Efficient Federated Learning Using Dynamic Update and Adaptive Pruning with Momentum on Shared Server Data

JI LIU, Hithink RoyalFlush Information Network Co., Ltd., Hangzhou, China
JUNCHENG JIA, School of Computer Science and Technology, Soochow University, Suzhou, China and Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China
HONG ZHANG, School of Computer Science and Technology, Soochow University, Suzhou, China
YUHUI YUN, Baidu Research, Beijing, China
LEYE WANG, Key Lab of High Confidence Software Technologies, Ministry of Education and Software Institute, Peking University, Beijing, China
YANG ZHOU, Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, USA
HUAIYU DAI, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA
DEJING DOU, BEDI Cloud and School of Computer Science, Fudan University, Beijing, China

Despite achieving remarkable performance, Federated Learning (FL) encounters two important problems, i.e., low training efficiency and limited computational resources. In this article, we propose a new FL framework, i.e., FedDUMAP, with three original contributions, to leverage the shared insensitive data on the server in addition to the distributed data in edge devices so as to efficiently train a global model. First, we propose a simple dynamic server update algorithm, which takes advantage of the shared insensitive data on the server while dynamically adjusting the update steps on the server in order to speed up the convergence and improve the accuracy. Second, we propose an adaptive optimization method with the dynamic server update algorithm to exploit the global momentum on the server and each local device for superior accuracy. Third, we develop a layer-adaptive model pruning method to carry out specific pruning operations, which is adapted to the diverse features of each layer so as to attain an excellent tradeoff between effectiveness and efficiency. Our proposed FL model, FedDUMAP, combines the three original techniques and has a significantly better performance compared with baseline approaches in terms of efficiency (up to 16.9 times faster), accuracy (up to 20.4% higher), and computational cost (up to 62.6% smaller).

Ji Liu, Juncheng Jia, and Hong Zhang contributed equally to this research.

Authors' Contact Information: Ji Liu (corresponding author), Hithink RoyalFlush Information Network Co., Ltd., Hangzhou, China; e-mail: jiliuwork@gmail.com; Juncheng Jia, School of Computer Science and Technology, Soochow University, Suzhou, China and Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China; e-mail: jiajuncheng@suda.edu.cn; Hong Zhang, School of Computer Science and Technology, Soochow University, Suzhou, China; e-mail: hzhanghzhang@stu.suda.edu.cn; Yuhui Yun, Baidu Research, Beijing, China; e-mail: yunyh0331@gmail.com; Leye Wang, Key Lab of High Confidence Software Technologies, Ministry of Education and Software Institute, Peking University, Beijing, China; e-mail: leyewang@pku.edu.cn; Yang Zhou, Department of Computer Science and Software Engineering, Auburn University, Auburn, Alabama, USA; e-mail: yangzhou@auburn.edu; Huaiyu Dai, Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, North Carolina, USA; e-mail: hdai@ncsu.edu; Dejing Dou, BEDI Cloud and School of Computer Science, Fudan University, Beijing, China; e-mail: dejingdou@gmail.com.

ACM 2157-6912/2024/11-ART122
https://doi.org/10.1145/3690648

CCS Concepts: • **Theory of computation → Parallel computing models**; **Distributed computing models**; **Convex optimization**; • **Computing methodologies → Machine learning**; • **Computer systems organization → Heterogeneous (hybrid) systems**;

Additional Key Words and Phrases: Federated learning; Distributed machine learning; Model pruning; Heterogeneity; Momentum

ACM Reference format:
Ji Liu, Juncheng Jia, Hong Zhang, Yuhui Yun, Leye Wang, Yang Zhou, Huaiyu Dai, and Dejing Dou. 2024. Efficient Federated Learning Using Dynamic Update and Adaptive Pruning with Momentum on Shared Server Data. *ACM Trans. Intell. Syst. Technol.* 15, 6, Article 122 (November 2024), 28 pages.
https://doi.org/10.1145/3690648

1 Introduction

With a large quantity of data distributed in numerous edge devices, cloud servers may have some shared insensitive data for efficient processing [1]. Due to data privacy and security concerns, multiple legal restrictions [2, 3] have been put into practice, making it complicated to aggregate the distributed sensitive data within a single high-performance server. As big amounts of training data generally lead to high performance of machine learning models [4], **Federated Learning (FL)** [5–7] becomes a promising approach to collaboratively train a model with considerable distributed data while avoiding raw data transfer.

Traditional FL exploits **non-Independent and Identically Distributed (non-IID)** data on mobile devices to collaboratively train a global model [5]. Within the training process, the weights or the gradients of the global model are transferred between the devices and the server while the raw data stays in each device. FL typically utilizes a parameter server architecture [6, 8], where a parameter server (server) coordinates the distributed training process. The training process generally consists of multiple rounds, each composed of three stages. First, the server chooses a set of devices and broadcasts the global model to them. Second, each of the chosen devices trains the received model using local data and then sends back the updated model. Third, the server gathers the updated models and aggregates them to form a new global model. This process continues when the predefined condition, e.g., a maximum round number or the convergence of the global model, is not achieved.

Although keeping data within the devices can protect data privacy and security, FL encounters two major challenges, which hinder its application in real-world environments. The first challenge is the limited data within a single edge device, which leads to ineffective local training while potential large amounts of shared insensitive data remain useless on the server. The second challenge is the modest edge devices with limited communication and computation capacity [9], which corresponds to inefficient local training on devices. To address the first challenge, we leverage the shared insensitive data on the server and design a new FL framework to improve the accuracy of the global model. To address the second challenge, we propose an adaptive optimization method and an adaptive pruning method within the proposed FL framework to improve model accuracy and reduce training cost.

In addition to the distributed data in edge devices, insensitive data can be transferred to the servers in the cloud to leave space for critical sensitive personal data [10, 11], e.g., Amazon Cloud [12], Microsoft Azure [13], and Baidu Artificial Intelligence Cloud [14]. The insensitive data may exist on the server by default before the training process. For instance, the server may have prior learning tasks, either training or inference, which have already collected some data to be re-used for the new task of interest. In addition, some end users may offload some insensitive data to the server with certain incentive mechanism such as for crow-sourcing tasks [15], or simply to free

up some local storage space [16]. Some other recent works are based on the insensitive shared data on the server [17–19]. The shared insensitive data can help improve the efficiency of the FL training process without the restriction of data distribution (see details in Section 3.2). The shared insensitive data can be transferred to the devices for local training in order to improve the accuracy of the global model [1, 20]. Although the shared data on the server is not sensitive, they still contain some important personal information. Thus, transferring the shared data to devices may still incur privacy or security problem. In addition, this approach leads to significantly high communication overhead. Furthermore, the existing approaches are inefficient when the shared server data is simply processed as that in edge devices [11] or when knowledge transfer is utilized to deal with heterogeneous models [21, 22].

Adaptive optimization methods, such as **Stochastic Gradient Descent with Momentum (SGDM)** [23], **Adaptive Moment Estimation (Adam)**, and **Adaptive Gradient (AdaGrad)**, have gained superb advance in accelerating the training speed and the final accuracy. Most of the existing FL literature adopts the adaptive optimization methods either on the server side [24–27] or on the device side [27–31]. However, applying the adaptive optimization on either of them often leads to inefficient learning results with inferior accuracy. The adaptive optimization methods can be exploited on both sides, as shown in [32], but the momentum transfer may incur severe communication costs with limited bandwidth between the devices and the server.

Model pruning reduces the size of a model into a slim one while the accuracy remains acceptable. These kinds of methods can be exploited in the training process to improve the efficiency of the training and to significantly reduce the overhead brought by a big model [33]. However, existing pruning approaches incur severe accuracy degradation. The simple application of existing pruning methods in FL does not consider the diverse dimensions and features of each layer [34]. In addition, as a big model may consist of numerous neurons, simple model pruning strategies cannot choose proper portions of the model to prune and may lead to low training efficiency and inferior accuracy [1].

In this work, we introduce a novel efficient FL framework, i.e., *FedDUMAP*, which enables collaborative training of a global model based on the sensitive device data and insensitive server data with a powerful server and multiple modest devices. We denote the sensitive data as the data that contain personal information and cannot be shared or transferred according to the users. In addition, the insensitive data can contain personal information while it can be transferred to a cloud server. In order to handle the two aforementioned problems, we utilize the shared insensitive data on the server to improve the accuracy of the global model with adaptive optimization with the consideration of the non-IID degrees, which represent the difference between a dataset (either on the server or a device) and the global dataset, i.e., the data on all the devices.

FedDUMAP consists of three modules, i.e., FedDU, FedDUM, and FedAP. FedDU is an FL algorithm, which dynamically updates the global model based on the distributed sensitive device data and the shared insensitive server data. In addition, FedDU dynamically adjusts the global model based on the accuracy and the non-IID degrees of the server data and the device data. Furthermore, we propose a novel adaptive optimization method on top of FedDU, i.e., FedDUM, to improve the accuracy of the model without transferring the momentum from devices to the server or from the server to the devices. We decouple the optimization on the server and the device sides while exploiting the model generated with adaptive optimization from each device to enable the adaptive optimization on the server side. Besides, FedAP removes useless neurons in the global model to reduce the model size so as to reduce the computational overheads and the computation costs on devices without degrading noticeable accuracy. FedAP considers the features of each layer to identify a proper pruning rate, based on the non-IID degrees. Furthermore, FedAP prunes the neurons according to the rank values, which can preserve the performance of the model. To the best of our knowledge, we are among the first to propose exploiting non-IID degrees for dynamic global

model updates while utilizing adaptive optimization and adaptive pruning for FL. This manuscript is an extension of a conference version [35]. In this article, we make four following contributions:

(1) A novel dynamic FL algorithm, i.e., FedDU, which utilizes both shared insensitive server data and distributed sensitive device data to collaboratively train a global model. FedDU dynamically updates the global model with the consideration of the model accuracy, normalized gradients from devices, and the non-IID degrees of both the server data and the device data.
(2) A new adaptive optimization method, i.e., FedDUM, which decouples the optimization between the server side and the device side while exploiting the models generated from devices for the optimization on the server side without additional communication cost.
(3) An original adaptive pruning method, i.e., FedAP, which considers the features of each layer to identify a proper pruning rate based on the non-IID degrees. FedAP prunes the model based on the rank values to preserve the performance of the global model.
(4) Extensive experimentation demonstrates significant advantages of FedDUMAP, including FedDU, FedDUM, and FedAP, in terms of efficiency, accuracy, and computational cost based on three typical models and two real-life datasets.

The rest of this article is organized as follows. Section 2 explains the related work. Section 3 proposes our framework, i.e., FedDUMAP, including FedDU, FedDUM, and FedAP. Section 4 presents the experimental results using three typical models and two real-life datasets. Finally, Section 5 concludes the article.

2 Related Work

FL was proposed to train a model using the distributed data within multiple devices while only transferring the model or gradients [5]. Some works ([36–45] and references therein) either focus on the device scheduling or the model aggregation within the server or even with a hierarchical architecture to improve the accuracy of the global model, which only deals with the distributed device data. In order to leave space for sensitive data on devices and with incentive mechanisms [11], some insensitive data are transferred to the server or the cloud, which can be directly utilized for training. When all the data are transferred to the server, the server data can be considered as IID data [11]. However, transferring all the data including the sensitive data incurs severe privacy and security issues with significant communication costs, which is not realistic [46]. Some existing works utilize the insensitive server with heterogeneous models [47], or within one-shot model training [4], based on knowledge transfer methods [48, 49], or label-free data [22, 50]. However, the aforementioned approaches are inefficient [4, 47] or ineffective [22] without the consideration of the non-IID degrees or big models. Furthermore, while it may lead to significant communication overhead, the transfer of the insensitive server data to devices [1, 20] may incur severe privacy and security issues. The proper devices can be selected for training based on the server data [51], which can be integrated with our proposed approach.

Without adaptive optimization within the model aggregation process, traditional methods, e.g., **Federated averaging (FedAvg)** [5], may incur the client drift problem, which makes the global model over-fitted to local device data [52]. While control parameters [52] can help alleviate the problem, they require the devices to participate all through the training process [52, 53], which may not be feasible in cross-device FL. Adaptive optimization methods, e.g., AdaGrad [24], Yogi [25], Adam [26, 27], and momentum [32, 54–56] can be exploited to address the client drift problem. However, the existing approaches generally consider only one side, i.e., either server side [57] or device side [28], which may lead to inferior accuracy. Furthermore, the devices may

have heterogeneous non-IID data, which makes the direct application of the momentum method insufficient [29]. Although adaptive optimization is applied on both the device side and the server side in some recent works [32], the communication of momentum between the server and devices may incur high communication costs.

Model pruning can reduce the size of the model, which corresponds to smaller communication overhead and computation costs compared with the original model within the training process [33, 58–60] and the inference process [61] of FL. However, the shared insensitive server data is seldom utilized. Two types of techniques exist for the pruning process, including filter (structured) pruning and weight (unstructured) pruning. The unstructured pruning set the chosen parameters to 0 while the structure of the original model remains unchanged. Although the accuracy remains almost the same as that of the original one [62] while reducing communication overhead [33], the unstructured pruning cannot reduce computational cost with general-purpose computing resources, e.g., edge devices [34]. On contrary, structured pruning can modify the model structure by abandoning selected neurons or filters in the original model. As some parts of the original model are removed, the structured pruning leads to significantly smaller computational costs and communication overhead [34]. However, it is non-trivial to determine the pruning rate, which hinders the application of unstructured pruning in the training process of FL. Sparsification, quantization [63, 64], or dropout [65, 66] are efficient techniques to reduce the communication overhead as well, which is out of the scope of this article and can be combined with our proposed approach for better performance.

In this article, we utilize both the shared insensitive server data and the distributed sensitive device data for FL training. We propose an FL framework, i.e., FedDUMAP, consisting of FedDU, FedDUM, and FedAP. FedDU dynamically adjusts the global model with the shared insensitive server data based on the non-IID degrees and model accuracy. FedDUM considers an adaptive optimization method on both the device and the server sides without additional communication overhead. Furthermore, FedAP prunes the global model with the consideration of the features of each layer and removes the filters based on the rank values to reduce communication and computational costs while achieving high accuracy. The pruning rate is calculated based on the non-IID degrees. FedDUMAP can achieve high accuracy and efficient training at the same time.

3 Method

In this section, we propose our FL framework, i.e., FedDUMAP. We first present the system model. Then, we detail FedDU, which dynamically updates the global model. Afterward, we explain the adaptive momentum-based optimization, i.e., FedDUM, to improve the accuracy of the global model. Finally, we reveal the design of FedAP, which reduces the size of the global model to reduce communication and computational costs.

3.1 System Model

Figure 1 shows the training process of FedDUMAP, where the FL system consists of a server and N edge devices. The description of major notations is summarized in Table 1. We consider a powerful server and modest devices. The shared insensitive data is stored on the server, and the distributed sensitive data is dispersed in edge devices. Both the server data and the device data can be utilized to update the model during the FL training process of FL. Let us assume that a dataset $D_k = \{x_{k,j}, y_{k,j}\}_{j=1}^{n_k}$, composed of n_k samples, resides on Device k. We take D_0 as the server data, and $D_k, k \neq 0$ as the device data. $x_{k,j}$ refers to the jth sample on Device k, while $y_{k,j}$ represents

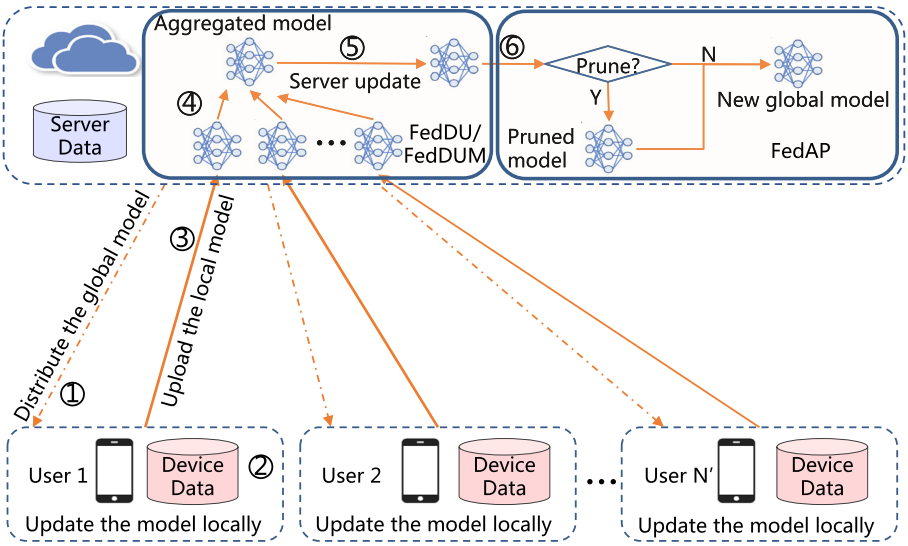


Fig. 1. The training process of FedDUMAP Framework.

Table 1. Summary of Main Notations

Notation	Definition
N; N'	The number of edge devices; the number of devices selected for an iteration; the number of selected devices
D_k; D_0; k	The local dataset on Device k; the server data; the index of a device
n_k; n_0	The number of samples on Device k; the number of samples on the server
$x_{k,j}$; $y_{k,j}$	The jth input data sample in Device k; the label of $x_{k,j}$
$\mathcal{X}$; $\mathcal{Y}$	The data sample set; the label set
w; w^t; $w^{t-\frac{1}{2}}$	The parameters of the global model; the global model in Round t; the aggregated model in Round t
$P_k = \{P_k(y) \mid y \in \mathcal{Y}\}$	The distribution of data in Device k with 0 representing the server
$P_k(y)$	The probability that a data sample corresponds to Label y in Device k
P_m; $\overline{P}$	The average distribution of two datasets; the distribution of all the device data
$\overline{P'}^t$	The distribution of all the data in the selected devices in Round t
$\mathcal{D}(P_k)$	The non-IID degree of data on Device k
$\mathscr{D}$; $\mathscr{D}^t$	The set of all devices and the server; the set of selected devices in Round t
$F_k(w)$	The local loss function in Device k
E; B; η	The local epoch; the local batch size; the learning rate
τ; τ^t_{eff}	The number of iterations performed in the server update; the effective steps for the server update in Round t
$\overline{g}_0^t(\cdot)$	The normalized stochastic gradients on the server in Round t
n'; n_0	The total number of samples in all the selected devices; the number of samples in the server data
$w^{t-\frac{1}{2},i}$	The parameters of the ith local iteration of the updated aggregated model in Round t
acc^t	The accuracy of the aggregated model in Round t
W_k; W'_k	The initial parameters of the local model in Device k; the parameters of the local model in Device k after T rounds
$H(W'_k)$	The Hessian matrix of the loss function on Device k
λ_k^m; d_k	The mth eigenvalue in the Hessian matrix on Device k; the rank of the Hessian matrix on Device k
p^*; p_k^*; p_l^*	The expected global pruning rate; the expected pruning rate on Device k; the expected pruning rate of Layer l
ϵ	A small value to avoid the division by 0
$\mathscr{M}$; q_l	The number of parameters in the model; the number of parameters in Layer l
r_l^j; d_l	The rank of the jth filter in Layer l; the number of filters in Layer l

the corresponding label. We utilize $\mathcal{X}$ to represent the set of samples and $\mathcal{Y}$ to represent the set of labels. Then, we formulate the objective of the training process as follows:

$$\min_w F(w), \text{ with } F(w) \triangleq \frac{1}{n}\sum_{k=1}^{N} n_k F_k(w), \tag{1}$$

where $F_k(w) \triangleq \frac{1}{n_k} \sum_{\{x_{k,j}, y_{k,j}\} \in \mathcal{D}_k} f(w, x_{k,j}, y_{k,j})$ represents the loss function on Device k with $f(w, x_{k,j}, y_{k,j})$ capturing the error of the inference results based on the model with the sample pair $\{x_{k,j}, y_{k,j}\}$ and w refers to the global model.

With the distributed non-IID data in FL [36], we utilize the Jensen–Shannon divergence [67] to quantize the non-IID degree of a dataset as shown in Formula (2)

$$\mathcal{D}(P_k) = \frac{1}{2}\mathcal{D}_{\mathrm{KL}}(P_k||P_m) + \frac{1}{2}\mathcal{D}_{\mathrm{KL}}(\overline{P}||P_m), \tag{2}$$

where $P_m = \frac{1}{2}(P_k + \overline{P})$, $\overline{P} = \frac{\sum_{k=1}^{N} n_k P_k}{\sum_{k=1}^{N} n_k}$, $P_k = \{P_k(y)|y \in \mathcal{Y}\}$, $P_k(y)$ refers to the possibility that a sample is related to Label y, and $\mathcal{D}_{\mathrm{KL}}(\cdot||\cdot)$ corresponds to the **Kullback–Leibler (KL)** divergence [68] as defined in Formula (3)

$$\mathcal{D}_{\mathrm{KL}}(P_i||P_j) = \sum_{y \in \mathcal{Y}} P_i(y) \log\left(\frac{P_i(y)}{P_j(y)}\right). \tag{3}$$

When the distribution of a dataset differs from that of the global dataset more significantly, the corresponding non-IID degree becomes higher. Although the raw data is not transferred from devices to the server or from the server to the devices in FedDUMAP, we assume that P_k and n_k incur little privacy concern [69], and can be transferred from devices to the server before the FL training process. When this statistical meta information, i.e., $P_k(y)$, cannot be shared because of restrictions, we can utilize gradients to calculate the data distribution of the dataset on each device [70].

The FL training process contains multiple rounds in FedDUMAP, each of which consists of 6 steps. In Step ①, a set of devices ($\mathscr{D}^t$ with t referring to the round number) is randomly selected to participate in the training process of Round t and the server broadcasts the global model to the device of $\mathscr{D}^t$. Afterward, each device updates the received model with the adaptive momentum-based optimization based on the local data in Step ②. Then, in Step ③, the selected devices upload updated models to the server, and the server aggregates all the received models based on FedAvg [5] in Step ④. In addition, the server updates (see details of the server update in Section 3.2) aggregated model utilizing the shared insensitive server data and the adaptive momentum-based optimization method (see details in Section 3.3) in Step ⑤. Furthermore, in a predefined round, the server performs model pruning based on the accuracy of the global model and the non-IID degrees (see details in Section 3.4). While Steps ①–④ resemble those within traditional FL, we propose exploiting the insensitive server data to improve the performance of the global model (in FedDU) with adaptive momentum-based optimization (in FedDUM) on both the server and device sides (Step ⑤) and an adaptive model pruning method (in FedAP) to improve the efficiency of the training process (Step ⑥).

3.2 Server Update

In this section, we explain the design of FedDU, which exploit both the shared insensitive server data and the distributed sensitive device data to adjust the global model. We exploit the non-IID degrees and the model accuracy to determine the weights of aggregated model and those of the normalized gradients based on the server data in FedDU, so as to avoid over-fitting to the server data.

We assume that the size of the insensitive server data is significantly bigger than the dataset within a single device. In this case, the straightforward combination of the aggregated model from the devices and the gradients calculated based on the shared insensitive server data leads to inferior accuracy due to objective inconsistency [71]. Inspired by [71], we normalize the gradients generated

using the server data to deal with this issue. The model aggregation in FedDU is formulated as Formula (4)

$$w^t = w^{t-\frac{1}{2}} - \tau_{eff}^{t-1}\eta\overline{g}_0^{(t-1)}(w^{t-\frac{1}{2}}), \tag{4}$$

where w^t refers to the weights of the global model in Round t, $w^{t-\frac{1}{2}}$ is the weights of the aggregated model from the selected devices in Round t as formulated in Formula (5) [5], τ_{eff}^{t-1} corresponds to the size of effective steps for normalized gradients calculated based on the shared insensitive server data as shown in Formula (7), η represents the learning rate, $\overline{g}_0^t(\cdot)$ refers to the normalized gradients generated based on the server data in Round t as formulated in Formula (6)

$$w^{t-\frac{1}{2}} = \sum_{k\in\mathscr{D}^t}\frac{n_k}{n'}(w^{t-1} - \eta g_k^{t-1}(w^{t-1})), \tag{5}$$

where $n' = \sum_{k\in\mathscr{D}^t} n_k$ refers to the size of the dataset on all the selected devices, $g_k^{t-1}(\cdot)$ corresponds to the gradients generated using the dataset on Device k

$$\overline{g}_0^{(t-1)}(w^{t-\frac{1}{2}}) = \frac{\sum_{i=1}^{\tau} g_0^{(t-1)}(w^{t-\frac{1}{2},i})}{\tau}, \tag{6}$$

where $w^{t-\frac{1}{2},i}$ represents the parameters of the global model after aggregating the gradients based on the server data, at ith iteration of Round t, $g_0^{(t-1)}(\cdot)$ refers to the gradients calculated using the server data, and $\tau = \lceil\frac{|n_0|E}{B}\rceil$ corresponds to the number of iterations of Round t, E is the number of local epochs, B is the batch size. Each round corresponds to multiple iterations with a mini-batch of sampled shared insensitive server data.

While τ_{eff}^t has a significant impact on the training process, we dynamically choose an effective step size with the consideration of the model accuracy, the non-IID degrees of both the server data and the device data, with the number of rounds, as formulated in Formula (7)

$$\tau_{eff}^t = f'(acc^t) * \frac{n_0 * \mathcal{D}(\overline{P'}^t)}{n_0 * \mathcal{D}(\overline{P'}^t) + n' * \mathcal{D}(P_0)} * C * decay^t * \tau, \tag{7}$$

where acc^t represents the model accuracy calculated based on the server data in Round t, i.e., $w^{t-\frac{1}{2}}$ defined in Formula (5), n_0 corresponds to the size of the shared insensitive server data, $\mathcal{D}(\cdot)$ is shown in Formula (2), $\overline{P'}^t = \frac{\sum_{k\in\mathscr{D}^t} n_k P_k}{\sum_{k\in\mathscr{D}^t} n_k}$ refers to the distribution of the distributed sensitive data in all the chosen devices in Round t, P_0 is the distribution of the shared insensitive server data, $decay \in (0, 1)$ is utilized to ensure the convergence of the global model toward the solution of Formula (1), and C corresponds to a hyper-parameter. $f'(acc)$ is calculated using acc. acc is small at the beginning of the training. In this stage, the value of $f'(acc)$ should be prominent so as to take advantage of the server data to improve the accuracy of the global model. Then, at a late stage of the training process, $f'(acc)$ should be small to attain the objective defined in Formula (1) while avoiding over-fitting to the shared insensitive server data.

When the distribution of server data resembles that of the overall device data, i.e., $\mathcal{D}(P_0)$ is small, or the distribution of the data of the chosen devices differs much from the overall device data, i.e., $\mathcal{D}(\overline{P'}^t)$ is significant, we take a significant value of τ_{eff}^t to improve the accuracy of the global model. Furthermore, the data size improves the importance of the dataset, as well. In addition, the weight of the device data is $\frac{n'}{\mathcal{D}(\overline{P'}^t)}$ and that of the server data is $\frac{n_0}{\mathcal{D}(P_0)}$. Finally, we have $\frac{\frac{n_0}{\mathcal{D}(P_0)}}{\frac{n_0}{\mathcal{D}(P_0)} + \frac{n'}{\mathcal{D}(\overline{P'}^t)}}$ as the importance of the server data, which is the second part of Formula (7).

Algorithm 1: Federated Dynamic Server Update (FedDU)

Input:

$\mathscr{D}^t$: The set of selected devices in Round t
$\mathcal{D}_k$: The dataset on Device k with 0 representing that on the server
w^{t-1}: The global model in Round $t-1$
E: The number of local epochs
B: The local batch size
$decay$: The decay rate
P: The set of data distribution $\{P_k | k \in \{0\} \cup \mathscr{D}^t\}$ with 0 representing the server
η: The learning rate

Output:

w^t: The global model in Round t

1: **for** k in $\mathscr{D}^t$ (in parallel) **do**
2: Calculate $g_k^{t-1}(w^{t-1})$ using w^{t-1}, $\mathcal{D}_k$
3: **end for**
4: Calculate $w^{t-\frac{1}{2}}$ using w^{t-1}, η according to Formula 5
5: Calculate w^t using $w^{t-\frac{1}{2}}, decay, E, B, P, \eta$ according to Formula 4

Algorithm 1 explains FedDU. The model is updated using the local dataset on each chosen device (Lines 1–3). Afterward, the models are aggregated using Formula (5) (Line 4). Finally, the aggregated model is updated utilizing the shared insensitive server data based on Formula (4) (Line 5).

Let us assume that the expected squared norm of stochastic gradients on the server is bounded, i.e., $\mathbb{E}\,||\overline{g}_0||^2 \leq G^2$. Then, in Round t, $\tau_{eff}^t \leq C \cdot decay^t \cdot \tau$. Afterward, the server update term can be less than $C \cdot decay^t \cdot \tau \cdot \eta \cdot G$. In this case, after sufficient steps, the server update becomes negligible, i.e., $\mathbb{E}[\lim_{t\to\infty} \tau_{eff}^{t-1} \eta \overline{g}_0^{t-1}(w^{t-\frac{1}{2}})] \leq \lim_{t\to\infty} C \cdot decay^{t-1} \cdot \tau \cdot \eta \cdot G = 0$, with $0 < decay < 1$. Finally, FedDU degrades to FedAvg [5], the convergence of which is guaranteed with a decaying learning rate η [72, 73].

3.3 Momentum-Based Optimization

In order to further improve the accuracy of the global model, we exploit momentum within the server update on the server and within the local iteration on each device. In this section, we propose a simple adaptive momentum approach, i.e., FedDUM, which enables the optimization on both the server and the devices without additional communication cost.

The centralized SGDM can be formulated as

$$m^t = \beta * m^{t-1} + (1-\beta) * g(w^{t-1}), w^t = w^{t-1} - \eta * m^t, \tag{8}$$

where m is momentum. A simple application of the momentum into the FL environment is to decompose the SGDM to each device while aggregating the momentum as formulated in Formula (9)

$$\begin{aligned} m_k^t &= \beta * m_k^{t-1} + (1-\beta) * g_k(w_k^{t-1}), \\ w_k^t &= w_k^{t-1} - \eta * m_k^t, \\ m^t &= \sum_{k\in\mathscr{D}^t} \frac{n_k}{n'} m_k^{t-1}, \\ w^t &= \sum_{k\in\mathscr{D}^t} \frac{n_k}{n'} w_k^{t-1}. \end{aligned} \tag{9}$$

In order to extend the optimization into multiple local epochs, we can formulate the process as follows:

$$\begin{aligned} {m'}_k^{t,t'} &= \beta' * {m'}_k^{t,t'-1} + (1-\beta') * {g'}_k({w'}_k^{t,t'-1}), \\ {w'}_k^{t,t'} &= {w'}_k^{t,t'-1} - \eta' * {m'}_k^{t,t'}, \\ m^t &= \sum_{k\in\mathscr{D}^t} \frac{n_k}{n'} {m'}_k^{t,E-1}, \\ w^t &= \sum_{k\in\mathscr{D}^t} \frac{n_k}{n'} {w'}_k^{t,E-1}, \end{aligned} \tag{10}$$

where ${m'}_k^{t,t'}$ represents the local momentum within Device k at global Round t and local Iteration t', β' is the local weight for gradients, ${g'}_k$ refers to the gradients with the parameters of the model ${w'}_k^{t,t'}$, and η' is the local learning rate. Please note that ${g'}_k$ and η' are local hyper-parameters on each device and are different from those in Formulas (8) and (9), which refer to global parameters on the server. We have the first momentum equal to the global momentum, i.e., ${m'}_k^{t,0} = m^t$, and the first parameters of the model equal to the global parameters, i.e., ${w'}_k^{t,0} = w^t$. However, this simple decomposition may deviate from the momentum of the centralized optimizer and harm the local training [32]. In addition, the communication of momentum, i.e., ${m'}_k^{t,E-1}$ and m^t, may incur significant communication cost. Inspired by [74], we restart the local adaptive optimization and set the momentum buffers to zero at the beginning of local iterations. Furthermore, we calculate the difference between the updated model and the original model as the total gradients as defined in Formula (12). Then, we can utilize Formula (11) below to perform the local iteration in each selected device, which replaces Formula (5) in Algorithm 1 with adaptive momentum

$${m'}_k^{t,t'} = \beta' * {m'}_k^{t,t'-1} + (1-\beta') * {g'}_k({w'}_k^{t,t'-1}), {w'}_k^{t,t'} = {w'}_k^{t,t'-1} - \eta' * {m'}_k^{t,t'}, \tag{11}$$

with ${m'}_k^{t,0} = 0$ and ${w'}_k^{t,0} = w^t$. This operation only requires the parameters of the model transferred from the server to devices without additional communication costs compared with FedAvg. Afterward, we exploit Formula (12) below to calculate the gradients on the server, and utilize Formula (8) to replace Formula (4) in Algorithm 1 with adaptive momentum

$$w^{t-\frac{1}{2}} = \sum_{k\in\mathscr{D}^t} \frac{n_k}{n'} {w'}_k^{t-1,E-1}, g(w^{t-1}) = w^{t-\frac{1}{2}} + \tau_{eff}^{t-1}\eta\bar{g}_0^{(t-1)}(w^{t-\frac{1}{2}}) - w^t, \tag{12}$$

where $\tau_{eff}^{t-1}\eta\bar{g}_0^{(t-1)}(w^{t-\frac{1}{2}})$ is the same as that in Formula (4). This only requires the transfer of the parameters of the local updated model, which does not incur additional communication cost compared with FedDU.

FedDUM is shown in Algorithm 2. The local model update is performed according to Formula (11) in parallel in each selected device (Lines 1–3). Then, the aggregated gradient is calculated using Formula (12) (Line 4). Afterward, the new global model is updated using the gradients and momentum according to Formula (8) (Line 5).

In order to analyze the adaptive optimization on devices, we have the following assumptions:

Assumption 1 (Lipschitz Gradient). *There exists a constant L_g such that $\|g_k(w_1) - g_k(w_2)\| \leq L_g\|w_1 - w_2\|$ for any w_1, w_2 and $k = 1, \cdots, N$.*

Assumption 2 (Bounded Gradient). *$\|g_k\|_{L^\infty} < \infty$ for any $k = 1, 2, \cdots, N$.*

Assumption 3 (Bounded Momentum). *$\|m_k\|_{L^\infty} < \infty$ for any $k = 1, 2, \cdots, N$.*

Algorithm 2: Federated Dynamic Update with Momentum (FedDUM)

Input:
$\mathscr{D}^t$: The set of selected devices in Round t
$\mathcal{D}_k$: The dataset on Device k with 0 representing that on the server
w^{t-1}: The global model in Round $t-1$
E: The number of local epochs
B: The local batch size
$decay$: The decay rate
P: The set of data distribution $\{P_k | k \in \{0\} \cup \mathscr{D}^t\}$ with 0 representing the server
η: The learning rate
Output:
w^t: The global model in Round t
1: **for** k in $\mathscr{D}^t$ (in parallel) **do**
2: Calculate ${w'}_k^{t-1,E-1}$ using w^{t-1}, $\mathcal{D}_k$, B, E using Formula 11
3: **end for**
4: Calculate $g(w^{t-1})$ using ${w'}_k^{t-1,E-1}$, $decay$, E, B, P, η according to Formula 12
5: Calculate w^t using $g(w^{t-1})$ according to Formula 8

Then, we can get the following theorem:

Theorem 3.1. *Local momentum deviates from the centralized one at linear rate* $O(e^{\lambda^+ E})$.

Proof. From the perspective of dynamics of Ordinary Differential Equations [32], we can formulate the difference between the local optimization and the centralized optimization as

$$\begin{pmatrix} \|m_k^{t,E-1} - m^{t,E-1}\| \\ \|w_k^{t,E-1} - w^{t,E-1}\| \end{pmatrix} \leq e^{AE} \begin{pmatrix} \|m_k^{t,0} - m^t\| \\ \|w_k^{t,0} - w^t\| \end{pmatrix} + (1-\beta') \int_0^{E-1} e^{A(E-t')} \begin{pmatrix} \|R^{t,t'}\|/\eta' \\ 0 \end{pmatrix} dt'. \tag{13}$$

where $m^{t,t'}$ represents the momentum in centralized optimization, $w^{t,t'}$ represents the model in centralized optimization, $R^{t,t'} = \sum_{j \in \mathscr{D}^t, j \neq k} \frac{n_j}{n'} \big(g_j(w^{t,t'}) - g_k(w^{t,t'})\big)$, and $A = \begin{pmatrix} -(1-\beta')/\eta' & (1-\beta')L_g/\eta' \\ 1 & 0 \end{pmatrix}$. The eigenvalue of Matrix A is $\lambda^{\pm} = \frac{-(1-\beta') \pm \sqrt{(1-\beta')^2 + 4(1-\beta')L_g}}{2\eta'}$. Note that $m_k^{t,0} = 0$ and $w_k^{t,0} = w^t$. In addition, $\|e^{At}\| \leq C_A e^{\lambda^+ t'}$ for any $t' \geq 0$ for some constant C_A. Then, we have

$$\begin{aligned} & \|m_k^{t,t'} - m^{t,t'}\| + \|w_k^{t,t'} - w^{t,t'}\| \\ \leq & C_A \|m^t\| e^{\lambda^+ t'} + e^{\lambda^+ t'} (1-\beta') \int_0^{t'} e^{-\lambda^+ \tau} \left(\|R^{t,t'}\|/\eta'\right) d\tau \\ \leq & C_A \|m^t\| e^{\lambda^+ E} + e^{\lambda^+ E} (1-\beta') \int_0^{E-1} e^{-\lambda^+ \tau} \left(\|R^{t,t'}\|/\eta'\right) d\tau \\ = & O(e^{\lambda^+ E}). \end{aligned} \tag{14}$$

□

In the experimentation, we set the local epoch to a small number, e.g., 5, to reduce the deviation between local momentum and centralized optimization.

Algorithm 3: Federated Adaptive Structured Pruning (FedAP)

Input:
L: The list of convolutional layers to prune
$\mathscr{D}$: The set of all devices and the server
w: The initial model
w^*: The current model in Round t
$W = [v_1, v_2, \cdots, v_{\mathscr{M}}]$: The list of parameters in Model w^*
Output:
w': The pruned model in Round t
1: $w' \leftarrow w^*$
2: **for** $k \in \mathscr{D}$ (in parallel) **do**
3: Calculate the expected pruning rate p_k^*
4: **end for**
5: Calculate p^* according to Formula 15
6: $W = [v_{o_1}, v_{o_2}, \cdots, v_{o_R}] \leftarrow$ Sort W in ascending order of $|v|$
7: $\mathscr{V} = |v_{o_{\lfloor R * p^* \rfloor}}|$
8: **for** $l \in L$ **do**
9: $W_l = [v_1, v_2, \cdots, v_{q_l}] \leftarrow$ The parameters in Layer l
10: $W_l' \leftarrow [v_1, v_2, \cdots, v_{q_l'}]$ with each $|v_q| < \mathscr{V}$
11: $p_l^* \leftarrow \frac{Cardinality(W_l')}{q_l}$
12: Calculate the ranks R_l of each filter in Layer l
13: Sort the filters according to R_l in an ascending order
14: $w_l' \leftarrow$ Preserve the last $d_l - \lfloor p_l^* * d_l \rfloor$ filters in R_l
15: $w' \leftarrow$ Replace the lth layer of w' with w_l'
16: **end for**

3.4 Adaptive Pruning

In this section, we present our layer-adaptive pruning method, i.e., FedAP. FedAP considers the diverse features of each layer, the non-IID degrees of both the sensitive shared server data and the distributed sensitive device data, to remove useless filters in the convolutional layers while preserving the accuracy. FedAP can reduce the communication overhead and the computational costs of the training process. Please note that FedAP is performed only once on the server in a predefined round.

Algorithm 3 details FedAP. On the server and each device (Lines 2–4), we calculate a proper pruning rate based on the shared sensitive server data or the distributed sensitive device data (Line 3). We denote the initial model W_k on the server or Device k. At Round T, the updated model is denoted by W_k'. Then, the difference between the current model and the original one is $\Delta_k = W_k - W_k'$. Afterward, we can calculate the Hessian matrix, i.e., $H(W_k')$. We sort the eigenvalues of $H(W_k')$ in ascending order, i.e., $\{\lambda_k^m | m \in (1, d_k)\}$ with m referring the index of an eigenvalue and d_k referring to the rank of the Hessian matrix. We take $B_k(\Delta_k) = H(W_k') - \nabla L(\Delta_k + W_k')$ as a base function with $\nabla L(\cdot)$ corresponding to the gradients. We denote the Lipschitz constant of $B_k(\Delta_k)$ as $\mathscr{L}_k$. Inspired by [62], we take the first m_k, which meets the condition of $\lambda_{m_{k+1}} - \lambda_{m_k} > 4\mathscr{L}_k$, to reduce accuracy degradation. Then, we can calculate the proper pruning rate by $p_k^* = \frac{m_k}{d_k}$, which is the ratio between the size of pruned eigenvalues and that of all the eigenvalues. As the proper pruning rate significantly differs in diverse devices due to the non-IID distribution, we utilize

Formula (15) to generate an aggregated proper pruning rate for the global model (Line 5)

$$p^* = \sum_{k=0}^{n} \frac{\frac{n_k}{\mathcal{D}(P_k)+\epsilon}}{\sum_{k'=0}^{n} \frac{n_{k'}}{\mathcal{D}(P_{k'})+\epsilon}} * p_k^*, \tag{15}$$

where ϵ represents a small value to avoid the division of zero. We take a global threshold value ($\mathscr{V}$) calculated based on the aggregated proper pruning rate to serve as a baseline value for generating the pruning rate of each layer. $\mathscr{V}$ is the absolute value of the $\lfloor R * p^* \rfloor$-th smallest parameter in all the parameters (Lines 6 and 7). Afterward, for each convolutional layer (Line 8), we calculate the proper pruning rate by calculating the ratio between the number of parameters with inferior absolute values than $\mathscr{V}$ and the number of all the parameters (Lines 9–11). Inspired by [34], we remove the filters corresponding to the smallest ranks in feature maps (the output of filters) based on the proper pruning rate in the layer (Lines 12–15). We take $R_l = \{r_l^j | j \in (1, d_l)\}$ to represent the ranks of feature maps at Layer l, where d_l refers to the number of filters in this layer. As the feature maps are almost the same for a given model [34], we take the ranks calculated based on the server data and perform the pruning operations on the server because of its powerful computation capacity. We calculate the feature maps in R_l in Line 12 and sort them in Line 13. We keep the filters having the last $d_l - \lfloor p_l^* * d_l \rfloor$ ranks in the sorted R^l, so as to attain the highest pruning rate $p_l \leq p_l^*$ (Line 14). In the end, we take the preserved filters in the original model as the layer of the pruned model (Line 15). When there is no available server data, the pruning process can also be performed on the server based on the rank values R_l calculated using the dataset on a device, which are transferred to the server, with the execution of Line 12 being carried out at that device.

4 Experiments

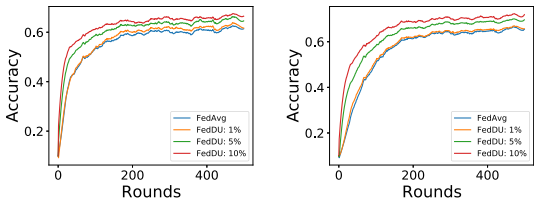
In this section, we demonstrate the experimental results to show the advantages of FedDUMAP by comparing it with state-of-the-art baselines, i.e., FedAvg [5], FedKT [4], FedDF [22], Data-sharing [1], Hybrid-FL [11], server-side momentum [25], device-side momentum [75], FedDA [32], HRank [34], IMC [62], and PruneFL [33].

4.1 Experimental Setup

We evaluate FedDUMAP using an FL system consisting of a parameter server and 100 devices. In each round, we randomly choose 10 devices. We utilize two real-life datasets, i.e., CIFAR-10 and CIFAR-100 [76], in the experimentation. We exploit four models, i.e., a simple synthetic **Convolutional Neural Network (CNN)**, ResNet18 (ResNet) [77], **Visual Geometry Group (VGG)** 11 [78], and LeNet5 (LeNet) [79]. The experimentation of CNN, VGG, and LeNet is carried out on both CIFAR-10 and CIFAR-100, while that of ResNet is performed on CIFAR-100.

CNN consists of three $3 * 3$ convolution layers, a fully connected layer, and a final softmax output layer. The first layer contains 32 channels, while the second and third layers have 64 channels. The first and the second layers are followed by $2 * 2$ max pooling. The fully connected layer contains 64 units and exploits ReLu activation. CNN contains 122,570 parameters in total. The mini-batch size is set to 10 for the local model update, and the selected device executes $E = 5$ local epochs. The local learning rate η is 0.1 with a decay rate of 0.99. C is set to 1. The experimentation is carried out using 33 Tesla V100 GPUs to simulate an FL environment composed of a parameter server and 100 devices. We carry out the pruning process in Round 30 in all the experimentation with adaptive pruning. In addition, we exploit the Savitzky–Golay filter [80] to smooth the accuracy in figures.

The CIFAR-10 dataset contains 60,000 images (50,000 for training and 10,000 for testing), each corresponding to one of ten categories. We take 40,000 images in the training dataset as device data and randomly select p * 40,000 images from the remaining 10,000 images as server data, with



(a) CNN (b) VGG

Fig. 2. The accuracy of FedDU with diverse amounts of server data on CIFAR-10. About 1%, 5%, and 10% represent the value of p (see details in Section 4.1).

$0\% < p < 25\%$. p represents the ratio between the size of the shared insensitive server and that of the distributed sensitive device data. For the non-IID setting, we sort the device data according to the label and then divide these data evenly into $2 * 100$ fractions. Each device is randomly assigned 2 fractions, and most devices contain the data with 2 labels. We utilize the same method to handle the CIFAR-100 dataset.

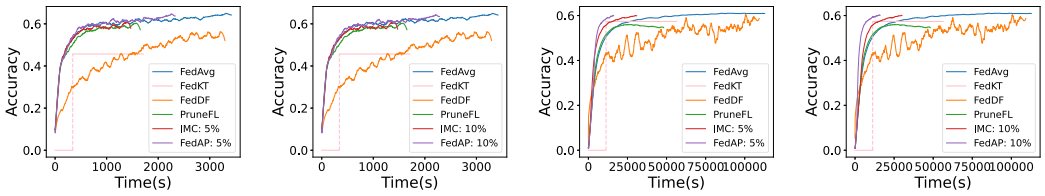
4.2 Evaluation with Non-IID Data

We first conduct comparison of FedDU with FedAvg, FedKT, FedDF, Data-sharing, and Hybrid-FL in terms of model accuracy. Then, we compare FedDUM with server-side momentum, device-side momentum, and FedDA. Afterward, we show the advantages of the adaptive pruning method, i.e., FedAP, with the comparison of FedAvg, IMC, HRank, and PruneFL, in terms of model efficiency. Last but not least, we demonstrate that FedDUMAP, which consists of both FedDUM and FedAP, significantly outperforms the eight state-of-the-art baselines in terms of accuracy, efficiency and computational cost. Finally, we present our ablation study.

4.2.1 Evaluation on FedDU. In this part, we compare FedDU with five baseline methods, i.e., FedAvg, FedKT, FedDF, Data-sharing, and Hybrid-FL. Afterward, we analyze the effect of τ_{eff}, the effect of $f'(acc)$, the effect of C, and the effect of the non-IID degree of server data. Data-sharing transfers the shared server data to devices in order to combine the local data and the server data, which may have a smaller non-IID degree compared with the original local data, so as to improve the accuracy of the updated model on the device. Hybrid-FL takes the shared server data of significant size as an ordinary dataset on a device and utilizes the FedAvg algorithm to perform the training process.

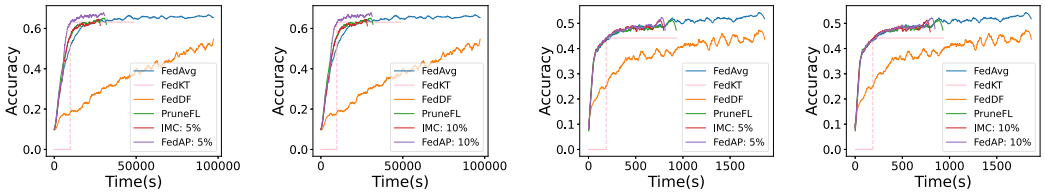
In order to take advantage of the server data, we have significant effective steps at the beginning, which leads to a quick increase of accuracy. Then, at the end of the training, we reduce the effective steps and focus on the device data to achieve high accuracy. Figure 2 shows that FedDU corresponds to excellent efficiency and high accuracy compared with baseline methods with CIFAR-10. In addition, with more server data, FedDU can achieve better performance in terms of accuracy (up to 5.3% higher) for both CNN and VGG. As shown in Figures 3 and 4, FedDU leads to a higher accuracy compared with FedAvg (up to 5.7%), FedKT (up to 22.6%), FedDF (up to 5.0%), Data-sharing (up to 11.7%), and Hybrid-FL (up to 19.5%) for both CNN and ResNet when $p = 5\%$ and 10%. In contrast, Data-sharing has slightly higher accuracy (up to 1.2%) compared with FedDU for VGG, as the more complex model can be better trained with augmented data. Compared with FedDU, Data-sharing needs to transfer the server data to devices, which may incur privacy issues and corresponds to high communication overhead. In addition, Data-sharing leads to a much longer training time to achieve the accuracy of 0.6 (for CNN, up to 15.7 times slower) and 0.4 (for LeNet, up to 28.9 times slower) than FedDU on CNN.

We have similar results on CIFAR-100 as that of CIFAR-10. As shown in Figure 5, FedDU corresponds to a higher accuracy compared with Data-sharing (up to 14.0%), FedKT (up to 16.0%),



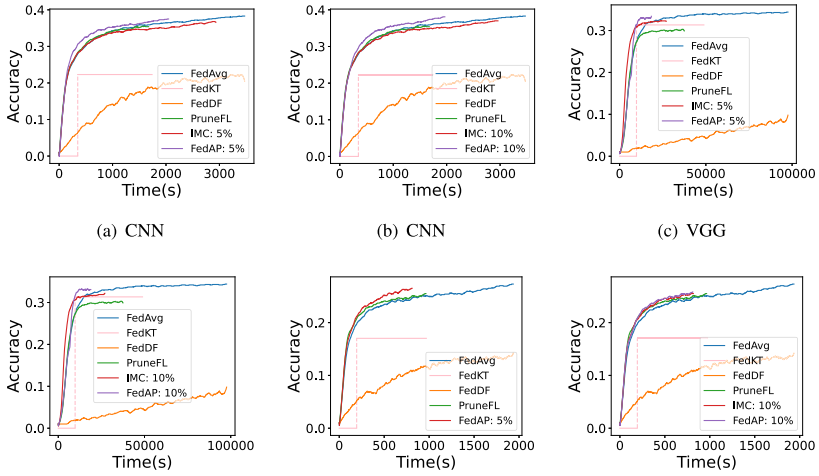
(a) CNN (b) CNN (c) ResNet (d) ResNet

Fig. 3. The accuracy and training time with diverse model update methods corresponding to FedDU with $p = 5\%$ and $p = 10\%$. CNN is with CIFAR-10 and ResNet is with CIFAR-100.



(a) VGG (b) VGG (c) LeNet (d) LeNet

Fig. 4. The accuracy and training time with diverse model update methods corresponding to FedDU based on CIFAR-10.



(a) CNN (b) CNN (c) VGG

(d) VGG (e) LeNet (f) LeNet

Fig. 5. The accuracy and training time with diverse model update methods corresponding to FedDU based on CIFAR-100.

FedDF (up to 17.0%), Hybrid-FL (up to 20.4%), and FedAvg (up to 2.0%) with CNN, VGG, LeNet and ResNet, when $p = 5\%$ and $p = 10\%$. In addition, Data-sharing suffers from a much longer training time to achieve the accuracy of 0.2 (up to 25.2 times slower) than FedDU.

Effect of τ_{eff}. First, we fix the number of effective steps, and we denote this setting **FedDU-Static (FedDU-S)**. We carry out the experiments with diverse values, i.e., $\tau_{eff} = 5$, $\tau_{eff} = 10$, $\tau_{eff} = 20$, and $\tau_{eff} = \frac{n_0 E}{B} > 20$. As shown in Table 2, when the effective number of steps is small, i.e., $\tau_{eff} = 5$ or 10, the corresponding accuracy is higher than FedAvg with a small margin, e.g., up to 1.7% for

Table 2. The Accuracy with Diverse Effective Steps on CIFAR-10

Method	p	τ_{eff} (CNN)				τ_{eff} (VGG)			
		5	10	20	$\frac{n_0 E}{B}$	5	10	20	$\frac{n_0 E}{B}$
$\mathcal{A}$		0.626				0.666			
$\mathcal{D}$	1%	**0.638**				**0.670**			
$\mathcal{S}$		0.628	**0.638**	0.631	0.577	0.668	**0.672**	0.668	0.627
$\mathcal{D}$	5%	**0.663**				**0.700**			
$\mathcal{S}$		0.636	0.639	**0.648**	0.436	0.674	0.684	**0.690**	0.651
$\mathcal{D}$	10%	**0.675**				**0.723**			
$\mathcal{S}$		0.640	0.643	**0.658**	0.429	0.682	0.689	**0.707**	0.664

"$\mathcal{A}$" represents FedAvg. "$\mathcal{D}$" represents FedDU. "$\mathcal{S}$" represents FedDU-S. Bold values represent the best performance.

Table 3. The Accuracy with $f'(acc) = 1 - acc$ and $f'(acc) = \frac{1}{acc+\epsilon}$ on CIFAR-10

Method	p	CNN		VGG	
		Accuracy		Accuracy	
		$1 - acc$	$\frac{1}{acc+\epsilon}$	$1 - acc$	$\frac{1}{acc+\epsilon}$
FedAvg		0.626		0.666	
FedDU	1%	**0.638**	0.637	0.670	**0.677**
	5%	**0.663**	0.660	**0.700**	0.697
	10%	**0.675**	0.668	**0.723**	0.711

Bold values represent the best performance.

CNN and 2.3% for VGG. However, when the number of effective steps is huge, e.g., $\frac{n_0 E}{B} > 200$, the accuracy becomes the lowest. This is expected as the global model is updated too much toward the central data, which degrades the final performance. We find that dynamical adjustment of τ_{eff}, i.e., FedDU, can significantly improve the accuracy (up to 1.7% higher accuracy compared with the static method).

Effect of $f'(acc)$. We can choose a synthetic function of $f'(acc)$ between $1 - acc$ and $\frac{1}{acc+\epsilon}$. As shown in Table 3, we empirically find that the performance corresponding to $1 - acc$ is similar to that corresponding to $\frac{1}{acc+\epsilon}$ in terms of accuracy, while $1 - acc$ is slightly better (0.32% on average). Thus, we choose $1 - acc$ in FedDU.

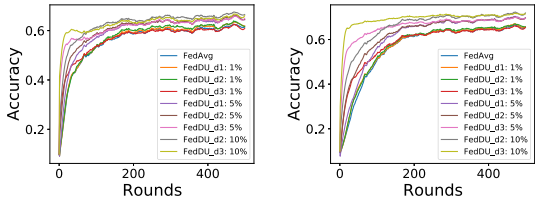
Effect of C. In order to choose an appropriate value of C, we empirically analyze the performance corresponding to diverse values of C, i.e., 1.5, 1, 0.5. As shown in Table 4, we find that the performance of $C = 1$ has the highest accuracy on average. Then, we choose $C = 1$ in FedDU.

Influence of the Non-IID Degree of Server Data. The non-IID degree of server data can have a significant influence on the training process. When the non-IID degree is small, the distribution of the server data is similar to that of the global distribution of all the device data, which is beneficial to the training process. Otherwise, the server data is of less help. In order to analyze the influence of the non-IID degree of the server data, we carry out the experiment with diverse non-IID degrees, e.g., 0.61, 0.31, and 9.0×10^{-6}. As shown in Figure 6, when the non-IID degree is smaller, the training

Table 4. The Accuracy with Diverse Values of C on CIFAR-10

Method	p	C (CNN)			C (VGG)		
		1.5	1	0.5	1.5	1	0.5
FedAvg		0.626			0.666		
FedDU	1%	0.636	**0.638**	0.634	**0.673**	0.670	0.672
	5%	0.651	**0.663**	0.647	0.685	**0.700**	0.690
	10%	0.669	**0.675**	0.668	**0.724**	0.723	0.715

Bold values represent the best performance.



(a) CNN (b) VGG

Fig. 6. The accuracy of FedDU with the server data of diverse non-IID degrees on CIFAR-10. About 1%, 5%, and 10% represent the value of p (see details in Section 4.1). "d" represents the non-IID degree. $d1 = 0.61$, $d2 = 0.31$, and $d3 = 9.0 \times 10^{-6}$.

Table 5. The Accuracy of FedDU with the Server Data of Diverse Non-IID Degrees on CIFAR-10

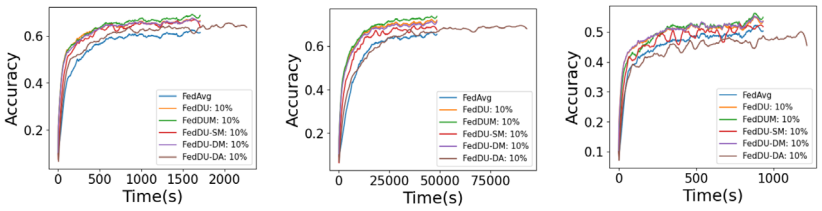
Method	p	d (CNN)			d (VGG)		
		0.61	0.31	9.0×10^{-6}	0.61	0.31	9.0×10^{-6}
FedAvg		0.626			0.666		
FedDU	1%	0.635	**0.638**	0.625	**0.671**	0.670	0.662
	5%	0.659	**0.663**	0.659	**0.704**	0.700	0.700
	10%	0.645	**0.675**	0.667	0.696	**0.723**	0.718

"d" represents the non-IID degree. See details of p in Section 4.1. Bold values represent the best performance.

is more efficient and it takes much shorter time to achieve a target accuracy (up to 6.6 times faster to achieve the accuracy of 0.6). However, as shown in Table 5, the accuracy is not directly related to the non-IID degrees.

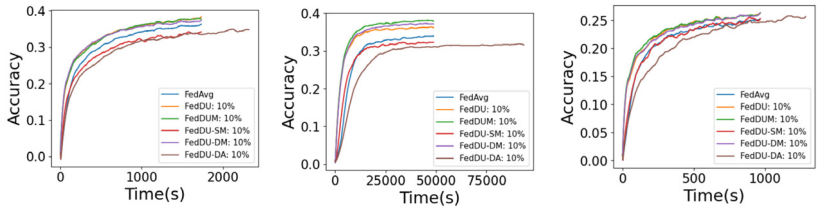
4.2.2 Evaluation on FedDUM. In this section, we compare FedDUM with two adapted adaptive optimization methods, i.e., server-side momentum (FedDU-SM) [25] and device-side momentum (FedDU-DM) [75], as well as FedDA (FedDU-DA) [32]. For a fair comparison, each optimization method is combined with FedDU to utilize the server data in order to improve the accuracy.

As shown in Figure 7, compared with FedAvg and FedDU, FedDUM achieves significantly higher accuracy within the same training time for CNN (up to 6.5% for FedAvg, 1.6% for FedDU), VGG (up to 7.2% for FedAvg, 1.5% for FedDU), and LeNet (up to 4.0% for FedAvg, 1.1% for FedDU) on CIFAR-10. FedDUM leads to a higher accuracy compared with FedDU-SM (up to 4.4%), FedDU-DM



(a) CNN (b) VGG (c) LeNet

Fig. 7. The accuracy of FedDUM with diverse adaptive optimization methods on CIFAR-10.



(a) CNN (b) VGG (c) LeNet

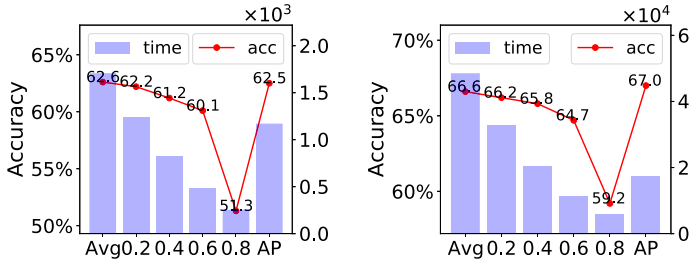
Fig. 8. The accuracy of FedDUM with diverse adaptive optimization methods on CIFAR-100.

(up to 1.6%), and FedDU-DA (up to 6.2%). In addition, FedDU-DA corresponds to a longer total training time (up to 91.8% compared with other methods) because of additional communication cost. Furthermore, FedDUM corresponds to a shorter training time to achieve the target accuracy (up to 1.9 times faster than FedAvg, 44.2% faster than FedDU-SM, 42.9% faster than FedDU-DM and 3.1 times faster than FedDA).

We have similar results on CIFAR-100. As shown in Figure 8, compared with FedAvg and FedDU, FedDUM achieves significantly higher accuracy for CNN (up to 1.6% for FedAvg), VGG (up to 4.2% for FedAvg, 1.8% for FedDU), and LeNet (up to 1.1% for FedAvg, 0.1% for FedDU), except one case, i.e., slightly lower (0.4%) for CNN compared with FedDU. The advantages of FedDUM become significant compared with FedDU-SM (up to 4.4%), FedDU-DM (up to 1.6%), and FedDU-DA (up to 6.2%), in terms of accuracy. In addition, FedDU-DA corresponds to a longer total training time (up to 91.8% compared with other methods). Furthermore, FedDUM corresponds to a shorter training time to achieve the target accuracy (up to 1.4 times faster than FedAvg, 1.6 times faster than FedDU-SM, 51.8% times faster than FedDU-DM, and 4.0 times faster than FedDA).

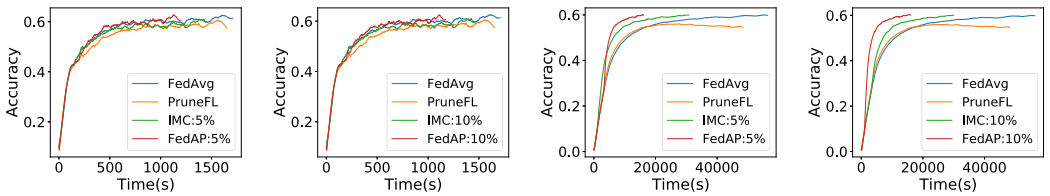
4.2.3 Evaluation on FedAP. In this section, we compare FedAP with three baseline methods, i.e., HRank [34], IMC [62], and PruneFL [33]. We utilize HRank with the shared insensitive server data with multiple pruning rates, e.g., 0.2, 0.4, 0.6, and 0.8, within the FL training process. Similarly, we exploit IMC based on the server data in the FL training process. HRank is a structured pruning method, while PruneFL and IMC are unstructured techniques.

Figure 9 reveals the results based on CNN and VGG with CIFAR10 in terms of training time. FedAP can generate a proper pruning rate (37.8% on average for CNN and 72.0% on average for VGG), with a faster training speed (up to 1.8 times faster) and an excellent accuracy (up to 0.1% accuracy reduction compared with that of FedAvg). However, both the training time and the accuracy decrease at the same time when the pruning rate increases with HRank. FedAP generates adaptive pruning rates for each layer while the accuracy is almost the same as that of FedAvg. Then, we compare FedAP with FedAvg, IMC, and PruneFL. Figure 10 demonstrates that the training speed of FedAP is significantly higher than baselines. In addition, FedAP achieves the highest accuracy



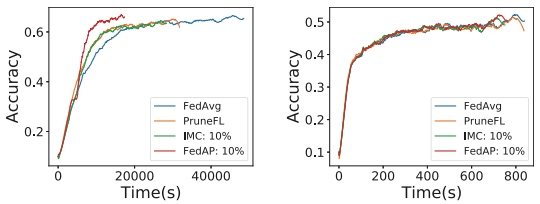
(a) CNN (b) VGG

Fig. 9. The accuracy and the training time with FedAvg, HRank of diverse pruning rates, and FedAP on CIFAR-10.



(a) CNN (b) CNN (c) ResNet (d) ResNet

Fig. 10. The accuracy and the training time with FedAvg, IMC, PruneFL, and FedAP with$p = 5\%$ and $p = 10\%$. CNN is with CIFAR-10 and ResNet is with CIFAR-100.



(a) VGG (b) LeNet

Fig. 11. The accuracy and training time with FedAvg, IMC, PruneFL, and FedAP based on CIFAR-10.

while its training time is significantly shorter to achieve the target accuracy of 0.55. As unstructured pruning techniques cannot reduce computational costs with general-purpose hardware, the training time remains unchanged. On the contrary, FedAP corresponds to a much smaller computational cost (up to 55.7% reduction).

As shown in Figure 11 and Tables 6 and 7, FedAP achieves the highest accuracy and incurs a negligible accuracy reduction (up to 0.1% than FedAvg) while its training time is significantly shorter to achieve the accuracy of 0.6 (up to one time faster than FedAvg, 50.3% faster than IMC, and 49.1% faster than PruneFL).

As shown in Figure 12 and Table 8, with CNN and VGG on CIFAR-100, FedAP achieves the highest accuracy and incurs a negligible accuracy reduction (up to 0.2% and 0.7%) while its training time is significantly shorter to achieve the accuracy of 0.3 (up to 8.2% faster than IMC, 1.4 times faster than PruneFL, and 66.4% faster than FedAvg). As shown in Table 9, similarly with LeNet, FedAP achieves the highest accuracy while its training time is significantly shorter to achieve the accuracy of 0.25 (up to 13.0% faster than IMC, 47.3% faster than PruneFL, and 64.3% faster than FedAvg).

Table 6. The Final Accuracy, Training Time, and Computational Cost of VGG with FedAvg, IMC, PruneFL, and FedAP Based on CIFAR-10

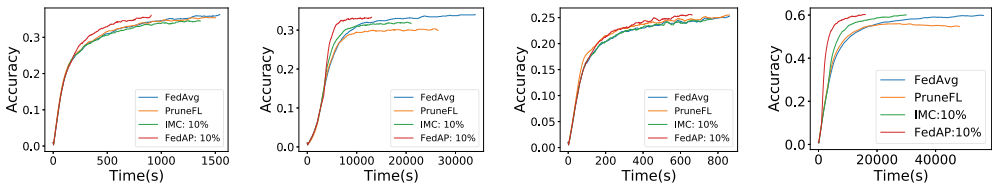
Methods	Accuracy	Time(0.6)	MFLOPs
FedAvg	0.666	15,953	153.4
IMC	0.645	11,992	153.4
PruneFL	0.652	11,896	153.4
FedAP	**0.670**	**7,980**	**56.1**

Bold values represent the best performance. "MFLOPs" represents the computational cost of devices.

Table 7. The Final Accuracy, Training Time, and Computational Cost of LeNet with FedAvg, IMC, PruneFL, and FedAP Based on CIFAR-10

Methods	Accuracy	Time(0.49)	MFLOPs
FedAvg	**0.523**	502	0.7
IMC	0.513	583	0.7
PruneFL	0.515	529	0.7
FedAP	**0.522**	**483**	**0.6**

Bold values represent the best performance.



(a) CNN (b) VGG (c) LeNet (d) ResNet

Fig. 12. The accuracy and training time with FedAvg, IMC, PruneFL, and FedAP based on CIFAR-100.

Table 8. The Final Accuracy, Training Time, and Computational Cost with FedAvg, IMC, PruneFL, and FedAP Based on CIFAR-100

Method	CNN			VGG		
	Accuracy	Time(0.3)	MFLOPs	Accuracy	Time(0.3)	MFLOPs
FedAvg	**0.363**	494	4.6	**0.340**	12,798	153.5
IMC	0.347	480	4.6	0.321	8,317	153.5
PruneFL	0.355	457	4.6	0.303	18,817	153.5
FedAP	**0.361**	**360**	**2.5**	**0.333**	**7,689**	**59.7**

Bold values represent the best performance.

Table 9. The Final Accuracy, Training Time, and Computational Cost of LeNet with FedAvg, IMC, PruneFL, and FedAP Based on CIFAR-100

Methods	Accuracy	Time(0.25)	MFLOPs
FedAvg	0.253	897	0.7
IMC	0.257	617	0.7
PruneFL	0.255	804	0.7
FedAP	**0.259**	**546**	**0.6**

Bold values represent the best performance.

Table 10. The Final Accuracy, Training Time, and Computational Cost with Diverse Approaches on CIFAR-10

Method	CNN			VGG			LeNet		
	Accuracy	Time	MFLOPs	Accuracy	Time	MFLOPs	Accuracy	Time	MFLOPs
FedAvg	0.626	838	4.5	0.666	15,953	153.3	0.523	824	0.7
D-S	0.634	4,447	4.5	**0.735**	6,726	153.3	0.435	NaN	0.7
FedKT	0.458	942	4.5	0.631	11,772	153.3	0.442	523	0.7
FedDF	0.634	1,047	4.5	0.547	12,035	153.3	0.474	513	0.7
Hybrid-FL	0.480	NaN	4.5	0.656	14,785	153.3	0.400	NaN	0.7
ServerM	0.670	383	4.5	0.694	10,797	153.3	0.528	346	0.7
DeviceM	0.676	431	4.5	0.722	10,700	153.3	0.560	280	0.7
FedDA	0.655	582	4.5	0.695	20,333	153.3	0.501	1,171	0.7
IMC	0.608	1,316	4.5	0.645	11,992	153.3	0.513	733	0.7
PruneFL	0.605	1,536	4.5	0.652	11,896	153.3	0.515	837	0.7
FedDUMAP	**0.684**	**248**	**2.7**	0.728	**4,677**	**57.3**	**0.561**	**249**	**0.6**

"Accuracy" represents the accuracy of the final global model. "Time" represents the training time (s) to achieve the accuracy of 0.6 for CNN and VGG, and 0.5 for LeNet. "NaN" represents that the accuracy does not achieve the required accuracy. "D-S" represents Data-sharing. Bold values represent the best performance.

4.2.4 Evaluation on FedDUMAP. In this section, we compare FedDUMAP, consisting of both FedDUM and FedAP, with the baseline approaches. We present the comparison results of CNN, VGG, LeNet with CIRAR-10 and CIFAR-100.

As shown in Table 10, FedDUMAP with CNN achieves higher accuracy compared with FedAvg (5.8%), Data-sharing (5.0%), Hybrid (20.4%), ServerM (1.4%), DeviceM (0.8%), FedDA (2.9%), IMC (7.6%), and PruneFL (7.9%) for CNN. In addition, FedDUMAP has a shorter training time to achieve the target accuracy (up to 2.4 times faster than FedAvg, 16.9 times faster than Data-sharing, 4.3 times faster than IMC, and 5.2 times faster than PruneFL) and a much smaller computational cost (up to 41.3% compared with others).

As shown in Table 10, FedDUMAP with VGG achieves a higher accuracy compared with FedAvg (6.2%), IMC (8.3%), PruneFL (7.6%), Hybrid-FL (7.2%), ServerM (3.4%), DeviceM (0.6%), FedDA (3.3%), IMC (8.3%), and PruneFL (7.6%). Although slightly outperforming FedDUMAP (0.7%), Data-sharing incurs severe data security problems. In addition, FedDUMAP corresponds to a shorter training time to achieve the accuracy of 0.6 (up to 2.4 times faster than FedAvg, 43.8% faster than Data-sharing,

Table 11. The Final Accuracy, Training Time, and Computational Cost with Diverse Approaches Based on CIFAR-100

Method	CNN			VGG			LeNet		
	Accuracy	Time	MFLOPs	Accuracy	Time	MFLOPs	Accuracy	Time	MFLOPs
FedAvg	0.363	494	4.6	0.340	12,798	153.3	0.253	897	0.7
D-S	0.243	NaN	4.6	0.340	11,842	153.3	0.159	NaN	0.7
FedKT	0.223	484	4.5	0.313	9,770	153.3	0.170	718	0.7
FedDF	0.213	476	4.5	0.098	11,247	153.3	0.171	694	0.7
Hybrid-FL	0.179	NaN	4.6	0.308	22,177	153.3	0.134	NaN	0.7
ServerM	0.342	667	4.6	0.323	13,775	153.3	0.256	862	0.7
DeviceM	**0.386**	306	4.6	0.364	8,011	153.3	0.264	558	0.7
FedDA	0.349	777	4.6	0.319	26,422	153.3	0.258	891	0.7
IMC	0.347	480	4.6	0.321	8,317	153.3	0.257	617	0.7
PruneFL	0.355	457	4.6	0.303	18,817	153.3	0.255	804	0.7
FedDUMAP	0.383	**202**	**2.8**	**0.396**	**3,849**	**61.8**	**0.270**	**393**	**0.6**

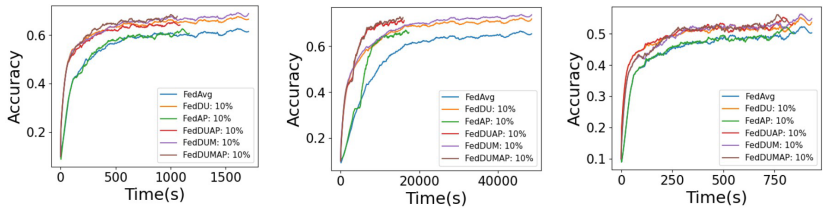
"Accuracy" represents the a ccuracy of the final global model. "Time" represents the training time (s) to achieve the accuracy of 0.3 for CNN and VGG, and 0.25 for LeNet. "NaN" represents that the accuracy does not achieve the required accuracy. "D-S" represents Data-sharing. Bold values represent the best performance.

2.2 times faster than Hybrid-FL, 1.6 times faster than IMC and 1.5 times faster than PruneFL) and a much smaller computational cost (up to 62.6% reduction for FedAvg, Data-sharing, Hybrid-FL, IMC, and PruneFL).

We obtain similar findings with LeNet. As shown in Table 10, FedDUMAP achieves a higher accuracy compared with FedAvg (3.8%), Data-sharing (12.6%), Hybrid-FL (16.1%), ServerM (3.3%), DeviceM (0.1%), FedDA (6.0%), IMC (4.8%) and PruneFL (4.6%). In addition, FedDUMAP corresponds to a shorter training time to achieve the accuracy of 0.5 (up to 2.3 times faster than FedAvg, 1.9 times than IMC, 2.4 times than PruneFL) and a much smaller computational cost (up to 14.3% reduction for FedAvg, Data-sharing, Hybrid-FL, IMC, and PruneFL).

As shown in Tables 11, with CNN, VGG, and LeNet on CIFAR-100, FedDUMAP achieves a higher accuracy compared with FedAvg (up to 5.6%), Data-sharing (up to 14.0%), Hybrid-FL (up to 20.4%), ServerM (up to 7.3%), DeviceM (up to 3.2%), FedDA (up to 7.7%), IMC (up to 7.5%) and PruneFL (up to 9.3%), except for a single case, i.e., slightly lower than DeviceM (0.3%) for CNN. In addition, FedDUMAP corresponds to a shorter training time to achieve the accuracy of 0.3 (up to 2.3 times faster than FedAvg, 2.1 times faster than Data-sharing, 4.8 times faster than Hybrid-FL, 1.4 times faster than IMC and 3.9 times faster than PruneFL) and a much smaller computational cost (up to 59.7% reduction compared with other methods).

4.2.5 Ablation Study. We conduct the ablation study by measuring the final accuracy, the training time to achieve the accuracy of 0.6 for CNN and VGG, 0.3 for LeNet with CIFAR-10, and the computational cost of FedAvg, FedDU, FedDUM, FedAP, FedDUAP, and FedDUMAP. As shown in Figure 13 and Table 12, the efficiency of FedDUMAP significantly outperforms FedDU, FedDUM, and FedAvg (up to 2.4 times faster), while the accuracy of FedDUMAP is much higher than that of FedDU, FedAP, and FedAvg (up to 6.2%). In addition, the computational cost of FedDUMAP is the smallest for CNN, while it is slightly higher (2.1%) than FedAP for VGG due to the dynamic server



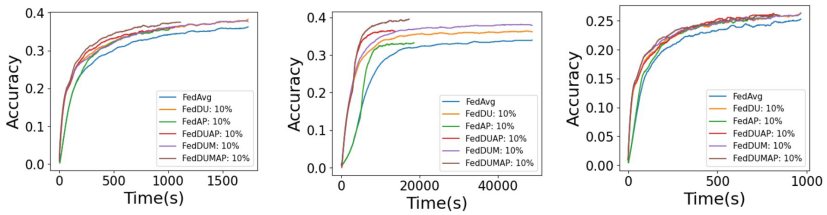
(a) CNN (b) VGG (c) LeNet

Fig. 13. The accuracy of FL with FedAvg, FedDU, FedAP, and FedDUAP based on CIFAR-10.

Table 12. The Final Accuracy, Training Time, and Computational Cost with FedAvg, FedDU, FedAP, and FedDUAP with CIFAR-10

Method	CNN			VGG			LeNet		
	Accuracy	Time	MFLOPs	Accuracy	Time	MFLOPs	Accuracy	Time	MFLOPs
FedAvg	0.626	838	4.5	0.666	15,953	153.3	0.523	824	0.7
FedDU	0.675	284	4.5	0.723	8,268	153.3	0.552	284	0.7
FedDUM	**0.691**	311	4.5	**0.738**	7,490	153.3	**0.563**	284	0.7
FedAP	0.625	696	3.0	0.670	7,980	**56.0**	0.522	754	**0.6**
FedDUAP	0.662	278	2.8	0.714	4,719	58.4	0.553	267	**0.6**
FedDUMAP	0.684	**248**	**2.7**	0.728	**4,677**	57.3	0.561	**249**	**0.6**

"Accuracy" represents the accuracy of the final global model. "Time" represents the training time (s) to achieve the accuracy of 0.6 for CNN and VGG, and 0.5 for LeNet. Bold values represent the best performance.



(a) CNN (b) VGG (c) LeNet

Fig. 14. The accuracy of FL with FedAvg, FedDU, FedAP, and FedDUAP based on CIFAR-100.

update. Although FedDUMAP achieves slightly lower accuracy compared with FedDUM (up to 1.0%), FedDUMAP is much more efficient than FedDU and FedDUM (up to 76.8% faster).

In order to reveal the advantages of FedDUMAP, we also conduct the ablation study with CIFAR-100. We measure the final accuracy, the training time to achieve the accuracy, and the computational cost of diverse methods.

As shown in Figure 14 and Table 13, the efficiency of FedDUMAP significantly outperforms FedDU (up to 2.3 times faster), FedDUM (up to 43.1% faster), and FedAvg (up to 2.3 times faster), while the accuracy of FedDUMAP is much higher than that of FedDU (up to 0.2%), FedAP (up to 6.3%), and FedAvg (up to 5.6%). FedDUMAP corresponds to the shortest training time with CNN, VGG, and LeNet. In addition, the computational cost of FedDUMAP is the smallest for LeNet while it is slightly higher than FedAP for CNN (16.0%) and VGG (3.5%) due to the dynamic server update. Furthermore, FedDUMAP has slightly higher accuracy (up to 1.4% for VGG) compared with that of

Table 13. The Final Accuracy, Training Time, and Computational Cost with FedAvg, FedDU, FedAP, and FedDUAP Based on CIFAR-100

Method	CNN			VGG			LeNet		
	Accuracy	Time	MFLOPs	Accuracy	Time	MFLOPs	Accuracy	Time	MFLOPs
FedAvg	0.363	494	4.6	0.340	12,798	153.3	0.253	897	0.7
FedDU	0.373	667	4.6	0.364	6,155	153.3	0.263	574	0.7
FedDUM	0.379	355	4.6	0.382	5,276	153.3	0.264	572	0.7
FedAP	0.361	360	2.4	0.333	7,689	**59.6**	0.259	546	**0.6**
FedDUAP	0.363	269	2.6	0.366	3,913	62.6	0.263	485	**0.6**
FedDUMAP	**0.383**	**202**	**2.8**	**0.396**	**3,849**	61.8	**0.270**	**393**	**0.6**

"Accuracy" represents the accuracy of the final global model. "Time" represents the training time (s) to achieve the accuracy of 0.3 for CNN and VGG, and 0.25 for LeNet. Bold values represent the best performance.

FedDUM, and the accuracy of FedDUMAP is much higher (up to 5.6% for VGG) than that of FedAvg. In addition, FedDUMAP is much more efficient than FedDUM (up to 43.1% faster for CNN).

5 Conclusion

In this article, we propose a novel FL framework FedDUMAP, which leverages both the shared insensitive server data and the distributed sensitive device data to train the global model. Furthermore, the non-IID degrees of the data are also considered in the FL training process. FedDUMAP consists of a dynamic update FL algorithm, i.e., FedDU, a simple yet efficient adaptive optimization method on top of FedDU, i.e., FedDUM, and an adaptive pruning method, i.e., FedAP. We conduct extensive experimentation to evaluate the performance of FedDUMAP with different models and real-life datasets. According to the experimental results, FedDUMAP significantly outperforms the baseline approaches in terms of accuracy (up to 20.4% higher), efficiency (up to 16.9 times faster), and computational cost (up to 62.6% smaller).

References

[1] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. arXiv:1806.00582. Retrieved from https://doi.org/10.48550/arXiv.1806.00582

[2] Official Journal of the European Union. 2016. General Data Protection Regulation. Retrieved February 12, 2021 from https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679

[3] 2018. California Consumer Privacy Act Home Page. Retrieved February 14, 2021 from https://www.caprivacy.org/

[4] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Practical one-shot federated learning for cross-silo setting. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI '21)*, 1484–1490.

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics (AISTATS '17)*, 1273–1282.

[6] Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. 2022. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems* 64, 4 (2022), 885–917.

[7] Ji Liu, Chunlu Chen, Yu Li, Lin Sun, Yulun Song, Jingbo Zhou, Bo Jing, and Dejing Dou. 2024. Enhancing trust and privacy in distributed networks: A comprehensive survey on blockchain-based federated learning. *Knowledge and Information Systems* 66 (2024), 1–27.

[8] Ji Liu, Zhihua Wu, Danlei Feng, Minxu Zhang, Xinxuan Wu, Xuefeng Yao, Dianhai Yu, Yanjun Ma, Feng Zhao, and Dejing Dou. 2023. Heterps: Distributed deep learning with reinforcement learning based scheduling in heterogeneous environments. *Future Generation Computer Systems* 148 (2023), 106–117.

[9] He Li, Kaoru Ota, and Mianxiong Dong. 2018. Learning IoT in edge: Deep learning for the internet of things with edge computing. *IEEE Network* 32, 1 (2018), 96–101.

[10] Jian Shen, Jun Shen, Xiaofeng Chen, Xinyi Huang, and Willy Susilo. 2017. An efficient public auditing protocol with novel dynamic structure for cloud data. *IEEE Transactions on Information Forensics and Security* 12, 10 (2017), 2402–2415.
[11] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani. 2020. Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data. In *Proceedings of the IEEE International Conference on Communications (ICC '20)*, 1–7.
[12] Amazon. 2021. Amazon Cloud. Retrieved December 30, 2021 from https://aws.amazon.com
[13] Microsoft Azure. 2021. Microsoft Azure. Retrieved December 30, 2021 from https://azure.microsoft.com/
[14] Baidu. 2021. Baidu AI Cloud. Retrieved December 30, 2021 from https://cloud.baidu.com/
[15] Kun Wang, Xin Qi, Lei Shu, Der-jiunn Deng, and Joel J. P. C. Rodrigues. 2016. Toward trustworthy crowdsourcing in the social internet of things. *IEEE Wireless Communications* 23, 5 (2016), 30–36.
[16] Xi Ye, Yushu Zhang, Xiangli Xiao, Shuang Yi, and Rushi Lan. 2023. Usability enhanced thumbnail-preserving encryption based on data hiding for JPEG images. *IEEE Signal Processing Letters* 30 (2023), 793–797.
[17] Kun Yang, Shengbo Chen, and Cong Shen. 2022. On the convergence of hybrid server-clients collaborative training. *IEEE Journal on Selected Areas in Communications* 41, 3 (2022), 802–819.
[18] Hang Gu, Bin Guo, Jiangtao Wang, Wen Sun, Jiaqi Liu, Sicong Liu, and Zhiwen Yu. 2022. FedAux: An efficient framework for hybrid federated learning. In *Proceedings of the IEEE International Conference on Communications (ICC '22)*. IEEE, 195–200.
[19] Zhuotao Lian, Qingkui Zeng, Weizheng Wang, Thippa Reddy Gadekallu, and Chunhua Su. 2022. Blockchain-based two-stage federated learning with non-IID data in IoMT system. *IEEE Transactions on Computational Social Systems* 10 (2022), 1701-1710.
[20] Tehrim Yoon, Sumin Shin, Sung Ju Hwang, and Eunho Yang. 2021. FedMix: Approximation of Mixup under Mean Augmented Federated Learning. In *Int. Conf. on Learning Representations (ICLR)*.
[21] Jie Zhang, Song Guo, Xiaosong Ma, Haozhao Wang, Wenchao Xu, and Feijie Wu. 2021. Parameterized knowledge transfer for personalized federated learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS '21)*, Vol. 34, 1–13.
[22] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. 2020. Ensemble distillation for robust model fusion in federated learning. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS '20)*, 1–13.
[23] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *Proceedings of the International Conference on Machine Learning (ICML '13)*, 1139–1147.
[24] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, 7 (2011), 2121–2159.
[25] S. Reddi, Manzil Zaheer, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. 2018. Adaptive methods for nonconvex optimization. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS '18)*, 1–17.
[26] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://doi.org/10.48550/arXiv.1412.6980
[27] Jed Mills, Jia Hu, and Geyong Min. 2019. Communication-efficient federated learning for wireless edge intelligence in IoT. *IEEE Internet of Things Journal* 7, 7 (2019), 5986–5994.
[28] Honglin Yuan, Manzil Zaheer, and Sashank Reddi. 2021. Federated composite optimization. In *Proceedings of the International Conference on Machine Learning (ICML '21)*, Vol. 139, 12253–12266.
[29] Wei Liu, Li Chen, Yunfei Chen, and Wenyi Zhang. 2020. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 31, 8 (2020), 1754–1766.
[30] Hongchang Gao, An Xu, and Heng Huang. 2021. On the convergence of communication-efficient local SGD for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35, 7510–7518.
[31] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. 2020. SlowMo: Improving communication-efficient distributed SGD with slow momentum. In *Proceedings of the International Conference on Learning Representations (ICLR '20)*, 1–27.
[32] Jiayin Jin, Jiaxiang Ren, Yang Zhou, Lingjuan Lyu, Ji Liu, and Dejing Dou. 2022. Accelerated federated learning with decoupled adaptive optimization. In *Proceedings of the International Conference on Machine Learning*. PMLR, 10298–10322.
[33] Yuang Jiang, Shiqiang Wang, Víctor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. 2023. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems* 34, 12 (2023), 10374–10386. DOI: https://doi.org/10.1109/TNNLS.2022.3166101
[34] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. 2020. HRank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '20)*, 1529–1538.

[35] Hong Zhang, Ji Liu, Juncheng Jia, Yang Zhou, Huaiyu Dai, and Dejing Dou. 2022. FedDUAP: Federated learning with dynamic update and adaptive pruning using shared data on the server. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '22)*, 1–7. In press.

[36] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, and Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G.L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning* 14, 1 (2021), 1–121.

[37] Chendi Zhou, Ji Liu, Juncheng Jia, Jingbo Zhou, Yang Zhou, Huaiyu Dai, and Dejing Dou. 2022. Efficient device scheduling with multi-job federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, 9971–9979.

[38] Guanghao Li, Yue Hu, Miao Zhang, Ji Liu, Quanjun Yin, Yong Peng, and Dejing Dou. 2022. FedHiSyn: A hierarchical synchronous federated learning framework for resource and data heterogeneity. In *Proceedings of the International Conference on Parallel Processing (ICPP '22)*, 1–10. In press.

[39] Ji Liu, Tianshi Che, Yang Zhou, Ruoming Jin, Huaiyu Dai, Dejing Dou, and Patrick Valduriez. 2024. AEDFL: efficient asynchronous decentralized federated learning with heterogeneous devices. In *Proceedings of the SIAM International Conference on Data Mining (SDM '24)*. SIAM, 833–841.

[40] Ji Liu, Juncheng Jia, Tianshi Che, Chao Huo, Jiaxiang Ren, Yang Zhou, Huaiyu Dai, and Dejing Dou. 2024. FedASMU: Efficient asynchronous federated learning with dynamic staleness-aware model update. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, 13900–13908.

[41] Tianshi Che, Yang Zhou, Zijie Zhang, Lingjuan Lyu, Ji Liu, Da Yan, Dejing Dou, and Jun Huan. 2023. Fast federated machine unlearning with nonlinear functional theory. In *Proceedings of the International Conference on Machine Learning*. PMLR, 4241–4268.

[42] Ji Liu, Xuehai Zhou, Lei Mo, Shilei Ji, Yuan Liao, Zheng Li, Qin Gu, and Dejing Dou. 2023. Distributed and deep vertical federated learning with big data. *Concurrency and Computation: Practice and Experience* 35, 21 (2023), e7697.

[43] Ji Liu, Juncheng Jia, Beichen Ma, Chendi Zhou, Jingbo Zhou, Yang Zhou, Huaiyu Dai, and Dejing Dou. 2022. Multi-job intelligent scheduling with cross-device federated learning. *IEEE Transactions on Parallel and Distributed Systems* 34, 2 (2022), 535–551.

[44] Tianshi Che, Zijie Zhang, Yang Zhou, Xin Zhao, Ji Liu, Zhe Jiang, Da Yan, Ruoming Jin, and Dejing Dou. 2022. Federated fingerprint learning with heterogeneous architectures. In *Proceedings of the IEEE International Conference on Data Mining (ICDM '22)*. IEEE, 31–40.

[45] Ji Liu, Daxiang Dong, Xi Wang, An Qin, Xingjian Li, Patrick Valduriez, Dejing Dou, and Dianhai Yu. 2023. Large-scale knowledge distillation with elastic heterogeneous computing resources. *Concurrency and Computation: Practice and Experience* 35, 26 (2023), e7272.

[46] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. 2018. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. arXiv:1811.11479. Retrieved from https://doi.org/10.48550/arXiv.1811.11479

[47] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. Group knowledge transfer: Federated learning of large CNNs at the edge. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS '20)*, Vol. 33, 1–17.

[48] Daxiang Dong, Ji Liu, Xi Wang, Weibao Gong, An Qin, Xingjian Li, Dianhai Yu, Patrick Valduriez, and Dejing Dou. 2022. Elastic deep learning using knowledge distillation with heterogeneous computing resources. In *Proceedings of the European Conference on Parallel Processing workshop (European Conference on Parallel Processing workshop)*, 116–128.

[49] Ji Liu, Daxiang Dong, Xi Wang, Weibao Gong, An Qin, Xingjian Li, Patrick Valduriez, Dejing Dou, and Dianhai Yu. 2022. Large-scale knowledge distillation with elastic heterogeneous computing resources. *Concurrency and Computation: Practice and Experience* 35 (2022), 1–16. In press.

[50] Wonyong Jeong, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. 2020. Federated semi-supervised learning with inter-client consistency & disjoint learning. In *Proceedings of the International Conference on Learning Representations (ICLR '20)*, 1–15.

[51] Lokesh Nagalapatti and Ramasuri Narayanam. 2021. Game of gradients: Mitigating irrelevant clients in federated learning. *AAAI Conference on Artificial Intelligence* 35, 10 (2021), 9046–9054.

[52] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of the International Conference on Machine Learning (ICML '20)*, 5132–5143.

[53] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. 2021. Federated learning based on dynamic regularization. In *Proceedings of the International Conference on Learning Representations (ICLR '21)*, 1–36.

[54] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. 2020. FetchSGD: Communication-efficient federated learning with sketching. In *Proceedings of the International Conference on Machine Learning (ICML '20)*, 8253–8265.

[55] Jed Mills, Jia Hu, Geyong Min, Rui Jin, Siwei Zheng, and Jin Wang. 2021. Accelerating federated learning with a global biased optimiser. arXiv:2108.09134. Retrieved from https://doi.org/10.48550/arXiv.2108.09134

[56] Tianshi Che, Ji Liu, Yang Zhou, Jiaxiang Ren, Jiwen Zhou, Victor S Sheng, Huaiyu Dai, and Dejing Dou. 2023. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. arXiv:2310.15080. Retrieved from https://doi.org/10.48550/arXiv.2310.15080

[57] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. 2021. Adaptive federated optimization. In *Proceedings of the International Conference on Learning Representations (ICLR '21)*, 1–38.

[58] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. 2021. Hermes: An efficient federated learning framework for heterogeneous mobile clients. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 420–437.

[59] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. 2021b. Fedmask: Joint computation and communication-efficient personalized federated learning via heterogeneous masking. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 42–55.

[60] Yongheng Deng, Weining Chen, Ju Ren, Feng Lyu, Yang Liu, Yunxin Liu, and Yaoxue Zhang. 2022. TailorFL: Dual-personalized federated learning under system and data heterogeneity. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, 592–606.

[61] Sixing Yu, Phuong Nguyen, Ali Anwar, and Ali Jannesari. 2021. Adaptive dynamic pruning for non-IID federated learning. arXiv:2106.06921. Retrieved from https://www.cs.iastate.edu/swapp/files/inline-files/yu_ea_flpruning_arxiv-june-2021.pdf

[62] Zeru Zhang, Jiayin Jin, Zijie Zhang, Yang Zhou, Xin Zhao, Jiaxiang Ren, Ji Liu, Lingfei Wu, Ruoming Jin, and Dejing Dou. 2021. Validating the lottery ticket hypothesis with inertial manifold theory. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS '21),* Vol. 34, 1–15.

[63] Jakub Konečnỳ, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. arXiv:1610.05492. Retrieved from https://doi.org/10.48550/arXiv.1610.05492

[64] Juncheng Jia, Ji Liu, Chendi Zhou, Hao Tian, Mianxiong Dong, and Dejing Dou. 2024. Efficient asynchronous federated learning with sparsification and quantization. *Concurrency and Computation: Practice and Experience* 36, 9 (2024), e8002.

[65] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. 2021. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '21)*, 1–6.

[66] Yuanyuan Chen, Zichen Chen, Pengcheng Wu, and Han Yu. 2022. FedOBD: Opportunistic block dropout for efficiently training large-scale neural networks through federated learning. arXiv:2208.05174. Retrieved from https://doi.org/10.48550/arXiv.2208.05174

[67] Bent Fuglede and Flemming Topsoe. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *Proceedings of the International Symposium on Information Theory (ISIT '04)*, 31.

[68] Solomon Kullback. 1997. *Information Theory and Statistics*. Courier Corporation.

[69] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient federated learning via guided participant selection. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI '21)*, 19–35.

[70] Miao Yang, Ximin Wang, Hua Qian, Yongxin Zhu, Hongbin Zhu, Mohsen Guizani, and Victor Chang. 2022. An improved federated learning algorithm for privacy-preserving in cybertwin-driven 6G system. *IEEE Transactions on Industrial Informatics* 18 (2022), 6733–6742.

[71] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H. Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS '20)*, 1–13.

[72] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2020. On the convergence of fedavg on non-IID data. In *Proceedings of the International Conference on Learning Representations (ICLR '20)*, 1–26.
[73] Fan Zhou and Guojing Cong. 2018. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '18)*, 3219–3227.
[74] Jianyu Wang, Zheng Xu, Zachary Garrett, Zachary Charles, Luyang Liu, and Gauri Joshi. 2021. Local adaptivity in federated learning: Convergence and consistency. In *Proceedings of the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML (FL-ICML '21)*, 1–22.
[75] Sai Praneeth Karimireddy, Martin Jaggi, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. 2020. Mime: Mimicking centralized stochastic algorithms in federated learning. arXiv:2008.03606. Retrieved from https://doi.org/10.48550/arXiv.2008.03606
[76] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning Multiple Layers of Features From Tiny Images*. Technical Report, *University of Toronto* (2009).
[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR '16)*, 770–778.
[78] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR '15)*, 1–14.
[79] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.
[80] William H. Press and Saul A. Teukolsky. 1990. Savitzky-Golay smoothing filters. *Computers in Physics* 4, 6 (1990), 669–672.

Received 29 December 2022; revised 2 January 2024; accepted 31 July 2024