng, J. Liu, H. Xu, Z. Wang, Y. Xu, and Y. Zhao, "Enhanced federated learning with adaptive block-wise regularization and knowledge distillation," in *Proc. IEEE/ACM 31st Int. Symp. Quality Service (IWQoS)*, Jun. 2023, pp. 1–4.

[2] K. L. Lueth et al., "State of the IoT 2018: Number of IoT devices now at 7B—Market accelerating," *IoT Analytics*, vol. 8, pp. 135–139, Jan. 2018.

[3] G. Gao, M. Xiao, J. Wu, H. Huang, S. Wang, and G. Chen, "Auction-based VM allocation for deadline-sensitive tasks in distributed edge cloud," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 1702–1716, Nov. 2021.

[4] W. Huang, M. Ye, Z. Shi, H. Li, and B. Du, "Rethinking federated learning with domain shift: A prototype view," in *Proc. CVPR*, 2023, pp. 1–10.

[5] B. M. Mahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (PMLR)*, 2017, pp. 1273–1282.

[6] Y. Liao, Y. Xu, H. Xu, L. Wang, and C. Qian, "Adaptive configuration for heterogeneous participants in decentralized federated learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2023, pp. 13903–13905.

[7] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2016, pp. 20–26.

[8] J. Liu, Y. Xu, H. Xu, Y. Liao, Z. Wang, and H. Huang, "Enhancing federated learning with intelligent model migration in heterogeneous edge computing," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 1586–1597.

[9] W. Huang, M. Ye, and B. Du, "Learn from others and be yourself in heterogeneous federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10133–10143.

[10] Z. Wang, H. Xu, J. Liu, Y. Xu, H. Huang, and Y. Zhao, "Accelerating federated learning with cluster construction and hierarchical aggregation," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3805–3822, Jul. 2023.

[11] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[12] X. Fang and M. Ye, "Robust federated learning with noisy and heterogeneous clients," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10062–10071.

[13] K. Bonawitz et al., "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, vol. 1, 2019, pp. 374–388.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[15] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 7, 2020, pp. 13001–13008.

[16] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," 2020, *arXiv:2010.13723*.

[17] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Comput.*, vol. 7, no. 2, pp. 219–269, Mar. 1995.

[18] M. Duan et al., "Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications," in *Proc. IEEE 37th Int. Conf. Comput. Design (ICCD)*, Nov. 2019, pp. 246–254.

[19] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2022, pp. 1739–1748.

[20] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[21] J. Kim, G. Kim, and B. Han, "Multi-level branched regularization for federated learning," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2022, pp. 11058–11073.

[22] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[23] A. Mora, I. Tenison, P. Bellavista, and I. Rish, "Knowledge distillation for federated learning: A practical guide," 2022, *arXiv:2211.04742*.

[24] 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 *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 5972–5984.

[25] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *IEEE Trans. Signal Process.*, vol. 70, pp. 1142–1154, 2022.

[26] W. Huang, G. Wan, M. Ye, and B. Du, "Federated graph semantic and structural learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2023, pp. 139–143.

[27] J. Liu, H. Xu, G. Zhao, C. Qian, X. Fan, and L. Huang, "Incremental server deployment for scalable NFV-enabled networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 2361–2370.

[28] N. Ivkin et al., "Communication-efficient distributed SGD with sketching," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1930–1935.

[29] J. Liu et al., "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 674–690, Feb. 2023.

[30] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2021, pp. 1–10.

[31] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1839–1843.

[32] J. M. Joyce, "Kullback-leibler divergence," in *International Encyclopedia of Statistical Science*. Cham, Switzerland: Springer, 2011, pp. 720–722.

[33] D. A. E. Acar et al., "Debiasing model updates for improving personalized federated training," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2021, pp. 21–31.

[34] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. neural Inf. Process. Syst.*, vol. 32, 2019, pp. 103–105.

[35] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 1698–1707.

[36] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[38] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10708–10717.

[39] N. Al-Falahy and O. Y. Alani, "Technologies for 5G networks: Challenges and opportunities," *IT Prof.*, vol. 19, no. 1, pp. 12–20, Jan. 2017.

[40] Z. Shen, J. Cervino, H. Hassani, and A. Ribeiro, "An agnostic approach to federated learning with class imbalance," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1038–1042.

[41] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2021, pp. 2089–2099.

[42] J. Oh, S. Kim, and S.-Y. Yun, "Fedbabu: Toward enhanced representation for federated image classification," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 294–296.

[43] H.-Y. Chen and W.-L. Chao, "On bridging generic and personalized federated learning for image classification," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 392–397.

[44] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, "Personalized federated learning using hypernetworks," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2021, pp. 9489–9502.

[45] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2020, pp. 5132–5143.

[46] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," 2021, *arXiv:2111.04263*.

[47] Z. Ma, Y. Xu, H. Xu, Z. Meng, L. Huang, and Y. Xue, "Adaptive batch size for federated learning in resource-constrained edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 37–53, Jan. 2023.

[48] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, and J. Liu, "Adaptive control of local updating and model compression for efficient federated learning," *IEEE Trans. Mobile Comput.*, early access, Jun. 28, 2022, doi: 10.1109/TMC.2022.3186936.

[49] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[50] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.

[51] M. R. Sprague et al., "Asynchronous federated learning for geospatial applications," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2018, pp. 21–28.

[52] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-IID data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 15–24.

[53] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1394–1398.

[54] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 123–128.

[55] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 145–150.

[56] K. Ozkara, N. Singh, D. Data, and S. Diggavi, "Quped: Quantized personalization via distillation with applications to federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 3622–3634.

[57] D. Yao et al., "Local-global knowledge distillation in heterogeneous federated learning with non-IID data," 2021, *arXiv:2107.00051*.

[58] M. Chen et al., "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3579–3605, Dec. 2021.

[59] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2021, pp. 12878–12889.

[60] L. Zhang, L. Shen, L. Ding, D. Tao, and L.-Y. Duan, "Fine-tuning global model via data-free knowledge distillation for non-iid federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, May 2022, pp. 10174–10183.

Jianchun Liu (Member, IEEE) received the Ph.D. degree from the School of Data Science, University of Science and Technology of China, in 2022. He is currently an Associate Researcher with the School of Computer Science and Technology, University of Science and Technology of China. His main research interests are software defined networks, network function virtualization, edge computing, and federated learning. He is a member of ACM.

Qingmin Zeng received the B.S. degree from Zhejiang Gongshang University in 2021. He is currently pursuing the master's degree with the School of Computer Science and Technology, University of Science and Technology of China (USTC). His main research interests are edge computing and federated learning.

Hongli Xu (Member, IEEE) received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China (USTC), China, in 2002 and 2007, respectively. He is currently a Professor with the School of Computer Science and Technology, USTC. He has published more than 100 papers in famous journals and conferences, including the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Infocom, and ICNP. He has also held more than 30 patents. His main research interests are software defined networks, edge computing, and the Internet of Things. He was awarded the Outstanding Youth Science Foundation of NSFC in 2018. He has won the best paper award or the best paper candidate in several famous conferences.

Yang Xu (Member, IEEE) received the B.S. degree from the Wuhan University of Technology in 2014 and the Ph.D. degree in computer science and technology from the University of Science and Technology of China in 2019. He is currently an Associate Researcher with the School of Computer Science and Technology, University of Science and Technology of China. His research interests include ubiquitous computing, deep learning, and mobile edge computing.

Zhiyuan Wang received the B.S. degree from Jilin University in 2019. He is currently pursuing the master's degree with the School of Computer Science, University of Science and Technology of China (USTC). His main research interests are edge computing, deep learning, and federated learning.

He Huang (Senior Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, University of Science and Technology of China (USTC), in 2011. He is currently a Professor with the School of Computer Science and Technology, Soochow University, China. His current research interests include traffic measurement, computer networks, and algorithmic game theory. He is a member of ACM.