FedPAGE: Pruning Adaptively Toward Global Efficiency of Heterogeneous Federated Learning

Guangmeng Zhou, Qi Li, *Senior Member, IEEE*, Yang Liu, Yi Zhao, *Member, IEEE, ACM*, Qi Tan, Su Yao, and Ke Xu, *Senior Member, IEEE*

***Abstract*— When workers are heterogeneous in computing and transmission capabilities, the global efficiency of federated learning suffers from the straggler issue, i.e., the slowest worker drags down the overall training process. We propose a novel and efficient federated learning framework named *FedPAGE*, where workers perform *distributed pruning* adaptively towards global efficiency, i.e., fast training and high accuracy. For fast training, we develop a pruning rate learning approach generating an adaptive pruning rate for each worker, making the overall update time approximate to the fastest worker's update time, i.e., no stragglers. For high accuracy, we find that structural similarity between sub-models is essential to global model accuracy in the *distributed pruning*, and thus propose the *CIG_X* pruning scheme to ensure maximum similarity. Meanwhile, we adopt the sparse training and design model aggregating of different size sub-models to cope with *distributed pruning*. We prove the convergence of *FedPAGE* and demonstrate the effectiveness of *FedPAGE* on image classification and natural language inference tasks. Compared with the state-of-the-art, *FedPAGE* achieves higher accuracy with the same speedup ratio.**

***Index Terms*— Federated learning, straggler issue, global efficiency, distributed pruning.**

I. Introduction

DEEP learning has made great progress in many fields (e.g., computer vision and natural language processing). It is generally agreed that the effect of the model is closely related to the amount of data used for training. However, due to data privacy or the high cost of data migration (e.g., face recognition data, mobile phone behavior data, and cross-domain network traffic data), data cannot be collected centrally for training in some scenes. As a new machine learning paradigm to solve this problem, federated learning [1] has received considerable attention. Instead of putting data to the model location, federated learning pushes the model to the data location.

In practical applications, workers of federated learning are typically resource-constrained and thus heterogeneous in computing and transmission capabilities (e.g., edge devices are equipped with different computing chips and located in different domains). Even with the same physical equipment, the resource (e.g., memory and bandwidth) allocated to a specific task is usually limited. In federated learning, the bulk synchronous parallel (BSP) policy [2] [1] has been widely applied, thus causing the straggler issue [3], [4]. More specifically, given the heterogeneity of workers, the slowest worker in collaboration drags down the entire training process. The root cause of the straggler issue is different capabilities workers are required to do the same thing (e.g., training the same size model). Thus different capabilities workers should train models of different sizes.

Existing studies [5], [6], [7] introduce different sub-models which are generated from a base model into federated learning. In Helios [5] and HeteroFL [6], the worker takes the number of parameters within its capabilities from the base model for training. The difference is that for a given worker, Helios changes the taken parameters every round, but HeteroFL keeps the parameters the same. SplitMix [7] first divides the base model into several sub-models and the worker randomly selects the corresponding number of sub-models for training each round according to its capability. However, the above studies suffer from two limitations, i.e., unrealistic assumption about known worker capabilities and lacking investigation about tailoring sub-models for global accuracy.

Unrealistic assumption about known worker capabilities. A key feature of federated learning is that the participants come from different entities. To ensure the privacy and security of each participant, the server cannot directly access the capability information of each participant. Moreover, the capabilities of participants may vary depending on the situation, such as when a PC is also performing other tasks that consume communication or CPU resources, or when a user's phone battery is low, they will have fewer resources to contribute to the federated learning task. Therefore, it is crucial to model the capabilities of participants when starting federated learning. The sooner the appropriate sub-models are generated, the

Manuscript received 25 August 2022; revised 2 April 2023 and 23 July 2023; accepted 20 October 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Liu. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant U22B2031, Grant 61932016, Grant 62132011, and Grant 62202258; in part by the National Science Foundation for Distinguished Young Scholars of China under Grant 61825204; in part by the Beijing Outstanding Young Scientist Program under Grant BJJWZYJH01201910003011; in part by the China Postdoctoral Science Foundation under Grant 2021M701894; in part by the China National Postdoctoral Program for Innovative Talents, Shuimu Tsinghua Scholar Program; and in part by the Ant Group through the CCF-Ant Innovative Research Program under Grant CCF-AFSGRF20210023. *(Corresponding authors: Yi Zhao; Ke Xu.)*

Guangmeng Zhou, Yi Zhao, Qi Tan, and Ke Xu are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: zgm19@mails.tsinghua.edu.cn; zhao_yi@tsinghua.edu.cn; tanq20@mails.tsinghua.edu.cn; xuke@tsinghua.edu.cn).

Qi Li is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China (e-mail: qli01@tsinghua.edu.cn).

Yang Liu is with Huawei, Shenzhen 518129, China (e-mail: liuyang19@mails.tsinghua.edu.cn).

Su Yao is with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China (e-mail: yaosu@tsinghua.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2023.3328632, provided by the authors.

Digital Object Identifier 10.1109/TNET.2023.3328632

[1] The server updates the global model until updates of all workers or at least a certain number of workers have committed.

higher speedups can be achieved. Therefore, worker capability modeling needs to be fast and accurate with the limited information available to the server.

Lacking investigation about tailoring sub-models for global accuracy. Network pruning studies [8], [9], [10] remove some parameters from the stand-alone base model to obtain a sub-model and maintain its accuracy, which is the most efficient way to achieve models of different sizes. The above studies directly apply stand-alone network pruning techniques for each worker. However, it only guarantees the accuracy of the pruned local model, not the global model after aggregation. *Distributed workers prune their own models while pursuing the accuracy of the aggregated global model on the server, such a scenario, is referred to as distributed pruning.* The above studies are unaware of the challenge of *distributed pruning* and neglect to investigate it.

Therefore, our key design goal is to **speed up the global training process with worker capability agnostic, and ensure global accuracy under *distributed pruning* to achieve superior global efficiency of federated learning**. In this paper, we propose a novel and efficient federated learning framework named *FedPAGE*, pruning adaptively towards global efficiency, i.e., fast training and high accuracy. It generates adaptive sub-models dynamically from the global base model for workers via our customized pruning scheme.

To speed up the global training process (fast training), we develop a pruning rate learning approach. First, we collect the only information that the server can observe, i.e., model retention size and corresponding update time (time interval between two adjacent updates received). Then, we model worker capabilities by the Newton interpolation method. Finally, we set the adaptive pruning rate depending on the capacity gap between the worker and the fastest worker. Thus all workers achieve approximately identical update time as the fastest worker, i.e., no stragglers. To ensure global accuracy (high accuracy), we investigate the important principles for global model accuracy in *distributed pruning*. Experiments reveal that *identical* and *constant* are two essential principles for global model accuracy because they ensure maximum structural similarity between local models after pruning. After introducing a well-established pruning principle *global*, we propose the *CIG-X* pruning scheme tailored for *distributed pruning*. In addition, we adopt the sparse training and design model aggregating of different size local models to cope with *distributed pruning*.

We conduct extensive experiments on multiple datasets (i.e., CIFAR, Tiny-ImageNet, MNLI and Digits) with different models (i.e., VGG16, ResNet50, and BERT). Moreover, we take multiple data heterogeneity cases into account, including Non-IID (independent and identically distributed) and cross domains heterogeneity. Empirical results show that *FedPAGE* outperforms state-of-the-art approaches to improve federated learning efficiency under various settings. Besides, *FedPAGE* can be combined with other approaches to achieve higher speedups.

To summarize, we make the following contributions:

- We propose an efficient federated learning framework *FedPAGE*, which performs customized *distributed pruning* for workers to speed up the entire training process while maintaining satisfactory accuracy.
- We propose a dynamic and adaptive pruning rate learning approach that enables each worker's update time to achieve identical under worker capability agnostic.
- We specify the concept of *distributed pruning* and its challenge for the first time. To address the challenge, we investigate and find the essential pruning principles in *distributed pruning* and tailor a novel and efficient pruning scheme *CIG-X*.
- We give the convergence analysis of *FedPAGE*. Further, we analyze the impact of the pruning rate and make comparisons with other federated learning algorithms.
- Extensive experiments show that *FedPAGE* exhibits efficiency in various data-model settings and different degrees of heterogeneous environments.

II. Definition and Related Work

A. Definition of Federated Learning

Federated learning, a new machine learning paradigm, is closely related to distributed machine learning. However, the relatively lower level of trust and a higher degree of heterogeneity between workers enable federated learning to be extensively studied in terms of security [11], [12], [13], privacy [14], [15], efficiency [16], [17], etc.

In the classic federated learning such as FedAVG, the training process of each round includes the following steps:

1) The server sends the current global model to workers.
2) Workers perform local model updates on the models and send the updated local models to the server.
3) The server aggregates local models from workers forming the new global model.

The problem solved by federated learning is shown in Eq. (1). $F(\theta)$ is the global optimization function, and θ is the model parameter. $f_w(\theta)$ is the optimization function on worker w. p_w is the model weight of worker w, which is usually determined by the amount of data in the worker. W is the number of workers.

$$\min_{\theta} \ \{F(\theta) \triangleq \sum_{w=1}^{W} p_w f_w(\theta)\}, \quad where \quad \sum_{w=1}^{W} p_w = 1 \tag{1}$$

We define $\theta_w^{t,e}$ to denote the parameter of the wth worker in round t after e local updates. θ_g^t is the global parameter at round t (after the t-1th round of aggregation). The local gradient update can be described as in Eq. (2). E is the number of local updates and η is the learning rate.

$$\begin{aligned} \theta_w^{t,e+1} &= \theta_w^{t,e} - \eta \nabla f_w(\theta_w^{t,e}), \quad e = 0, 1, \ldots, E-1, \\ where \quad \theta_w^{t,0} &= \theta_g^t \end{aligned} \tag{2}$$

The tth round local model aggregation can be described as

$$\theta_g^{t+1} = \sum_{w=1}^{W} p_w \theta_w^{t,E}, \quad where \quad \sum_{w=1}^{W} p_w = 1 \tag{3}$$

We illustrate the parameters changes of federated learning under two workers setting in Fig. 1. Eq. (2) corresponds to the arrows of local update and download, and Eq. (3) corresponds to the arrows of upload and aggregation.

B. Related Work

Network Pruning.Compared to existing techniques such as neural architecture search (NAS) [18], [19], network pruning is the most efficient way to achieve models of different

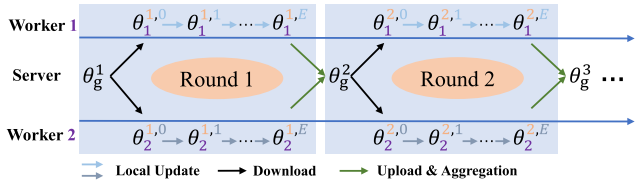


Fig. 1. The parameters changes illustration of federated learning under two workers setting.

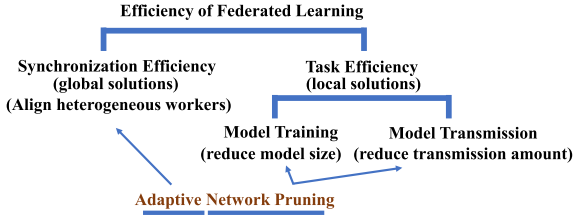


Fig. 2. The illustration of related work about efficient federated learning.

sizes because it brings significantly less additional overhead to the generation and aggregation of sub-models. Network pruning is based on the premise that deep neural networks are over-parameterized, and thus proper pruning can cut off a large number of parameters to accelerate model inference while maintaining accuracy. Network pruning consists of three stages: training, pruning, and fine-tune. According to the pruned object, network pruning can be divided into non-structural pruning, which only cuts off weights, and structural pruning, which cuts off units such as neurons and filters. However, non-structural pruning can only achieve acceleration on specialized software [20] or hardware [21]. In contrast, structural pruning can achieve acceleration on arbitrary software and hardware, and thus we adopt it.

Structural pruning usually cuts off units of low importance predefined, such as the percentage of zero activation [8], the $\ell 1$-norm [22], the scaling factor of Batch Normalization (BN) layer [23], the distance from the geometric median of filters of same layer [24], the rank of feature map [25], or the importance indicators introduced in training [9], [26]. Considering that different network layers have different sensitivities to parameter pruning [27], so the pruning rate per layer should be different for a given pruning budget. [9], [28], [29] try to get global rank for all filters, and [30] proposes to learn pruning thresholds for different layers.

However, directly applying these stand-alone techniques is not suitable. On the one hand, they are based on pre-trained models to speed up inference as opposed to our intention to speed up training and based on the model being trained. While some ideas could be adopted, we expect the pruning and fine-tune to be extremely time-efficient and can be done early in the training process, which is not considered in previous works. On the other hand, there is a gap between what we do and what we pursue in *distributed pruning*, i.e., we prune the local model but pursue the accuracy of the aggregated global model. These all motivates us to design customized pruning methods for the distributed scenario.

Efficient Federated Learning. Federated learning can be divided into general federated learning and personalized federated learning. Personalized federated learning aims to obtain personalized model suitable for each worker, and the common approaches include personalizing the trained global model [31] or focusing on learning with workers that are similar to itself [32]. In this paper, we are interested in general federal learning, i.e., seeking a superior global model. We divide the causes of inefficient training into local and global causes as illustrated in Fig. 2. The local cause is that the model is too large, leading to extremely time-consuming model training and transmission. The global cause is the heterogeneity between workers, leading to the stragglers, which in turn affects synchronization efficiency.

Local solutions. To speed up model transmission, gradient quantization and sparsification are extensively studied. Gradient quantization is achieved by quantizing the gradients to low-precision values [33]. Gradient sparsification is usually done by selecting significant parameters to transmit [16], [34], [35], [36] or adding constraints to get a sparse model [37]. In addition to reducing transmitted parameters to speed up single transmission, efficiency can also be improved by adjusting the frequency of parameter aggregation [38]. In contrast, little research has been done on speed up training. Adding new loss items to improve the training accuracy at the same time can be seen as a kind of training speedup [17], [39]. Network pruning can speed up model transmission, but almost all network pruning studies do not pay attention to speed up training because they retain the original dense model during pruning. Reference [40] solve the problem by reconfiguring the model into smaller models during pruning. We draw on the idea of network reconfiguration.

Global solutions. Compared with local solutions, global solutions take a global view of efficiency issue and focus on the heterogeneity. Therefore, other server synchronization policies have been extensively studied except for BSP to solve the straggler issue. The asynchronous parallel (ASP) policy is the opposite of BSP. In ASP, the server updates the global model as soon as it receives an update. The stale synchronous parallel (SSP) policy [41], [42] is a trade-off between BSP and ASP. When the fastest worker is ahead of the slowest worker by predefined threshold s rounds, the fastest worker needs to stop and wait. Reference [43] develops a unified synchronization policy framework that can cover BSP, ASP, and SSP by designing the collection of active workers of the next round, and then uses reinforcement learning to learn the collection to minimize the total time cost. However, these works do not work well in environments with high heterogeneity.

Sub-model solutions. Some works have also introduced sub-models into federated learning. PruneFL [44] and AFD [45] try to find the same size optimal sub-model for all workers without considering the straggler issue. Helios [5], HeteroFL [6] and Split-Mix [7] consider the straggler issue, but both assume that the worker capabilities are known and do not consider *distributed pruning* from the perspective of global model accuracy. As illustrated in Fig. 2, our approach obtains adaptive small models for workers according to their capacities. The small models enable faster training and transmission addressing local causes. The adaptive size models align worker update times, eliminate stragglers and thus address global issues.

III. FedPAGE

In this section, we overview our framework *FedPAGE* firstly, then elaborate on the critical components of the framework, including model training and aggregating, pruning rate learning approach, and network pruning approach. The pruning

TABLE I

SUMMARY OF MAIN NOTATIONS. THE NOTATIONS FROM TOP TO BOTTOM ARE THE FORMAL REPRESENTATION, HYPERPARAMETERS, AND INTERMEDIATE PARAMETERS TO BE CALCULATED

D_w	Data of worker w
θ_g^t	Global model at round t
$\theta_w^{t,e}$	Worker w's model at round t after of e local updates
$f_w(\theta)$	The optimization function on worker w
μ	$f_w(\theta)$ is μ-Lipschitz
L	$f_w(\theta)$ is L-smooth
σ	$\mathbb{E}[\|\nabla f_w(\theta_w^{t,e})\|^2] \leq \sigma^2$
κ_{t+1}	The pruned parameters threshold in round t
$U_{retained}$	The retained units after pruning
I_w^t	Unit indexes in worker w's sub-model corresponding to the global model at round t
T	Number of transmission rounds
W	Number of workers
E	Number of local updates
H	Heterogeneity
PI	Pruned interval
p_w	The model weight of worker w when aggregating models
β	Ratio of the first training part
γ_{min}	Minimum model retention ratio set for pruning
ρ_{min}	Minimum pruning rate set for pruning
ρ_{max}	Maximum pruning rate set for pruning
P_w^t	Pruning rate of worker w at round t, which is calculated by pruning rate learning algorithm Alg. 2
ϕ_w^t	Update time of worker w at round t, which is the time difference between the server sending and receiving models

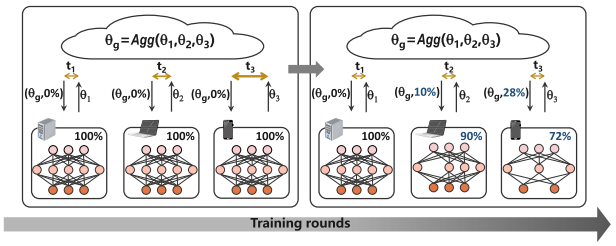


Fig. 3. *FedPAGE* process: the cloud server keeps issuing pruning rates to resize workers' models during training until all workers have the same update time. t_1, t_2 and t_3 represent the update time of the above three workers, respectively.

rate learning approach determines the worker pruning rate in order to achieve consistency with the fastest update time. The network pruning approach determines which units to prune at the determined pruning rate to minimize the impact on the global model. Model training and aggregating are tailored under distributed pruning, and also aim to minimize the impact on the global model. Before elaborating on our framework *FedPAGE*, we summarize the main notations in Tab. I.

A. Overview

The process of *FedPAGE* is illustrated in Fig. 3. In the learning process, there is one server for model aggregating, as well as several workers with different capabilities holding various data, such as PC, laptop, and phone. The workers communicate directly with the server, but the bandwidth is diverse. The task aims to leverage the data on each worker to get a superior global model. When training begins, the parameter server distributes the same model θ_g to each worker conservatively since workers' capabilities are unknown. After obtaining the worker capability signal, i.e., the update time, the server generates adaptive pruning rates (e.g., 0%, 10%, 28%)

Algorithm 1 FedPAGE: Pruning Adaptively Towards Global Efficiency

Server:

1: **for** each round $t = 1, \ldots, T$ **do**
2: **while** not receiving all updates **do**
3: Server receives θ_w^t, I_w^t from worker w,
4: and calculate worker w' update time ϕ_w^t
5: $\theta_g^t \leftarrow$ ModelAggregating(θ_1^t, θ_2^t,..., θ_W^t)
6: **if** it is pruning round **then**
7: Obtain $\{P_1^{t+1}, \ldots, P_W^{t+1}\}$ with Pruning Rate Learning approach Alg. 2
8: **for** each worker $w = 1, \ldots, W$ **do**
9: Extract parameters of θ_g^t in I_w^t forming θ_w^{t+1}
10: Server send θ_w^{t+1} and P_w^{t+1} to worker w

Worker:

1: Worker w receives θ_w^{t+1} and P_w^{t+1} from server
2: $\theta_w^{t+1} \leftarrow$ ModelTraining($\beta E, \theta_w^{t+1}, D_w$)
3: **if** $P_w^{t+1} > 0$ **then**
4: $U_{retained} \leftarrow$ NetworkPruning(P_w^{t+1})
5: θ_w^{t+1}, $I_w^{t+1} \leftarrow$ NetworkReconfigure(θ_w^{t+1}, $U_{retained}$)
6: **else**
7: $I_w^{t+1} \leftarrow I_w^t$
8: $\theta_w^{t+1} \leftarrow$ ModelTraining($(1-\beta)E, \theta_w^{t+1}, D_w$)
9: Worker w send θ_w^{t+1} and I_w^{t+1} to server

for workers. The worker receives the pruning rate and then prunes its model locally. Then the workers get sub-models with different sizes, e.g., 100%, 90%, and 72%. Next, the worker performs training with its sub-model and sends the sub-model to the server for aggregating. The process proceeds dynamically with training until the update time tends to be identical. The pseudo-code of the framework is shown in Algorithm 1. As we can see, *FedPAGE* consists of two types of participants, server and worker.

On the server-side, we adopt the synchronous approach that the server starts aggregating (in Sec. III-B) until all workers' updates are received as in federated learning [1]. As shown in Server line 2, the updates include parameters θ_w^t and global index I_w^t of the local model, where I_w^t is to align the units in the local model and the global model if the local model is pruned. Meanwhile, the server calculates the update time ϕ_w^t of worker w, which is the time difference between the server sending and receiving models. ϕ_w^t is used to model worker capability and obtain pruning rate as shown in Alg. 2. If the pruned round arrives, the pruning rate learning approach 2 (in Sec. III-C) is used to get the pruning rate P_w^{t+1} for each worker. *FedPAGE* adopts iterative pruning, pruning after each pruning interval (PI), and pruning from the very beginning of training, making the overall training process as time-saving as possible and leaving more rounds to recover the model. Finally, the server obtains the parameters θ_w^{t+1} of the worker's sub-model by extracting parameters of θ_g^t in position I_w^t, and sends the parameters and pruning rate to the worker.

On the worker-side, the worker trains part of the epochs (βE, $0 \leq \beta \leq 1$) with its dataset D_w after receiving parameters (in Sec. III-B). If pruning is not required, the worker continues training the other part of the epochs $(1-\beta)E$ on the previous model as shown in Worker line 8. When

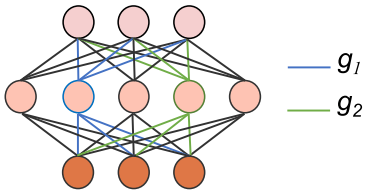


Fig. 4. The illustration of parameters group g.

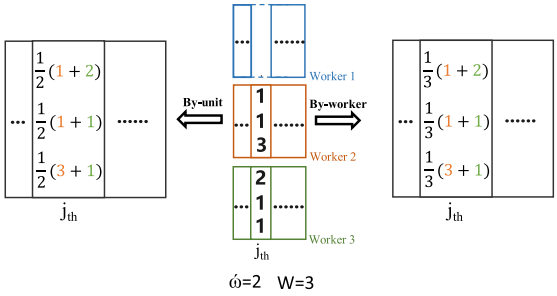


Fig. 5. Two approaches to aggregate parameters. The three squares in the middle represent one weight matrix of three workers, respectively. The squares on either side represent the results of the two ways of aggregation. The top worker cuts out a unit, resulting in a missing column of the matrix.

pruning is required, we follow a specific pruning order to prune network (in Sec. III-D). As shown in Worker line 4 in Alg. 1, we obtain retained units $U_{retained}$ according to the pruning order and pruning rate P_w^{t+1}. Then we rebuild and initialize a smaller sub-model as in [40] (network reconfigure). And the units' global index I_w^{t+1} of current sub-model is updated. The step is shown in Worker line 5, where θ_w^{t+1} is the parameters of current sub-model. In the end, the current sub-model's parameters and global unit index are sent to the server. It is worth noting that, compared to federated learning, the content of the transmission is only more model's global index as well as pruning rate, which introduces little transmission overhead.

B. Model Training and Aggregating

Model training. As introduced in Sec. II-B, we adopt structural pruning, i.e., cut off units for achieving acceleration on arbitrary software and hardware. For mitigating the impact caused by cutting off units, we adopt the sparse training approach in [40]. The loss function is shown in Eq. (4), consisting of cross-entropy loss and group lasso loss. The loss function is used for sparse training in each round. With the introduction of group lasso regularization, the parameters associated with a unit are viewed as a group g as illustrated in Fig. 4. The group lasso loss makes the parameters in a group smaller at the same time, thus mitigating the impact caused by cutting off the unit. In addition, we divide the training process into two parts to explore the effects of different pruning positions in training in Sec. V-F.

$$\min_{\theta}\{f_w(\theta) \triangleq \frac{1}{|\mathcal{B}|}(\underbrace{\sum_{(x,y)\in\mathcal{B}} l(y,h(x,\theta))}_{\text{cross-entropy loss}} + \lambda \cdot \underbrace{\sum_{g\in\mathcal{G}} \sqrt{|g|}\,\|\theta_g\|_2}_{\text{group lasso loss}})\} \tag{4}$$

where $\mathcal{B}$ represents a batch of data, $\mathcal{G}$ represents groups of parameters, $h(x,\theta)$ is the predicted label, λ is the coefficient balancing the above two losses.

Model aggregating. If we ignore the differences in the amount of data between workers, the parameter aggregation coefficient p_w is $\frac{1}{W}$. However, as a result of network pruning, some parameters may only exist in w' ($w' < W$) local models. Take the example in Fig. 5, the local model of worker 1 (referred to as local model 1 below) does not have the parameters in column j, i.e., the parameters in column j only exist in two local models ($w' = 2$). There are two approaches to set the aggregation coefficient for parameters in column j, i.e., parameters pruned at some workers, which are illustrated in Fig. 5.

One way is **By-unit** with a coefficient of $\frac{1}{w'}$, which is the way in HeteroFL [6]. By-unit is equivalent to treating the pruned parameters (e.g., jth column of parameters in local model 1) as the mean of the parameters at the corresponding position in the other local models. For example, in Fig. 5, the first weight in By-unit is $\frac{1}{2}(1+2) = \frac{1}{3}(1+2+1.5)$, where $1.5 = \frac{1}{2}(1+2)$ is the mean of the jth column of parameters in local model 2 and local model 3. We propose the other way **By-worker**, whose coefficient is still $\frac{1}{W}$. By-worker is equivalent to treating the pruned parameters as 0. For example, in Fig. 5, the first weight in By-worker is $\frac{1}{3}(1+2) = \frac{1}{3}(1+2+0)$. Considering that the pruned parameters are relatively small, treating them as 0 is closer to the real value. In addition, [46] points out that the reason why lottery tickets behave well is that mask operation freezes some values to zero and can make those values reach the end of their optimization process faster. In conclusion, treating pruned parameters as 0 when aggregating is not only closer to the original values, but may also speed up the optimization. Therefore, we adopt by-worker to do model aggregating in *FedPAGE*. We demonstrate the superiority of By-worker in Sec. V-F.

Algorithm 2 Pruning Rate Learning

Input: Suppose the round is the nth pruning round, i.e., $n = \frac{t}{PI}$. $\hat{\gamma}_w^n$ and $\hat{\phi}_w^n$ represent the model retention ratio and corresponding average update time in the last pruning interval (PI). That is, $\hat{\phi}_w^n = \mathsf{Average}(\phi_w^{t-PI+1}, \ldots, \phi_w^t)$, $\hat{\gamma}_w^n = \gamma_w^{t-PI+1} = \ldots = \gamma_w^t$.

Output: $P_1^{t+1}, P_2^{t+1}, \ldots, P_W^{t+1}$

1: $\phi_{min} \leftarrow \mathsf{Min}(\hat{\phi}_1^n, \hat{\phi}_2^n, \ldots, \hat{\phi}_W^n)$
2: **for** each worker $w = 1, \ldots, W$ **do**
3: **if** $n > 1$ **then** ▷ the worker has been pruned
4: Get γ_w^{target} by Newton interpolation as Eq. (5)
5: $\gamma_w^{target} \leftarrow \mathsf{Max}(\gamma_w^{target}, \gamma_{min})$
6: $P_w^{t+1} \leftarrow \frac{\hat{\gamma}_w^n - \gamma_w^{target}}{\hat{\gamma}_w^n}$
7: **else**
8: $P_w^{t+1} \leftarrow \frac{\hat{\phi}_w^n - \hat{\phi}_{min}}{\alpha * \hat{\phi}_w^n}$
9: $P_w^{t+1} \leftarrow \mathsf{Max}(P_w^{t+1}, \rho_{min})$
10: $P_w^{t+1} \leftarrow \mathsf{Min}(P_w^{t+1}, \rho_{max})$

C. Pruning Rate Learning

Pruning rate learning gives the adaptive pruning rate for each worker to achieve approximately identical update time. The worker's update time consists of two parts: model transmission time and model training time. The transmission time can be considered to vary linearly with the size of transmitted parameters if the bandwidth is relatively stable. Many researchers use FLOPs (floating point operations) as an indicator of training capability. However, training time is affected by

many practical factors (e.g., data loading speed, parallel optimization in the computing chip, underlying optimization of the computing platform) [47]. That is, no capability indicator can give the precise update time when the local model is pruned. Moreover, the capability information is private and difficult to obtain by the server. Thus, approximate and dynamic capacity estimation is needed. Considering that collaborating parties usually establish a relatively stable collaboration environment, e.g., Google lets phone join the collaboration when the phone is idle and connected to WIFI [48], the capacity estimation of the workers is feasible.

Our pruning rate learning approach is shown in Alg. 2. First, we take the current minimum average update time as the target time ϕ_{min} for the next round of pruning. Then we use the data that have been collected, i.e., the model retention ratio and averaged update time each round $(\hat{\gamma}_w^0, \hat{\phi}_w^0)$, ..., $(\hat{\gamma}_w^n, \hat{\phi}_w^n)$, to construct a polynomial R that satisfies $R_w(\hat{\phi}_w^i) = \hat{\gamma}_w^i,\ i = 0, \ldots, n$. Finally, $\gamma_w^{target} = R_w(\phi_{min})$. Averaged update time avoids the influence of some random factors and makes capacity estimation more accurate. There are many interpolation methods that have been studied. Among them, Newton interpolation is solid and fast, so we adopt Newton interpolation in *FedPAGE* as shown in Eq.(5). According to Main Theorem of Polynomial Interpolation [49], R_w is existent and unique. The Newton interpolation may have the Runge phenomenon at higher orders causing larger errors. However, since approximately identical update time can be achieved with three or four prunings, the n is small in our scenario. Thus the Runge phenomenon does not occur.

$$\begin{aligned}
\gamma_w^{target} &= R_w(\phi_{min}) = R_w[\hat{\phi}_w^0] + R_w[\hat{\phi}_w^0, \hat{\phi}_w^1] \\
&\quad \times (\phi_{min} - \hat{\phi}_w^0) \\
&\quad + \ldots + R_w[\hat{\phi}_w^0, \hat{\phi}_w^1, \ldots, \hat{\phi}_w^n](\phi_{min} - \hat{\phi}_w^0) \\
&\quad \ldots (\phi_{min} - \hat{\phi}_w^{n-1}) \\
where \quad R_w[\hat{\phi}_w^i] &= R_w(\hat{\phi}_w^i) = \hat{\gamma}_w^i, i = 0, \ldots, n \\
R_w[\hat{\phi}_w^0, \ldots, \hat{\phi}_w^n] &= \frac{R_w[\hat{\phi}_w^0, \ldots, \hat{\phi}_w^{n-1}] - R_w[\hat{\phi}_w^1, \ldots, \hat{\phi}_w^n]}{\hat{\phi}_w^n - \hat{\phi}_w^0}
\end{aligned} \tag{5}$$

If no pruning has been done before, we conservatively assume that the update time ϕ varies linearly with the model retention ratio γ, with a slope of α, and the pruning rate is obtained as in line 8. The value of α can be set slightly larger, avoiding pruning too much at the beginning. Besides, we set a minimum model retention ratio γ_{min}, a maximum pruning rate ρ_{max} to prevent excessive pruning, and a minimum pruning rate ρ_{min} to prevent overly frequent minor pruning. Our algorithm introduces little computational overhead to the server and quickly adapts to changing environments in different heterogeneous environments as shown in Sec. V-F.

D. Network Pruning

Network pruning tries to find pruning orders of units for workers. The intuitive approach is that each worker prunes their models individually according to the learned pruning rate. In this way, previous network pruning works can be applied to each worker individually to obtain pruning orders. However, previous network pruning works only guarantee the accuracy of the obtained sub-model, i.e., the accuracy of each worker's local model, but not the accuracy of the global model after aggregation. How can we obtain sub-models by pruning while taking into account the global model accuracy? This is the essence of the *distributed pruning*.

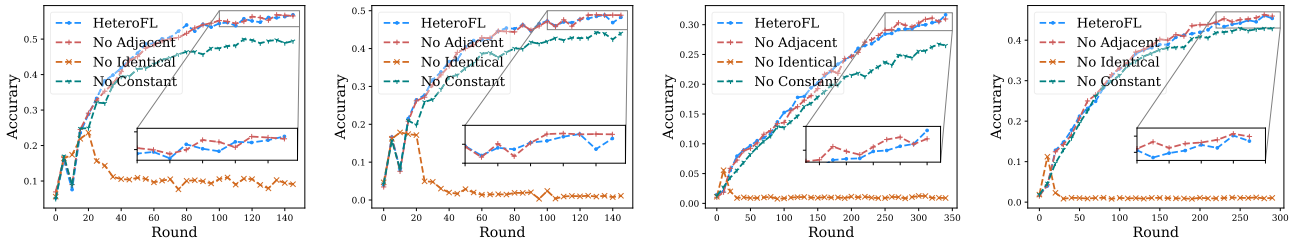
We noticed that HeteroFL [6] adopts a pruning approach that ignores the unit importance in models for a certain task, but also achieves good performance. The pruned units are adjacent in the unit index, and the pruning order is identical between each worker, constant between each round. We attempt to investigate the underlying reasons for its good performance and find the key to guarantee the accuracy of the global model. The good performance may be related to the three characteristics, i.e., adjacent (pruned units are adjacent), identical (all workers share the pruning order), and constant (all rounds share the pruning order). We keep all other treatments the same and perform a number of variants as follows (we show the examples of HeteroFL and three variants in Appendix B, see the Supplementary Material):

- *No adjacent*: generate a random order at the beginning and keep it *identical* and *constant*.
- *No identical*: generate different but *adjacent* orders randomly for different workers and keep it *constant*.
- *No constant*: generate different but *adjacent* order randomly at each pruning and keep it *identical*.

We show results of the above three variants and HeteroFL on an image classification task (CIAFR100 dataset) under multiple Non-IID settings in Fig. 6. As we can see, both the experiments on the IID dataset (Fig. 6a) and the Non-IID dataset (Fig. 6b, Fig. 6c, Fig. 6d) show similar results. *No identical* has the worst result, and the global model does not converge, which reveals that *identical* is the crucial reason. Followed by *No constant*, the global model converges but converges to a lower accuracy, which reveals that *constant* is also important. *No adjacent* behaves almost the same as HeteroFL, which reveals that *adjacent* doesn't matter at all. Why *Identical* and *constant* are crucial? [50] indicates that sharing the same model structure results in better performance when tasks are more similar in multi-task training. In federated learning, the tasks of individual workers are extremely similar, although workers may have different data distributions. *Identical* and *constant* ensure maximum structural similarity between sub-models. We also demonstrate that structural similarity is essential to global model accuracy through experiments in Sec. V-D.

Identical and *constant* are two principles to guarantee the accuracy of the global model. Considering that different layers express different semantics, we introduce the well-established principle that the unit importance needs to be global. Thus, we design a *constant*, *identical*, and *global* (*CIG-X*) pruning scheme, where *X* represents a method to calculate the global importance of a unit which can be borrowed from deeply studied network pruning works. The generation of pruning order is as follows.

1) A unit importance metric *X* is selected, which is used to compare units in all layers together (*global-G*).
2) At the first pruning round, each worker calculates the unit importance in its model based on the importance metric *X*, and sends unit importance to the server.
3) The server averages the unit importance received from workers obtaining the unit importance in the global model. All units are ranked in the ascending order of importance to form the pruning order.



(a) Sort-and-partition, $W = 10$, s=0 (b) Sort-and-partition, $W = 10$, s=80 (c) Sort-and-partition, $W = 100$, s=80 (d) Dirichlet, $W = 100$, s=0.1

Fig. 6. The investigation of the effects of three characteristics in *distributed pruning* on CIFAR100. The pruning rates of workers are pre-set as in Sec. V-D. Sort-and-partition and dirichlet partition are two common Non-IID distribution partition ways and the s represents the degree of Non-IID, see Sec. V-A and Sec. V-E for details.

4) The pruning order is sent to each worker (*identical-I*). Each worker adopts the pruning order in each round (*constant-C*).

CIG-X considers both unit importance and structural maximum similarity, and thus obtain higher accuracy.

IV. Convergence Analysis of *FedPAGE*

We analyze the convergence of *FedPAGE* in this section. To facilitate the analysis, we first introduce some definitions, then give assumptions and some simple lemmas.

A. Definition, Assumptions and Lemmas

We use $\theta_w^{t,e} \odot m_w^{t,e}$ to denote the remaining parameter after $\theta_w^{t,e}$ is pruned. $m_w^{t,e}$ is a matrix with the same dimension as $\theta_w^{t,e}$, containing only 0 and 1. 0 means pruning the parameter at that position and 1 means keeping the parameter at that position. $\overline{m}_w^{t-1,e}$ is the mask obtained by swapping the 0 and 1 in $m_w^{t-1,e}$. That is, $\theta_w^{t,e} \odot \overline{m}_w^{t-1,e}$ represents the pruned parameter.

In the following proof, we focus on pruning before aggregation, i.e., $\beta = 1.0$. The proof is similar when β is other values. At round $t-1$, if the parameters of all workers are pruned, the global aggregation can be described as

$$\hat{\theta}_g^t = \sum_{w=1}^{W} (p_w \theta_w^{t-1,E} \odot m_w^{t-1,E}) \tag{6}$$

Assumption 1: We assume the following for all w, t, e.

1) *$f_w(\theta)$ is μ-Lipschitz, i.e., $\|f_w(\theta) - f_w(\theta')\| \leq \mu\|\theta - \theta'\|$ for any θ, θ'.*
2) *$f_w(\theta)$ is L-smooth, i.e., $\|\nabla f_w(\theta) - \nabla f_w(\theta')\| \leq L\|\theta - \theta'\|$ for any θ, θ'.*
3) *Bounded Gradients,[2] i.e., $\mathbb{E}[\|\nabla f_w(\theta_w^{t,e})\|^2] \leq \sigma^2$*

Assumption 2: We assume the following for all w, t.

1) *The pruned parameters in round t are below the threshold κ_{t+1}.*

Lemma 1: $F(\theta)$ is μ-Lipschitz and L-smooth.

Proof of Lemma 1. Straightforwardly from the $f_w(\theta)$'s assumption, Eq. (1), and triangle inequality.

Lemma 2: In a fully connected neural network using the relu activation function, if all the relevant parameters of a neuron are 0, the gradient of these parameters is also 0.

Proof of Lemma 2. The proof is in Appendix A-A, see the Supplementary Material.

Remark 1: The operation of convolution can also be achieved by matrix multiplication [49], so the above conclusion also holds for convolutional neural networks. In the following, we assume that for $f_w(\theta)$ the above conclusion holds. Thus, we do not need to add mask to the gradient of parameters obtained after structured pruning, i.e., $\nabla f_w(\theta_w^{t,e} \odot m_w^{t,e}) = \nabla f_w(\theta_w^{t,e} \odot m_w^{t,e}) \odot m_w^{t,e}$.

B. Convergence Proof

Theorem 1: Under Assumption 1 and Assumption 2, if $\eta = \frac{1}{\sqrt{T}}$ and $T \geq (EL)^2$, T rounds yields the following bound[3]:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\hat{\theta}_g^t)\|^2] \\
&\leq \frac{F(\theta_g^0) - F(\theta^*)}{\sqrt{T}E - E^2L} + \frac{LE\sigma^2}{(\sqrt{T} - EL)} \\
&\quad + \frac{\mu}{\sqrt{T}E - E^2L}\sum_{t=0}^{T-1}\mathbb{E}[\|\theta_g^t - \hat{\theta}_g^t\|] \\
&\quad + \frac{\mu\sqrt{\zeta}}{\sqrt{T}E - E^2L}\sum_{t=0}^{T-1}\kappa_t + \frac{3L\zeta}{2(\sqrt{T}E - E^2L)}\sum_{t=0}^{T-1}\kappa_t^2
\end{aligned} \tag{7}$$

where $\hat{\theta}_g^t = \sum_{w=1}^{W} p_w(\theta_w^{t-1,E} \odot m_w^{t-1,E})$, $\theta^* := arg\ min\ F(\theta)$, and ζ denotes the dimensionality of the initial parameter θ_g^0.

Proof of Theorem 1. The proof is in Appendix A-B, see the Supplementary Material.

In the right of the Eq. (7), the first two items go to zero as $T \to \infty$. Note that pruning is only done a couple of times in the early part of the training, so $\hat{\theta}_g^t = \theta_g^t$ and $\kappa_t = 0$ always hold in almost all rounds. In the pruning round, $\hat{\theta}_g^t \neq \theta_g^t$ and $\kappa_t \neq 0$. However, since the group lasso penalty is applied to the weight value, the pruned weight is small, i.e., $\|\theta_g^t - \hat{\theta}_g^t\|$ and κ_t are small. So the last three terms also go to zero as $T \to \infty$. Further, the left items of Eq. (7) go to zero as $T \to \infty$, i.e., the algorithm converges. Our experiments also demonstrate the convergence.

C. Convergence Discussion

We do some discussion of Theorem 1 here, including the impact of P_w^t and the comparions with other federated learning methods.

[2]Bounded gradients assumption is commonly used [51], [52], [53].

[3]This convergence analysis method is commonly used [44], [51], [53].

Introduction of P_w^t. The pruning rate P_w^t is related with $m_w^{t,e}$. When P_w^t is introduced to the proof, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\hat{\theta}_g^t)\|^2] \leq \frac{F(\theta_g^0)-F(\theta^*)}{\sqrt{T}E-E^2L} + \frac{LE\sigma^2}{(\sqrt{T}-EL)} + \frac{2\mu\sqrt{\zeta}}{\sqrt{T}E-E^2L}\sum_{t=0}^{T-1}\sqrt{C_t}\kappa_t + \frac{3L\zeta}{2(\sqrt{T}E-E^2L)}\sum_{t=0}^{T-1}C_t\kappa_t^2$$
$$where \quad C_t = \sum_{w=1}^{W} p_w(P_w^{t-1}\prod_{i=0}^{t-2}(1-P_w^i)) \leq 1 \tag{8}$$

Proof when introducing P_w^t. The proof is in Appendix A-C, see the Supplementary Material.

The P_w^{t-1} has an impact on C_t. Given fixed previous pruning rates $P_w^i, i < t-1, t \leq T$, a smaller P_w^{t-1} leads to a smaller C_t. Consequently, the right-hand side of Eq. (8) also becomes smaller, which implies that the upper bound of the mean of the gradients of T rounds becomes lower. This may mean that the algorithm converges faster with a smaller P_w^{t-1}.

Comparison with FedAVG. In the classic FedAVG framework, under Assumption 1, if $\eta = \frac{1}{\sqrt{T}}$ and $T \geq (EL)^2$, T rounds yields the following:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\hat{\theta}_g^t)\|^2] \leq \frac{F(\theta_g^0)-F(\theta^*)}{\sqrt{T}E-E^2L} + \frac{LE\sigma^2}{2(\sqrt{T}-EL)} \tag{9}$$

Proof of FedAVG. The proof is in Appendix A-C, see the Supplementary Material.

Comparing Eq. (7) with Eq. (9), the last three items are due to the involvement of pruning, and their values are positive. Thus, the upper bound of the mean of the gradients of T rounds becomes higher with the involvement of pruning. This may mean that the involvement of pruning makes the algorithm need more rounds to converge. However, the time for each round is significantly reduced after pruning models. From the perspective of convergence time, pruning speeds up convergence.

Comparison with PruneFL and AFD. PruneFL [44] and AFD [45] try to find the same optimal sub-model, i.e., the pruning rate and pruning order are consistent between workers. Under Assumption 1, if $\eta = \frac{1}{\sqrt{T}}$ and $T \geq (EL)^2$, T rounds yields the following:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla F(\hat{\theta}_g^t)\|^2] \leq \frac{F(\theta_g^0)-F(\theta^*)}{\eta TE(1-\eta EL)} + \frac{L\eta E\sigma^2}{2(1-\eta EL)} + \frac{\mu}{\eta TE(1-\eta EL)}\sum_{t=0}^{T-1}\mathbb{E}[\|\theta_g^t-\hat{\theta}_g^t\|] \tag{10}$$

Proof of PruneFL and AFD. The proof is in Appendix A-C, see the Supplementary Material.

Comparing Eq. (7), Eq. (9) and Eq. (10), the first two items of Eq. (7) derive from the federated average training approach, the third item from model pruning, and the last two items from the inconsistent pruning between workers. The upper bound of the mean of the gradients of T rounds is higher compared with the FedAVG and lower compared with the *FedPAGE*. This may mean that the consistent pruning makes the algorithm need fewer rounds to converge. However, inconsistent pruning can make full use of the worker's capabilities and achieve better performance.

V. Experiments

In this section, we intend to answer the following questions.

1) The superiority of *FedPAGE* compared to other efficiency improvement solutions, including local solutions, global solutions and sub-model solutions as clarified in Sec. II-B (Sec. V-B, V-C, V-D).
2) Extensive evaluation on larger number of workers and diverse data heterogeneity settings (Sec. V-E).
3) The deep dive of the *FedPAGE* through ablation experiments (Sec. V-F).

A. Experimental Settings

Datasets, models. We evaluate *FedPAGE* on CIFAR10, CIFAR100 [54], Tiny-ImageNet[4] and MultiNLI (MNLI) Matched.[5] We train a variation of VGG16 on CIFAR10 and CIFAR100, as in [25], ResNet50 [55] on Tiny-ImageNet, and BERT (bert-base-uncased) [56] on MNLI. CIFAR10 and CIFAR100 consist of 50,000 training images and 10,000 validation images in 10 and 100 classes, respectively. Tiny-ImageNet has 200 classes, and each class contains 500 training images, 50 validation images. MNLI has three classes.

Baselines. We compare *FedPAGE* with 12 solutions for efficiency improvement, including six local solutions (FedAVG [1], FedRC [38], FetchSGD [16], FedGen [17]), two global solutions (FedAsync [57], SSP [42]), and six sub-model solutions (Taylor [29], FPGM [24], HRank [25], HeteroFL [6], Helios [5], SplitMix [7]). We adopt the *CIG-BNscalor* for VGG and ResNet, i.e., X is the Batch Normalization's Scalor (*BNscalor*) [23], [58]. Since Layer Normalization is used in BERT, we adopt the *CIG-HIS* for BERT, i.e., X is the Head Importance Score (HIS) [59].

Heterogeneous setting. The heterogeneity H is defined based on the distribution of worker update time ϕ_w as Eq. (11). The definition ensures that H is between 0 and 1, and the closer to 1 the higher the heterogeneity.

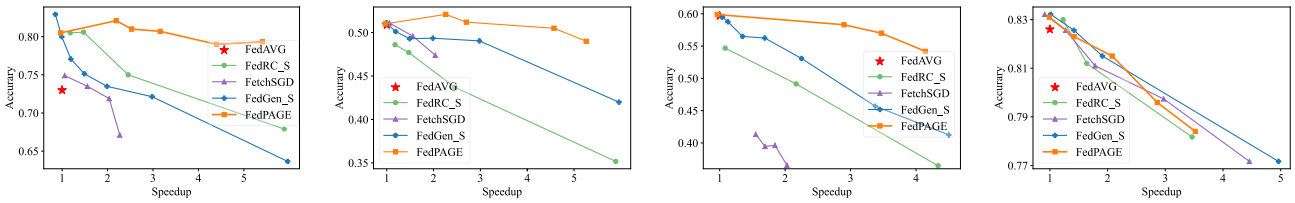
$$H = 1 - \frac{1}{W-1}\sum_{w=1}^{W-1}\frac{\phi_W}{\phi_w}$$
$$Assume\ \phi_W = Min(\phi_1, \ldots, \phi_W) \tag{11}$$

Our experiments achieve heterogeneous computing capabilities by making different workers run on different computing chips, i.e., CPU or GPU, and heterogeneous transmission capabilities by setting different bandwidths.

Non-IID setting. The Non-IID setting is reflected in the uneven distribution of classes. Each worker has the same amount of data but a different number for each class. Specifically, we partition the Non-IID dataset in the same way as [60], [61], and [62]. We divide the (1-s%) of the IID dataset equally

[4]Tiny-ImageNet visual recognition challenge, https://tiny-imagenet.herokuapp.com

[5]https://gluebenchmark.com/tasks



(a) CIFAR10, Non-IID (s=80) (b) CIFAR100, Non-IID (s=80) (c) ImageNet, Non-IID (s=80) (d) MNLI, Non-IID (s=80)

Fig. 7. Comparison with local efficiency improvement solutions on multiple datasets.

TABLE II
HYPERPARAMETERS IN THE EXPERIMENTS

Datasets	Hyperparameters
CIFAR	W=10, T=150, E=2, batchSize=64, PI=10 optimizer=SGD, lr=0.01, weight decay=5e-4,
Tiny-ImageNet	W=10, T=350, E=1, batchSize=64, PI=10 optimizer=SGD, lr=0.1, weight decay=5e-4
MNLI	W=3, T=50, E=0.04, batchSize=32, PI=3 optimizer=Adam, lr=2e-5, weight decay=0.01

to each worker, and the remaining $s\%$ of the IID dataset is sorted by the label and divided sequentially to each worker. We call the partition sort-and-partition scheme.

Configurations. We build *FedPAGE* using PyTorch. The CPUs and GPUs used in experiments are Intel E5-2699 v4 and NVIDIA V100, respectively. The general hyperparameters in the experiments, and the changed hyperparameters are specified in the corresponding sections. In our experiments, β=1.0, γ_{min}=0.1, ρ_{min}=0.2, ρ_{max}=0.5. We adopt the same method in [40] to set sparse coefficient λ by sparsification strength. Sparsification strength is set to 0.9 for CIFAR10, IID CIFAR100 and 0.1 for the others. The model weight p_w of worker w when aggregating models is the proportion of worker data amount to total data amount. The rest shared hyperparameters are presented in Tab. II.

The training of the BERT model follows a triangular learning rate schedule with a warmup ratio of 0.1, a peak learning rate of 2e-5, and a final learning rate of 1.5e-5. For VGG, we do not prune the last fully connected layer. For ResNet, we do not prune the first convolutional layer and the last layer of each residual block. For BERT, we prune the multi-head attention layer and the intermediate layers.

B. Comparison With Local Solutions

Local efficiency improvement solutions focus on speedup the overall transmission or training time. In experiments, we put five workers on the CPU and the remaining five on the GPU. And the ten workers are set different bandwidths. So we achieve heterogeneous initial update time for all workers, even if some run on the same computing chip (see detailed heterogeneity setting in Appendix C, see the Supplementary Material. We report the speedup ratio of the total time compared to FedAVG and the corresponding accuracy.

For FedRC, we enable the algorithm to vary the number of local epochs per round to achieve different speedups by setting different total time limits. For FetchSGD, we achieve different speedups by setting the size of the sketch data structure. FedGen cannot adjust the speedup, but it can speed up the optimization process, i.e., achieve the same accuracy with fewer rounds. So we report the corresponding time and accuracy at different rounds in the figure. For *FedPAGE*, we adjust the minimum retention ratio γ_{min}, the maximum pruning rate ρ_{max}, and the pruned interval PI to achieve different speedups. A larger γ_{min}, smaller ρ_{max} and larger PI both can bring a higher accuracy and a longer total time. More details about the parameters balancing the speedup and accuracy are deferred to Sec. V-F. The suffix *_S* represents the use of the sparse training approach mentioned above. We found that sparse training has no gain for FetchSGD in experiments, so we did not use it for FetchSGD.

CIFAR. For CIFAR, we set a heterogeneous environment where $H \approx 0.82$ according to defined heterogeneity. The results of VGG16 on CIFAR10 and CIFAR 100 are presented in Fig. 7. As we can see, *FedPAGE* outperforms all baselines, achieving the highest accuracy on all speedup ratios.

For CIFAR10 (Fig. 7a), all methods can achieve accuracy improvement compared to FedAVG at lower speedups. Although FedGen achieves the highest accuracy, it consumes more time compared to FedAVG, i.e., the speedup ratio is less than 1. The high accuracy may be related to the knowledge distillation within FedGen. The leftmost point in the *FedPAGE* line shows the result of performing sparse training but no pruning. After moderate pruning, the accuracy of *FedPAGE* has some improvement compared to the accuracy without pruning. The improvement probably has to do with the fact that the pruned parameters are treated as zero in the aggregation, and most of them are optimized toward zero as well. So pruning accelerates the optimization of most of the pruned parameters which is similar to the conclusion of [46]. As the speedup ratio increases, the accuracies of baselines start to drop significantly. The speedup ratio of FetchSGD is constrained by its inability to decrease the training time, which dominates the total time, despite its reduction of the transmission time. For FedRC and FedGen, their accuracies drop by 12% and 20%, respectively, compared to their accuracies at the minimum speedup ratio. *FedPAGE* can do about 6x speedup with a slight drop in accuracy compared with the minimum speedup ratio.

As we can see from Fig. 7b, the accuracies of the baselines do not improve significantly compared to FedAVG, and the speedups they can achieve while maintaining accuracy are pretty low. For FedRC and FedGen, their accuracies drop by 14% and 8%, respectively, compared to their accuracies at the minimum speedup ratio. This is related to the increase in the difficulty of the task, i.e., from a 10-way classification task to a 100-way classification task. In comparison, *FedPAGE* still maintains a better performance. *FedPAGE* achieves the highest accuracy and the accuracy drops by only about 1% at a speedup ratio of about 5.

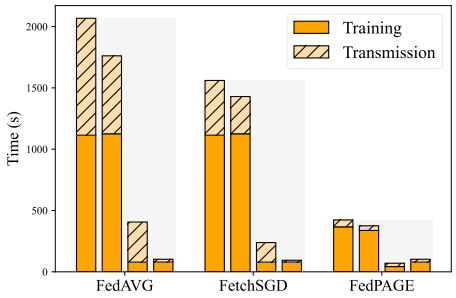


Fig. 8. Comparison of update time composition in a single round. The four bars represent four workers with heterogeneous abilities.

Tiny-ImageNet. For Tiny-ImageNet, we set a heterogeneous environment where $H \approx 0.58$. The results of ResNet50 on Tiny-ImageNet are presented in Fig. 7c. As the task's difficulty increased again (from a 100-way classification task to a 200-way classification task), *FedPAGE* still shows high overall performance, exhibiting its robustness. FedRC and FedGen still have a significant drop in accuracy when the speedup ratio increases, -13% and -8%, respectively. The accuracy of FetchSGD is low, and we infer that the Non-IID ImageNet task is too difficult for it. *FedPAGE* has a drop in accuracy, e.g., -3% with 3.6x, -6% with 4.2x.

MNLI. Since there are only three data classes in the MNLI dataset, we set the number of workers W =3 in the experiment, with one worker running on CPU and the rest on GPU. We set a heterogeneous environment where $H \approx$ 0.8. Since the training is performed on a pre-trained model, all methods quickly achieve high accuracy, so we report the results for different rounds of a single experiment. As we can see from Fig. 7d, all methods achieve good performance. However, the accuracy drop is more notable compared with the previous experiments. We consider that the reason is related to the structure of the pre-trained BERT model. We will conduct further research on the pruning and aggregation of the BERT model.

Internal comparison. We show the update time composition of FedAVG, FetchSGD and *FedPAGE* in Fig. 8. The worker's update time per round includes both model transmission time and training time. The four bars represent four workers with heterogeneous abilities which is illustrated by that the bars of their transmission time and training time are different. The gray bar actually represents the time of each round, and its height is determined by the slowest worker (the straggler), i.e., the worker with the highest bar. In Fig. 8, the transmission times of FetchSGD are significantly decreased for all four workers, but the training times remain unchanged compared with FedAVG. Because FetchSGD and similar methods aiming to reduce the transmission overhead can only reduce the transmission time, not the training time. The speedup that can be achieved is bounded when the worker's training time occupies a certain percentage in the initial update time. As we can see, the speedup ratio of FetchSGD is only about 1.3. We also observe that the transmission time reduction ratio is the same for all workers. This indicates that FetchSGD ignores the heterogeneous abilities of workers. In contrast, *FedPAGE* provides different size networks to each worker by network pruning, which reduces both transmission time and training time differently. Meanwhile, the update time of all workers is converging to the fastest worker. As shown in Fig. 8, the second worker on the right of *FedPAGE* has converged to the fastest worker (the leftmost worker).

TABLE III
COMPARISON WITH GLOBAL EFFICIENCY IMPROVEMENT BASELINES (CIFAR10 AND CIFAR100)

Dataset	Approaches	IID(s=0)		Non-IID(s=80)	
		Acc(%)	Time(min)	Acc(%)	Time(min)
CIFAR10	FedAVG	84.91	270.77	72.06	271.28
	FedAVG_S	87.18	279.37	79.60	279.86
	FedAsync_S	81.70	264.72	69.93	203.40
	SSP_S	85.61	273.08	75.45	282.48
	FedPAGE (Ours)	**87.35**	**171.80**	**80.90**	**157.58**
CIFAR100	FedAVG	55.80	270.40	51.26	271.53
	FedAVG_S	61.65	280.67	51.67	280.63
	FedAsync_S	50.67	211.22	42.98	202.44
	SSP_S	60.66	283.27	50.60	285.03
	FedPAGE (Ours)	**62.17**	**152.45**	**51.85**	**154.32**

C. Comparison With Global Solutions

Global solutions focus on the heterogeneous environments, and try to mitigate the straggler issue from synchronization policy. FedAsync_S [57] is an asynchronous FedAVG using the sparse training approach, where the aggregation weights for the local and global models are set in a polynomial way, and the hyperparameter a is set to 0.5. The aggregation coefficient in SSP [42] is set to $1/W$. The threshold s in SSP is set to 2, 4, and 8, respectively, and we chose the best result as the result of SSP. For FedAsync and SSP, each worker runs T rounds, resulting in $W * T$ aggregations, and we report the best accuracy of $W * T$ aggregations and the corresponding finished time for that round. We put ten workers all on the GPU, and set the bandwidth of workers differently to achieve needed heterogeneity (see detailed heterogeneity setting in Appendix C, see the Supplementary Material. The heterogeneity $H \approx 0.32$.

The results of VGG16 on CIFAR10 and CIFAR 100 are presented in Tab. III. Two asynchronous baselines, FedAsync and SSP have a shorter overall time but lower accuracy compared with FedAVG_S. The lower accuracy of FedAsync and SSP is caused by the gradient staleness issue, which is common in asynchronous ways. The staleness issue is that the model trained at the slowest worker is many rounds behind the latest global model, and the updates obtained from the slowest worker may damage the latest global model or even lead to the dilemma of non-convergence [63]. As we can see, the impact of gradient staleness increases gradually with the difficulty of the task (from CIFAR10 to CIFAR100), the degree of data Non-IID (from IID to Non-IID). We note that the overall time of SSP in some cases also increases, which is because the server needs to do more aggregations, i.e., $W * T$ aggregations. If the server is busy, the worker must wait longer for the new global model after committing the updated model. In contrast, *FedPAGE* has higher accuracy and shorter overall time compared with all baselines. *FedPAGE* still adopts the synchronous way, and thus does not have the gradient staleness issue. The reason for accuracy improvement has been analyzed in Sec. V-C. Meanwhile, *FedPAGE* shortens the overall time by making workers train adaptive models.

D. Comparison With Sub-Model Solutions

Sub-model solutions introduce sub-models of different sizes. Since these works all assume that the worker capabilities are known, for a fair comparison, we set the pruning rate per worker per round before the experiment (see detailed heterogeneity setting in Appendix C, see the Supplementary Material. Thus, each approach assigns sub-model of the same

TABLE IV
COMPARISON WITH SUB-MODEL SOLUTIONS (CIFAR100). THE PRUNING SCHEME *CIG_X* USED IN *FedPAGE* IS *CIG_BNscalor* HERE. sim IS THE SUB-MODEL SIMILARITY OF TWO WORKERS WITH THE SAME PRUNING RATE (E.G., WORKER 4 AND WORKER 6 IN THE FIGURE OF APPENDIX C, SEE THE SUPPLEMENTARY MATERIAL) AS DEFINED IN EQ. (12)

Approaches	IID(s=0)		Non-IID(s=80)	
	Acc(%)	sim(%)	Acc(%)	sim(%)
Taylor [29]	57.06	98.36	49.08	98.11
HRank [25]	54.17	77.27	45.03	64.46
FPGM [24]	57.45	99.88	49.38	99.94
Helios [5]	53.33	34.60	40.99	34.04
HeteroFL [6]	56.43	100	49.28	100
SplitMix [7]	53.24	-	50.08	-
FedPAGE (Ours)	**59.53**	100	**50.11**	100

size to a certain worker per round, and the difference between baselines lies in how the sub-model is obtained. We can assume that all approaches have the same update time per round, so we only compare the accuracy.

For Taylor, FPGM, HRank, HeteroFL and Helios, workers obtain their sub-models individually according to the pruning importance defined by themselves. Other settings are the same as *FedPAGE*. Note that Taylor, FPGM, and HRank are just the stand-alone works of network pruning, and they are applied directly into our framework for comparison. In addition, for Helios, we followed its training approach, i.e., workers reassemble units from the global model to construct their sub-models individually at each round. In Helios, 95% remaining units are selected randomly. For SplitMix, the size of the atomic models is x0.125, i.e., the model's units per layer are 12.5% of those in the base model. We decide the number of atomic models based on the model retention rate. For example, when the model retention ratio of a worker is 0.26, we randomly select two of the eight atomic models and send them to the worker for training. Meanwhile, we define the similarity of two sub-models as in Eq. (12) to help understand the results.

Definition 1: Given sub-model I_w^t of worker w' at round t, the similarity of two workers' ($w1$, $w2$) sub-models is:

$$sim = \frac{1}{N} \sum_{n=1,2,...N} \frac{|I_{w1}^t[n] \cap I_{w2}^t[n]|}{|I_{w1}^t[n] \cup I_{w2}^t[n]|} \tag{12}$$

where $I_w^t[n]$ is the units set of nth layer. N is the number of layers. $|I_{w1}^t[n] \cup I_{w2}^t[n]|$ gets the size of the set.

We show the results on the CIFAR100 dataset in Tab. IV. *FedPAGE* achieves the highest accuracy in both the IID and Non-IID cases. SplitMix shows average performance in IID case, but performs well in Non-IID case. This is related to the fact that the atomic models in it can access all data distributions. And we note that there is a strong correlation between accuracy and sub-model structural similarity. Both HeteroFL and *FedPAGE* guarantee the *identical* and *constant*, i.e., have the maximum similarity. But *FedPAGE* has the advantage that the pruning order takes into account the unit importance. The remaining sub-model similarity of FPGM is slightly higher than that of Taylor, and its accuracy is also slightly higher than that of Taylor. The similarity of HRank is relatively low compared to the above methods, and its accuracy is also relatively low. Helios randomly selects a large percentage of units each round, so its remaining sub-models have the lowest similarity. Correspondingly, Helios has the lowest accuracy. Therefore, this again demonstrates that structural similarity between sub-models is the key to *distributed pruning*, and *identical* and *constant* ensure maximum similarity between sub-models.

E. Evaluation Under Extensive Settings

In this section, we evaluate the performance of *FedPAGE* under extensive settings. We extend the number of workers to 100 and introduce two new realistic data heterogeneity settings for evaluation, i.e., dirichlet distribution partition scheme and cross domains heterogeneity.

Dirichlet distribution partition scheme. We follow the Non-IID partition as in [64], [65], [66]. For each class c, $p_{w,c}$ proportion of the instances is allocated to the worker w, where $p_{w,c} \sim Dir(s)$. When s approaches infinity, each class is more evenly distributed on each worker. Conversely, as s approaches zero, the distribution gets more skewed. We set $s = 0.1$ and $s = 0.8$ in experiments. We visualize two data heterogeneity (Non-IID) settings in Appendix B, see the Supplementary Material.

Cross domains heterogeneity. We introduce another data heterogeneity here, i.e., the datasets are collected from different domains. We adopt a subset of Digits dataset, a benchmark for domain adaption [67] as in [7]. Digits also serves as a commonly used benchmark for FL [7], [39], [68]. The dataset is from five domains: MNIST [69] (handwritten digits), SVHN [70] (cropped from pictures of house number plates), USPS [71] (scanned from envelopes), SynthDigits [72] (generated from Windows fonts), MNIST-M [72] (difference-blended digits over non-uniform background). Each domain has 7438 training images. Similar to the partition in [7], each domain of Digits is split into 20 workers, and therefore 100 workers in total. In this way, each worker has samples for only one domain.

We set the number of workers $W = 100$, and the sparsification strength is set to 0.9. The other settings are the same as in Tab. II. The baselines here are the sub-model solutions that are closer to our framework. Due to the poor performance of HRank and Helios, we do not choose them as a baseline here. The heterogeneity setting is similar to Sec. V-D (see detailed heterogeneity setting in Appendix C, see the Supplementary Material. In addition, we also introduce the standard FedAVG for comparison. Note that the models in FedAVG are not pruned.

Non-IID settings. We set local epoch $E = 5$, update rounds $T = 600$ for $s = 80$, and $T = 300$ for other settings. As presented in the Tab. V, *FedPAGE* achieves the best overall performance. *FedPAGE* achieves the highest accuracy under moderate heterogeneity, and has a relatively large gap (1.15% in $s = 0$ and 1.57% in $s = 0.8$) with the second-place approach. When the heterogeneity increases, *FedPAGE* has a small gap (0.18% in $s = 0.1$) with the second-place approach SplitMix. This may be related to the fact that the atomic models in SplitMix can access all data distributions. Thus, the benefit is more prominent when the data distribution between workers is more different. However, when the heterogeneity continues to increase ($s = 80$), the accuracy gap between SplitMix and *FedPAGE* widens again. This indicates *FedPAGE* is also well suited to highly heterogeneous environments. FedAVG performs the worst in all settings, especially when

TABLE V
COMPARISON WITH BASELINES ON MULTIPLE NON-IID SETTINGS WITH 100 WORKERS (CIFAR100). THE PRUNING SCHEME *CIG_X* USED IN *FedPAGE* IS *CIG_BNscalor* HERE

Approaches	Sort-and-partition		Dirichlet partition	
	$s = 0$	$s = 80$	$s = 0.8$	$s = 0.1$
FedAVG [1]	45.92	26.41	46.35	44.14
Taylor [29]	47.07	32.79	46.69	44.88
FPGM [24]	47.97	35	46.97	45.99
HeteroFL [6]	47.96	34.98	48.46	46.11
SplitMix [7]	47.64	34.93	47.52	47.03
FedPAGE (Ours)	**49.12**	**36.99**	**50.03**	**47.21**

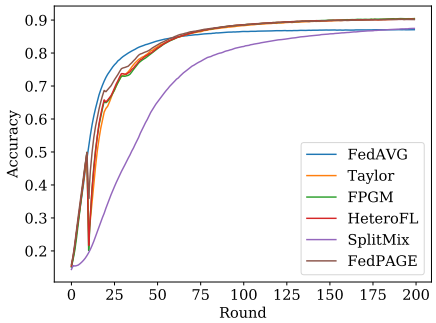


Fig. 9. Accuracy of different rounds on the digits dataset.

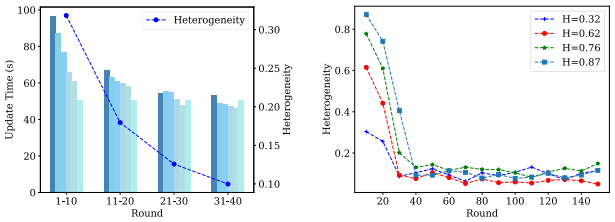
the degree of worker heterogeneity increases. For example, when $s = 80$, its accuracy is 10% lower than that of *FedPAGE*. We speculate that this may be related to the ability of sub-models to learn more refined knowledge.

Cross domains settings. We set local epoch $E = 2$, update rounds $T = 200$. As illustrated in Fig. 9, emFedPAGE, Taylor, FPGM, and HeteroFL have similar performance, and have higher accuracy compared with FedAVG and SplitMix. This indicates that *FedPAGE* is also applicable to cross domains data heterogeneity. FedAVG converges the fastest, which is consistent with our theoretical analysis in Sec. IV-C. SplitMix converges the slowest, and eventually reaches the same accuracy as FedAVG. Since SplitMix does not prune model, but changes the number of atomic models, the accuracy curve is relatively smooth, and there is no sudden drop in accuracy like *FedPAGE*.

F. Deep Dive of FedPAGE

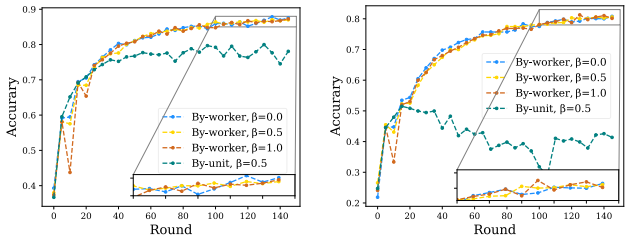
In this section, we show more details about the *FedPAGE*, including the performance of the pruning rate learning approach, the impact of pruning position, model aggregating approach, speedup adjusting parameters and compatibility with local solutions (see heterogeneity settings in Appendix C, see the Supplementary Material).

Pruning rate learning approach. We selected six workers to show their average update time during the first four pruning intervals, as shown in Fig. 10a. As training proceeds, the pruning rate learning approach assigns an adaptive pruning rate to each worker, and the update time of all workers gradually tends to the fastest worker. Meanwhile, the heterogeneity of update time between workers rapidly decreases. There are no more stragglers in the system by internal adjusting, and the system's efficiency increases dramatically. We show multiple



(a) Mean update time of par-tial workers. PI=10, H=0.32. (b) Multiple cases of initial heterogeneity.

Fig. 10. Pruning rate learning approach performance.



(a) CIFAR10, IID (b) CIFAR10, Non-IID (s=80)

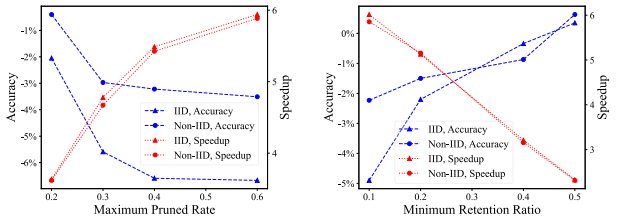
Fig. 11. Comparison of different pruning positions and model aggregation methods.

cases of initial heterogeneity in Fig. 10b. In *FedPAGE*, the heterogeneity of update time between workers rapidly decreases and stabilizes quickly regardless of the initial degree of heterogeneity. This reflects that our pruning rate learning approach can dynamically give an adaptive pruning rate to make the worker update time converge to the minimum update time.

Impact of pruning position and model aggregating. For a fair comparison, we set the pruning rate per worker per round before the experiment. We report the performance under different pruning positions and model aggregation approaches in Fig. 11. The smaller the β, the accuracy drop is smaller after each pruning. When $\beta = 1.0$, since no local fine-tune is done after local pruning, accuracy dropped sharply at first but recovered quickly later. Overall, the pruning position has little effect on accuracy. Under the IID setting (Fig. 11a), when By-unit is used for model aggregation, the model does not have a drop in accuracy after pruning. The accuracy continues to rise after pruning but soon stops rising. However, accuracy no longer rises after pruning and stays at a low value under the Non-IID setting (Fig. 11b). In our opinion, By-unit treats the pruned weights (those weights are pruned due to extremely low values) as the mean of the unpruned weights of the other local models at the corresponding location. Thus the global model no longer reflects the information from local models.

Impact of adjusting parameters. Here, we analyze the impact of the main adjusting parameters inside *FedPAGE* on performance, i.e., maximum pruning rate ρ_{max} and minimum retention ratio γ_{min}. Fig. 12 reports the performance of accuracy and speedup on the CIFAR100 dataset under different adjusting parameters, taking FedAVG_S as a comparison.

For maximum pruning rate ρ_{max} (Fig. 12a), the model achieves better accuracy when ρ_{max} is smaller, but more rounds are required for the pruning, causing the overall time to rise (e.g., -0.4% with 3.62x when $\rho_{max} = 0.2$ vs. -3.51% with 5.88x when $\rho_{max} = 0.6$, Non-IID). For minimum retention ratio γ_{min} (Fig. 12b), the model achieves better accuracy when γ_{min} is bigger, but more parameters are left behind resulting



(a) Maximum pruning rate ρ_{max} (b) Minimum retention ratio γ_{min}

Fig. 12. Performance of *FedPAGE* on CIFAR100 comparing to FedAVG_S under different adjusting parameters (H=0.87).

TABLE VI
FEDPAGE+DGC (CIFAR10, NON-IID, s=80). SPARSITY REPRESENTS THE RATIO OF UNCOMMITTED WEIGHTS

Sparsity	Acc(%)	Transmission Amount
0.00	80.90%	52.20%
0.70	81.49%	35.55%
0.90	**81.95%**	12.50%
0.99	79.55%	2.13%

in higher overall time (e.g., -4.9% with 6.01x when $\gamma_{min} = 0.1$ vs. 0.35% with 2.32x when $\gamma_{min} = 0.5$, IID). As we can see from the results above, the adjusting parameters allow us to do a trade-off between accuracy and speedup. When accuracy is more of a concern, a low maximum pruning rate as well as a high minimum retention ratio can be set, and vice versa.

Compatibility with Local Solutions. *FedPAGE* is orthogonal to local solutions, e.g., we can introduce the gradient quantization or the optimization way of FedGen to achieve further speedup.

We show an example of the combination with DGC [36] here. DGC reduces the transmission overhead by committing only some of the essential gradients, and the uncommitted gradients are accumulated locally until a certain threshold is reached. We use DGC to compress weights after pruning and report the *FedPAGE*+DGC results in Tab. VI. When not combined with DGC, i.e., sparsity is 0.0, the transmission amount is already 52.2% of parameters transmitted in FedAVG. When sparsity is 0.9, the accuracy is highest and the transmission amount is further reduced to 12.5%. And we note that when only 2.13% parameters are transmitted, the accuracy is also acceptable. In conclusion, *FedPAGE*+DGC can further reduce the amount of transmission and bring an accuracy improvement.

VI. DISCUSSION

Addressable Market Analysis. *FedPAGE* can improve the training speed of federated learning while maintaining accuracy, which is applicable to most federated learning scenarios and extremely beneficial for some scenarios. For example, in the situation where data is constantly generated, such as industrial internet of things scenario, *FedPAGE* enables fast mining of the value of the latest data in a timely manner. In the situation where workers can only participate in training for a short time, such as phones are allowed to participate in training only when idle, *FedPAGE* can efficiently utilize phones' data.

Limitations. (i) When the gap between worker capabilities is too large, the speedup ratio that *FedPAGE* can achieve is limited. Because in order to ensure accuracy, the set minimum model retention ratio will limit the model size of the slow worker, and the update time of the slow worker is difficult to converge to that of the fastest worker, i.e., its update time may still be relatively long. However, compared with the FedAVG, our method can still guarantee a certain speedup ratio. (ii) When the worker capabilities fluctuate greatly, our pruning rate learning algorithm cannot model the worker capabilities well. For example, when multiple other tasks occupy the CPU and memory of the worker at the same time, the training capability of worker will fluctuate greatly. When other tasks that occupy bandwidth or the worker is in motion, the transmission capability of the worker will fluctuate greatly. However, the collaborating parties usually establish a relatively stable collaboration environment, e.g., Google lets phone join the collaboration when the phone is idle and connected to WIFI [48]. When the worker capabilities change periodically, we can extend our method to achieve local model expansion or contraction.

VII. CONCLUSION

In this paper, we propose a novel and efficient federated learning framework named *FedPAGE*, which generates an adaptive sparse sub-model dynamically from the global base model for each worker based on its capability. By equipping capability-different workers with adaptive size sub-models, all workers commit model updates near-synchronously, thus avoiding the straggler issue. We discuss in detail model training, pruning, and aggregation in the framework and the design of the dynamic pruning rate learning approach. In addition, we give the convergence proof for *FedPAGE*. Extensive experiments on various models and datasets demonstrate the efficiency of *FedPAGE*. In the future, we will do more theoretical research on distributed adaptive pruning and explore more efficient and precise pruning approaches adapting to the dynamic heterogeneous environment.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273–1282.
[2] L. G. Valiant, "A bridging model for parallel computation," *Commun. ACM*, vol. 33, no. 8, pp. 103–111, Aug. 1990.
[3] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments," in *Proc. USENIX OSDI*, 2008, pp. 1–7.
[4] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: Attack of the clones," in *Proc. USENIX NSDI*, 2013, pp. 185–198.
[5] Z. Xu, F. Yu, J. Xiong, and X. Chen, "Helios: Heterogeneity-aware federated learning with dynamically balanced collaboration," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 997–1002.
[6] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," in *Proc. ICLR*, 2021, pp. 1–24.
[7] J. Hong, H. Wang, Z. Wang, and J. Zhou, "Efficient split-mix federated learning for on-demand and in-situ customization," in *Proc. ICLR*, 2022, pp. 1–18.
[8] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*.
[9] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, "Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks," in *Proc. NIPS*, 2019, pp. 2133–2144.
[10] C. T. Tan and M. Motani, "DropNet: Reducing neural network complexity via iterative pruning," in *Proc. ICML*, 2020, pp. 9356–9366.

[11] Z. Zhang et al., "SecCL: Securing collaborative learning systems via trusted bulletin boards," *IEEE Commun. Mag.*, vol. 58, no. 1, pp. 47–53, Jan. 2020.

[12] Q. Zhang, B. Gu, C. Deng, and H. Huang, "Secure bilevel asynchronous vertical federated learning with backward updating," in *Proc. AAAI*, 2021, pp. 10896–10904.

[13] S. Yao et al., "Blockchain-empowered collaborative task offloading for cloud-edge-device computing," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3485–3500, Dec. 2022.

[14] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[15] B. Zhao, P. Sun, T. Wang, and K. Jiang, "FedInv: Byzantine-robust federated learning by inversing local model updates," in *Proc. AAAI*, 2022, pp. 9171–9179.

[16] D. Rothchild et al., "FetchSGD: Communication-efficient federated learning with sketching," in *Proc. ACM ICML*, 2020, pp. 8253–8265.

[17] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. ICML*, 2021, pp. 12878–12889.

[18] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. AAAI*, 2018, pp. 1–8.

[19] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, and J. Yan, "DARTS-: Robustly stepping out of performance collapse without indicators," in *Proc. ICLR*, 2021, pp. 1–22.

[20] J. Park et al., "Faster CNNs with direct sparse convolutions and guided pruning," in *Proc. ICLR*, 2017, pp. 1–12.

[21] 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.

[22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proc. ICLR*, 2017, pp. 1–13.

[23] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2755–2763.

[24] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4335–4344.

[25] M. Lin et al., "HRank: Filter pruning using high-rank feature map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1526–1535.

[26] S. Lin et al., "Towards optimal structured CNN pruning via generative adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2785–2794.

[27] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. NIPS*, 2015, pp. 1135–1143.

[28] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1515–1525.

[29] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. ICLR*, 2017, pp. 1–17.

[30] A. Kusupati et al., "Soft threshold weight reparameterization for learnable sparsity," in *Proc. ACM ICML*, 2020, pp. 5544–5555.

[31] C. T. Dinh et al., "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021.

[32] J. Lu, H. Liu, R. Jia, J. Wang, L. Sun, and S. Wan, "Towards personalized federated learning via group collaboration in IIoT," *IEEE Trans. Ind. Informat.*, vol. 19, no. 8, pp. 8923–8932, Aug. 2023.

[33] W. Wen et al., "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. NIPS*, 2017, pp. 1508–1518.

[34] K. Hsieh et al., "GAIA: Geo-distributed machine learning approaching LAN speeds," in *Proc. USENIX NDSI)*, 2017, pp. 629–647.

[35] J. Konecny, 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*.

[36] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. ICLR*, 2018, pp. 1–14.

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

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

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

[40] S. Lym, E. Choukse, S. Zangeneh, W. Wen, S. Sanghavi, and M. Erez, "PruneTrain: Fast neural network training by dynamic sparse model reconfiguration," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2019, pp. 1–13.

[41] W. Dai, A. Kumar, J. Wei, Q. Ho, G. A. Gibson, and E. P. Xing, "High-performance distributed ML at scale through parameter server consistency models," in *Proc. AAAI*, 2015, pp. 1–9.

[42] Q. Ho et al., "More effective distributed ML via a stale synchronous parallel parameter server," in *Proc. NIPS*, 2013, pp. 1223–1231.

[43] R. Zhu, S. Yang, A. Pfadler, Z. Qian, and J. Zhou, "Learning efficient parameter server synchronization policies for distributed SGD," in *Proc. ICLR*, 2019, pp. 1–10.

[44] Y. Jiang et al., "Model pruning enables efficient federated learning on edge devices," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 25, 2022, doi: 10.1109/TNNLS.2022.3166101.

[45] N. Bouacida, J. Hou, H. Zang, and X. Liu, "Adaptive federated dropout: Improving communication efficiency and generalization for federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–6.

[46] H. Zhou, J. Lan, R. Liu, and J. Yosinski, "Deconstructing lottery tickets: Zeros, signs, and the supermask," in *Proc. NIPS*, 2019, pp. 3597–3607.

[47] H. Wang et al., "HAT: Hardware-aware transformers for efficient natural language processing," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 1–14.

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

[49] T. Sauer, *Numerical Analysis*. Reading, MA, USA: Addison-Wesley, 2006.

[50] T. Sun et al., "Learning sparse sharing architectures for multiple tasks," in *Proc. AAAI*, 2020, pp. 8936–8943.

[51] 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*, 2019, pp. 5693–5700.

[52] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. ICLR*, 2020, pp. 1–26.

[53] J. Perazzone, S. Wang, M. Ji, and K. S. Chan, "Communication-efficient device scheduling for federated learning using stochastic optimization," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 1449–1458.

[54] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

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

[56] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

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

[58] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *Proc. ICLR*, 2018, pp. 1–11.

[59] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?" in *Proc. NIPS*, 2019, pp. 14037–14047.

[60] F. Sattler, S. Wiedemann, K.-R. Müller, 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.

[61] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. ACM ICML*, 2020, pp. 5132–5143.

[62] Z. Shen, J. Cervino, H. Hassani, and A. Ribeiro, "An agnostic approach to federated learning with class imbalance," in *Proc. ICLR*, 2022, pp. 1–12.

[63] W. Dai, Y. Zhou, N. Dong, H. Zhang, and E. Xing, "Toward understanding the impact of staleness in distributed machine learning," in *Proc. ICLR*, 2019, pp. 1–19.

[64] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proc. ACM ICML*, 2019, pp. 7252–7261.
[65] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *Proc. ICLR*, 2021, pp. 1–43.
[66] Y. Guo, Y. Sun, R. Hu, and Y. Gong, "Hybrid local SGD for federated learning with heterogeneous communications," in *Proc. ICLR*, 2022, pp. 1–42.
[67] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," in *Proc. ICLR*, 2019, pp. 1–19.
[68] S. Caldas et al., "LEAF: A benchmark for federated settings," 2018, *arXiv:1812.01097*.
[69] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[70] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9.
[71] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.
[72] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proc. ICML*, 2015, pp. 1180–1189.

Guangmeng Zhou received the B.E. degree from the School of Computer Science and Technology, Tianjin University, Tianjin, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Tsinghua University, Beijing, China. His main research interests include collaborative learning, network security, and programmable switch.

Qi Li (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University. He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include internet and cloud security, mobile security, and big data security. He is an Editorial Board Member of IEEE TRANSACTIONS ON DEPENDABLE AND SECURITY COMPUTING and *ACM DTRAP*.

Yang Liu received the B.E. degree from the School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2019, and the master's degree from the School of Computer Science and Technology, Tsinghua University, Beijing, in 2022. He is currently with Huawei Technologies Company Ltd. His main research interests include collaborative learning.

Yi Zhao (Member, IEEE) received the B.Eng. degree from the School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, China, in 2016, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2021. He is currently an Assistant Researcher and a Post-Doctoral Fellow with the Department of Computer Science and Technology, Tsinghua University. His research interests include next-generation internet, network security, machine learning, and game theory. He is a member of ACM. He was a recipient of the Shuimu Tsinghua Scholar Program.

Qi Tan received the B.Eng. degree in 2012 and the master's degree from Tsinghua University, Beijing, China, in 2019, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology. His research interests include machine learning, network security, and data privacy.

Su Yao received the Ph.D. degree from the National Engineering Laboratory for Next Generation Internet, Beijing Jiaotong University. He is currently with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University. His research interests include future network architecture and the IoT security.

Ke Xu (Senior Member, IEEE) received the Ph.D. degree from Tsinghua University, Beijing, China. He is currently a Full Professor with the Department of Computer Science, Tsinghua University. He has published more than 200 technical articles and holds 11 U.S. patents in the research areas of next-generation internet, blockchain systems, the Internet of Things, and network security. He serves as the Steering Committee Chair for IEEE/ACM IWQoS. He has guest-edited several special issues for IEEE and Springer journals. He is an Editor of IEEE INTERNET OF THINGS JOURNAL.