Fed-RAA: Resource-Adaptive Asynchronous Federated Edge Learning with Theoretical Guarantee

Ruirui Zhang, Xingze Wu, Yifei Zou, *Member, IEEE*, Zhenzhen Xie, *Member, IEEE*, Peng Li, *Senior Member, IEEE*, Xiuzhen Cheng, *Fellow, IEEE*, Falko Dressler, *Fellow, IEEE*, and Dongxiao Yu, *Senior Member, IEEE*,

***Abstract*—This paper studies an efficient federated learning (FL) problem involving multiple edge-based clients with heterogeneous constrained resources. Compared with numerous training parameters, the computing and communication resources of clients in edge scenarios are usually insufficient for fast local training and real-time knowledge sharing. Besides, training on clients with heterogeneous resources may result in the straggler problem, which delays the convergence of FL. To address these issues, we propose Fed-RAA: a Resource-Adaptive Asynchronous Federated learning algorithm. Different from vanilla FL methods, where all parameters are trained by each participating client regardless of resource diversity, Fed-RAA adaptively allocates submodels of the global model to clients based on their computing and communication capabilities. Each client then individually trains its assigned submodel and asynchronously uploads the updated result. Theoretical analysis confirms the convergence of our approach. Additionally, an online greedy-based algorithm is designed for asynchronous submodel assignment in Fed-RAA, improving the convergence of Fed-RAA by optimal minimization on the training delay bound of submodels. Compared to state-of-the-art methods, our Fed-RAA algorithm reduces the time required to achieve the target accuracy by an average of 30.89%, demonstrating its superior efficiency on heterogeneous constrained computing and communication resources. To the best of our knowledge, this paper is the first resource-adaptive asynchronous method for submodel-based FL with guaranteed theoretical convergence.**

***Index Terms*—Asynchronous Federated learning, Resource heterogeneity, Resource-constrained, Resource adaptive federated learning**

I. Introduction

Federated edge learning (FEEL) enables multiple clients to collaboratively train machine learning (ML) models on the edge side. This approach not only provides users with personalized and real-time inference services but also strengthens user privacy, since raw data is not transferred to the cloud [1]–[3]. Consider that ML models often consist of billions or even trillions of parameters [4], [5], which are time-consuming or even unaffordable for the edge-side clients[1] with limited computing and communication resources to locally train and transmit in FEEL. How to improve the efficiency of edge resources in FEEL for fast model training becomes important to implement the ML models in real-time decision-making scenarios, such as healthcare [6] and autonomous driving [7].

Due to its significance, recent works have been proposed, including [8], [9] improving the computing efficiency, [10]–[16] reducing the communication cost, and [17]–[20] optimizing both. For instance, in HeteroFL [17] and PruneFL [18], the global ML model is shrunk or pruned into smaller submodels, respectively. These submodels are transmitted through the edge network and trained by the clients, improving both computational and communication efficiency compared to vanilla federated learning methods based on full-model training.

Even with the existing works mentioned above, improving the computing and communication efficiencies of FEEL among multiple clients with heterogeneous resources is not an easy task. The limited computing and networking resources directly undermine the strategies adopted in [8]–[16], in which the full ML models are directly transmitted through the edge networks or trained by the clients. Moreover, all the methods mentioned above are designed under a synchronous framework. Since faster clients must wait until the slowest completes its task during each local training step in the synchronous setting, the computing resources of faster clients are not fully utilized and the slowest client becomes a bottleneck on efficiency, known as the straggler problem [24]. Furthermore, synchronous global model dissemination and local model aggregation lead to the communication contention problem [25]: the edge network spikes when all clients download/upload models synchronously while it is vacant in the local training steps. Without careful consideration, the struggler problem and the network contention problem reduce the convergence speed of FEEL.

To make full use of computing and communication resources in edge-based clients and improve the convergence speed of FEEL, this paper investigates the resource-efficient FEEL problem in an asynchronous setting and proposes a **R**esource-**A**daptive **A**synchronous **Fed**erated learning algorithm, named **Fed-RAA**. Specifically, considering the heterogeneous and constrained computing and communication resources of various clients, our method deploys a full global model on the PS but only designates submodels to clients for local training, which are tailored from the full model according to the computing and communication capabilities of clients. For each client, once it receives a submodel from the PS, it trains the submodel using its local data. Considering the disadvantage of synchronous setting, the asynchronous aggregation is adopted in Fed-RAA, i.e. once a client uploads its submodel, the parameters of the

Ruirui Zhang, Xingze Wu, Yifei Zou, Zhenzhen Xie, Xiuzhen Cheng, and Dongxiao Yu are with the School of Computer Science and Technology, Shandong University, Qingdao, Shandong 266237, China. E-mail: {sherryz, wuxingze}@mail.sdu.edu.cn,{yfzou, xiezz21, xzcheng, dxyu}@sdu.edu.cn

Peng Li is with the Department of Computer Science, Xi'an Jiaotong University, China. E-mail: pengli@xjtu.edu.cn

Falko Dressler is with the School of Electrical Engineering and Computer Science, TU Berlin, Berlin, 10587, Germany. E-mail: dressler@ccs-labs.org

(Corresponding author: Yifei Zou.)

[1]For simplicity, the *clients* in the remaining of this paper indicates *edge-side clients* unless otherwise specified.

TABLE I
COMPUTATION AND COMMUNICATION EFFICIENCY OF VARIOUS FEDERATED LEARNING METHODS

Methods	Global Model on Parameter Server	Local Training on Clients	Download/Upload between Parameter Server and Clients	Efficiency		Async.
				Comp.	Comm.	
[8] (SplitFed)	submodel	submodel	submodel and hidden states	yes	no	no
[9] (RAM-Fed)	full model	submodel	full model/submodel	yes	no	no
[10], [11] (OMC, DGC)	compressed model	full model	compressed model	no	yes	no
[12]–[14] (FedSQ, QSGD, Atomo)	quantized gradients	full model	quantized gradients	no	yes	no
[15] (FedScalar)	scaled model	full model	scaled model	no	yes	no
[16] (FedSL)	full model	full model	submodel	no	yes	no
[17]–[19] (IST, PruneFL, HeteroFL)	submodel	submodel	submodel	yes	yes	no
[21]–[23] (FJORD, FedRolex, NeFL)	submodel	submodel	submodel	yes	yes	no
Fed-RAA	**submodel**	**submodel**	**submodel**	**yes**	**yes**	**yes**

The methods are grouped by their primary techniques: **Split Methods** (e.g., SplitFed, FedSL), **Compression-based Methods** (e.g., OMC, FedSQ), **Scaling Methods** (e.g., FedScalar), **Submodel and Pruning-based Methods** (e.g., RAM-Fed, IST, FJORD), and **Our Approach** (Fed-RAA).
Each method is compared on its computation (Comp.), communication (Comm.), and asynchronous (Async.) settings.

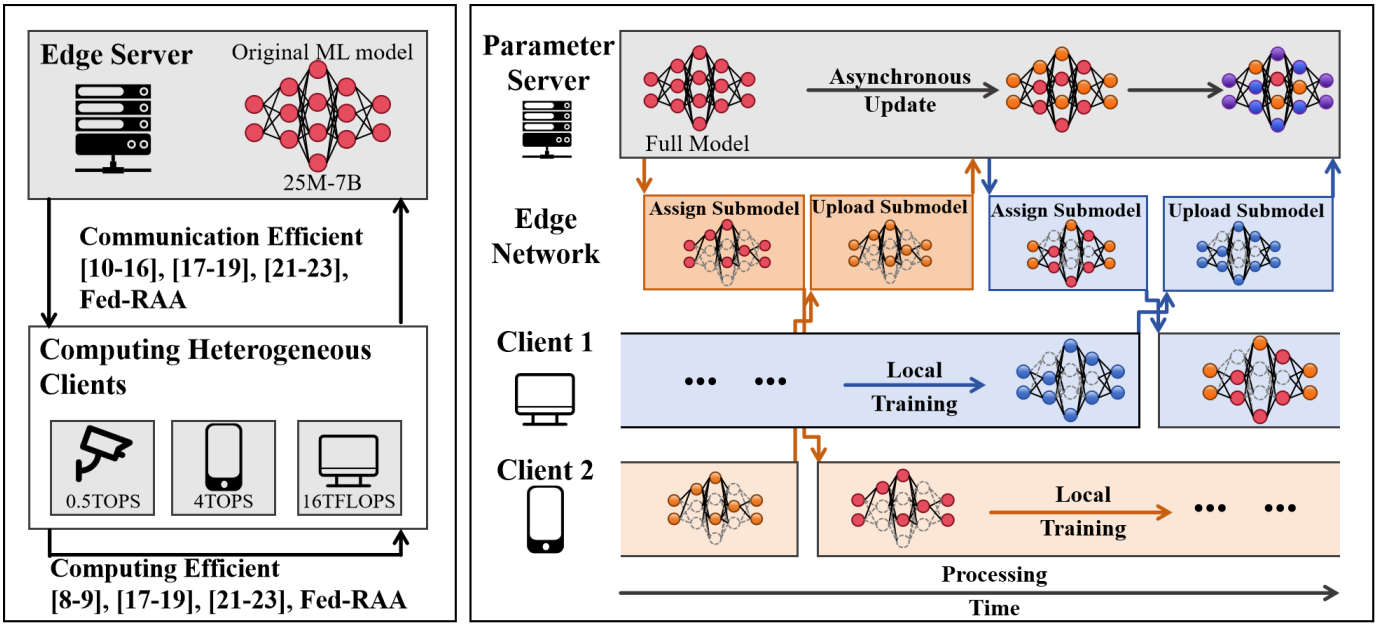


Fig. 1. Flowchart of Fed-RAA. In FL algorithms, [8], [9] improve the computing efficiency, [10]–[16] reduce the communication cost, and [17]–[19], [21]–[23] optimizing both in the synchronous setting. Fed-RAA improves the computing and communication efficiency of FL in the asynchronous setting, which is more efficient. In particular, Fed-RAA asynchronously assigns different submodels to heterogeneous clients. After receiving the submodels, clients train them locally and upload the updated ones via the edge network to the parameter server.

submodel are immediately updated to the global model, and the new submodel is tailored and disseminated. We illustrate the flowchart of Fed-RAA in Fig. 1 and compare it with previous works in Tab. I.

Even though our resource-adaptive asynchronous strategy improves the computing and communication efficiencies of FEEL, two key technical questions may arise. The first question is the convergence of the new FEEL algorithm in which the submodels are asynchronously assigned, locally trained, and aggregated. The absence of a synchronous global model undermines the existing convergence analysis under asynchronous setting [26]. Even for two clients uploading their submodels simultaneously, their newly assigned submodels may differ since they have different computing and communication capabilities. Secondly, in existing resource-efficient methods based on submodel training, those submodels are synchronously assigned to all the clients at the beginning of each training step. Whereas, in our asynchronous setting, once a client uploads its local submodel, a new tailored submodel is immediately assigned, which involves an online decision-making problem ensuring efficient resource utilization on parameters updating. In this paper, we answer the first question by providing a theoretical analysis of the convergence of our Fed-RAA even when only submodels are asynchronously aggregated and solve the second question by designing an online submodel assignment strategy that ensures optimal performance on improving the convergence.

Our main contributions can be summarized as follows:

- This paper studies the computing and communication efficient federated edge learning problem under an asynchronous setting. Unlike the existing resource-efficient methods under the synchronous setting, the asynchronous setting in our work can solve the well-known straggler problem and communication contention problem under

the synchronous setting and further improve the efficiency of FEEL. As a trade-off, it results in a harder convergence analysis on the asynchronous submodel training and aggregating, and an online submodel assignment problem that involves the convergence of FEEL.
- We propose **Fed-RAA**, **R**esource-**A**daptive **A**synchronous **Fed**erated edge learning algorithm on edge-based clients with limited computing and communication resources. Instead of transmitting and training the full ML model on individual clients, Fed-RAA adaptively assigns submodels to the clients for local training under the asynchronous setting. Specifically, once a client completes its local training, its submodel will be aggregated to the global model and a new submodel will be assigned immediately according to its computing and communication capability. Rigorous theoretical analyses are presented to prove the convergence of Fed-RAA.
- A greedy-based strategy is designed as a component of Fed-RAA, to address the online submodel assignment problem. We prove that our online greedy-based strategy has an optimal performance on minimizing the training delay bound of submodels, which improves the convergence of Fed-RAA.

Numerical results with necessary comparisons and ablation studies are presented, showing that our Fed-RAA has promising performance in accuracy and time cost. In comparison with state-of-the-art approaches, our Fed-RAA algorithm achieves a reduction in the time needed to reach the target accuracy by an average of 30.89%, highlighting its superior efficiency on heterogeneous and resource-constrained computing and communication environments.

Roadmap. The rest of this paper is organized as follows. Section II reviews related work on resource-efficient federated learning. Section III outlines the formal problem setting and the necessary background. Section IV describes the Fed-RAA algorithm and its theoretical analysis on convergence. Section V introduces a greedy-based algorithm for online submodel assignment. Numerical results are reported in Section VI. Finally, Section VII concludes the paper.

II. Related Work

To federated learn a large ML model on resource-constrained clients, improving the computing and communication efficiencies is much of important for fast and accurate convergence. In the following, we detail the related works on the optimization of computing and communication efficiency.

- **Computing Efficiency.** SplitFed [8] splits the global model into the client-side model and server-side model. Each client only trains the client-side model, which reduces the computing cost of clients. However, in SplitFed, each client not only has to upload the client-side model but also exchange intermediate results with the server during each forward and backward pass, allowing the server-side model to continue training. In [9], the global neural network on the PS is decomposed into a collection of disjoint subnetworks, which are assigned to clients for local training according to their local computing resources. Since the global model is required for local training, the entire ML model must be transmitted between the PS and all clients in each training round, leading to significant network strain.
- **Communication Efficiency.** Techniques such as model compression [10], [11], [27], sparsification [12], quantization [13], [14], and scaling [15] are proposed to address communication overhead in FL. For example, Yang *et al.* [10] introduce an online model compression techique, to transmit model parameters in a compressed format and decompress them on clients for local training. DGC [11] combines gradient compression and multiple optimization techniques to reduce communication costs with comparable accuracy. FedSQ [12] combines gradient sparsity, quantization, and error compensation to reduce communication costs while maintaining comparable accuracy. The works in [13], [14] employ model quantization techniques, transmitting quantized models during communication and performing dequantization during local training to maintain accuracy, thereby reducing communication costs between the PS and clients. However, these methods typically incur a high computational cost since all the parameters of the ML model are updated by the clients in local training.
- **Computing and Communication Efficiency.** HeteroFL [17] decomposes the large full model into multiple submodels, which are disseminated and trained by the clients synchronously in the FL process. Moreover, PruneFL [18] adapts the model size during FL to reduce both communication and computation overhead and minimize the overall training time. FJORD [21] introduces the Ordered Dropout technique to dynamically adapt submodel size for heterogeneous FL clients, enhancing computational efficiency and enabling participation from devices with varying resources. FedRolex [22] proposes a rolling submodel extraction scheme to ensure that all the parameters of the global server model are evenly trained over the local data of client devices, while NeFL [23] divides deep neural networks into submodels through both depthwise and widthwise scaling. IST [19] partitions fully connected neural networks into independent submodels, which are trained separately across workers to overcome computing and communication challenges.

However, the aforementioned works are built upon a synchronous setting. In this setup, the efficiency of FL in each training step is constrained by the slowest client, often referred to as the straggler problem [24]. Additionally, the synchronous model dissemination and aggregation between the PS and clients give rise to the communication contention issue [25]: the edge network experiences congestion when all clients download/upload ML models simultaneously, while the network remains underutilized during the local training steps. Different from the existing works, this paper investigates the asynchronous resource-efficient FEEL problem, in which the submodels are online assigned to the clients for resource-adaptive local training and aggregated asynchronously for efficient convergence.

TABLE II
TABLE OF NOTATIONS

Symbol	Meaning
M	The number of submodels divided from the global model
N	The number of clients
θ	Global model, consisting of submodels $\theta^1, \theta^2, \ldots, \theta^M$
θ^j	The j-th submodel
D^n	The local dataset at client n
$f_n(\theta)$	The local loss function at client n
ξ_n	Sample data at client n
p_n	The computation power of client n
b_n	The bandwidth between client n and the PS
$c_{n,j}$	The delay time for training submodel θ^j at client n
$I(n)$	The number of local training iterations at client n
α	Hyperparameter controlling the model update ratio
γ	Learning rate
$q(j)$	The number of updates of submodel θ^j
$s(t,\hat{t})$	Function of staleness

III. System Model and Problem Definition

We consider an FEEL system consisting of a PS and N clients on the edge side with reliable communication. The PS holds an ML model θ which serves as the global model in FEEL. Each client n ($n = 1, \ldots, N$) holds its own local dataset D^n of size $|D^n|$, with computation power denoted as p_n and bandwidth to the PS denoted as b_n. The computation power p_n typically refers to the amount of computation the client can perform per second, expressed in terms of FLOPS (floating-point operations per second). Moreover, the time required for client n to upload or download a unit of data to/from the PS can be expressed as $1/b_n$.

Global Model Partition. As discussed in the *Related Work Section*, methods for partitioning the global model into multiple submodels have been previously explored in works such as [17], [19], [28]. However, none of them considers the asynchronous setting and its associated online submodel assignments. In this paper, we investigate the asynchronous FEEL problem with the global model partitioning technique as the background. Specifically, we assume that the global model θ has been divided into M submodels, denoted by $\{\theta^1, \theta^2, \ldots, \theta^M\}$, where each submodel corresponds to a subset of the model parameters. The partitioning of the global model can be done in various ways (e.g., width/depth scaling, channel/block grouping, or structured pruning) [17], [18], [21]–[23]. An additional requirement is that all the submodels must together cover the entire parameter space of the global model.

Asynchronous FEEL Setting. For each client n, once a submodel θ^j is assigned, n starts local training on θ^j based on its local dataset D^n. The updated submodel θ^j will be uploaded to the PS after the local training is completed. Upon receiving the updated submodel θ^j, PS aggregates θ^j into the global model θ and a new submodel $\theta^{j'}$ will be assigned to client n for the next round of local training. Both parameters N and M can vary from 2 to arbitrarily large values, and we assume that $N > M$. The submodel assignment, local training, and uploading process are repeated until the predesigned training rounds or accuracy is achieved. The above asynchronous setting enables non-blocking and independent local training on clients and submodel uploading/assignment on the edge network, which improves the efficiency of FEEL.

Objective of Asynchronous FEEL Problem. The objective of the system is to minimize the global loss function through empirical risk minimization, expressed as:

$$\min_{\theta \in \mathbb{R}^d} F(\theta) := \frac{1}{N} \sum_{n=1}^{N} f_n(\theta),$$

where $f_n(\theta) := \mathbb{E}_{\xi_n \sim D_n}[f_n(\theta, \xi_n)]$ represents the local loss function of client n with respect to its local dataset D_n.

IV. Algorithm Design for Fed-RAA

In this section, we design the resource-adaptive asynchronous FEEL algorithm, named Fed-RAA, in which the submodels $\{\theta^1, \theta^2, \ldots, \theta^M\}$ are asynchronously assigned/aggregated and locally trained by the clients.

A. Algorithm Description

In Fed-RAA, the PS holds a global model θ that has been divided into submodels of $\{\theta^1, \theta^2, \ldots, \theta^M\}$. The execution of Fed-RAA is divided into successive epochs with $t = 0, 1, \ldots$. Initially, at $t = 0$, the PS assigns submodels θ^j and a time stamp of 0 to each client n. The submodel assignment, i.e., which submodel will be assigned to client n, is determined by a Model-Assignment-Function (θ) executed by the PS. How to optimize the Model-Assignment-Function to improve the convergence of Fed-RAA is presented in the next section. In this section, we mainly focus on the convergence of Fed-RAA with a given Model-Assignment-Function, even if it works with a random assignment strategy.

For clarity, we use θ^j to denote the submodel held by the PS and $\hat{\theta}^j$ to denote the corresponding local model trained by a client if θ^j was assigned. $\hat{\theta}^j_i$ is the submodel trained by a client in its i-th iteration. In the following epochs, once the PS assigns its submodel θ^j to client n at epoch $\hat{t}$ and receives the updated result $\hat{\theta}^j$ from client n at epoch t, it updates the submodel θ^j using the following rules:

$$\theta^j \leftarrow [1 - \alpha \cdot s(t, \hat{t})] * \theta^j + \alpha \cdot s(t, \hat{t}) * \hat{\theta}^j,$$

in which $\alpha \in (0, 1)$ is a hyperparameter, and $s(t, \hat{t}) = \frac{1}{t - \hat{t} + 1}$ is a function to determine the value of the error caused by staleness [29]. Then, the PS executes the Model-Assignment-Function to assign a new submodel $\theta^{j'}$ to client n.

For each client n, when receiving a submodel θ^j from PS at epoch t, it sets its local model $\hat{\theta}^j_0 \leftarrow \theta^j$ and solves the regularized optimization problem:

$$g_{\theta^j}(\hat{\theta}^j; z) = f_n(\hat{\theta}^j; z) + \frac{\rho}{2} \|\hat{\theta}^j - \theta^j\|^2, \ \rho > \mu,$$

in the following $I(n)$ local iterations. μ is the hyperparameter defined in our Definition 2 for weak convexity. In each iteration $i = 1, \ldots I(n)$, client n samples data z^n_i from its local dataset D^n and updates the model using gradient descent:

$$\hat{\theta}^j_i \leftarrow \hat{\theta}^j_{i-1} - \gamma \nabla g_{\theta^j}(\hat{\theta}^j_{i-1}; z^n_i).$$

After completing the local updates, the client n has $\hat{\theta}^j \leftarrow \hat{\theta}^j_{I(n)}$ and uploads the submodel $\hat{\theta}^j$ back to the PS for aggregation.

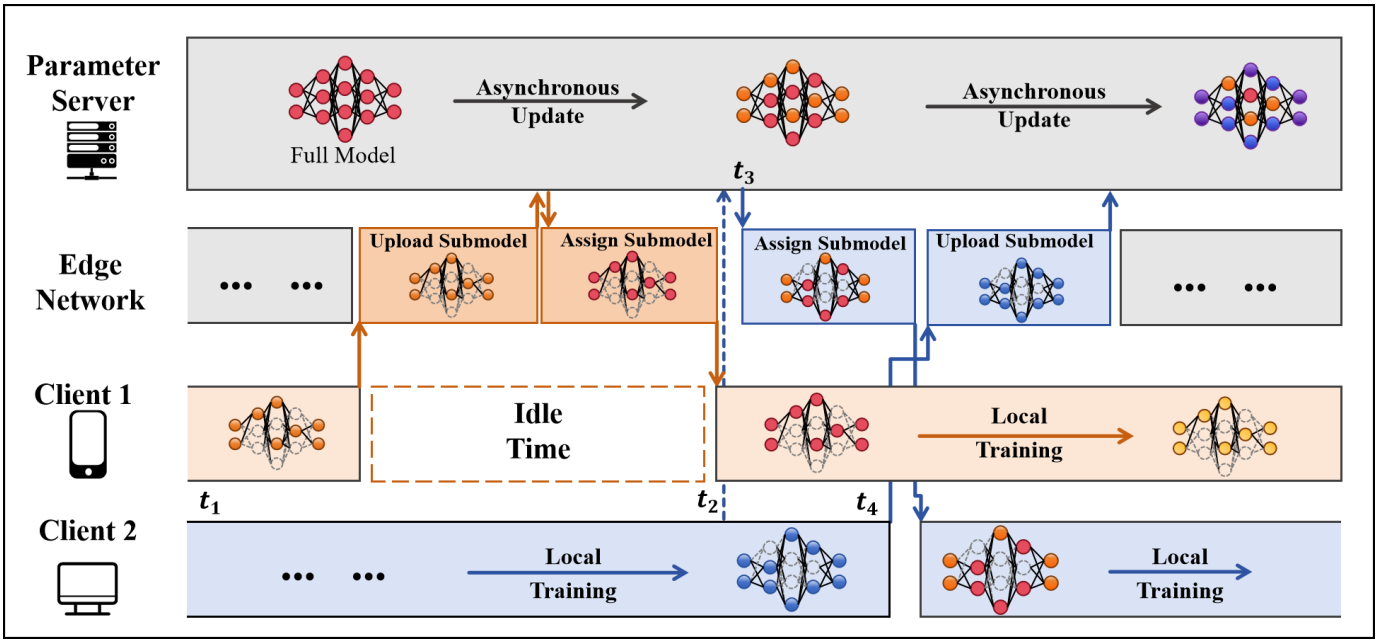


Fig. 2. The pipeline of Fed-RAA. We show the different ways clients transmit with PS in Vanilla FL and our Fed-RAA.

Submodel Assignment before Aggregation. According to the setting in vanilla FL, a new submodel will be assigned to the client n until its previous submodel has been uploaded and aggregated into the global model. In other words, when the submodel is being uploaded/assigned via the edge network, the computing resource on client n is idle and gets wasted. Client 1 in Fig. 2 is given as an example. To address this issue, a pipeline is designed in Fed-RAA to make sure the submodel uploading/assignment and local training can be executed in parallel.

Without loss of generality, we take client 2 in Fig. 2 as an example, which starts its local training based on submodel θ^j from epoch t_1 and ends at epoch t_4. Firstly, the client n conducts a statistical prediction on its ending epoch t_4 during its local training. Specifically, when the client n executes its i-th iteration at epoch t_2, it can infer that the total $I(n)$ iterations end at epoch $t_4 = t_1 + (t_2 - t_1)I(n)/i$, because the computing overhead in each iteration is the same. At epoch t_2, a new submodel request will be sent to the PS with the inferred ending epoch t_4. After receiving the new submodel request and inferred ending epoch, the PS assigns a new submodel $\theta_{j'}$ to client n at epoch t_3. Considering the limited bandwidth b_n defined in the model section, the time cost of submodel assignment is $\frac{|\theta^{j'}|}{b_n}$ according to our system model. By setting i sufficiently small and $t_3 = t_4 - \frac{|\theta^{j'}|}{b_n}$, we have $t_1 \leq t_2 \leq t_3 < t_4$, and the new submodel $\theta^{j'}$ arrives at the client n when its current local training ends. At epoch t_4, when the client completes its local training, it immediately uploads the updated submodel $\hat{\theta}^j$ and starts the new local training based on $\theta_{j'}$.

With the strategy mentioned above, a client can immediately start the next round of local training after its current round of local training ends because the new submodel assigned by the PS has already arrived. This is possible in most cases because the time spent on local training is typically longer than the communication time for uploading/ downloading the submodel, ensuring that the client can proceed without delay. However, in very rare cases, when the communication time exceeds the local training time, the client may need to wait for the submodel before starting the next round of training. Even in these cases, the idle time is still shorter than it would be without the pipeline strategy, where the client would otherwise experience significant delays between local training rounds. A detailed discussion is provided in the Appendix.

B. Theoretical Analysis on Convergence

To make the proof clear and easier to follow, we redefine some of the variables involved in the FL process separately in the analysis section. In Fed-RAA, when a client n receives a submodel θ^j assigned by the server at epoch τ, it updates the submodel θ^j for $I(n)$ iterations. $\theta^j_{\tau,i-1}$ is used to denote the model θ^j at the beginning of the i-th iteration. Initially, we have $\theta^j_{\tau,0} \leftarrow \hat{\theta}^j_0$. For the parameters $I(n)$ for all clients, $I_{\min}$ and $I_{\max}$ are their lower bound and upper bound, respectively.

Definition 1. *(Smoothness) A differentiable function f is L-smooth if for $\forall x, y$, $f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2$, where $L > 0$.*

Definition 2. *(Weak convexity) A differentiable function f is μ-weakly convex if the function g with $g(\theta) = f(\theta) + \frac{\mu}{2}\|\theta\|^2$ is convex, where $\mu \geq 0$. f is convex if $\mu = 0$, and non-convex if $\mu > 0$.*

Definition 3. *(Training delay) For any submodel θ^j assigned by PS to client n for local training at epoch τ, and later received by PS from client n for model aggregation at epoch t, its training delay is defined as the time gap between epochs τ and t. For all the submodels already assigned and received*

Algorithm 1 Fed-RAA

Definition
D^n: local dataset of client n;
θ: global model held by the PS, $\theta = \{\theta^1, \theta^2, \ldots, \theta^M\}$;
θ^j: submodels held by the PS, $j \in \{1, 2, \ldots, M\}$;
$\hat{\theta}^j$: submodels updated by the clients, $j \in \{1, 2, \ldots, M\}$;
$\hat{\theta}_i^j$: submodels trained by a client in its i-th iteration;
$s(t, \hat{t})$: function of staleness;
α: hyperparameter, $\alpha \in (0, 1)$;

For parameter server:
if $t = 0$ **then**
  for $n = 1, \cdots, N$ **do**
    $\theta^j \leftarrow$ Model-Assignment-Function (θ);
    PS sends submodel θ^j and time stamp 0 to client n;
  end for
end if *// Initial submodel assignment*
for $t = 1, \cdots, T$ **do**
  if receive $(\hat{\theta}^j, \hat{t})$ from client n **then**
    $\theta^j \leftarrow [1 - \alpha * s(t, \hat{t})] * \theta^j + \alpha * s(t, \hat{t}) * \hat{\theta}^j$;
    $\theta^{j'} \leftarrow$ Model-Assignment-Function (θ);
    PS sends submodel $\theta^{j'}$ and time stamp t to client n;
  end if *// Asynchronous submodel assignment*
end for

For each client n**:**
if receive submodel θ^j and time stamp t from PS **then**
  Define: $g_{\theta^j}(\hat{\theta}^j; z) = f_n(\hat{\theta}^j; z) + \frac{\rho}{2}\|\hat{\theta}^j - \theta^j\|^2$, $\rho > \mu$;
  $\hat{\theta}_0^j \leftarrow \theta^j$;
  for local iteration $i = 1, \cdots, I(n)$ **do**
    Sample $z_i^n \sim D^n$;
    Update local model: $\hat{\theta}_i^j \leftarrow \hat{\theta}_{i-1}^j - \gamma \nabla g_{\theta^j}(\hat{\theta}_{i-1}^j; z_i^n)$;
  end for *// Local training for $I(n)$ iterations*
  $\hat{\theta}^j \leftarrow \hat{\theta}_{I(n)}^j$;
  Send updated submodel and time stamp $(\hat{\theta}^j, t)$ to server;
end if

by the PS, let variable K be an upper bound for the training delays of these submodels.

Definition 4. *(Imbalance ratio of local updates) The imbalance ratio δ of local updates is defined as the ratio between the maximum and minimum number of local updates performed by each client before pushing the model to the server: $\delta = I_{max}/I_{min}$, where I_{min} and I_{max} are the minimum and maximum numbers of local updates, respectively.*

Assumption 1. *(Gradient bounds) For all $\theta \in \mathbb{R}^d$, $n \in [N]$, and $z \sim D^n$, the gradient of the loss functions $f(\theta; z)$ and $g_{\theta'}(\theta; z)$ are bounded by constants V_1 and V_2, respectively:*

$$\|\nabla f(\theta; z)\|^2 \leq V_1 \quad \textit{and} \quad \|\nabla g_{\theta'}(\theta; z)\|^2 \leq V_2.$$

Assumption 2. *(Condition on ρ) For any small constant $\epsilon > 0$, we assume that ρ is large enough such that $\rho > \mu$ and the following inequality holds:*

$$-(1 + 2\rho + \epsilon)V_2 + \rho^2\|\theta_{\tau,i-1}^j - \theta_\tau^j\|^2 - \frac{\rho}{2}\|\theta_{\tau,i-1}^j - \theta_\tau^j\|^2 \geq 0,$$

for all $\theta_{\tau,i-1}^j, \theta_\tau^j$, and $j \in [M]$.

Assumption 3. *(Condition on γ) We assume that the step size γ is sufficiently small, such that $\gamma < 1/L$, where L is the smoothness constant of the objective function.*

Theorem 1. *Let all assumptions hold. If every submodel θ^j has been updated for at least Q times until epoch T, Fed-RAA converges to a critical point:*

$$\min_{q=0}^{T-1} \mathbb{E}\|\nabla F(\theta_q)\|^2 \leq \frac{\mathbb{E}[F(\theta_0) - F(\theta_T)]}{\alpha\gamma\epsilon QM I_{min}} + \mathcal{O}\left(\frac{\gamma I_{max}^3 + \alpha K I_{max}}{\epsilon I_{min}}\right) + \mathcal{O}\left(\frac{\alpha^2\gamma K^2 I_{max}^2 + \gamma K^2 I_{max}^2}{\epsilon I_{min}}\right).$$

where θ_q denotes the global model at epoch q.

Theorem 1 shows the convergence rate of algorithm Fed-RAA by giving the upper bound on the gradient of all clients for all trained submodels.

Remark 1. *Impact of the training delay bound K. Our convergence result shows that the smaller the upper bound K of staleness $(t - \tau)$ would lead to a faster convergence rate and better performance in our federated learning process.*

Corollary 1. *Let all assumptions hold. Using $\delta = \frac{I_{max}}{I_{min}}$, and taking $\alpha = \frac{1}{\sqrt{I_{min}}}$, $\gamma = \frac{1}{\sqrt{QM}}$, $I_{min} = (QM)^{\frac{1}{5}}$, we have the convergence rate as follows:*

$$\min_{q=0}^{T-1} \mathbb{E}[\|\nabla F(\theta_q)\|^2] \leq \mathcal{O}\left(\frac{1}{(QM)^{\frac{3}{5}}}\right) + \mathcal{O}\left(\frac{\delta^3}{(QM)^{\frac{1}{10}}}\right) + \mathcal{O}\left(\frac{K\delta}{(QM)^{\frac{1}{10}}}\right) + \mathcal{O}\left(\frac{K^2\delta^2}{(QM)^{\frac{1}{2}}}\right) + \mathcal{O}\left(\frac{K^2\delta^2}{(QM)^{\frac{3}{10}}}\right).$$

Corollary 1 indicates that the term $\mathcal{O}\left(\frac{\delta^3 + K\delta}{(QM)^{\frac{1}{10}}} + \frac{K^2\delta^2}{(QM)^{\frac{3}{10}}}\right)$ will dominate the convergence rate. The proof of Theorem 1 is given in the Appendix.

V. Online strategy for Submodel Assignment

In Fed-RAA, the PS assigns submodels to clients for local training with a Model-Assignment-Function. According to our theoretical analysis in Theorem 1, when the submodels have been updated for a sufficient time (i.e., Q is sufficiently large), the convergence of Fed-RAA is mainly impacted by the training delay bound K, and a smaller K results in a smaller global loss. Thus, an online submodel assignment strategy that minimizes the delay bound K is important for the convergence of Fed-RAA. In the following, we first formulate the algorithm design for submodel assignment as an optimization problem on the training delay bound K with the constraint on Q. Later, an online **Greedy** and **Resource Adaptive** strategy is presented, abbreviated as *Greedy*, and proved to be optimal.

A. Problem Formulation for Submodel Assignment

The primary goal of submodel assignment is to optimize the training delay bound K, i.e., minimize the maximum training delay in Fed-RAA for all submodels. According to Algorithm 1, the training delay of a submodel θ^j assigned to client n consists of three components: (1) the time cost for client n to download the submodel θ^j, (2) the time cost on $I(n)$ iterations in local

updating, and (3) the time cost for client n to upload the submodel. Since the network bandwidth for client n is b_n, the time cost to download/upload the submodel is $|\theta^j|/b_n$. According to our model definition, the computation power of client n is p_n, then the time cost for local updating is $I(n)|z_i^n|C/p_n$, where C is the computational complexity of local training on each sample, $|z_i^n|$ is the size of data sampled from its local dataset in each iteration, and $I(n)$ is the number of iterations in local training. Let $c_{n,j} = 2\frac{|\theta^j|}{b_n} + \frac{I(n)|z_i^n|C}{p_n}$ be the training delay if a submodel θ^j is assigned to client n.

With the above formulation, the submodel assignment strategy aims to minimize the training delay bound $K = \max_{n,j}(c_{n,j})$ across any client n and submodel j if submodel θ^j is assigned to clients n for local training, subject to the constraint that each submodel must be trained at least Q times for convergence. We define $x_{n,j,t}$ as a binary variable that indicates whether submodel θ^j is assigned to client n at epoch t. The optimization problem can be formally written as:

$$\begin{aligned} \min_{\{x_{n,j,t}\}} \quad & \max_{n,j,t}(x_{n,j,t}c_{n,j}) \\ \text{subject to:} \quad & \sum_{t=1}^{T}\sum_{n=1}^{N} x_{n,j,t} \geq Q, \quad \forall j \in \{1,\ldots,M\} \\ & x_{n,j,t} \in \{0,1\}, \quad \forall n,j,t \end{aligned}$$

B. Algorithm Design for Online Submodel Assignment

As illustrated in Fig. 2, when the PS receives a submodel request from client n, a new submodel will be assigned to client n with the following online strategy:

- In step 1, the PS figures out a set of submodels $S = \{\theta^j | c_{n,j} \leq K, \forall j \in [M]\}$, which means that assigning every submodel from the set S will not increase the current training delay bound K. $K \leftarrow K+1$ until S is non-empty
- In step 2, the PS selects a submodel θ_j from the set S with the smallest $q(j)$, in which $q(j)$ is the number of times θ^j has been updated by the clients. The submodel θ^j will be assigned to client n for its next-round local training. If there are more than one submodel θ^j with the smallest update count (i.e., $q(j)$), then randomly select one and assign it to client n.

Initially, we have $K = 1$. Obviously, step 1 is designed as the greedy strategy to minimize the training delay bound K. The strategy in step 2 prioritizes those submodels with fewer updates while also handling ties in a random manner to avoid deterministic bias in the selection process. With step 2, we make sure that each submodel can be trained at least Q times when minimizing the training delay bound K. The pseudocode of our online model assignment strategy is given in Algorithm 2.

C. Algorithm Analysis

Theorem 2. `Algorithm` *2 reaches an optimal performance on minimizing the training delay bound* K.

Proof. Let K_{OPT} and K_{ALG} be the training delay bounds obtained by an optimal offline submodel assignment strategy and our online greedy-based model assignment algorithm, `Algorithm` 2. In the online setting, the PS makes decisions according to the history and current information when it receives the submodel requests from clients. While, in the offline setting, the PS knows all the information of clients and when they will require submodels for local training during the execution of Fed-RAA. We now prove that $K_{\text{ALG}} = K_{\text{OPT}}$.

Step 1: Proving $K_{\text{OPT}} \geq K_{\text{ALG}}$. In `Algorithm` 2, parameter $K = 1$ initially. When PS receives a submodel request from client n, $K \leftarrow K+1$ until the set S is non-empty. Then, a submodel θ^j with the smallest update count $q(j)$ from set S is assigned to client n. When Fed-RAA terminates, the parameter K records the largest training delay bound obtained by our online strategy. We then obtain a result that when the value of K increases to $\max_{j=1}^{M}\min_{n=1}^{N} c_{n,j}$ in the first time, K no longer increases. This is because, when $K = \max_{j=1}^{M}\min_{n=1}^{N} c_{n,j}$, we can always find a submodel θ^j satisfying that $c_{n,j} \leq K$ for arbitrary client $n \in [M]$. Then, S is always a non-empty set for the following clients and K no longer increases. With the above analysis, we have $K_{ALG} \leq \max_{j=1}^{M}\min_{n=1}^{N} c_{n,j}$.

Since K_{OPT} is the training delay bound obtained by an optimal offline strategy, we obtain the result that $K_{OPT} \geq \max_{j=1}^{M}\min_{n=1}^{N} c_{n,j}$. Otherwise, we can at least find a submodel θ_j satisfying that $\min_{n=1}^{N} c_{n,j} > K_{OPT}$, which means the submodel θ^j was never updated by any clients. Obviously, this inference violates the constraint that each submodel must be updated at least Q times and results in a contradiction.

With the above results, we obtain the result that $K_{\text{OPT}} \geq \max_{j=1}^{M}\min_{n=1}^{N} c_{n,j} \geq K_{\text{ALG}}$

Step 2: Proving $K_{\text{OPT}} \leq K_{\text{ALG}}$. By directly executing our online `Algorithm` 2 in the offline setting, we obtain a feasible offline solution with training delay bound K_{ALG}. Since K_{OPT} is defined as the optimal training delay bound in the offline setting, we have $K_{\text{OPT}} \leq K_{\text{ALG}}$.

From the two steps above, we have shown that:

$$K_{\text{ALG}} \leq K_{\text{OPT}} \quad \text{and} \quad K_{\text{OPT}} \leq K_{\text{ALG}}$$

Therefore, we conclude that $K_{\text{ALG}} = K_{\text{OPT}}$, meaning that the upper bound of the training delay produced by our algorithm

Algorithm 2 Model-Assignment-Function (θ) on the PS

Definition
$c_{n,j}$: the training delay if submodel θ^j was assigned to client n for local training;
$q(j)$: the number of times θ^j has been updated by clients;
K: training delay bound. Initially, $K = 1$;

For PS:
if receive a submodel request from client n **then**
  while set $\{\theta^j | c_{n,j} \leq K, \forall j \in [M]\} = \emptyset$ **do**
    $K \leftarrow K + 1$;
  end while
  $S \leftarrow \{\theta^j | c_{n,j} \leq K, \forall j \in [M]\}$;
  Select θ^j with the smallest $q(j)$ from the set S;
  Assign θ^j to client n;
  $q(j) \leftarrow q(j) + 1$;
end if

is equal to the upper bound produced by the optimal offline model assignment strategy. □

VI. Experiment

In this section, we present the experimental setup and performance evaluation of the Fed-RAA algorithm. The experiment setups describe the simulation environment, hardware, datasets, models, and partitioning strategies used for comparison. We then assess Fed-RAA's performance through comparisons with baseline algorithms, ablation studies on model assignment strategies, and scalability analysis, highlighting the trade-offs between efficiency, accuracy, and resource usage.

A. Experiment Setup

In this part, we present the experiment setups for evaluating Fed-RAA, which include the simulation environment, hardware specifications, and the setup of devices with varying computing capabilities and network bandwidth. We also outline the datasets, models, and model partitioning strategy used to compare Fed-RAA with baselines.

1) Implementation: We implement the algorithmic simulation using PyTorch with multi-threading, where each thread maintains a local client model for individual training, while the master program manages the global model. The simulation software runs on Linux servers equipped with an AMD EPYC 9654 96-core processor, an NVIDIA RTX 4090 GPU with 24GB VRAM, and 60GB of memory.

TABLE III
Computing Capability and Network Bandwidth Level Setting

Level	CPU	GPU	Bandwidth
1	3.0 GHz - 10.0 GHz	2.0 GHz - 10.0 GHz	200 Mbps - 500 Mbps
2	2.0 GHz - 3.0 GHz	1.5 GHz - 2.0 GHz	50 - 200 Mbps
3	1.0 GHz - 2.0 GHz	500 MHz - 1.5 GHz	10 - 50 Mbps

We conduct the experiments across 100 clients. The capability of simulated clients consists of two parts: computing capability and network bandwidth, each with three levels, as shown in Table III. Level 1 clients have the highest performance in both computing resources and network bandwidth, while Level 3 clients have the lowest. The distribution of clients is set to reflect common network environments, with 40% low-performance, 30% medium-performance, and 30% high-performance clients. When a client is assigned to level 1/2/3, it randomly and uniformly selects its CPU/GPU/bandwidth capability from the corresponding intervals.

2) Datasets, Models and Model Splitting Strategy: To evaluate the performance of Fed-RAA and compare it with other algorithms, we selected four benchmark datasets commonly used in image classification: CIFAR-10 [30], CIFAR-100 [30], MNIST [31], and Tiny ImageNet [32]. We focus on observing the convergence rate and maximum achievable accuracy of the global model during the simulation.

For the independent and identically distributed (IID) setting, each client contains samples from all classes, with the class distribution on each client matching the overall distribution of the dataset. In the Non-IID setting, we refer to the class partitioning strategy proposed in [16], where each client contains only half of the classes, and the amount of data per class is kept equal across clients. To evaluate performance across different scenarios and datasets, we adopted dataset-specific backbones. For **MNIST**, we used a three-layer MLP model with 784 input units, resulting in approximately **158,000** parameters. For **CIFAR-10**, we employed an improved CNN architecture, totaling about **1.2 million** parameters. For **CIFAR-100**, we utilized a *ResNet-18* architecture with Group Normalization (GN) instead of Batch Normalization (BN), with a total of **11.2 million** parameters. Lastly, for **Tiny-ImageNet**, we adopted a *ResNet-50* architecture, consisting of approximately **25.6 million** parameters. In the Fed-RAA framework, submodels are created by randomly partitioning the global model into smaller segments like [9], enabling clients to train fewer parameters and speed up convergence. This partitioning is controlled by predefined hyperparameters, which segment the model into 2, 3, 4, or 5 parts, as detailed in Table IV. The partition strategy is non-uniform, where weaker devices receive smaller submodels to minimize training time, and more capable devices are assigned larger submodels to maximize their processing power. More details can be seen in the Appendix. We have made the source code publicly available at https://github.com/tdwxz/Fed_raa.

TABLE IV
Model Partition Strategy

Number of partitions	Partition ratio
2	50%, 100%
3	30%, 60%, 100%
4	25%, 50%, 75%, 100%
5	20%, 40%, 60%, 80%, 100%

B. Performance of Fed-RAA

In this subsection, we evaluate the performance of the Fed-RAA algorithm through comparisons with state-of-the-art federated learning methods, ablation studies, and scalability analysis. First, we benchmark Fed-RAA against several baseline algorithms, demonstrating its superior training speed and convergence. Next, we conduct an ablation study to assess the impact of different model assignment strategies and aggregation methods on Fed-RAA's efficiency and accuracy. Finally, we explore the scalability of Fed-RAA by analyzing the effects of user heterogeneity and submodel fragmentation on training performance, highlighting the key trade-offs between time, accuracy, and resource usage.

1) Comparisons with Baselines: To substantiate the superior performance of Fed-RAA, we benchmark it against classical or state-of-the-art methods in FL, namely, **FedAvg** [33], **FedSync** [29], **FedProx** [34], **FedPrun** [28], **SplitFed** [8], **RAM-Fed** [9], **FJORD** [21], **FedRolex** [22], and **NeFL** [23]. FedAvg represents a classical synchronous full-model FL framework, while FedSync exemplifies the classical asynchronous approach. FedProx, SplitFed, and RAM-Fed focus primarily

on the local computational limitations of users. In FedProx, users train the full model. In SplitFed and RAM-Fed, users employ a submodel for synchronous FL. FedPrun considers both computational and communication resources but operates within a synchronous FL framework. FJORD and NeFL handles system heterogeneity via ordered dropout, producing nested submodels matched to client resources. FedRolex enables model-heterogeneous FL with rolling submodel extraction, evenly updating the global model across rounds.

We conducted experiments comparing the Fed-RAA algorithm with baseline methods across multiple datasets, and the results in Table V and Fig. 3 demonstrate that Fed-RAA converges more rapidly. As shown in Table V, Fed-RAA outperforms other federated learning algorithms in most cases across both IID and Non-IID client distributions. On MNIST (IID), Fed-RAA reaches 80% accuracy in 3.07 seconds, markedly faster than FedProx with 14.77 seconds and RamFed with 56.33 seconds, and it is also the fastest to reach 90% accuracy (Fed-RAA with 21.80 s vs. FedAvg with 59.87 s and FedSync with 126.72 s). Under MNIST (Non-IID), Fed-RAA remains the quickest, showing strong robustness to data heterogeneity. On CIFAR-10, Fed-RAA dominates the higher targets in the IID setting: its *time-to-target* is 15.94 s, 36.46 s, and 80.41 s to reach 60%, 70%, and 75% accuracy, respectively. Whereas, several baselines are substantially slower or *did not reach these targets within the evaluation budget* (reported as "N/A"). Under Non-IID partitions, it is consistently the fastest (e.g., Fed-RAA with 12.22 s vs. FedSync with 80.57 s, FedAvg with 366.41 s, and FedRolex with 15.73 s to achieve 40% accuracy). For CIFAR-100 (IID), Fed-RAA is the fastest to achieve the accuracy 60% within 273.38 s and is the *only* method to reach 65% (within 732.85 s). For CIFAR-100 (Non-IID), it is the fastest to reach accuracy 45% and 50% (within 173.90 s and 314.68 s respectively), with other methods slower or failing to reach the targets. On Tiny-ImageNet, while FedSync has the best time-to-accuracy performance at the very low accuracy (10%) threshold, Fed-RAA has the shortest time-to-accuracy for essentially all higher accuracy thresholds (e.g., 55.18–280.19 seconds to achieve 20%–80% accuracy with IID data, 42.73–293.55 seconds to achieve 20%–70% accuracy with Non-IID data). Overall, these results show that Fed-RAA scales favorably to harder datasets and Non-IID partitions.

The experimental results in Fig. 3 indicate that Fed-RAA produces curves that are consistently left-shifted (faster to reach a given accuracy) and/or above competing methods (higher accuracy at the same time) across all four datasets and under both IID and Non-IID partitions. On MNIST, the Fed-RAA trajectory rises sharply and saturates early, staying above FedAvg, FedSync, and other baselines; the gap is even more visible under Non-IID splits, where alternatives plateau lower. On CIFAR-10, strong baselines (e.g., FedSync/FedRolex) climb quickly at the very beginning, but Fed-RAA sustains a steeper improvement and maintains the highest envelope over time; under Non-IID partitions, its curve dominates the entire horizon. On CIFAR-100, although one or two methods exhibit a brief early surge at very low accuracy, Fed-RAA soon overtakes and attains a higher terminal level; the advantage persists under Non-IID settings, particularly in the medium-to-high accuracy range. For Tiny-ImageNet, Fed-RAA preserves a clear mid–high accuracy advantage, the curve keeps improving while others flatten, showing better robustness as heterogeneity increases. Overall, the trajectories corroborate that Fed-RAA converges faster and to higher accuracy than competing algorithms, and this advantage is amplified when data are Non-IID, consistent with the design that reduces synchronization bottlenecks and better tolerates client heterogeneity.

In summary, Fed-RAA outperforms baseline algorithms in terms of both training speed and model convergence. It achieves faster training speed, and balances time and accuracy effectively.

2) Ablation Study: We designed the Greedy-based submodel online assignment algorithm (abbreviated as *Greedy*) to address the model assignment problem within Fed-RAA, as described in Algorithm 2. While *Greedy* has been theoretically proven to achieve the same upper bound of training delay as the optimal offline strategy, we conducted ablation studies to validate its impact on convergence speed and maximum accuracy in Fed-RAA. The study involved three variant algorithms: (1) *Random*, which randomly assigns a submodel when receiving the submodel request from a client; (2) *Minimum Priority*, which assigns the submodel with the least updating times; and (3) *Sync*, which uses synchronous aggregation while keeping all other aspects of Fed-RAA unchanged. The results of these variants on three datasets (MNIST, CIFAR-10, CIFAR-100, and Tiny-ImageNet) are shown in Fig. 4 and Table VI.

In our ablation study, the *Greedy* algorithm demonstrates high efficiency on the MNIST dataset with IID data distribution, achieving 92.85% accuracy in just 3.52 seconds. Similarly, *Greedy* also attains a high accuracy on the CIFAR-10 dataset in a very short amount of time, reaching 38.70% accuracy in 49.11 seconds. In contrast, other algorithms, such as *Minimum Priority*, take 50.00 seconds to achieve 37.96% accuracy, while *Sync* requires significantly more time (55.02 seconds) to achieve a slightly higher accuracy of 37.89%. On the CIFAR-100 dataset, *Greedy* and other algorithms still perform with comparable efficiency, requiring around 97.32 seconds to achieve similar accuracy (25.53%). For the Tiny-ImageNet dataset, the *Greedy* algorithm achieves 26.00% accuracy in 91.13 seconds, whereas the other algorithms require longer times (ranging from 154.29 to 210.58 seconds) for comparable accuracy levels. In the Non-IID scenario, *Greedy* also demonstrates the same advantage.

3) Scalability of Our Approach: In this part, we investigate the impact of two key hyperparameters on the performance of Fed-RAA: the proportion of lowest-capability users (β) and the number of submodels (M) after segmentation.

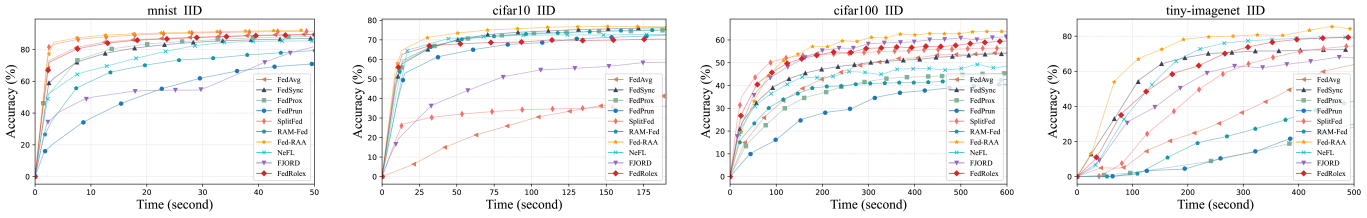
To quantify the impact of low-capability users on the overall training process of Fed-RAA, we introduce the parameter β, which represents the proportion of the lowest-capability users (Level 3) among all users. During user allocation, we set β as the proportion of Level 3 users, $\frac{1-\beta}{2}$ for Level 2 users, and $\frac{1-\beta}{2}$ for Level 1 users. Additionally, when comparing the effect of different submodel partitioning strategies on the overall Fed-RAA training, the corresponding partitioning strategy for various values of M is shown in Table IV.

TABLE V
COMPARISON OF TIME (IN SECONDS) REQUIRED TO ACHIEVE SPECIFIED ACCURACY LEVELS UNDER BOTH **IID** AND **NON-IID** CLIENT DATA DISTRIBUTIONS. N/A MEANS THE TARGET WAS NOT ACHIEVED WITHIN THE RECORDED HORIZON.

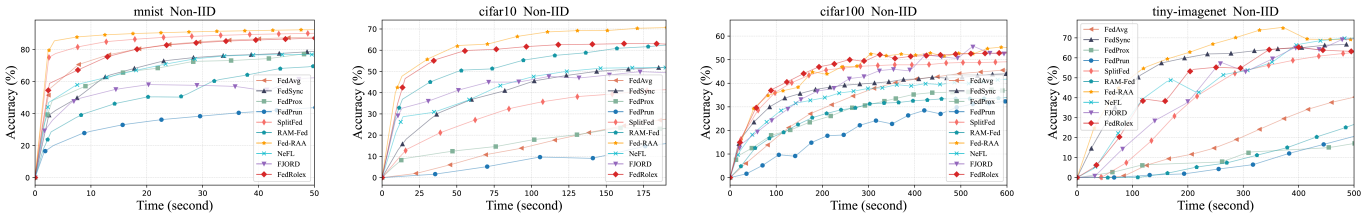
Dataset	Distribution	Accuracy Target	FedAvg	FedSync	FedProx	FedPrun	SplitFed	Ram-Fed	FJORD	FedRolex	NeFL	Fed-RAA
MNIST	IID	30	0.79	1.97	1.84	8.55	0.47	3.67	1.96	0.87	2.84	**0.30**
		40	1.06	2.91	2.46	14.40	0.63	5.42	5.72	1.20	3.52	**0.32**
		50	1.46	3.37	3.32	19.58	0.85	7.82	10.22	1.66	4.34	**0.49**
		60	2.15	5.00	4.99	28.92	1.12	11.33	38.84	2.41	5.74	**0.72**
		70	3.75	8.65	7.71	47.31	1.75	21.82	42.65	4.11	15.46	**1.38**
		80	8.45	18.45	14.77	135.24	3.51	56.33	48.72	9.55	27.13	**3.07**
		90	59.87	126.72	87.67	805.73	25.87	466.48	100.27	65.67	N/A	**21.80**
	Non-IID	30	1.05	2.40	3.59	13.91	0.54	5.32	2.30	0.83	3.18	**0.30**
		40	2.16	5.42	4.10	36.24	0.77	10.26	6.59	1.34	3.74	**0.31**
		50	3.85	10.20	7.38	81.33	1.15	21.26	10.25	2.45	5.09	**0.48**
		60	5.94	14.39	14.28	130.43	1.92	33.83	48.01	5.47	12.13	**0.67**
		70	9.70	20.60	25.29	193.16	2.95	53.95	170.06	11.61	24.39	**1.13**
		80	20.14	57.66	72.31	N/A	8.22	84.10	N/A	20.29	N/A	**2.54**
		90	100.99	N/A	202.14	N/A	49.99	N/A	N/A	118.14	N/A	**20.14**
CIFAR-10	IID	30	101.16	1.06	7.54	8.07	41.04	2.62	23.58	1.94	1.03	**0.74**
		40	175.38	5.22	7.57	16.61	N/A	3.40	44.70	**2.54**	5.52	2.87
		50	320.56	12.17	11.50	19.32	N/A	10.45	87.23	**4.80**	17.13	8.34
		60	N/A	26.83	24.84	45.38	N/A	23.19	370.61	17.13	21.64	**15.94**
		70	N/A	60.76	58.92	138.30	N/A	58.44	N/A	171.18	45.24	**36.46**
		75	N/A	134.76	N/A	N/A	N/A	195.53	N/A	N/A	N/A	**80.41**
	Non-IID	20	130.52	27.26	134.02	262.53	42.90	4.14	5.99	5.52	3.72	**1.98**
		30	218.24	45.96	279.83	495.58	83.55	13.92	12.93	9.28	30.50	**6.19**
		40	366.41	80.57	1049.86	3550.49	167.56	29.36	55.51	15.73	76.81	**12.22**
		50	3422.11	151.82	N/A	N/A	3623.31	60.46	N/A	29.75	131.90	**25.30**
		60	N/A	586.22	N/A	N/A	N/A	155.81	N/A	82.08	N/A	**51.71**
		70	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	**171.92**
CIFAR-100	IID	10	29.94	15.09	35.12	57.04	**9.67**	18.22	12.48	18.96	21.43	18.20
		20	58.36	35.39	76.21	122.48	**16.70**	48.59	30.44	25.97	34.91	29.63
		30	101.98	58.55	124.33	276.28	**29.42**	93.35	41.88	45.84	62.25	56.26
		40	177.39	118.02	266.77	572.41	**51.35**	256.38	80.43	68.88	126.56	78.60
		50	322.35	318.60	1423.29	N/A	**99.70**	N/A	144.28	136.73	756.89	115.81
		60	7125.43	5143.23	N/A	N/A	2946.46	N/A	443.06	685.01	N/A	**273.38**
		65	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	**732.85**
	Non-IID	10	65.33	21.73	33.66	109.65	19.57	51.51	22.18	28.28	29.59	**19.16**
		15	100.01	32.79	78.99	177.80	**30.02**	76.22	40.59	34.57	44.90	36.42
		20	130.52	48.20	134.02	262.53	44.71	132.53	74.96	**40.33**	68.98	41.77
		25	166.89	71.69	213.88	342.05	61.70	181.66	114.60	53.56	97.72	**52.07**
		30	218.24	100.49	279.83	495.58	82.90	286.96	141.50	64.18	136.75	**61.08**
		35	288.50	143.81	384.02	773.06	106.74	904.14	183.45	**88.84**	232.64	97.41
		40	366.41	318.73	1049.86	3550.49	142.71	4256.81	240.73	**114.48**	441.22	161.98
		45	567.47	855.93	2749.02	N/A	209.29	N/A	325.78	177.48	N/A	**173.90**
		50	3422.11	3783.14	N/A	N/A	772.71	N/A	430.69	358.63	N/A	**314.68**
Tiny-ImageNet	IID	10	87.27	**38.13**	259.33	254.86	88.08	158.50	65.16	45.66	53.44	47.12
		20	164.30	56.50	398.56	370.14	114.46	227.53	88.09	62.63	73.86	**55.18**
		30	253.96	84.27	521.56	500.18	147.32	351.48	99.89	79.32	95.93	**64.16**
		40	321.76	99.50	644.56	627.01	184.97	457.91	159.24	100.98	117.36	**74.26**
		50	387.18	120.43	767.56	753.84	221.28	590.52	175.00	141.80	140.18	**84.78**
		60	463.41	154.55	890.55	880.68	274.29	721.32	306.52	204.08	168.23	**104.43**
		70	563.26	302.33	1013.55	1007.51	382.84	852.12	583.84	277.16	214.71	**149.43**
		80	663.16	3678.09	1136.55	1134.34	809.59	982.93	N/A	589.68	282.29	**280.19**
	Non-IID	10	174.75	**35.49**	282.71	358.18	99.48	307.48	67.10	56.18	49.83	35.53
		20	295.33	55.97	N/A	490.92	139.28	428.75	115.04	87.64	62.26	**42.73**
		30	377.74	75.80	N/A	N/A	172.69	532.44	148.37	111.87	101.14	**63.19**
		40	495.78	100.78	N/A	N/A	212.72	N/A	229.32	132.74	151.40	**78.82**
		50	N/A	125.15	N/A	N/A	279.09	N/A	249.19	175.32	197.18	**111.15**
		60	N/A	212.17	N/A	N/A	417.38	N/A	361.54	355.05	352.33	**177.04**
		70	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	515.46	**293.55**

TABLE VI
ALGORITHM COMPARISON ON MNIST, CIFAR-10, CIFAR-100, AND TINY-IMAGENET UNDER BOTH **IID** AND **NON-IID** CLIENT DATA DISTRIBUTIONS.

Algorithm	**MNIST**				**CIFAR-10**				**CIFAR-100**				**Tiny-ImageNet**			
	IID		**Non-IID**		**IID**		**Non-IID**		**IID**		**Non-IID**		**IID**		**Non-IID**	
	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)	Acc. (%)	Time (s)
Greedy	92.85	3.52	87.37	6.91	38.70	49.11	44.31	47.62	26.00	91.13	15.08	95.83	48.90	183.62	56.11	150.67
Random	91.90	5.04	85.81	7.38	37.60	50.08	41.40	49.67	15.78	92.23	13.27	96.28	45.15	189.50	52.23	194.75
Minimum Priority	92.03	4.81	86.50	7.19	37.96	50.00	41.79	47.98	25.53	154.29	11.79	97.27	42.46	200.17	53.01	199.89
Sync	92.13	6.81	47.41	21.07	37.89	55.02	22.74	50.38	23.57	97.32	7.42	93.16	7.50	210.58	23.03	206.11

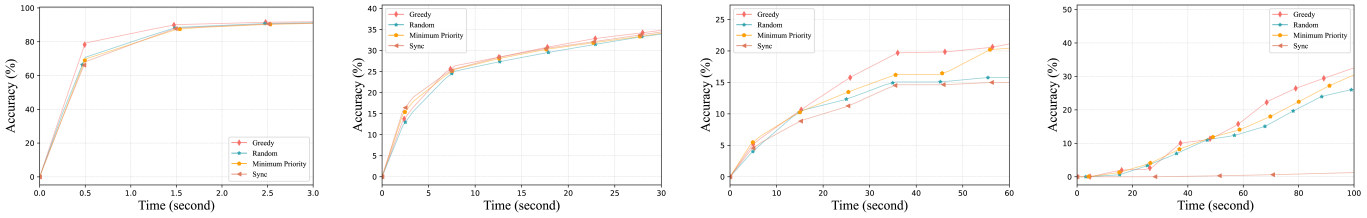


(a) MNIST (IID) (b) CIFAR-10 (IID) (c) CIFAR-100 (IID) (d) Tiny-ImageNet (IID)

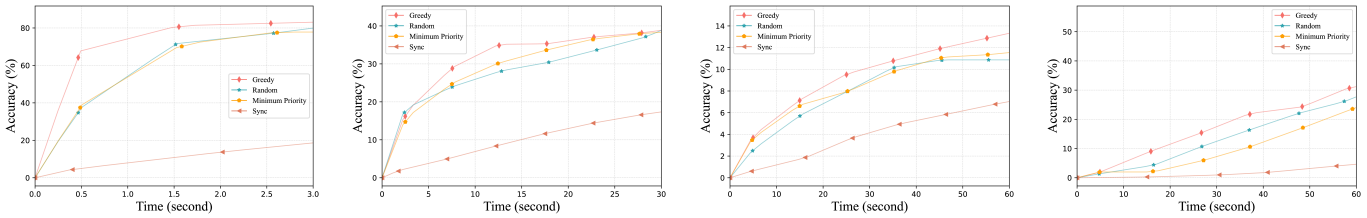


(e) MNIST (Non-IID) (f) CIFAR-10 (Non-IID) (g) CIFAR-100 (Non-IID) (h) Tiny-ImageNet (Non-IID)

Fig. 3. Detailed results of comparison among different algorithms. The top row shows the results for **IID**, while the bottom row shows those for **Non-IID**.



(a) MNIST (IID) (b) CIFAR-10 (IID) (c) CIFAR-100 (IID) (d) Tiny-ImageNet (IID)

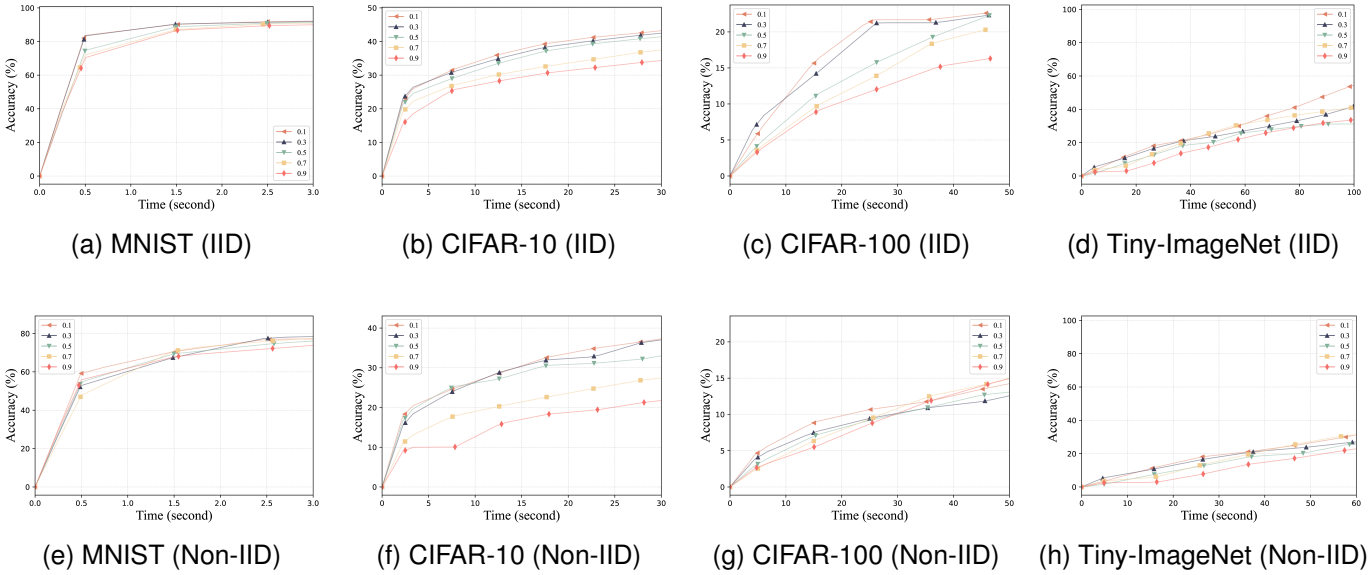


(e) MNIST (Non-IID) (f) CIFAR-10 (Non-IID) (g) CIFAR-100 (Non-IID) (h) Tiny-ImageNet (Non-IID)

Fig. 4. The comparison of ablation methods on different datasets. The top row shows the results for **IID**, while the bottom row shows those for **Non-IID**.

Impact of Client Heterogeneity (β). As shown in Fig. 5, as β increases, Fed-RAA continues to perform well even in weaker environments, still making efficient use of the computational power of weak clients with minimal impact. Specifically, in most cases (e.g., MNIST in both IID and Non-IID settings, CIFAR-10 in IID, CIFAR-100 in Non-IID, and Tiny-ImageNet in Non-IID scenarios), we observe that as β increases, Fed-RAA still converges at a very fast rate, with the data lines for $\beta = 0.9$ and $\beta = 0.1$ being very close, and with the difference between the largest curves being less than 5%. This indicates that Fed-RAA is little affected by the increase in the proportion of weak clients, and can still fully utilize their local computational power for fast convergence to high accuracy. However, in a few scenarios (such as CIFAR-10 in Non-IID, CIFAR-100 in IID, and Tiny-ImageNet in IID), increasing β does have some impact on the results. Nevertheless, the effect remains relatively limited, with the largest accuracy drop occurring in the Tiny-ImageNet IID case, where the accuracy decreases by 15% when β is 0.9 compared to when β is 0.1. Overall, this demonstrates that Fed-RAA is quite resilient to increases in the proportion of weak clients, particularly for most datasets.

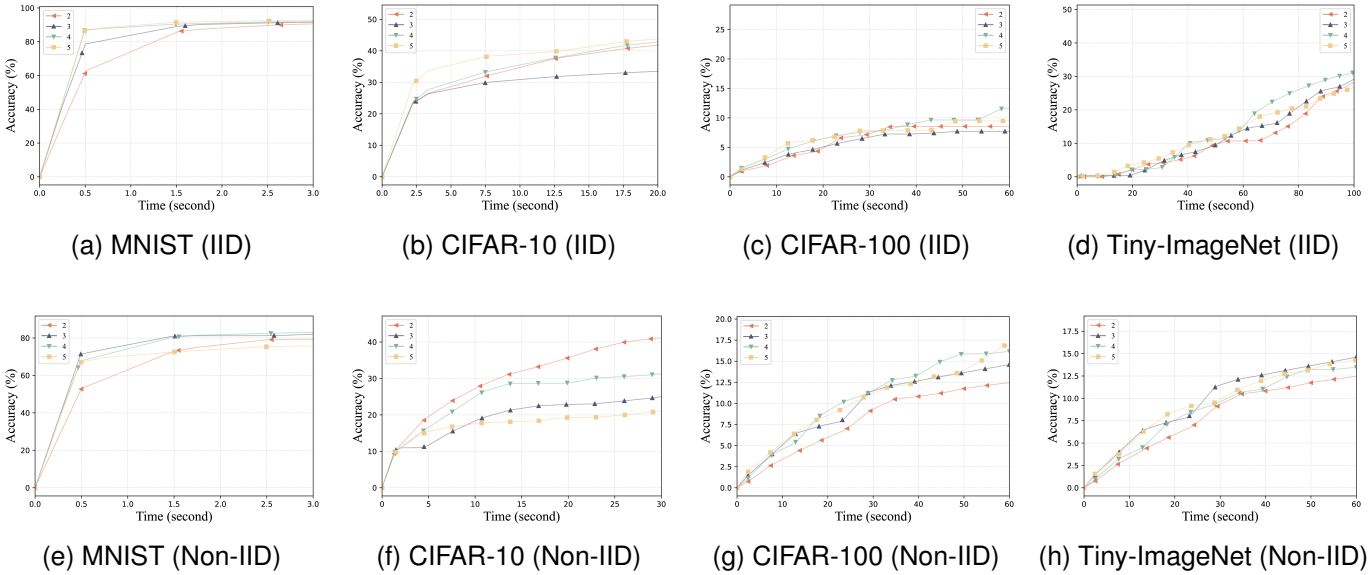
Impact of Submodel Number (M). The results from Fig. 6 highlight the varying impacts of the number of submodels M on training time and accuracy across different datasets and distributions. On the MNIST dataset, under the IID condition, the curves for different values of M are closely overlapping, indicating minimal differences in accuracy. However, under the Non-IID condition, the curve for $M = 4$ consistently



(a) MNIST (IID) (b) CIFAR-10 (IID) (c) CIFAR-100 (IID) (d) Tiny-ImageNet (IID)

(e) MNIST (Non-IID) (f) CIFAR-10 (Non-IID) (g) CIFAR-100 (Non-IID) (h) Tiny-ImageNet (Non-IID)

Fig. 5. The influence of β on different datasets. The top row shows the results for **IID**, while the bottom row shows those for **Non-IID**.



(a) MNIST (IID) (b) CIFAR-10 (IID) (c) CIFAR-100 (IID) (d) Tiny-ImageNet (IID)

(e) MNIST (Non-IID) (f) CIFAR-10 (Non-IID) (g) CIFAR-100 (Non-IID) (h) Tiny-ImageNet (Non-IID)

Fig. 6. The influence of M on different datasets. The top row shows the results for **IID**, while the bottom row shows those for **Non-IID**.

performs the best, achieving the highest accuracy. This trend is also observed on other datasets, such as CIFAR-100, where both IID and Non-IID distributions show $M = 4$ as yielding the optimal results. However, on the Tiny-ImageNet dataset, particularly under the Non-IID condition, $M = 3$ proves to be the most effective configuration, yielding the best accuracy. These findings suggest that the optimal value of M is not fixed and depends on both the dataset and the data distribution, with careful selection necessary to avoid unnecessary overhead and ensure optimal performance.

VII. Acknowledge

This work was supported in part by the National Natural Science Foundation of China under Grant 62572280, U24A20244, U23A20302, and the Shandong Science Fund for Excellent Young Scholars under Grant 2023HWYQ-007.

VIII. Conclusion

This paper presents Fed-RAA, a Resource-Adaptive Asynchronous Federated Learning algorithm designed to address the inefficiencies of traditional federated learning methods in resource-constrained edge environments. Unlike previous synchronous approaches, which suffer from the straggler and communication contention problems, Fed-RAA adapts to clients' heterogeneous resources by assigning tailored submodels and enabling asynchronous updates. This ensures that clients with stronger computing/communication abilities are not delayed by weaker ones, optimizing both computing and

communication efficiency. The greedy-based online submodel assignment strategy further enhances resource utilization in local training, offering a practical solution to the online decision-making challenge. Our theoretical analysis guarantees convergence, and extensive experiments on MNIST, CIFAR-10, CIFAR-100, and Tiny-ImageNet show significant improvements in training speed and accuracy compared to existing methods. Scaling Fed-RAA for large model architectures like Mixture of Experts will be our work in the future.

REFERENCES

[1] Q. Nguyen, H. H. Pham, K. Wong, P. L. Nguyen, T. T. Nguyen, and M. N. Do, "Feddct: Federated learning of large convolutional neural networks on resource-constrained devices using divide and collaborative training," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 1, pp. 418–436, 2024.

[2] J. Qin, X. Zhang, B. Liu, and J. Qian, "A split-federated learning and edge-cloud based efficient and privacy-preserving large-scale item recommendation model," *J. Cloud Comput.*, vol. 12, no. 1, p. 57, 2023.

[3] L. Song, J. Li, H. Jiang, S. Wei, and Y. Guo, "Chpfl: Clustered adaptive hierarchical federated learning for edge-level personalization," *High-Confidence Computing*, p. 100343, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667295225000479

[4] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity." *arXiv: Learning,arXiv: Learning*, Jan 2021.

[5] B. Yan, K. Li, M. Xu, Y. Dong, Y. Zhang, Z. Ren, and X. Cheng, "On protecting the data privacy of large language models (llms) and llm agents: A literature review," *High-Confidence Computing*, vol. 5, no. 2, p. 100300, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667295225000042

[6] R. Yang, T. F. Tan, W. Lu, A. J. Thirunavukarasu, D. S. W. Ting, and N. Liu, "Large language models in health care: Development, applications, and challenges," *Health Care Science*, vol. 2, no. 4, pp. 255–263, 2023.

[7] C. Cui, Y. Ma, X. Cao, W. Ye, Y. Zhou, K. Liang, J. Chen, J. Lu, Z. Yang, K.-D. Liao *et al.*, "A survey on multimodal large language models for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 958–979.

[8] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 2022, pp. 8485–8493.

[9] Y. Wang, X. Zhang, M. Li, T. Lan, H. Chen, H. Xiong, X. Cheng, and D. Yu, "Theoretical convergence guaranteed resource-adaptive federated learning with mixed heterogeneity," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, A. K. Singh, Y. Sun, L. Akoglu, D. Gunopulos, X. Yan, R. Kumar, F. Ozcan, and J. Ye, Eds. ACM, 2023, pp. 2444–2455.

[10] T. Yang, Y. Xiao, G. Motta, F. Beaufays, R. Mathews, and M. Chen, "Online model compression for federated learning with large models," in *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*. IEEE, 2023, pp. 1–5.

[11] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[12] Z. Long, Y. Chen, H. Dou, Y. Zhang, and Y. Chen, "Fedsq: Sparse-quantized federated learning for communication efficiency," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 4050–4061, 2024.

[13] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[14] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," *Advances in neural information processing systems*, vol. 31, 2018.

[15] M. Rostami and S. S. Kia, "Fedscalar: A communication efficient federated learning," 2024.

[16] W. Zhang, T. Zhou, Q. Lu, Y. Yuan, A. Tolba, and W. Said, "Fedsl: A communication-efficient federated learning with split layer aggregation," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 15 587–15 601, 2024.

[17] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *arXiv: Learning,arXiv: Learning*, Oct 2020.

[18] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices." *IEEE Transactions on Neural Networks and Learning Systems*, p. 1–13, Jan 2022.

[19] B. Yuan, C. R. Wolfe, C. Dun, Y. Tang, A. Kyrillidis, and C. Jermaine, "Distributed learning of fully connected neural networks using independent subnet training," *Proceedings of the VLDB Endowment*, p. 1581–1590, Apr 2022.

[20] M. Wu, M. Boban, and F. Dressler, "Flexible Training and Uploading Strategy for Asynchronous Federated Learning in Dynamic Environments," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 12 907–12 921, 12 2024.

[21] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane, "Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 876–12 889, 2021.

[22] S. Alam, L. Liu, M. Yan, and M. Zhang, "Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction," *Advances in neural information processing systems*, vol. 35, pp. 29 677–29 690, 2022.

[23] H. Kang, S. Cha, J. Shin, J. Lee, and J. Kang, "Nefl: Nested model scaling for federated learning with system heterogeneous clients," *IEEE Transactions on Mobile Computing*, 2025.

[24] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[25] F. Nikolaidis, M. Symeonides, and D. Trihinas, "Towards efficient resource allocation for federated learning in virtualized managed environments," *Future Internet*, vol. 15, no. 8, 2023.

[26] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *CoRR*, vol. abs/1903.03934, 2019.

[27] L. Gao, W. Li, H. Ma, Y. Liu, and C. Li, "Data cube-based storage optimization for resource-constrained edge computing," *High-Confidence Computing*, vol. 4, no. 4, p. 100212, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667295224000151

[28] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

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

[30] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[31] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[32] M. A. mnmoustafa, "Tiny imagenet," 2017. [Online]. Available: https://kaggle.com/competitions/tiny-imagenet

[33] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016.

[34] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

Ruirui Zhang received the B.S. degree from Taishan College, Shandong University, in 2022. She is currently pursuing a Ph.D. degree with the School of Computer Science and Technology, Shandong University. Her research interests include mobile edge computing, Internet of Things, and federated learning.

Xingze Wu received his Bachelor's degree from North China Electric Power University, China, in 2023. He is currently pursuing his Master's degree at the School of Computer Science and Technology, Shandong University. His research interests include edge computing, wireless sensor networks, and federated learning.

Yifei Zou (Member, IEEE) received the BEng degree from Wuhan University, China, in 2016, and the Ph.D. degree from The University of Hong Kong, China, in 2020. He is currently an assistant professor at the School of Computer Science and Technology, Shandong University, Qingdao, China. His research interests include wireless networks, ad hoc networks, and distributed computing.

Zhenzhen Xie (Member, IEEE) received her M.S. and Ph.D. degrees in Computer Science from Jilin University, China, in 2014 and 2021, respectively. She currently holds a post-doctoral position in the School of Computer Science and Technology, Shandong University. Her research areas are edge computing, the Internet of Things, and federated learning.

Peng Li (Senior Member, IEEE) is currently a Professor with the Xi'an Jiaotong University, China. He received the PhD degree in computer science from The University of Aizu, Japan. His research interests mainly focus on wired/wireless networking, cloud/edge computing, distributed AI systems, and blockchain. He has authored or co-authored over 100 papers in major conferences and journals. He won the 2020 Best Paper Award of IEEE Transactions on Computers. He serves as the chair of SIG on Green Computing and Data Processing in IEEE ComSoc Green Communications and Computing Technical Committee. He is a guest editor of IEEE Journal of Selected Areas on Communications, the editor of IEEE Open Journal of the Computer Society and IEICE Transactions on Communications. He is a senior member of IEEE.

Xiuzhen Cheng received her M.S. and Ph.D. degrees in computer science from the University of Minnesota Twin Cities in 2000 and 2002, respectively. She is a professor in the School of Computer Science and Technology, Shandong University, Qingdao, China. She has published more than 170 peer-reviewed papers. Her current research interests include cyber physical systems, wireless and mobile computing, sensor networking, wireless and mobile security, and algorithm design and analysis. She worked as a professor with the Department of Computer Science, the George Washington University, Washington, DC, USA, from 2013 to 2017. She worked as a program director for the US National Science Foundation (NSF) from April to October in 2006 (full time), and from April 2008 to May 2010 (part time). She received the NSF CAREER Award in 2004. She is Fellow of IEEE and a member of ACM.

Falko Dressler received his M.Sc. and Ph.D. degrees from the Dept. of Computer Science, University of Erlangen in 1998 and 2003, respectively. He is a full professor and Chair for Data Communications and Networking at the School of Electrical Engineering and Computer Science, TU Berlin. Dr. Dressler has been associate editor-in-chief for IEEE Trans. on Mobile Computing and Elsevier Computer Communications as well as an editor for journals such as IEEE/ACM Trans. on Networking, IEEE Trans. on Network Science and Engineering, Elsevier Ad Hoc Networks, and Elsevier Nano Communication Networks. He has been chairing conferences such as IEEE INFOCOM, ACM MobiSys, ACM MobiHoc, IEEE VNC, IEEE GLOBECOM. He authored the textbooks Self-Organization in Sensor and Actor Networks published by Wiley & Sons and Vehicular Networking published by Cambridge University Press. He has been an IEEE Distinguished Lecturer as well as an ACM Distinguished Speaker. Dr. Dressler is an IEEE Fellow as well as an ACM Distinguished Member. He is a member of the German National Academy of Science and Engineering (acatech). He has been serving on the IEEE COMSOC Conference Council and the ACM SIGMOBILE Executive Committee. His research objectives include adaptive wireless networking (radio, visible light, molecular communications) and embedded system design (from microcontroller to Linux kernel) with applications in ad hoc and sensor networks, the Internet of Things, and cooperative autonomous driving systems.

Dongxiao Yu (Senior Member, IEEE) received the B.S. degree in 2006 from the School of Mathematics, Shandong University and the Ph.D degree in 2014 from the Department of Computer Science, The University of Hong Kong. He became an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2016. He is currently a professor in the School of Computer Science and Technology, Shandong University. His research interests include wireless networks, distributed computing and graph algorithms.