Computation and Communication Efficient Federated Learning With Adaptive Model Pruning

Zhida Jiang, Yang Xu, *Member, IEEE*, Hongli Xu, *Member, IEEE*, Zhiyuan Wang, Jianchun Liu, Qian Chen, and Chunming Qiao, *Fellow, IEEE*

***Abstract*—Federated learning (FL) has emerged as a promising distributed learning paradigm that enables a large number of mobile devices to cooperatively train a model without sharing their raw data. The iterative training process of FL incurs considerable computation and communication overhead. The workers participating in FL are usually heterogeneous and the workers with poor capabilities may become the bottleneck of model training. To address the challenges of resource overhead and system heterogeneity, this article proposes an efficient FL framework, called FedMP, that improves both computation and communication efficiency over heterogeneous workers through adaptive model pruning. We theoretically analyze the impact of pruning ratio on training performance, and employ a Multi-Armed Bandit based online learning algorithm to adaptively determine different pruning ratios for heterogeneous workers, even without any prior knowledge of their capabilities. As a result, each worker in FedMP can train and transmit the sub-model that fits its own capabilities, accelerating the training process without hurting model accuracy. To prevent the diverse structures of pruned models from affecting the training convergence, we further present a new parameter synchronization scheme, called Residual Recovery Synchronous Parallel (R2SP). Besides, our proposed framework can be extended to the peer-to-peer (P2P) setting. Extensive experiments on physical devices demonstrate that FedMP is effective for different heterogeneous scenarios and data distributions, and can provide up to 4.1× speedup compared to the existing FL methods.**

***Index Terms*—Adaptive model pruning, edge computing, federated learning, heterogeneity.**

Manuscript received 5 March 2022; revised 14 November 2022; accepted 15 February 2023. Date of publication 22 February 2023; date of current version 5 February 2024. This work was supported by the National Key Research and Development Program of China under Grant 2021YFB3301501, in part by the National Science Foundation of China (NSFC) under Grants 61936015, 62132019, and 62102391, in part by the Jiangsu Province Science Foundation for Youths under Grant BK20210122. A preliminary version of this article titled "FedMP: Federated Learning through Adaptive Model Pruning in Heterogeneous Edge Computing" was accepted by the 38th IEEE International Conference on Data Engineering (ICDE 2022). [DOI: 10.1109/ICDE53745.2022.00062]. Recommended for acceptance by Dr. Gong Zhang. *(Corresponding author: Hongli Xu.)*

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

Chunming Qiao is with the Department of Computer Science and Engineering, University at Buffalo, State University of New York, Buffalo, NY 14260 USA (e-mail: qiao@buffalo.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2023.3247798, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3247798

I. Introduction

With the rapid development of the Internet of Things (IoT), mobile and embedded devices (e.g., smartphones, wearables, and sensors) now generate more data than ever before, expected to reach gigabytes or even terabytes every day [1]. However, the limited communication bandwidth, together with the desire for data privacy, makes it infeasible to collect all data to the remote cloud for processing. As an alternative, edge computing has gained much attention [2], which performs data processing on edge nodes close to the data sources. With more data and advanced applications (e.g., autonomous driving, virtual reality), there is a growing trend to use massive data generated at the network edge to boost machine learning (ML) performance, which is known as federated learning (FL) [3]. Among previous FL frameworks, the dominant one is the parameter server (PS) based framework [4]. The PS transmits the global model to the participating workers (i.e., edge nodes). These workers perform local training using stochastic gradient descent (SGD) [5] and send local updates to the PS for global aggregation. In this process, FL enables multiple workers to cooperatively train an ML model without having to reveal their local data, which can protect user privacy and relieve bandwidth burden. With the technical advantages and implemental feasibilities, FL has been applied in a variety of scenarios other than edge computing, including finance, health, digital assistance and personalized recommendations [6], [7], [8], [9]. Besides, some specific FL applications have been proposed to improve our daily life such as next word prediction [10], query suggestion [11], physical activity detection [12], and keyword trigger [13].

However, the FL paradigm will encounter some difficulties in practice for the following reasons. (1) *Scarce Communication Bandwidth*. Since the average wide area network (WAN) bandwidth between the PS and workers is much (e.g., 15×) lower than the local area network (LAN) bandwidth within the datacenters where the PS resides [14], the frequent communication between the PS and workers will overwhelm the scarce WAN bandwidth, and the communication may be frequently interrupted, slow or expensive. (2) *Limited On-Edge Resources*. Besides the bandwidth, the computation and storage resources of the workers are always limited in edge computing [3]. On the other hand, the parameter size of deep neural networks (DNNs) may reach tens or even hundreds of megabytes. The heavyweight architecture of DNNs makes them computationally expensive with excessive memory requirements and probably results in unbearable delay or breakdown when training DNNs on resource-constrained workers. (3) *Heterogeneous Edge Nodes*. The edge nodes may range from universal gateways to specialized base stations, and their communication, computation and storage capabilities are usually heterogeneous [15]. The workers with different capabilities will widen the performance gap between high- and low-performance nodes and deteriorate the training efficiency [7].

Some previous works have made efforts to separately address the above resource limitation and system heterogeneity challenges. On the one hand, many solutions have been proposed to reduce resource consumption by communication round reduction [5], [16] or model/gradient compression [17], [18], [19], [20], [21], [22], [23], [24], [25]. However, they just alleviate the total communication cost and do not essentially reduce the model complexity. As a consequence, huge computation overhead makes on-device training difficult and may become the principal constraint of these works. Moreover, an important factor is ignored in previous studies of efficient FL, i.e., heterogeneity. If the PS sends the identical model to heterogeneous workers like [26], [27], the workers with poor capabilities may become the bottleneck of model training and delay global aggregation, called straggler problem [28].

On the other hand, some optimization methods [29], [30], [31], [32], [33] have been proposed to mitigate the impact of heterogeneity on FL. Unfortunately, they cannot reduce both communication and computation overhead, thereby hindering efficient FL over resource constrained workers. The recent works [34], [35] enable the training of heterogeneous local models with varying complexities, improving communication and computation efficiency. However, HeteroFL [34] does not propose specific methods about how to adaptively determine the complexity levels of local models. They also require complete knowledge about device capabilities for model assignment. However, edge nodes are reluctant to share their private information (e.g., computing power) with the PS for privacy concerns. AdaptCL [35] does not provide the convergence analysis and introduces new hyper-parameters to prevent excessive pruning, which is difficult to achieve the trade-off between efficiency and accuracy.

In this article, we propose FedMP, an efficient FL framework to conquer the above challenges. Specifically, FedMP adopts model pruning technique which can reduce the resource demands of DNNs while maintaining the accuracy of the original models [27], [36], [37]. Compared with parameter quantization and compression, model pruning can realize both communication and computation savings. We develop an online learning algorithm to adaptively determine appropriate pruning ratios for heterogeneous workers. Given the pruning ratios, the PS performs distributed model pruning and sends the personalized sub-models to the workers for local training. In FedMP, each worker only transmits and trains a pruned model fitting its own capability, which can mitigate the effect of stragglers on training performance.

However, there are some challenges in designing such an FL framework. First, it is non-trivial to determine different pruning ratios for heterogeneous workers so as to achieve the trade-off between resource efficiency and model accuracy, especially when the prior knowledge of workers' capabilities is not available. Second, it is also challenging to aggregate sub-models of workers to derive the global model. Different from existing pruning methods, the sub-models generated by adaptive model pruning have diverse structures, thus direct aggregation by traditional synchronization schemes such as Bulk Synchronous Parallel (BSP) will degrade the overall accuracy [38]. In light of the above discussion, we state the key contributions as follows:

- We propose a novel FL framework, called FedMP, which simultaneously reduces computation and communication overhead through adaptive model pruning, enabling efficient FL over heterogeneous workers. Besides, FedMP can be extended to the P2P setting.
- FedMP adopts a Multi-Armed Bandit (MAB) based online learning algorithm to adaptively determine the pruning ratios for the workers even without knowing any prior knowledge of their capabilities, which achieves the trade-off between efficiency and accuracy.
- We also present a novel parameter synchronization scheme called Residual Recovery Synchronous Parallel (R2SP), which recovers the sub-models with diverse structures before model aggregation and ensures comprehensive model updates during the training process.
- We implement FedMP on a physical platform and extensively evaluate FedMP with typical FL tasks. The experimental results show that FedMP is effective for different heterogeneous scenarios and data distributions, and can provide up to 4.1× speedup compared to the existing FL methods.

The remainder of this article is organized as follows. Section II introduces the background and motivation. We describe the workflow of FedMP and provide the convergence analysis in Section III. Section IV presents an adaptive pruning ratio decision algorithm. In Section V, we extend FedMP to the P2P settings. Experimental results are given in Section VI. Section VII discusses the applicability of FedMP to other models. Section VIII reviews related works and Section IX concludes the article.

II. Background and Motivation

In this section, we first introduce the background of FL and then discuss the disadvantages of traditional model pruning methods for FL in heterogeneous edge computing, which motivates our design.

A. Federated Learning

In FL, the workers make full use of the rich data from IoT devices, mobile phones, edge servers, *etc*, and train DNNs collaboratively while keeping data locally [3], [5], [15]. Considering a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of workers (i.e., edge nodes) with local datasets $\{D_1, D_2, \ldots, D_N\}$, the goal of FL is to solve the following distributed optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) \triangleq \frac{1}{N}\sum_{n=1}^{N} F_n(\mathbf{x}), \qquad (1)$$

TABLE I
SUMMARY OF MAIN NOTATIONS

Symbol	Definition
N	Number of workers
D_n	Local dataset
F	Global loss function
F_n	Local loss function
φ_n	Training sample in the local dataset
K	Total number of training rounds
k	Round number index
τ	Number of local iterations in each round
t	Iteration number index
α_n^k	Pruning ratio of worker n in round k
$\mathbf{x}^k$	Global model in round k
$\hat{\mathbf{x}}_n^k$	Sub-model of worker n in round k
$\mathbf{x}_n^k$	Sparse model of worker n in round k
$\overline{\mathbf{x}}_n^k$	Residual model of worker n in round k
Q_n^k	Pruning error of worker n in round k
γ	Learning rate
T_n^k	Completion time of worker n in round k
T^k	Completion time of round k
ε	Convergence threshold of global loss function
$\mathbf{P_k}$	Sequence of partition regions
θ	Granularity of pruning ratio exploration
λ	Discount factor

where $\mathbf{x}$ is the model parameter vector, N is the number of workers, and $F_n(\mathbf{x})$ is the local loss function of worker n.

Such a distributed ML framework incurs huge computation and communication overhead, especially training modern DNNs with a large number of parameters [39]. Due to the limited computation and communication resources in edge computing, the considerable overhead becomes a bottleneck of speeding up the model training [17]. To overcome the bottleneck, some efficient FL solutions have been proposed to alleviate communication overhead by reducing the frequency of global aggregation [3], [5], quantization [17], [18], [19], [20], [21] and sparsification [22], [23], [24], [25].

However, the previous works mainly focus on reducing the communication overhead, but ignore the computation efficiency. In fact, the computation capability of edge nodes is also limited compared to that of the data center. It is notoriously expensive for DNNs to perform iterations of SGD. For example, the VGG-16 model [40], with 138.34 million parameters, requires 30.94 billion floating-point operations to process a single image. Consequently, local training is often very slow in modern DNNs due to the large amount of computation overhead. Furthermore, the edge nodes participating in FL are heterogeneous, i.e., with different computation and communication capabilities. The unbalanced capabilities of workers will exacerbate the straggler problem. Some "weak" workers may become the bottleneck of model aggregation and seriously limit the training speed of FL. In light of this fact, it is worthwhile to design more effective solutions, which can reduce the resource cost and mitigate the impact of device heterogeneity on FL.

B. Model Pruning

There are usually tens of millions of parameters in modern DNNs, and it could be infeasible to store and train these over-parameterized models on the resource-constrained devices. To address this challenge, model pruning is proposed to reduce the model parameters and resource demands of DNNs while achieving almost similar accuracy of the original model [36], [37]. In general, model pruning approaches can be classified into unstructured pruning and structured pruning. On the one hand, unstructured pruning removes the unimportant weights (i.e., the connections between the neurons) in the neural networks, which results in a high level of parameter sparsity [36]. However, due to the irregularization of the resulting sparse matrix, it is very difficult to compress the parameters in memory, and some specialized hardware/libraries are required to accelerate the model training [41]. On the other hand, structured pruning [37] is designed to directly remove some unimportant model structures (e.g., convolutional filters) without introducing sparsity. Therefore, the resulting model can be regarded as a sub-figure or sub-model of the original neural network, which contains fewer parameters and helps reduce both communication and computation overhead.

Most of the existing model pruning methods are proposed for centralized model training. However, the data in FL is scattered across workers, and the PS does not have access to data for pre-training. Thus these methods cannot be directly applied to distributed scenarios of FL [36], [37]. Although some works [27] focus on the application of unstructured model pruning in FL, they need the support of additional hardware and libraries. Besides, they ignore the heterogeneity of workers and the PS sends the identical pruned model to all workers, thus weak workers will become the bottleneck of model training. To this end, we are inspired to design an efficient FL framework through adaptive model pruning for heterogeneous workers.

III. Proposed Framework

In this section, we propose FedMP, an efficient FL framework, to improve both computation and communication efficiency in heterogeneous edge computing. We first introduce the overview of FedMP, then describe two key phases in detail. Finally, we theoretically analyze the convergence of FedMP.

A. Overview of FedMP

The model training usually involves a certain number of rounds. As shown in Fig. 1, each round in FedMP consists of the following phases:

① *Adaptive model pruning:* At the beginning of round $k \in \{0, 1, \ldots, K-1\}$, the PS adaptively determines the specific pruning ratio α_n^k according to the heterogeneous capabilities of worker n (Section IV). Then the PS prunes the global model $\mathbf{x}^k$ into the sub-models in terms of the pruning ratios. FedMP exploits a distributed method to remove less important filters and neurons from the original model (Section III-B). After model pruning, the PS sends the corresponding sub-model to each worker for local training.

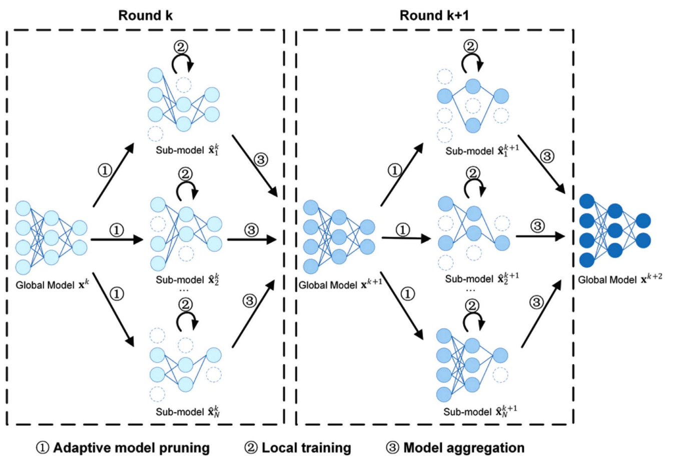

Round k
Round k+1
①
②
③
Global Model $\mathbf{x}^k$
Sub-model $\hat{\mathbf{x}}_1^k$
Sub-model $\hat{\mathbf{x}}_2^k$
Sub-model $\hat{\mathbf{x}}_N^k$
Global Model $\mathbf{x}^{k+1}$
Sub-model $\hat{\mathbf{x}}_1^{k+1}$
Sub-model $\hat{\mathbf{x}}_2^{k+1}$
Sub-model $\hat{\mathbf{x}}_N^{k+1}$
Global Model $\mathbf{x}^{k+2}$
① Adaptive model pruning
② Local training
③ Model aggregation

Fig. 1. FedMP overview. Each round consists of three phases, i.e., adaptive model pruning, local training, model aggregation. The workers are assigned different pruning ratios according to their heterogeneous capabilities. The pruning ratio of the same worker may vary in different rounds.

② *Local training:* Within a training round, each worker updates the sub-model over its local dataset via SGD for τ iterations. The larger the pruning ratio is (i.e., more global model parameters are pruned), the fewer CPU cycles it requires to process a data sample, and the fewer parameters it transmits. After τ local iterations, model aggregation will be activated, and each worker only uploads its trained sub-model rather than the entire model.

③ *Model aggregation:* After receiving the sub-models from all workers, the PS begins to perform global aggregation. In FedMP, we propose R2SP to recover the sub-models, then derive an updated global model by aggregating the recovered models (Section III-C).

The above procedure repeats until model convergence. Since only the pruned models are transmitted and trained, the computation and communication overhead is reduced significantly compared with training the entire model.

B. Adaptive Model Pruning Phase

In order to prevent weak workers from blocking the learning process, the PS determines different pruning ratios for heterogeneous workers, which will be elaborated in Section IV. The pruning ratio α_n^k represents the percentage of filters/neurons that are removed from the original model of worker n in round k. After determining the specific pruning ratios, the PS performs distributed model pruning on the same global model $\mathbf{x}^k$ for each worker, which is different from centralized model pruning methods. FedMP adopts the structured pruning approach rather than the unstructured pruning approach. The sparse network derived by the unstructured pruning approach is difficult to accelerate due to the lack of efficient sparse libraries and specialized hardware. Pruning weights does not efficiently reduce the computation time since most removed weights are from the fully-connected layers, but the computation cost is concentrated in the convolutional layers [42].

Specifically, the pruning strategy in FedMP is as follows. To avoid introducing layer-wise hyper-parameters, every layer uses the same pruning ratio [37]. The filters/neurons in the same layer are sorted by their importance scores. Given the pruning ratio, the filters/neurons with lower importance scores will be removed from each layer of the original model. FedMP chooses to use a simple yet effective metric (i.e., l_1-norm) to obtain the importance scores of filters/neurons [37]. For each filter in the convolutional layers, we calculate the sum of the absolute kernel weights as the filter's score. Previous studies [36] have shown that low-weight connections have a weak effect on model accuracy. Consequently, for each neuron in the fully-connected layers, we calculate the sum of the absolute weights that the neuron is connected to as the neuron's score. If the filters/neurons have relatively lower scores, they seem less important, and will be removed in terms of the pruning ratio, leading to a more compact network architecture.

When the filters with their feature maps are pruned, the corresponding channels of filters in the next layer are also removed. In addition, if a convolutional layer is pruned, the weights of the subsequent batch normalization layer are removed too. A smaller sub-model $\hat{\mathbf{x}}_n^k$ is created for worker n after distributed model pruning, and the remaining parameters of the modified global model are copied into the sub-model. The sub-models are sent from the PS to workers, then each worker starts local training over its local dataset. These pruned models have much fewer parameters compared to the original global model, thus both resource consumption and memory footprint will be reduced dramatically. Note that the sub-model $\hat{\mathbf{x}}_n^k$ of each worker is related to its pruning ratio α_n^k and may vary with different workers.

In a nutshell, our approach converts a cumbersome model into a slim model and does not introduce extra layer-wise metaparameters. l_1-norm is a good metric for filters and neurons evaluation considering that the PS is data free in FL setting. Moreover, FedMP does not rely on specialized libraries and hardware for acceleration performance, so it can work with existing deep learning libraries.

C. Model Aggregation Phase

After the workers complete the sub-model training over the local datasets, the local updates will be sent to the PS that synchronizes the parameters. Synchronization among workers is very critical in FL, and is a costly operation that may significantly reduce the benefits of data parallelism and model parallelism. Since the pruning ratios of different workers change dynamically, the sub-models involved in global aggregation have diverse structures in FedMP. Traditional synchronization schemes in distributed machine learning such as BSP may degrade the overall accuracy [38]. Therefore, we design a new synchronization scheme, called *Residual Recovery Synchronous Parallel (R2SP)* to guarantee the training convergence.

We next introduce the workflow of R2SP, as shown in Algorithm 1. In round k, the relatively unimportant filters and neurons are selected for worker n according to the pruning ratio α_n^k. The kernel weights of these filters and the weights connected to these

Algorithm 1: Residual Recovery Synchronous Parallel.

```
1: for Each round $k \in \{0, 1, \ldots, K-1\}$ do
2:     Processing at the PS:
3:     Prune the global model $\mathbf{x}^k$ to obtain the sparse
       model $\mathbf{x}_n^k$ and sub-model $\hat{\mathbf{x}}_n^k$;
4:     Obtain the residual model $\overline{\mathbf{x}}_n^k = \mathbf{x}^k - \mathbf{x}_n^k$;
5:     Record the indexes of the remaining parameters;
6:     Distribute the sub-model $\hat{\mathbf{x}}_n^k$ for local training;
7:     Receive the trained sub-models from all workers;
8:     Recover the sub-model $\hat{\mathbf{x}}_n^k$ based on the indexes;
9:     Add the recovered models and residual models;
10:    Perform parameter averaging;
```

neurons will be set to zero so as to obtain the sparse model $\mathbf{x}_n^k$. Note that the sparse model $\mathbf{x}_n^k$ is different from the sub-model $\hat{\mathbf{x}}_n^k$ that will be sent to worker n. That is, the physically removed parameters in the sub-model are set to 0 in the sparse model. The global model and the sparse model are subtracted to obtain the residual model $\overline{\mathbf{x}}_n^k$, i.e., $\overline{\mathbf{x}}_n^k = \mathbf{x}^k - \mathbf{x}_n^k$, and it works as an auxiliary variable for parameter synchronization. In practice, the indexes of the remaining parameters also need to be recorded for each worker n in R2SP, and we can use a binary mask to store the indexes, resulting in a negligible overhead for the resource-rich PS [14].

After receiving the trained sub-models from all workers, the PS begins to perform global aggregation. R2SP first performs model recovery for each sub-model based on the indexes stored in the PS. The recovered models have the same network structure as the global model. R2SP adds the recovered models and the corresponding residual models, then performs parameter averaging among all workers. Although the PS needs to maintain the residual models in R2SP, their occupied memory will be released after each global aggregation. In the current round, the residual models will not incur significant memory overhead on the PS. It is known that DNNs do not need full floating-point precision for inference [43]. When there are many workers, we can quantize each parameter in residual models with fewer bits to further reduce the memory overhead. The memory occupied by the residual model is only 10-20% of that by the original model, and thus leads to a small overhead for the resource-rich PS, which is usually a cloud or cloudlet in edge computing scenarios [44].

By the iterative pruning and recovery process, the PS maintains a rather complete model structure in the whole training process. With residual models, the filters and neurons pruned in the last round will be recovered during global aggregation, thus they may be trained in the next round. In this way, R2SP ensures that each model parameter has a chance to be trained. This is the major difference from traditional model pruning methods, which *permanently* remove filters and neurons from the network. The pruned parameters in traditional methods cannot update their weights, leading to a smaller and smaller global model. However, R2SP enables a fully functional model by recovering the sub-models and prevents the removal of important parameters from affecting subsequent training.

D. Convergence Analysis

In this section, we provide the convergence analysis of our proposed framework. We use an element-wise product to represent the pruned model, i.e., $\hat{\mathbf{x}}_n^k = \mathbf{x}^k \odot \mathbf{m}_n^k$, where $\mathbf{x}^k$ is the global model in round k and $\mathbf{m}_n^k$ denotes the mask vector that is zero if the corresponding parameter in $\mathbf{x}^k$ is pruned and one otherwise. Only the parameters that have not been pruned from the global model are accessible and trainable. After adaptive model pruning, each worker n receives the corresponding pruned model $\hat{\mathbf{x}}_n^k$ and performs τ local iterations by setting $\hat{\mathbf{x}}_n^k(0) = \hat{\mathbf{x}}_n^k$. At each iteration $t \in \{1, 2, \ldots, \tau\}$, the local model can be updated as

$$\hat{\mathbf{x}}_n^k(t) = \hat{\mathbf{x}}_n^k(t-1) - \gamma \nabla F_n(\hat{\mathbf{x}}_n^k(t-1); \varphi_n) \odot \mathbf{m}_n^k \tag{2}$$

where γ is the learning rate and φ_n is a training sample from the local dataset D_n. It is worth mentioning that $\nabla F_n(\hat{\mathbf{x}}_n^k(t-1); \varphi_n) \odot \mathbf{m}_n^k$ is the stochastic gradient computed by the remaining parameters in $\hat{\mathbf{x}}_n^k(t-1)$. Only the unpruned parameters are updated by the stochastic gradient while the pruned parameters will not be updated due to the element-wise product with mask $\mathbf{m}_n^k$. In this way, the training process of the pruned model $\hat{\mathbf{x}}_n^k$ (including forward and backward propagation) in the theoretical analysis is consistent with that of the sub-model in our FedMP framework. After τ iterations, the trained local models $\hat{\mathbf{x}}_n^k(\tau)$ are uploaded and then aggregated into the new global model.

To analyze the convergence of FedMP, we assume that our problem satisfies the following common assumption, which is widely used in previous works [45], [46], [47], [48] on the convergence analysis.

Assumption 1. Assume that our problem satisfies the following conditions:

1) Smoothness: Each function $F_n(\mathbf{x})$ is smooth with modulus L.
2) Bounded variances: There exists a constant $\sigma > 0$ such that

$$\mathbb{E}[\|\nabla F_n(\mathbf{x}; \varphi_n) - \nabla F_n(\mathbf{x})\|^2] \leq \sigma^2, \forall \mathbf{x}, \forall n$$

3) Bounded stochastic gradients and weights: There exist constants $G > 0$ and $\delta > 0$ such that

$$\mathbb{E}[\|\nabla F_n(\mathbf{x}; \varphi_n)\|^2] \leq G^2, \mathbb{E}[\| \mathbf{x} \|^2] \leq \delta^2, \forall \mathbf{x}, \forall n$$

The following lemma shows that the deviations between $\mathbf{x}^k$ and $\hat{\mathbf{x}}_n^k(t-1)$ can be controlled by selecting proper pruning ratios. In this article, we use the pruning error to measure how well the pruned model approximates the original model, which is defined as $Q_n^k \triangleq \mathbb{E}[\| \mathbf{x}^k - \hat{\mathbf{x}}_n^k \|^2]$.

Lemma 1. Under Assumption 1, the proposed framework ensures

$$\sum_{t=1}^{\tau} \sum_{n=1}^{N} \mathbb{E}[\| \mathbf{x}^k - \hat{\mathbf{x}}_n^k(t-1) \|^2] \leq \frac{2\tau^3 \gamma^2 G^2 N}{3} + 2\tau \sum_{n=1}^{N} Q_n^k$$

The proof of Lemma 1 can be found in Appendix A, available online.

Theorem 1. Let Assumption 1 and Lemma 1 hold, the convergence bound of our proposed framework is

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\| \nabla F(\mathbf{x}^k) \|^2]$$
$$\leq \frac{4}{K\gamma\tau}(F(\mathbf{x}^0)-F(\mathbf{x}^*)) + \frac{6\,L^2}{KN}\sum_{k=0}^{K-1}\sum_{n=1}^{N}Q_n^k$$
$$+\frac{6L\gamma\sigma^2}{N} + 2\tau^2\gamma^2\,L^2\,G^2$$

where $\mathbf{x}^*$ denotes the optimal model which minimizes the global loss function.

The proof of Theorem 1 can be found in Appendix B, available in the online supplemental material. In Theorem 1, we settle for a weaker notion of convergence and use the expected squared gradient norm to analyze the convergence property, as suggested in [18], [49].

Corollary 1. According to the pruning strategy and Assumption 1, we have

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\| \nabla F(\mathbf{x}^k) \|^2]$$
$$\leq \frac{4}{K\gamma\tau}(F(\mathbf{x}^0)-F(\mathbf{x}^*)) + \frac{6\,L^2\delta^2}{KN}\sum_{k=0}^{K-1}\sum_{n=1}^{N}\alpha_n^k + \frac{6L\gamma\sigma^2}{N}$$
$$+\,2\tau^2\gamma^2\,L^2\,G^2$$

The proof of Corollary 1 can be found in Appendix C, available in the online supplemental material. From Corollary 1, we find that the convergence bound is closely related to the pruning ratio of each worker. Larger pruning ratios result in a looser convergence bound, and vice versa. If the sub-model has fewer parameters (i.e., larger pruning ratios), the resource overhead can be reduced, but it will also have a negative impact on convergence.

IV. Algorithm for Pruning Ratio Decision

In this section, we propose an adaptive pruning ratio decision algorithm which is an important part in FedMP. We first formalize the problem and then identify two key challenges in designing such an algorithm. Finally, we propose an online learning algorithm to adaptively determine the pruning ratios for workers.

A. Problem Formulation

As discussed in Section III, the completion time T_n^k of worker n in round k includes local computation time $T_{n,comp}^k$ and transmission time $T_{n,comm}^k$, i.e.,

$$T_n^k = T_{n,comp}^k + T_{n,comm}^k \tag{3}$$

Both $T_{n,comp}^k$ and $T_{n,comm}^k$ are related to the pruning ratios. The larger the pruning ratios are, the more parameters will be removed from the original model, resulting in less transmission time and local computation time.

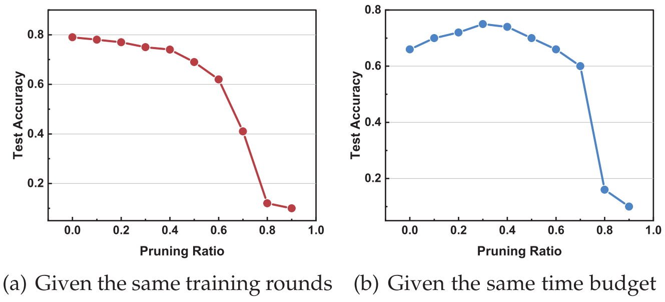


Fig. 2. Effect of different pruning ratios on test accuracy.

In the synchronous FL, the total latency is determined by the "slowest" worker. The total time of round k in FedMP is defined as

$$T^k = \max_n T_n^k \tag{4}$$

By (4), the completion time T^k of round k depends on the pruning ratios of all workers. Our goal is to determine the optimal pruning ratios for workers so as to minimize the completion time of the whole FL training with accuracy requirement. The optimization problem can be formulated as

$$\min \sum_{k=0}^{K-1} T^k$$
$$s.t. \begin{cases} F(\mathbf{x}^K) < \varepsilon \\ 0 \leq \alpha_n^k < 1, \quad \forall n, \forall k \end{cases} \tag{5}$$

The first inequality guarantees the model accuracy where ε is the convergence threshold of global loss function. The second inequality bounds the range of the pruning ratio.

B. Key Challenges

There are two key challenges in designing an effective algorithm for adaptive pruning ratio decision.

Challenge 1. *How should we determine optimal model pruning ratios for workers so as to achieve the trade-off between resource efficiency and training accuracy?* To verify the trade-off between efficiency and accuracy, we train AlexNet on the CIFAR-10 dataset over 10 workers. The learning rate, the batch size and the number of local iterations are set as 0.02, 16 and 50, respectively. Fig. 2(a) shows the accuracy results given the same training rounds when pruning ratios vary from 0 to 0.9. We observe that the model accuracy gradually decreases with the increase of pruning ratio. This is expected because more sensitive and meaningful filters/neurons are removed as the pruning ratio increases. Aggressive pruning will result in the negative impact on training performance with respect to the number of rounds. On the other hand, we report the accuracy results given the same time budget in Fig. 2(b), where the accuracy first increases and then decreases with the pruning ratio varying from 0 to 0.9. When the pruning ratio is relatively small, the pruned model can achieve better accuracy than the original one whose pruning ratio is 0. The explanation for the above phenomenon is the training time is equal to the number of rounds multiplied by the per-round completion time. Model pruning with a smaller ratio does not seriously hurt the model accuracy, and can reduce the per-round completion time [36], [37]. Thus under the given time, the pruned model can be trained for more rounds than the original model, resulting in better accuracy. According to the experimental results, the smaller pruning ratio ensures the model accuracy, but the computation and communication overhead is still high. On the contrary, the larger pruning ratio will contribute to less communication and computation overhead, and is more likely to deteriorate the model accuracy. The pruning ratio decision has a crucial impact on the trade-off between resource efficiency and training accuracy. However, it is difficult to capture the quantitative relationship among model pruning ratio, resource consumption and convergence performance.

Challenge 2. *How should we adaptively determine different pruning ratios for heterogeneous workers?* Considering edge heterogeneity, the pruning ratios should be adaptive to the training process of different workers. If the PS determines a uniform pruning ratio or sends a uniform pruned model to all workers, the high-performance workers will compromise to the low-performance ones and perform local training with large pruning ratios, leading to resource waste and accuracy loss. The low-performance workers are still the bottleneck of model training. However, due to the large solution space, it is challenging to simultaneously determine different pruning ratios for heterogeneous workers. Furthermore, the edge nodes probably are unwilling to share their private information such as computing power with the PS. The pruning ratio decision would be more complicated due to the lack of prior knowledge.

C. *Multi-Armed Bandit Based Algorithm*

To address the challenges above, we propose a Multi-Armed Bandit (MAB) based online learning algorithm to adaptively determine the pruning ratios for heterogeneous workers without any prior knowledge of their capabilities.

The decision optimization problem can be modeled as an MAB problem, where the PS and the pruning ratios can be regarded as the player and the arms, respectively. In each round, the PS makes the decision about which arm of the bandit is pulled, and a reward in (6) will be received after the decision. Upper Confidence Bound (UCB) policy is widely used to address the MAB problem [50]. Traditional UCB policy with the discrete arm setting only has a finite set of choices. However, the value range of pruning ratio in FedMP is a continuous space so that the arm space is infinite. In this article, we extend the UCB policy to the case where the arms are continuous, namely Extended Upper Confidence Bound (E-UCB), as described in Algorithm 2.

Specifically, E-UCB will create agents for the workers participating in FL and use the decision tree to adaptively learn arm space partitions. In round k, each agent maintains a sequence of finite partitions of the arm space $\mathbf{P_k} = \{P_k^1, P_k^2, \ldots, P_k^{l_k}\}$ with $\cup_{j=1}^{l_k} P_k^j = [0,1)$ and $\mathbf{P_0} = \{[0,1)\}$ (Line 1), where l_k is the number of partition regions and may vary over time. These partition regions could be viewed as leaves in the decision tree.

Algorithm 2: Extended Upper Confidence Bound (E-UCB) for Worker n.

```
1: Initialize $\mathbf{P_0} = \{[0,1)\}$;
2: for Each round $k \in \{0, 1, \ldots, K-1\}$ do
3:     for Each partition region $P_k^j \in \mathbf{P_k}$ do
4:         Calculate upper confidence bound
           $U_k(P_k^j) = \overline{R}_k(\lambda, P_k^j) + c_k(\lambda, P_k^j)$;
5:     Choose the partition region $P_k^j = \arg\max U_k(P_k^j)$;
6:     Select the pruning ratio $\alpha_n^k$ from $P_k^j$;
7:     if The diameter of $P_k^j$ is larger than $\theta$ then
8:         Split $P_k^j$ with $\alpha_n^k$ to form $\mathbf{P_{k+1}}$;
9:     else
10:        $\mathbf{P_{k+1}} = \mathbf{P_k}$;
11:    Record the completion time $T_n^k$ of worker $n$;
12:    Calculate the reward $R(\alpha_n^k)$;
```

E-UCB treats the problem as a finite-arm bandit problem with respect to the partition regions, and chooses the partition region that maximizes the upper bound of a confidence interval for the expected reward (Lines 3-5). All arms within the chosen region are treated as the same and selected randomly (Line 6). After determining the pruning ratio α_n^k for worker n, the chosen partition region is split into multiple regions to form the new partition $\mathbf{P_{k+1}}$ (Line 8), and the decision tree grows adaptively as the algorithm runs. E-UCB stops extending the decision tree once leaf diameters (region diameters) are below θ (Line 7), which represents the granularity of pruning ratio exploration. We will show the influence of θ on the model training process through experiments in Section VI. After receiving all sub-models from workers, the PS can get the completion time T_n^k of worker n (Line 11). The agent will observe a reward from its interactive environment (Line 12), which has a crucial impact on future decisions. The reward in E-UCB can be defined as

$$R(\alpha_n^k) = \frac{\Delta Loss}{|T_n^k - \frac{1}{N}\sum_{n'=1}^{N} T_{n'}^k|}, \tag{6}$$

where the numerator indicates the contribution of the workers to model convergence. The denominator represents the gap between the completion time of worker n and the average completion time. A smaller gap means that the selected pruning ratio fits the worker's capabilities better, leading to a higher reward.

To choose the most appropriate partition region, the agent should pursue a balance between the exploitation of arms that perform well in the past and the exploration of arms that might return higher rewards in the future. E-UCB uses the following definition of upper confidence bound to address the *exploitation versus exploration* challenge.

(1) Exploitation: Let $N_k(\lambda, P_k^j) = \sum_{s=1}^{k-1} \lambda^{k-s} \mathbb{1}_{\{\alpha_n^s \in P_k^j\}}$ record the number of times that the partition region P_k^j is chosen, where $\lambda \in (0,1)$ is a discount factor. The indicator function $\mathbb{1}_{\{\alpha_n^s \in P_k^j\}}$ is 1 when $\alpha_n^s \in P_k^j$ and 0 otherwise. The discounted

empirical average is given by

$$\overline{R}_k(\lambda, P_k^j) = \frac{1}{N_k(\lambda, P_k^j)} \sum_{s=1}^{k-1} \lambda^{k-s} R(\alpha_n^s) \mathbb{1}_{\{\alpha_n^s \in P_k^j\}} \tag{7}$$

The exploitation item averages all past rewards with a discount factor λ while giving more weight to recent observations.

(2) Exploration: If the agent always selects the pruning ratio from the partition region which it currently regards as the best, it might miss another partition region with a higher expected reward. Thus E-UCB adds an exploration item to the upper bound. Let $n_k(\lambda) = \sum_{j=1}^{l_k} N_k(\lambda, P_k^j)$ hold and the discounted padding function is defined as

$$c_k(\lambda, P_k^j) = \sqrt{\frac{2\log n_k(\lambda)}{N_k(\lambda, P_k^j)}} \tag{8}$$

where $N_k(\lambda, P_k^j)$ records the number of times that the partition region P_k^j is chosen and $n_k(\lambda)$ represents the total number of times that all partition regions are chosen. When the partition region P_k^j is not chosen for a long time, $N_k(\lambda, P_k^j)$ remains the same and $n_k(\lambda)$ keeps increasing, thus expanding the exploration item. As a result, this partition region P_k^j has a greater chance of being selected in the subsequent training process. The discounted padding function enables the agent to explore other potential partition regions, which prevents getting stuck in the local optimum.

The upper confidence bound in E-UCB is defined as

$$U_k(P_k^j) = \overline{R}_k(\lambda, P_k^j) + c_k(\lambda, P_k^j) \tag{9}$$

The region P_k^j with the largest U_k will be chosen. The key idea behind E-UCB is that the agent always chooses the best partition region that fits the worker's capabilities based on the estimated reward. The regions with the largest U_k are carefully split while worse regions are not explored anymore. By fitting the decision tree that grows adaptively, the partitions can be learned from last experience.

With the knowledge of heterogeneous capabilities, some more straightforward methods can be used to determine the pruning ratios. However, it is usually impractical for the PS to obtain these private information of workers in edge computing. To this end, E-UCB adaptively learns the partitions and determines the pruning ratios only through the completion time of workers without requiring any prior knowledge of heterogeneous capabilities. The performance of designed arm-pulling policy is measured by regret, which is defined as the difference between the expected reward by playing the optimal arms and that obtained by the given policy. The goal of E-UCB algorithm is to minimize the expected regret, i.e.,

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=0}^{K-1} (R(\alpha^*) - R(\alpha_n^k)) = 0 \tag{10}$$

where α^* is the optimal pruning ratio. Obviously, bounding the expected regret is essentially equivalent to controlling the number of pulling the sub-optimal arms. The algorithm performance can be guaranteed according to [51]. Considering the system performance, E-UCB algorithm only incurs negligible overhead on each worker while significantly speeding up the training process. The algorithm overhead will be quantified in the experiments of Section VI.

V. Extension to P2P Setting

In this section, we show how FedMP extends to the P2P setting and propose an algorithm of pruning ratio decision and neighbor selection for decentralized FL.

A. Workflow of P2P-FedMP

In the PS based FL framework, all distributed workers have to communicate with the PS, which could potentially incur the single-point of failure due to network jam or malicious attacks. This bottleneck makes the PS architecture difficult to scale to a large number of workers. To overcome the shortcomings of the PS architecture, decentralized FL is well worth investigating, where each participating worker trains local model and exchanges parameters with neighbors in the peer-to-peer (P2P) manner to reach a consensus. There is no central server to collect or distribute models under the P2P setting. Therefore, it can avoid the communication hot-spot and potential disastrous failure of the PS.

To this end, we extend FedMP to the P2P setting without the assistance of the PS (referred to as P2P-FedMP). At the beginning of round k, each worker n determines its pruning ratio α_n^k. Then worker n performs structured pruning for its local model and obtains a more compact local model $\hat{\mathbf{x}}_n^k$ that has much fewer parameters compared to the original model. After model pruning, the local models of workers in the P2P topology have different structures and sizes, which fit their own capabilities and thus mitigate the effect of stragglers on training performance. In each round, the pruned model $\hat{\mathbf{x}}_n^k$ is trained τ iterations over the local dataset D_n.

Unlike the PS architecture where the central server is responsible for collecting and aggregating the local models, the workers in decentralized FL exchange the models along the P2P links. The link speeds connecting the workers are mainly determined by the geographic distance. It is reported that the link speed between geographically-close workers can be up to 12× faster than that between distant ones [52]. Besides, the link speeds are usually time-varying due to workers' mobility or background noise. A high-speed link at one time may get slow at other time. In such heterogeneous and dynamic P2P networks, exchanging models via low-speed links will severely slow down the training process. Therefore, considering the pruning ratios and network conditions, P2P-FedMP will dynamically select the communication neighbors for workers in each round. In our framework, the workers tend to communicate through high-speed links in the P2P networks to accelerate decentralized training. The detailed policy of pruning ratio decision and neighbor selection is presented in Section V-C.

Afterward, the workers exchange the pruned models with their selected neighbors. The indexes of the reserved parameters are also sent to the neighbors. We adopt a binary mask to record the indexes and the binary mask only uses 1 b to indicate whether the parameter is reserved, resulting in negligible communication overhead [14]. After receiving the parameters from the neighbors, worker n recovers the pruned models of its neighbors in terms of the received indexes. Finally, the recovered models are aggregated into the latest local model. Since the pruned models are trained and transmitted in the P2P topology, the computation and communication overhead are drastically reduced. Besides, the workers tend to select neighbors with short communication time, thus P2P-FedMP has cheaper scalability and higher training efficiency.

B. System Model

In decentralized FL, a set $\mathcal{N} = \{1, 2, \ldots, N\}$ of workers collaboratively train models under the P2P setting. The P2P topology can be expressed as an undirected graph. We use the symmetric adjacency matrix $\mathbf{A}^k = [a_{n,m}^k] \in \mathbb{R}^{N \times N}$ to denote the undirected graph. If there exists a communication link (n, m) between worker n and worker m in round k, we have $a_{n,m}^k = a_{m,n}^k = 1$; otherwise $a_{n,m}^k = a_{m,n}^k = 0$. Each worker only exchanges models with its neighbors instead of the PS. The neighbor set of worker n can be represented by $\mathcal{M}_n^k = \{m \in \mathcal{N} | a_{n,m}^k = 1\}$. The degree matrix $\mathbf{E}^k = [e_{n,m}^k] \in \mathbb{R}^{N \times N}$ is a diagonal matrix, where $e_{n,n}^k = |\mathcal{M}_n^k|$. Combining the adjacency matrix and the degree matrix, the Laplacian matrix $\mathbf{L}^k$ can be expressed as $\mathbf{L}^k = \mathbf{E}^k - \mathbf{A}^k$. According to the spectral graph theory [53], there are the following important properties about the Laplacian matrix:

- The eigenvalues of $\mathbf{L}^k$ are denoted by $\Phi_1(\mathbf{L}^k) < \Phi_2(\mathbf{L}^k) < \cdots < \Phi_N(\mathbf{L}^k)$, where $\Phi_i(\mathbf{L}^k)$ is the i-th smallest eigenvalue of matrix $\mathbf{L}^k$. Then we have $\Phi_1(\mathbf{L}^k) = 0$.
- $\Phi_2(\mathbf{L}^k) > 0$ if and only if the P2P topology is connected. A larger value of $\Phi_2(\mathbf{L}^k)$ implies a denser graph.

Furthermore, due to the communication resource constraints, the available bandwidth on each worker is limited in the P2P networks. Let $b_{n,m}^k$ be the bandwidth occupied by communication from worker n to worker m and B_n^k denote the available bandwidth of worker n in round k. We need to ensure that the bandwidth used to communicate with neighbors does not exceed resource constraints, i.e., $\sum_{m \in \mathcal{M}_n^k} b_{n,m}^k \le B_n^k$. The time $T_{n,m}^k$ for transmitting a pruned model from worker n to worker m is defined as

$$T_{n,m}^k = \frac{(1 - \alpha_n^k) \cdot W}{v_{n,m}^k} \tag{11}$$

where W is the size of the original model and $v_{n,m}^k$ is the link speed from worker n to worker m in round k. The completion time T_n^k of worker n in round k includes the computation time $T_{n,comp}^k$ for local updating and the communication time $\max_{m \in \mathcal{M}_n^k} T_{n,m}^k$ with neighbors, which can be expressed as

$$T_n^k = T_{n,comp}^k + \max_{m \in \mathcal{M}_n^k} T_{n,m}^k \tag{12}$$

In the synchronous manner, all decentralized workers are required to enter the next round simultaneously. Thus, the round completion time is $T^k = \max_n T_n^k$. Finally, the optimization problem of pruning ratios and communication neighbors under the P2P setting can be formulated as

$$\min \sum_{k=0}^{K-1} T^k$$
$$s.t. \begin{cases} F(\mathbf{x}^K) < \varepsilon & \\ \Phi_2(\mathbf{L}^k) > 0, & \forall k \\ \sum_{m \in \mathcal{M}_n^k} b_{n,m}^k \le B_n^k, & \forall n, \forall k \\ 0 \le \alpha_n^k < 1, & \forall n, \forall k \end{cases} \tag{13}$$

The first inequality guarantees the model accuracy. The second inequality ensures a connected topology in each round. The third inequality expresses the communication resource constraints and the fourth inequality bounds the range of the pruning ratio.

Algorithm 3: Pruning Ratio Decision and Neighbor Selection in Round k.

```
1: Initialize $\mathbf{A}^k$ with the fully-connected topology;
2: Determine the initial pruning ratio $\alpha_n^k$ by E-UCB;
3: Construct a list of candidate links based on $\mathbf{A}^k$;
4: **while** The candidate list is not empty **do**
5:     Calculate the communication time $T_{n,m}^k$ based on the updated pruning ratio $\alpha_n^k$;
6:     Choose the link $(n, m)$ with the longest time $T_{n,m}^{k,max}$ from the candidate list;
7:     Remove the link $(n, m)$ from the candidate list;
8:     **if** $\sum_{i \in \mathcal{M}_n^k} b_{n,i}^k > B_n^k$ or $\sum_{i \in \mathcal{M}_m^k} b_{m,i}^k > B_m^k$ **then**
9:         $a_{n,m}^k = a_{m,n}^k = 0$;
10:        **if** $\Phi_2(\mathbf{L}^k) \le 0$ **then**
11:            $a_{n,m}^k = a_{m,n}^k = 1$;
12:    Calculate the reward $R(\alpha_n^k)$ based on the updated topology $\mathbf{A}^k$;
13:    Select the pruning ratio $\alpha_n^k$ from the partition region $P_k^j$ with probability $z(P_k^j)$;
14: Obtain the final pruning ratio $\alpha_n^k$ and neighbor set $\mathcal{M}_n^k$;
```

C. Algorithm Design

Considering the inherent properties of decentralized FL paradigm, P2P-FedMP not only needs to determine the appropriate pruning ratios for workers but also selects communication neighbors for them. The pruning ratio decision and neighbor selection are interactive, i.e., their decisions have an impact on each other. On the one hand, the pruning ratio should match the link speeds of the selected neighbors, which prevents the stragglers from affecting training efficiency. On the other hand, the workers tend to select some neighbors with short communication time for model exchange according to the corresponding pruning ratios. The tightly coupled problem demonstrates the need for joint optimization, which brings additional challenges for algorithm design under the P2P setting.

To this end, we take into account interactive decisions on the basis of E-UCB and optimize the pruning ratios and communication neighbors alternately via fixed-point iterations [54]. Only by optimizing the relationship between the two, can the coupled problem be solved overall. In the alternate optimization process, we remove the bottleneck links that slow down the training process under the updated pruning ratio scheme. Then the pruning ratios are adjusted in time based on the received rewards to adapt to changes in network topology. In this way, the pruning ratio and communication neighbors adapt to each other to get the final decision, which will better accommodate heterogeneous and dynamic network conditions under the P2P setting.

In the above joint optimization process, the contribution of pruning ratio to model convergence is agnostic in advance of actual training. Thus we need to update the design of the reward function in E-UCB to overcome the unavailability of future training information. The reward function of the joint optimization process is

$$R(\alpha_n^k) = \frac{1-\alpha_n^k}{|T_n^k - \frac{1}{N}\sum_{n'=1}^{N} T_{n'}^k|} \tag{14}$$

A smaller pruning ratio does not seriously hurt model accuracy and thus gets higher rewards. Meanwhile, if the completion time under this pruning ratio deviates from other workers, it will have a negative impact on the received rewards. In other words, we expect slight pruning and a small gap in completion time. Traditional MAB methods compute the actual reward of an arm by averaging the received rewards. However, the link speeds are time-varying in decentralized FL paradigm. Correspondingly, the optimal pruning ratios change dynamically over time, which results in a non-stationary MAB problem [55]. Therefore, it is not rational to directly average the rewards as traditional MAB methods. To make the reward estimation more stable and accurate, P2P-FedMP adopts exponential moving average to concentrate more on the recent rewards while the weights of past rewards decay [56]. In round k, each agent maintains a sequence of finite partitions of the arm space $\mathbf{P_k} = \{P_k^1, P_k^2, \ldots, P_k^{l_k}\}$ with $\cup_{j=1}^{l_k} P_k^j = [0,1)$. In the joint optimization process, if $\alpha_n^k \in P_k^j$, the discounted empirical average of partition region P_k^j will be gradually updated as

$$\overline{R}_k(\lambda, P_k^j) = \overline{R}_k(\lambda, P_k^j) + \lambda(R(\alpha_n^k) - \overline{R}_k(\lambda, P_k^j)) \tag{15}$$

where λ is the same discount factor as the E-UCB algorithm.

The goal of our MAB algorithm is to maximize the total received rewards by balancing exploitation (the use of acquired information) and exploration (acquiring new information). If the player always plays the arm he currently thinks is best, he may miss identifying another arm that actually has a higher reward. Conversely, if the gambler explores the environment too frequently in search of profitable actions, he cannot accumulate as many rewards. In P2P-FedMP, we use the Boltzmann strategy [57] to pursue a judicious trade-off between exploration and exploitation. Specifically, the probability of choosing partition region P_k^j is derived as

$$z(P_k^j) = \frac{e^{\overline{R}_k(\lambda, P_k^j)}}{\sum_{j'=1}^{l_k} e^{\overline{R}_k(\lambda, P_k^{j'})}} \tag{16}$$

Based on the updated reward function, we describe the joint optimization process of pruning ratios and communication neighbors in Algorithm 3. In each round k, we initialize the pruning ratios of alternate decision process (Lines 4-13) by E-UCB, and the initial topology is the fully-connected topology [58]. According to the initial topology, we can construct a list of candidate links. In heterogeneous and dynamic P2P networks, the workers with low-speed links will become the bottleneck of decentralized learning and restrict the training process. P2P-FedMP aims to optimize the utilization of high-speed links to accelerate decentralized training. Considering both the pruning ratios and link speeds, we calculate the communication time between worker n and worker m as $T_{n,m}^{k,max} = \max\{T_{n,m}^k, T_{m,n}^k\}$. Then we identify the bottleneck link (n,m) with the longest communication time $T_{n,m}^{k,max}$ from the candidate list. On the premise of ensuring resource constraints, the link (n,m) will remove from the P2P topology $\mathbf{A}^k$, which contributes to improving training efficiency. It is worth noting that a connected topology is enough to guarantee the training convergence under the P2P setting [59]. Therefore, we only remove the links that will not affect the connectivity of the P2P topology. Based on the updated topology, the pruning ratio α_n^k receives the corresponding reward $R(\alpha_n^k)$. Then the selection probability of partition regions can be obtained according to the returned reward. To balance exploitation and exploration, we choose the pruning ratio α_n^k from the partition region P_k^j with the corresponding probability $z(P_k^j)$. Considering the updated pruning ratios, we can again identify the bottleneck link with the longest communication time from the remaining candidate links. Upon traversing the candidate list, we generate the final pruning ratio α_n^k and neighbor set $\mathcal{M}_n^k$ for worker n in round k.

VI. Evaluation

A. Experimental Setup

Implementation. We implement a prototype of FedMP by employing an AMAX deep learning workstation as the PS and 30 NVIDIA Jetson TX2 devices as the workers. The workstation is equipped with one Intel Xeon Octa-core E5-2620 v4 processor and 4 NVIDIA GeForce RTX 2080Ti GPUs with 11 GB GDDR6 memory each. Every Jetson TX2 has 8 GB LPDDR4 main memory and features an NVIDIA Denver2 (dual-core) CPU cluster, an ARM Cortex-A57 (quad-core) CPU cluster and an NVIDIA Pascal GPU with 256 CUDA capable cores. Our software implementation is based on Pytorch v1.6, but can be easily extended to other ML frameworks such as TensorFlow.

Heterogeneous Workers. we consider both computation heterogeneity [60] and communication heterogeneity [61] in the experiments. (1) *Computation heterogeneity:* We set different computing modes for each Jetson TX2 to reflect the heterogeneous computation capabilities of workers, as summarized in Table II. From modes 0 to 3, the computation capability decreases gradually. (2) *Communication heterogeneity:* The communication capabilities of workers may also be different in practice. The workers usually connect to the PS via wireless links in edge computing, and the signal strength of wireless links may vary with the distance [62]. Hence, we place Jetson TX2 devices at different locations to simulate communication

TABLE II
DIFFERENT COMPUTING MODES FOR JETSON TX2

Mode	Denver2 (dual-core)	Cortex-A57 (quad-core)	GPU
0	2.0 GHz×2	2.0 GHz×4	1.30 GHz
1	—	2.0 GHz×4	1.12 GHz
2	1.4 GHz×2	1.4 GHz×4	1.12 GHz
3	—	1.2 GHz×4	0.85 GHz

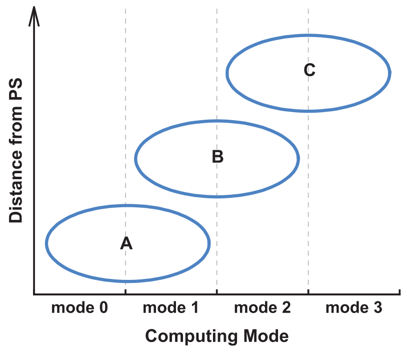


Fig. 3. Workers with different computing modes and locations.

heterogeneity. As shown in Fig. 3, based on different computing modes (X-axis) and locations (Y-axis), we partition the devices into three clusters (i.e., A, B, C). By selecting workers from these clusters, we can create different heterogeneous scenarios.

Models and Datasets. We evaluate the effectiveness of FedMP on four typical FL tasks: (1) *CNN on MNIST*, (2) *AlexNet on CIFAR-10*, (3) *VGG-19 on EMNIST* and (4) *ResNet-50 on Tiny-ImageNet*. The MNIST dataset contains 60,000 training and 10,000 test greyscale images of handwritten digits with size 28×28. The CIFAR-10 dataset includes 60,000 32×32 color images (50,000 for training and 10,000 for testing) of ten different types of objects. The EMNIST dataset is composed of 731,668 training images and 82,587 testing images which draw from 62 classes of objects [63]. The Tiny-ImageNet dataset contains 200 classes, where each class has 500 training images and 50 test images. The dimension of each image is 64×64×3. The CNN has two 5×5 convolutional layers, a fully-connected layer with 256 units, and a softmax output layer with 10 units [5].

Data Distribution. Considering that the workers in FL collect data from their physical locations directly, the data samples among workers are usually not independent and identically distributed (non-IID). As a measurement of non-IIDness in data distribution among workers, we use y to define the non-IID level.

- For MNIST and CIFAR-10: The non-IID level of y indicates that $y\%$ of the data on each worker belong to one label and the remaining data belong to other labels [64]. As a special case, we use the non-IID level of 0 to denote IID data distribution among workers.
- For EMNIST and Tiny-ImageNet: The non-IID level of y indicates that each worker lacks y classes of data samples, which is similar with the setting of [65], [66], [67].

Baselines. We compare our framework with the following baselines:

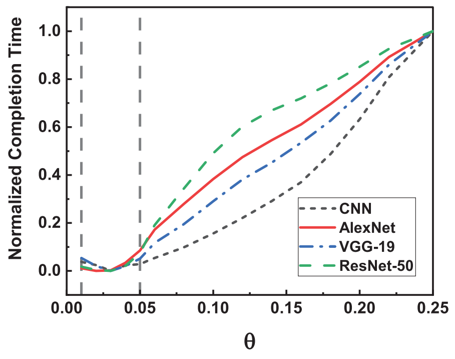


Fig. 4. Effect of pruning granularity θ on training performance.

- *Syn-FL* [3] transmits and trains the entire models. The PS aggregates the parameters after all of the heterogeneous workers have finished local updates.
- *UP-FL* [27] determines a uniform pruning ratio for all workers in each round, and the pruning ratio may vary in different rounds.
- *FedProx* [29] allows participating workers to perform different numbers of local iterations based on their heterogeneous capabilities.
- *FlexCom* [33] considers heterogeneous communication condition and enables flexible communication compression, which allows heterogeneous workers to compress the gradients to different levels before uploading.

Default Settings. Unless noted otherwise, the number of workers is 10 throughout the experiments. Half of them are selected from cluster A, and half are selected from cluster B. The data samples are assigned to each worker uniformly. We set the discount factor λ as 0.95 [56]. For a fair comparison, the baselines are implemented on the same platform as FedMP.

B. Experimental Results

1) Effect of Pruning Granularity θ: The parameter θ represents the granularity of pruning ratio exploration in E-UCB. In Section IV-C, E-UCB uses the decision tree to adaptively learn arm space partitions and determine the pruning ratios. The decision tree stops growing when region diameters are below θ. To investigate the effect of pruning granularity θ on training performance, we measure the completion time of different models to reach the target accuracy (e.g., 90% for CNN, 80% for AlexNet, 80% for VGG-19, 45% for ResNet-50) when θ varies from 0.01 to 0.25. For the sake of comparison, we normalize the completion time, and the results of the four models are shown in Fig. 4.

For all the four models, when the parameter θ is small (i.e., $\theta \in [0.01, 0.05]$), it only has a minor impact on training performance. As the parameter θ gets large (i.e., $\theta \in (0.05, 0.25]$), the completion time increases drastically. The reason lies in that large pruning granularity (i.e., larger θ) may cause the pruning ratio selected in a large range, leading to poor training performance. Therefore, we tend to explore the pruning ratio with a relatively small granularity. On the other hand, since model parameters

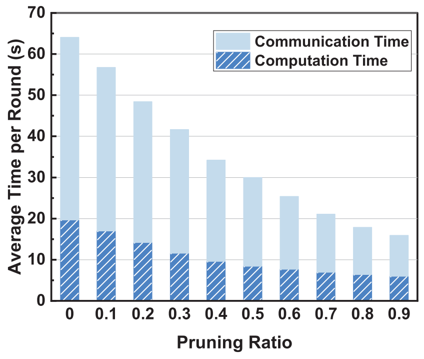


Fig. 5. Effect of different pruning ratios on training time.

TABLE III
ACCURACY OF DIFFERENT FL METHODS IN A GIVEN TIME

Models	Time Budget	Syn-FL	UP-FL	FedProx	FlexCom	**FedMP**
CNN	20000s	93.83%	94.31%	95.82%	96.21%	**97.17%**
AlexNet	30000s	81.59%	81.74%	81.78%	81.91%	**82.34%**
VGG-19	50000s	85.04%	84.93%	85.15%	85.33%	**85.66%**
ResNet-50	100000s	47.15%	46.43%	47.55%	47.37%	**47.85%**

have significant redundancy [37], if the pruning granularity is small enough, it does not affect the completion time too much. That is, $\theta \in [0.01, 0.05]$ achieves almost the same performance, which is demonstrated for different models. Motivated by this, choosing θ from [0.01,0.05] is modest for different scenarios.

2) Overall Effectiveness: We investigate the performance of FedMP and baselines when they are deployed across heterogeneous workers. Fig. 5 shows that the average per-round time of computation and communication decreases with the increase of pruning ratio. The reason lies in that model pruning significantly reduces the complexity of DNNs, leading to time reduction in computing and delivering local updates. Table III compares the test accuracy that different FL methods can achieve in a given time. We note that FedMP always converges to the similar accuracy as Syn-FL, and achieves higher accuracy in a given time on all the four models. It indicates that distributed model pruning with appropriate pruning ratios does not hurt the model accuracy and can achieve the goal of effective FL.

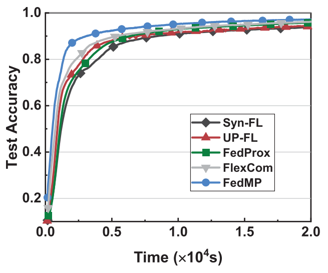
Fig. 6 shows the test accuracy with time passed on different datasets. From these results, we make three major observations. First, FedMP substantially outperforms Syn-FL. For example, FedMP takes 10,906 s to achieve 80% accuracy for AlexNet on CIFAR-10, while Syn-FL takes 24,017 s. FedMP provides 2.2× speedup compared to Syn-FL. Since Syn-FL does not adopt any parameter reduction method, the over-parameterized DNNs incur a great deal of computation and communication overhead. In contrast, the pruned models are transmitted and trained in FedMP, which reduce both communication and computation time while ensuring the accuracy.

Second, FedMP converges much faster than UP-FL on all the four datasets, and achieves a speedup of nearly 2× for AlexNet on CIFAR-10. This is because UP-FL accelerates model training by uniform model pruning, but ignores the heterogeneity of edge nodes. The pruned models are the same across workers, so that the workers with weak capabilities delay the global aggregation. However, FedMP adaptively prunes the global model according to the heterogeneity of edge nodes. The pruned models are customized for workers' capabilities, which accelerates the DNNs training by a significant margin.

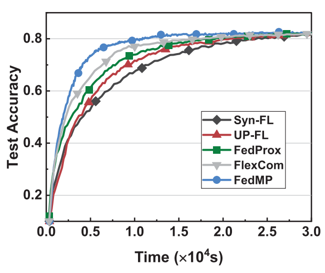
Third, FedMP has a clear and consistent advantage on training speed over the other heterogeneity-aware baselines (i.e., FedProx and FlexCom). Particularly, FedMP improves the performance over FedProx by 2.0× for CNN, 1.8× for AlexNet, 1.2× for VGG-19, and 1.1× for ResNet-50. Compared to FlexCom, FedMP achieves about 1.8×, 1.6×, 1.2× and 1.2× speedup, correspondingly. Although FedProx allows heterogeneous workers to perform different numbers of local iterations, it does not incorporate model compression or pruning techniques. As a result, the computation and communication overhead in FedProx is still large, leading to poor training efficiency. FlexCom reduces the communication overhead by assigning different compression ratios to heterogeneous workers. However, each worker trains the same local model, which cannot reduce the computation overhead and cope with the computation heterogeneity. Consequently, local training is still very slow due to the large amount of computation overhead. By contrast, the excellent performance of FedMP can be explained by the superiority of adaptive model pruning, which allows each worker to train and transmit the pruned model under its resource constraints. Therefore, FedMP can reduce both computation and communication overhead, thereby accelerating the training process compared with the baselines.

3) Effect of Heterogeneity: To understand how FedMP performs under heterogeneous scenarios, we deploy FedMP and baselines across the workers with different heterogeneity levels, i.e., *Low*, *Medium*, *High*. For *Low*, we select 10 workers from cluster A. For *Medium*, we select 5 workers from cluster A, and 5 workers from cluster B. For *High*, we select 3 workers from cluster A, 3 workers from cluster B, and 4 workers from cluster C. For our experiments, the desired target accuracies of CNN, AlexNet, VGG-19 and ResNet-50 are 90%, 80%, 80% and 45%, respectively. Fig. 7 shows the required time to reach the target accuracy under different heterogeneous scenarios.

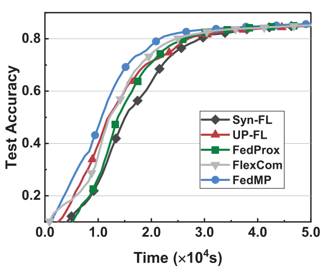
We observe that, from *Low* to *High*, the required time to reach the target accuracy increases accordingly. This is expected because less capable workers are introduced into the system. In the synchronous FL, the PS performs global aggregation after receiving all the local models. Thus the total latency is determined by the "slowest" worker, increasing the completion time of all methods. Nevertheless, FedMP still takes less time to reach the target accuracy compared with the baselines. When training AlexNet on CIFAR-10 in the heterogeneity level of *High*, FedMP achieves 3.6×, 3.0×, 2.3×, and 2.0× speedup compared to Syn-FL, UP-FL, FedProx, and FlexCom, respectively. Moreover, the performance gap can be enlarged with the increase of heterogeneity level. For example, FedMP improves the performance over Syn-FL by 1.3× in *Low*, 2.8× in *Medium*, and 4.1× in *High* for CNN on MNIST. This is because FedMP determines the appropriate pruning ratios for the weak workers that are newly introduced into the system, so that each worker



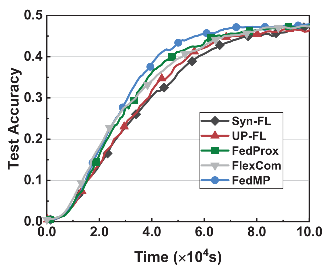
(a) CNN on MNIST



(b) AlexNet on CIFAR-10

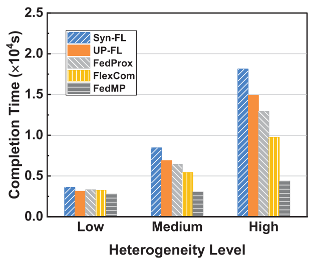


(c) VGG-19 on EMNIST

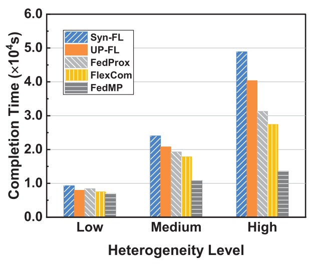


(d) ResNet-50 on Tiny-ImageNet

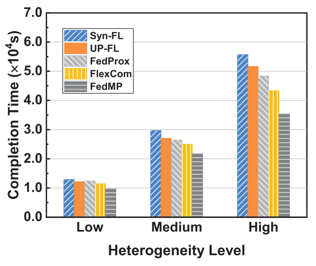
Fig. 6. Test accuracy of different FL methods.



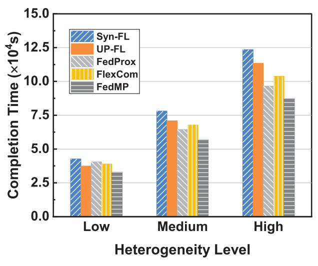
(a) CNN on MNIST



(b) AlexNet on CIFAR-10

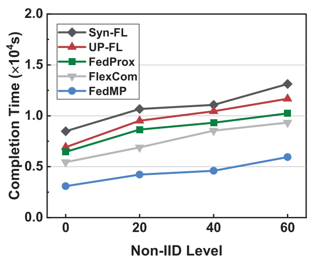


(c) VGG-19 on EMNIST

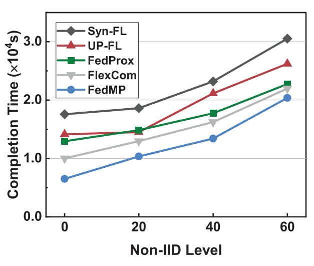


(d) ResNet-50 on Tiny-ImageNet

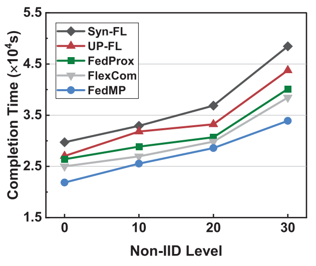
Fig. 7. Completion time under different heterogeneous scenarios.



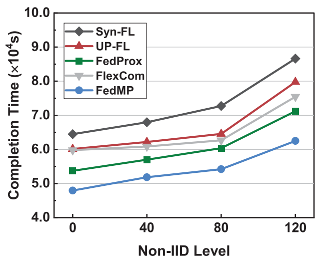
(a) CNN on MNIST



(b) AlexNet on CIFAR-10



(c) VGG-19 on EMNIST



(d) ResNet-50 on Tiny-ImageNet

Fig. 8. Completion time under different non-IID levels.

can still train the sub-model that best suits its capability, leading to a slight increase in completion time. These results demonstrate that FedMP is robust for different heterogeneous scenarios and can fully utilize the limited resources of workers.

4) Effect of non-IID Data: While the previous experiments are based on uniform data partitioning, we now discuss the impact of non-IID data on training performance. Fig. 8 shows the required time for FedMP and baselines to achieve the target accuracy in different non-IID levels. We set the target accuracy of CNN, AlexNet, VGG-19 and ResNet-50 as 90%, 77%, 80%, and 42%, respectively. From the results, we note that the required time for all methods to achieve the target accuracy increases with the increase of non-IID levels. Since the local models trained on non-IID data are different from each other, aggregating these divergent models may degrade the training performance and result in more communication rounds until convergence. Nevertheless, for all the four datasets, FedMP still outperforms the baselines in different non-IID levels. For example, even in the non-IID level of 30, FedMP can reduce the completion time by 30%, 23%, 16% and 12% compared to Syn-FL, UP-FL, FedProx and FlexCom for VGG-19 on EMNIST. These results demonstrate the effectiveness of our framework even under non-IID data distribution.

5) Effect of Worker Number: We evaluate the scalability of FedMP with different numbers of workers. In this set of experiments, half of the workers participating in FL are selected from cluster A, and half are from cluster B. We compare the required time of FedMP and baselines to achieve the target accuracy with the number of workers varying from 10 to 30. The target accuracy is the same as for Fig. 7(b). The results of training AlexNet on CIFAR-10 are shown in Fig. 9. The completion time of FedMP increases slightly with the increasing number of workers. When the number of workers is 30, FedMP still provides 2.4×, 2.0×, 2.0×, and 1.6× speedup compared to Syn-FL, UP-FL, FedProx, and FlexCom, indicating that our design is superior in scalability over other baselines.

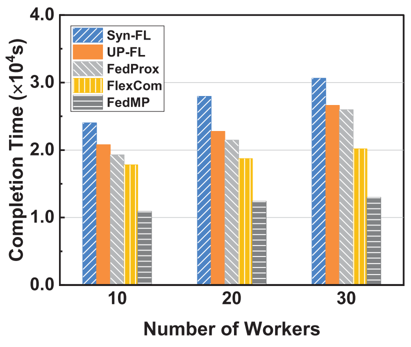


Fig. 9. Completion time with different numbers of workers.

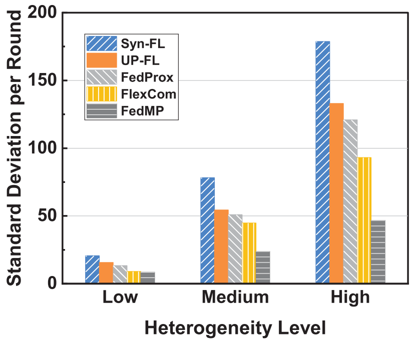


Fig. 11. Standard deviations under different heterogeneous scenarios.

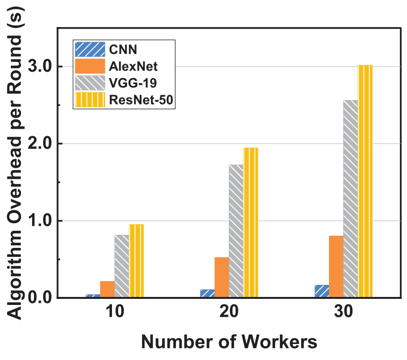


Fig. 10. Average algorithm overhead with different numbers of workers.

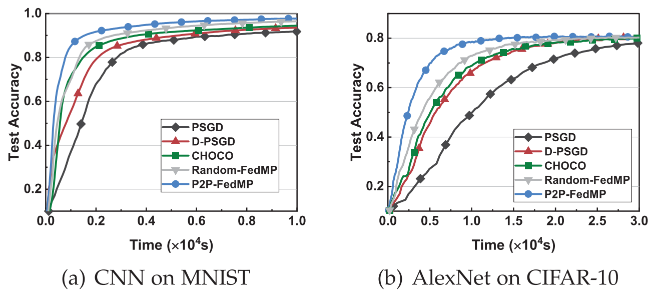


Fig. 12. Test accuracy of different decentralized methods.

Next, we quantify the algorithm overhead which plays a critical role in practical deployment. We measure the average per-round algorithm overhead including the pruning ratio decision time and model pruning time. The results for different numbers of workers are shown in Fig. 10. Apparently, the time overhead increases with the increasing number of workers. However, the maximum time overhead in our experiments is far less than the transmission time and local training time (e.g., hundreds of seconds), thus can be ignored. These results show that FedMP can be deployed in large-scale scenarios, with a small overhead in exchange for substantial training speedup.

6) Effect of E-UCB: In Section IV-C, we propose an MAB based online learning algorithm, called E-UCB, to adaptively determine different pruning ratios for workers. To show high efficiency of our proposed algorithm, we compare the standard deviation of the round completion time for workers before and after adopting E-UCB. The standard deviation is the arithmetic square root of the variance and reflects the degree of dispersion of the workers' round completion time. We measure the standard deviation of the workers' completion time in each round and then calculate their average. The results for AlexNet on CIFAR-10 are shown in Fig. 11, which is obtained by deploying FedMP and baselines across the workers with different heterogeneity levels. As the heterogeneity level increases (from *Low* to *High*), the capability gap between different workers becomes larger. The completion time of workers in each round becomes more dispersed, and thus the standard deviations of the five methods gradually increase. Nevertheless, FedMP still has the smallest standard deviation of the round completion time for workers compared to the baselines. For example, FedMP reduces the standard deviation by 58.1%-73.8% compared to Syn-FL. The reason is that adaptive pruning ratios obtained by E-UCB allow each worker to train a sub-model that fits its own capabilities, thereby mitigating the effect of stragglers on latency. After adopting E-UCB, the round completion time for workers becomes more concentrated. The above results show the effectiveness of our proposed pruning ratio decision algorithm.

7) Evaluation of P2P-FedMP: To avoid the possible bottleneck of the PS, we extend FedMP to P2P setting (i.e., P2P-FedMP). Since there is no central server or global model in decentralized FL paradigm, we introduce the following decentralized baselines for performance comparison:

- *PSGD* [68] trains and exchanges the complete model on each worker with all other workers.
- *D-PSGD* [49] trains and then exchanges the complete model with neighbors in the ring topology, which is constructed by arranging the N workers in the order of $1 \rightarrow 2 \rightarrow \cdots \rightarrow N \rightarrow 1$.
- *CHOCO* [69] as a gossip-based stochastic optimization algorithm employs compression techniques to reduce communication overhead of decentralized training.
- *Random-FedMP* is a simplified version of P2P-FedMP, where E-UCB algorithm is directly deployed on each worker and the communication topology is randomly constructed in each round.

We first compare the overall training performance of P2P-FedMP with decentralized baselines. Fig. 12 plots the test accuracy of different methods with respect to training time. We

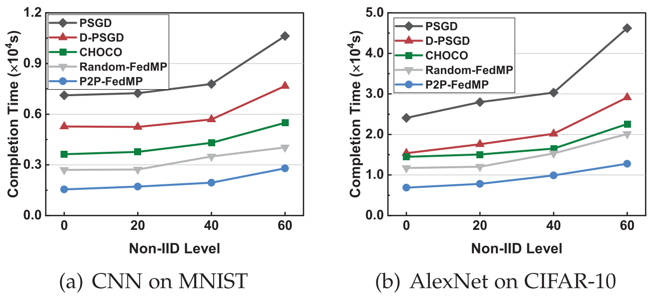


Fig. 13. Completion time of decentralized methods under different non-IID levels.

observe that P2P-FedMP converges faster and attains higher test accuracy after a fixed training time compared to baselines. For example, after training for 10,000 s, the accuracy of P2P-FedMP is 97.67% for CNN on MNIST while that of PSGD, D-PSGD, CHOCO, and Random-FedMP is 91.98%, 93.60%, 94.51%, and 96.50%, respectively. Besides, our framework can drastically reduce the completion time to achieve the target accuracy. In particular, P2P-FedMP takes 9,336 s to reach 78% accuracy for AlexNet on CIFAR-10. The completion time of PSGD, D-PSGD, CHOCO, and Random-FedMP is 29,825 s, 19,891 s, 19,456 s, and 15,941 s, respectively. The corresponding speedup provided by P2P-FedMP is 3.2×, 2.1×, 2.1×, and 1.7×.

The explanation for these observations is that each worker in PSGD and D-PSGD sends/receives the complete models to/from all its neighbors, leading to massive resource overhead. Their topologies are fixed and cannot adapt to network dynamics. Moreover, the sub-models in P2P-FedMP are individually pruned based on workers' capabilities while the compressed models in CHOCO ignore the device heterogeneity and incur high computation overhead. Although Random-FedMP assigns different pruning ratios to heterogeneous workers, the randomly constructed topology still suffers from the low-speed links and substantially restricts the training process. On the contrary, P2P-FedMP jointly optimizes the pruning ratios and communication neighbors considering the inherent properties of decentralized FL paradigm. The interactive decision of our algorithm enables the suitable sub-models to be trained and transmitted over the high-speed links, which contributes to reducing both computation and communication overhead in dynamic heterogeneous P2P networks, thus accelerating decentralized training.

We further evaluate the performance of P2P-FedMP in the non-IID scenario. Fig. 13 depicts the completion time for reaching the target accuracy under different non-IID levels. The target accuracies of CNN and AlexNet are set as 90% and 75% since they are the accuracies that all methods can achieve. We note that the superiority of our solution is still significant under high non-IIDness of data distribution. Specifically, when the non-IID level of MNIST increases from 0 to 60, P2P-FedMP provides up to 4.6× speedup, 4.2× speedup, 4.0× speedup, and 3.8× speedup compared with the decentralized baselines. For AlexNet on CIFAR-10, the corresponding speedup is 3.5×, 3.6×, 3.1×, and 3.6×. The above results verify the fact that the extension of FedMP is still effective under the P2P setting.

TABLE IV
PERPLEXITY OF DIFFERENT FL METHODS IN A GIVEN TIME AND SPEEDUP FOR REACHING TARGET PERPLEXITY

Methods	Perplexity (validate, test)	Speedup
Syn-FL	(156.35, 148.15)	1.0×
UP-FL	(158.94, 149.81)	0.8×
FedMP	(155.47, 146.95)	1.6×

VII. DISCUSSION

In this section, we discuss the adaptability and potential extensions of FedMP. FedMP can accommodate diverse neural networks by easily replacing different pruning strategies. Given some other neural networks, the PS still determines different pruning ratios for heterogeneous workers even without knowing any prior knowledge of their capabilities. According to the pruning ratios, the PS performs distributed model pruning with the specialized strategy so that each worker only trains and transmits a pruned model suiting its own capability. Under these circumstances, FedMP can reduce computation and communication overhead for different models.

We take Recurrent Neural Networks (RNNs) as an example to show the applicability of FedMP to other models. Compared with Convolutional Neural Networks (CNNs), pruning RNNs is more challenging. Since a recurrent unit is shared across all the time steps in sequence, independently removing the unit will result in mismatch of dimensions and then inducing invalid recurrent units. Following the intrinsic sparse structure method [70], we update the pruning strategy in Section III-B and keep other designs unchanged. Specifically, we remove weights associated with one component of intrinsic sparse structures, and then the sizes/dimensions (of basic structures) are simultaneously reduced by one. As a result, the obtained RNN has the original schematic with dense connections but with smaller sizes of these basic structures.

To study the benefits of FedMP, we train a RNN with two stacked LSTM layers on the Penn TreeBank dataset [71]. The performance of the models is measured by the metric of perplexity, which is the exponent of cross-entropy loss. The results are reported in Table IV. We note that FedMP has lower perplexity in a given time compared to baselines and provides 1.6× speedup for reaching the target test perplexity (e.g., 150). These results prove that FedMP can achieve efficient FL for diverse neural networks.

VIII. RELATED WORK

A. Federated Learning for Resource Constraints

To make full use of isolated data from IoT devices, FL has been proposed to enable collaborative model training over multiple workers without leaking their respective local data [3]. However, due to the constrained resources in edge computing, FL imposes massive computation and communication overhead, which limits its efficiency in practical deployment. Some previous works have made efforts to alleviate the communication burden without sacrificing model performance. FedAvg [3] and its variants [5] enlarge the communication interval to significantly reduce the communication overhead. These proposals allow workers to train local models for multiple iterations before global aggregation instead of aggregating local updates every iteration, thereby reducing total communication rounds.

Another natural solution to ease the communication overhead is to reduce the size of transmitted data. The compression strategies in FL can be roughly divided into two categories: quantization [17], [18], [19], [20], [21] and sparsification [22], [23], [24], [25]. Specifically, quantization aims to represent each original element of parameters by using fewer number of bits. Alistarh et al. propose QSGD [18], where the workers can trade-off the number of bits communicated per round with the variance added to the process. SignSGD [19] sends the sign of the local updates to the PS and aggregates the gradient signs using majority voting, which can achieve 32 times less communication compared with the standard FL method. Considering that signSGD does not converge to the optimum, Karimireddy et al. [20] present an error-feedback mechanism to overcome the intrinsical bias of signSGD. FedPAQ [21] is proposed to periodically average the local updates and quantize the updates before uploading to the PS. On the other hand, sparsification only transmits a subset of important elements from the original model parameters. Han et al. [22] present a fairness-aware bidirectional top-k gradient sparsification approach where the near-optimal k was determined by online learning techniques. Sun et al. [23] propose a General Gradient Sparsification (GGS) framework consisting of gradient correction and Batch Normalization update with local gradients. Sattler et al. [25] introduce sparse ternary compression (STC) that can compress both upstream and downstream communications as well as ternarization and optimal Golomb encoding of the weight updates. However, the above works only focus on improving communication efficiency and cannot reduce computation overhead. In fact, the computation capabilities of workers in FL are also limited, and thus huge computation overhead hinders efficient FL over resource-constrained workers.

In addition, the workers participating in training need to share the limited resources in FL. The resource allocation problems have been studied by a series of works. Tran et al. [72] jointly optimize the CPU frequency, transmit power and model accuracy to minimize the weighted sum of energy cost and learning time. Yang et al. [73] propose an iterative algorithm to address the problem of energy-efficient transmission and computation resource allocation. Yao et al. [74] investigate both the CPU frequency and wireless transmission power control to balance the trade-off between the energy consumption and learning time in fog-aided IoT networks. The authors in [75] develop resource allocation mechanisms for delay-constrained FL in the non-orthogonal multiple access (NOMA) enabled and relay-assisted IoT networks. Wu et al. [76] jointly optimize the radio resource allocation for NOMA transmission and computation resource allocation for FL training, with the objective of minimizing the total energy consumption and convergence latency. Though the above works provide some novel insights, they do not consider any parameter reduction techniques. Training and transmitting the complete models incur huge computation and communication overhead. In these methods, the PS sends the identical model to heterogeneous workers. Consequently, the workers with poor capabilities make it difficult to train the over-parameterized DNNs. They may become the bottleneck of model aggregation and eventually degrade the training performance of FL. It is worth noting that our work is orthogonal to the resource allocation methods, which can be applied alongside FedMP to further boost training efficiency.

B. *Federated Learning for System Heterogeneity*

The configuration of workers may differ due to variability in CPU, GPU, memory, and so on, leading to system heterogeneity. The performance of FL will be significantly affected by stragglers which finish training much slower than others. To this end, some heterogeneity-aware solutions have been proposed to adapt to workers with different capabilities. Li et al. [29] propose to support running different numbers of local iterations according to workers' heterogeneous capabilities. The asynchronous FL methods [30], [31], [32] let workers upload the local updates to the PS for model aggregation immediately after local training without waiting for other slow workers. However, the aforementioned heterogeneity-aware methods also train and transmit the complete models, thus heavy communication and computation overhead make the overall training process inefficient.

Very few efforts have been made to address system heterogeneity and resource overhead challenges simultaneously. To improve communication efficiency, Li et al. [33] develop a convergence-guaranteed FL algorithm enabling flexible compression, which assigns different compression ratios to heterogeneous workers. Similar to other compression based FL methods, this solution cannot reduce computation overhead and ignore computation heterogeneity. HeteroFL [34] improves communication and computation efficiency by assigning different sub-models to heterogeneous workers. However, they do not provide adaptive decision algorithms for heterogeneous workers and require complete information on device heterogeneous capabilities. Although AdaptCL [35] designs a solution of dynamic and adaptive pruning for efficient collaborative learning, the proposed method lacks the convergence analysis and may not achieve the trade-off between resource overhead and model accuracy. Compared to the existing works, our proposed framework can (1) reduce both computation and communication overhead, (2) address system heterogeneity, and (3) adaptively provide device-specific solutions to achieve the trade-off between overhead and accuracy.

IX. Conclusion

In this article, we design and implement FedMP, which performs federated learning through adaptive model pruning. Specifically, we adopt a structured model pruning approach for federated learning so as to simultaneously reduce computation and communication overhead. We then propose an MAB based online learning algorithm to adaptively determine the pruning ratios for different workers to conquer their heterogeneity. Extensive experiments on the classical models and datasets show the high effectiveness of FedMP. In future work, we could consider joint optimization of client selection and model pruning to further improve resource efficiency. In each round, we select only a fraction of workers to participate in FL rather than all workers, which can reduce the resource overhead associated with the proposed framework.

REFERENCES

[1] K. L. Lueth, "State of the IoT 2018: Number of IoT devices now at 7b–market accelerating," *IoT Analytics*, vol. 8, 2018.

[2] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 2, pp. 869–904, Second Quarter 2020.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[4] M. Li et al., "Scaling distributed machine learning with the parameter server," in *Proc. 11th USENIX Symp. Operating Syst. Des. Implementation*, 2014, pp. 583–598.

[5] S. Wang et al., "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun.*, 2018, pp. 63–71.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[8] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 2031–2063, Third Quarter 2020.

[9] J. Wang et al., "A field guide to federated optimization," 2021, *arXiv:2107.06917*.

[10] A. Hard et al., "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[11] T. Yang et al., "Applied federated learning: Improving Google keyboard query suggestions," 2018, *arXiv:1812.02903*.

[12] A. Doherty et al., "Large scale population assessment of physical activity using wrist worn accelerometers: The UK biobank study," *PLoS One*, vol. 12, no. 2, 2017, Art. no. e0169649.

[13] A. Hard et al., "Training keyword spotting models on non-IID data with federated learning," in *Proc. 21st Annu. Conf. Int. Speech Commun. Assoc. Virtual Event*, Shanghai, China, 2020, pp. 4343–4347.

[14] K. Hsieh et al., "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. 14th USENIX Symp. Netw. Syst. Des. Implementation*, 2017, pp. 629–647.

[15] D. Li and J. Wang, "FedMD: Heterogenous federated learning via model distillation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019.

[16] Y. Liu et al., "A communication efficient vertical federated learning framework," *Scanning Electron Microsc Meet at*, 2019.

[17] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.

[18] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1709–1720.

[19] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 560–569.

[20] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes signSGD and other gradient compression schemes," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3252–3261.

[21] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2021–2031.

[22] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 300–310.

[23] H. Sun, S. Li, F. R. Yu, Q. Qi, J. Wang, and J. Liao, "Towards communication-efficient federated learning in the Internet of Things with edge computing," *IEEE Internet Things J.*, vol. 7, no. 11, pp. 11053–11067, Nov. 2020.

[24] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1306–1316.

[25] F. Sattler, S. Wiedemann, K. R. Muller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[26] S. Caldas, J. Konečný, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," 2018, *arXiv:1812.07210*.

[27] Y. Jiang, S. Wang, B. J. Ko, W.-H. Lee, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," 2019, *arXiv:1909.12326*.

[28] H. Wang, S. Guo, B. Tang, R. Li, and C. Li, "Heterogeneity-aware gradient coding for straggler tolerance," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 555–564.

[29] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, Austin, TX, USA, 2020.

[30] 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*, 2020, pp. 15–24.

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

[32] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.

[33] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[34] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," in *Proc. Int. Conf. Learn. Representations*, 2021.

[35] G. Zhou, K. Xu, Q. Li, Y. Liu, and Y. Zhao, "AdaptCL: Efficient collaborative learning with dynamic and adaptive pruning," 2021, *arXiv:2106.14126*.

[36] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[37] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[38] B. Yuan, A. Kyrillidis, and C. M. Jermaine, "Distributed learning of deep neural networks using independent subnet training," 2019, *arXiv:1910.02120*.

[39] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.

[41] S. Han et al., "EIE: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 243–254, 2016.

[42] J. Cong and B. Xiao, "Minimizing computation in convolutional neural networks," in *Proc. Int. Conf. Artif. Neural Netw.*, 2014, pp. 281–290.

[43] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2849–2858.

[44] 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. 2021.

[45] P. Prakash et al., "IoT device friendly and communication-efficient federated learning via joint model pruning and quantization," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13 638–13 650, Aug. 2022.

[46] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms," 2018, *arXiv:1808.07576*.

[47] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 5693–5700.

[48] S. Liu, G. Yu, R. Yin, and J. Yuan, "Adaptive network pruning for wireless federated learning," *IEEE Wireless Commun. Lett.*, vol. 10, no. 7, pp. 1572–1576, Jul. 2021.

[49] 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.*, 2017, pp. 5330–5340.

[50] G. Gao, J. Wu, M. Xiao, and G. Chen, “Combinatorial multi-armed bandit based unknown worker recruitment in heterogeneouscrowdsensing,” in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 179–188.

[51] T. Wang, W. Ye, D. Geng, and C. Rudin, “Towards practical lipschitz bandits,” in *Proc. ACM-IMS Foundations Data Sci. Conf.*, 2020, pp. 129–138.

[52] A. C. Zhou, Y. Xiao, Y. Gong, B. He, J. Zhai, and R. Mao, “Privacy regulation aware process mapping in geo-distributed cloud data centers,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 8, pp. 1872–1888, Aug. 2019.

[53] F. R. Chung, *Spectral Graph Theory*. Providence, RI, USA: American Mathematical Soc., 1997.

[54] B. Gao, Z. Zhou, F. Liu, and F. Xu, “Winning at the starting line: Joint network selection and service placement for mobile edge computing,” in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1459–1467.

[55] O. Besbes, Y. Gur, and A. Zeevi, “Stochastic multi-armed-bandit problem with non-stationary rewards,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 199–207.

[56] A. Garivier and E. Moulines, “On upper-confidence bound policies for non-stationary bandit problems,” 2008, *arXiv:0805.3415*.

[57] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, “Boltzmann exploration done right,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6284–6293.

[58] Z. Meng, H. Xu, M. Chen, Y. Xu, Y. Zhao, and C. Qiao, “Learning-driven decentralized machine learning in resource-constrained wireless edge computing,” in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[59] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, “A unified theory of decentralized SGD with changing topology and local updates,” in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5381–5393.

[60] C. Wang, Y. Yang, and P. Zhou, “Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 394–410, Feb. 2021.

[61] P. Zhou, Q. Lin, D. Loghin, B. C. Ooi, Y. Wu, and H. Yu, “Communication-efficient decentralized machine learning over heterogeneous networks,” in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 384–395.

[62] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[63] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, “EMNIST: Extending MNIST to handwritten letters,” in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 2921–2926.

[64] 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.*, 2020, pp. 1698–1707.

[65] Q. Li, Y. Diao, Q. Chen, and B. He, “Federated learning on non-IID data silos: An experimental study,” in *Proc. IEEE 38th Int. Conf. Data Eng.*, 2022, pp. 965–978.

[66] L. Wang, Y. Xu, H. Xu, M. Chen, and L. Huang, “Accelerating decentralized federated learning in heterogeneous edge computing,” *IEEE Trans. Mobile Comput.*, early access, May 27, 2022, doi: 10.1109/TMC.2022.3178378.

[67] 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.*, 2022, pp. 1739–1748.

[68] J. Wang and G. Joshi, “Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD,” *Proc. Mach. Learn. Syst.*, vol. 1, pp. 212–229, 2019.

[69] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, “Decentralized deep learning with arbitrary communication compression,” in *Proc. Int. Conf. Learn. Representations*, 2020.

[70] W. Wen et al., “Learning intrinsic sparse structures within long short-term memory,” in *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[71] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, “Building a large annotated corpus of english: The penn treebank,” *Comput. Linguistics*, vol. 19, pp. 313–330, 1993.

[72] N. H. Tran, W. Bao, A. Zomaya, N. M. NH, and C. S. Hong, “Federated learning over wireless networks: Optimization model design and analysis,” in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 1387–1395.

[73] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, “Energy efficient federated learning over wireless communication networks,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Mar. 2021.

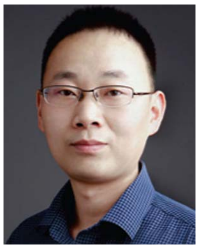
[74] J. Yao and N. Ansari, “Enhancing federated learning in fog-aided IoT by CPU frequency and wireless power control,” *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3438–3445, Mar. 2021.

[75] M. S. Al-Abiad, M. Z. Hassan, and M. J. Hossain, “Energy efficient resource allocation for federated learning in noma enabled and relay-assisted Internet of Things networks,” *IEEE Internet Things J.*, vol. 9, no. 24, pp. 24736–24753, Dec. 2022.

[76] Y. Wu, Y. Song, T. Wang, L. Qian, and T. Q. Quek, “Non-orthogonal multiple access assisted federated learning via wireless power transfer: A cost-efficient approach,” *IEEE Trans. Commun.*, vol. 70, no. 4, pp. 2853–2869, Apr. 2022.

Zhida Jiang received the BS degree from the Hefei University of Technology, in 2019. He is currently working toward the PhD degree with the School of Computer Science and Technology, University of Science and Technology of China (USTC). His research interests include mobile edge computing and federated learning.

Yang Xu (Member, IEEE) received the BS degree from the Wuhan University of Technology, in 2014, and the PhD 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.



Hongli Xu (Member, IEEE) received the BS degree in computer science from the University of Science and Technology of China, China, in 2002, and the PhD degree in computer software and theory from the University of Science and Technology of China, in 2007. He is a professor with the School of Computer Science and Technology, University of Science and Technology of China. 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. He has published more than 100 papers in famous journals and conferences, including *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, Infocom and ICNP, etc. He has also held more than 30 patents. His main research interests include software defined networks, edge computing, and Internet of Thing.

Zhiyuan Wang received the BS degree from the Jilin University, in 2019. He is currently working toward the PhD degree with the School of Computer Science and Technology, University of Science and Technology of China (USTC). His main research interests include edge computing, federated learning, and distributed machine learning.

Jianchun Liu received the PhD 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 include software defined networks, network function virtualization, edge computing, and federated learning.

Qian Chen received the BS degree from the Hefei University of Technology. He is currently working toward the master's degree with the School of Computer Science and Technology, University of Science and Technology of China (USTC). His main research interests include edge computing and federated learning.

Chunming Qiao (Fellow, IEEE) is a SUNY distinguished professor and also the current chair of the Computer Science and Engineering Department, University at Buffalo, Buffalo, NY. His contributions to optical and wireless network architectures and protocols. His current focus is on connected and autonomous vehicles. He has published extensively with an h-index of more than 69. Two of his papers have received the best paper awards from IEEE and Joint ACM/IEEE venues. He also has seven US patents and served as a consultant for several IT and Telecommunications companies since 2000. His research has been funded by a dozen major IT and telecommunications companies including Cisco and Google, and more than a dozen NSF grants.