FedDUAP: Federated Learning with Dynamic Update and Adaptive Pruning Using Shared Data on the Server

Hong Zhang[1†], **Ji Liu**[2†*], **Juncheng Jia**[1*], **Yang Zhou**[3], **Huaiyu Dai**[4], **Dejing Dou**[2]
[1]School of Computer Science and Technology, Soochow University, [2]Baidu Inc., China,
[3]Auburn University, [4]North Carolina State University, United States

Abstract

Despite achieving remarkable performance, Federated Learning (FL) suffers from two critical challenges, i.e., limited computational resources and low training efficiency. In this paper, we propose a novel FL framework, i.e., FedDUAP, with two original contributions, to exploit the insensitive data on the server and the decentralized data in edge devices to further improve the training efficiency. First, a dynamic server update algorithm is designed to exploit the insensitive data on the server, in order to dynamically determine the optimal steps of the server update for improving the convergence and accuracy of the global model. Second, a layer-adaptive model pruning method is developed to perform unique pruning operations adapted to the different dimensions and importance of multiple layers, to achieve a good balance between efficiency and effectiveness. By integrating the two original techniques together, our proposed FL model, FedDUAP, significantly outperforms baseline approaches in terms of accuracy (up to 4.8% higher), efficiency (up to 2.8 times faster), and computational cost (up to 61.9% smaller).

1 Introduction

While a large amount of data are generally dispersed over numerous edge devices, some data may be available at the server side to facilitate more efficient processing [Zhao *et al.*, 2018]. Due to the concerns of data privacy and security, multiple legal restrictions [Official Journal of the European Union, 2016; CCP, 2018] have been put into practice, which makes it complicated to aggregate the distributed sensitive data into a single server or data center. With the size of training data having a significant influence on the quality of machine learning models [Li *et al.*, 2021], **Federated Learning** (**FL**) [McMahan *et al.*, 2017; Liu *et al.*, 2022] becomes a promising approach to collaboratively train a model without transferring raw data.

Conventional FL was proposed to train a global model using non-Independent and Identically Distributed (non-IID) data on mobile devices [McMahan *et al.*, 2017]. During the training process of FL, the raw training data is kept within each device while the gradients or the weights of a model are communicated. FL typically utilizes a parameter server architecture [Liu *et al.*, 2022], where a parameter server (server) coordinates the distributed training process. The training process generally consists of multiple rounds, each of which is composed of three steps. First, the server randomly selects several devices and sends the global model to the selected devices. Second, each selected device updates the global model with local data and uploads the updated model to the server. Third, the server aggregates all uploaded local models to generate a new global model. These steps are repeated until the global model converges or predefined conditions are attained.

Although FL helps preserve the privacy and the security of distributed device data, two unsolved challenges limit the applicability of the FL paradigm in real-world scenarios. On the one hand, although a large amount of data are collected from numerous devices, the data over a single device may be limited, which results in low effectiveness of local training. On the other hand, local devices often have limited computing capacity and communication capacity [Li *et al.*, 2018], which leads to low efficiency in the local training process.

While sensitive data are not allowed to be transferred among the devices and the server, some insensitive data may reside in the servers or the cloud [Yoshida *et al.*, 2020] , e.g., Amazon Cloud [Amazon, 2021], which can be exploited to address the low-effectiveness challenges. Existing works that introduce the insensitive server data to improve the accuracy of the global model generally encounter high communication costs due to data transfer between the devices and the server [Zhao *et al.*, 2018]. In addition, they may suffer from low efficiency due to simple processing of the server data as that in an ordinary device [Yoshida *et al.*, 2020] or knowledge transfer for heterogeneous models [Lin *et al.*, 2020b].

Pruning techniques are adopted within the training process of FL to reduce the communication costs between the devices and the server, as well as to accelerate the training process [Jiang *et al.*, 2019]. However, most model pruning approaches in the FL context are lossy pruning methods with

[†]H. Zhang and J. Liu contributed equally to the paper. This work was done when H. Zhang was an intern at Baidu Inc.
[*]Corresponding authors (liuji04@baidu.com and jiajuncheng@suda.edu.cn).

a single pruning strategy for all layers of neural networks, without considering their individual dimensions and importance [Lin *et al.*, 2020a]. The aforementioned straightforward model pruning strategies may fail to generate a good approximation to prune appropriate parts and thus lead to non-trivial failure probability of FL training [Zhao *et al.*, 2018].

In this paper, we propose **FedDUAP**, an efficient federated learning framework that collaboratively trains a global model using the device data and the server data. To address the two challenges in FL, we take advantage of the server data and the computing power of the server to improve the performance of the global model while considering the non-IID degrees of the server data and the device data. The non-IID degree represents the difference between the distribution of a dataset and that of all the devices.

Under this framework, we propose an FL algorithm, i.e., FedDU, to dynamically update the model using the device data and the server data. Within FedDU, we propose a method to dynamically adjust the centralized training within the server according to the accuracy of the global model and the non-IID degrees. Furthermore, we design an adaptive model pruning method, i.e., FedAP, to reduce the size of the global model, so as to further improve the efficiency and reduce the computational and communication costs of the training process while ensuring comparable accuracy. FedAP performs unique pruning operations for each layer of specific dimensions and importance to achieve a good balance between efficiency and effectiveness, based on the non-IID degrees and the server data. To the best of our knowledge, we are among the first to incorporate non-IID degrees in the dynamic server update and the adaptive pruning for FL. We summarize our contributions as follows:

1. We propose a dynamic FL algorithm, i.e., FedDU, to take advantage of both server data and device data within FL. The algorithm dynamically adjusts the global model according to the accuracy of the global model and the non-IID degrees with normalized stochastic gradients.
2. We propose an adaptive pruning method, i.e., FedAP, which performs unique pruning operations adapted to the different dimensions and importance of layers based on the non-IID degrees, to further improve the efficiency with an accuracy comparable to the original model.
3. We carry out extensive experiments to show the advantages of our proposed approach. FedDUAP significantly outperforms state-of-the-art approaches on four typical models and two datasets in terms of accuracy, efficiency, and computational cost.

The rest of this paper is organized as follows. In Section 2, we present related work. In Section 3, we propose our framework, i.e., FedDUAP, including FedDU and FedAP. In Section 4, we show the experimental results using two typical models. Finally, Section 5 concludes.

2 Related Work

FL is first proposed to collaboratively train a global model with the non-IID data across mobile devices [McMahan *et al.*, 2017]. Some works ([Kairouz *et al.*, 2021; Zhou *et al.*, 2022] and references therein) already exist to increase the performance of global models, while they only consider the device data. Because of incentive mechanisms [Yoshida *et al.*, 2020] or the convenience of cloud services, some insensitive data may reside on the server and can be exploited for the training process. While the server data can be assumed to be IID by gathering the data from devices and can be treated as that in an ordinary device with a simple average method [Yoshida *et al.*, 2020], the IID distribution of the server data is not realistic, and the upload of device data to the server may incur privacy or security concerns and excessive communication cost [Jeong *et al.*, 2018]. In the existing literature, the server data is mainly exploited for some specific objectives, e.g., heterogeneous models [He *et al.*, 2020], label-free data [Lin *et al.*, 2020b], and one-shot model training [Li *et al.*, 2021], based on knowledge transfer methods, while these methods may suffer from low accuracy [He *et al.*, 2020; Li *et al.*, 2021], or long training time [Lin *et al.*, 2020b]. The server data could also be transferred to the devices to further improve the model performance [Zhao *et al.*, 2018], which incurs significant communication costs. Finally, the server data can be exploited to select relevant clients for the training [Nagalapatti and Narayanam, 2021], which is orthogonal to and can be combined with our approach.

Model pruning can help reduce the computation and communication cost in the training process of FL [Jiang *et al.*, 2019], for which the server data is rarely considered. There are two major types of pruning, i.e., weight (unstructured) pruning and filter (structured) pruning. With unstructured pruning, some parameters are set to 0 without changing the model structure. While it can achieve accuracy comparable to the original model [Zhang *et al.*, 2021] and helps reduce communication costs in FL [Jiang *et al.*, 2019], unstructured pruning has limited advantages on general-purpose hardware in terms of computational cost [Lin *et al.*, 2020a]. In contrast, structured pruning directly changes the structure of the model by removing some neurons (filters in the convolution layers). While it can significantly reduce both computational cost and communication cost [Lin *et al.*, 2020a], it is complicated to determine the number of filters to preserve in the structured pruning method, and the existing methods are not well adapted to FL. Gradient compression or sparsification can be exploited in FL to reduce communication cost [Konečný *et al.*, 2016], which has been extensively researched in literature and won't be further explored in this paper.

3 Method

In this section, we first present the system model of our framework, i.e., FedDUAP. Then, we present our FL algorithm, i.e., FedDU, for the collaborative FL model update, and our adaptive model pruning method, i.e., FedAP, to reduce the computational and communication cost.

3.1 System Model

As shown in Figure 1, we consider an FL system composed of a server and N edge devices. We assume that the server is much more powerful than the devices. The server data resides in the server, and the device data is distributed in multiple devices, both of which can be locally exploited for the training

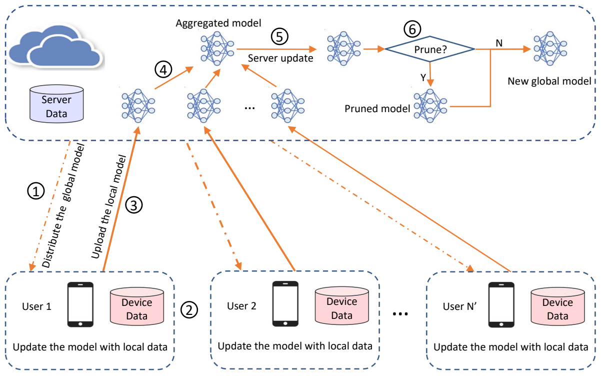


Figure 1: The training process of FedDUAP Framework.

process of FL without being transferred. We assume that Device k has a local dataset $D_k = \{x_{k,j}, y_{k,j}\}_{j=1}^{n_k}$, consisting of n_k samples. D_0 denotes the server data. $x_{k,j}$ is the j-th input data sample on Device k, and $y_{k,j}$ is the label of $x_{k,j}$. We denote the input data sample set by $\mathcal{X}$ and the label set by $\mathcal{Y}$. The objective of the FL training can be formulated as follows:

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

where w is the parameters of the global model, $F_k(w) \triangleq \frac{1}{n_k}\sum_{\{x_{k,j},y_{k,j}\}\in\mathcal{D}_k} f(w, x_{k,j}, y_{k,j})$ is the local loss function of Device k, and the loss function $f(w, x_{k,j}, y_{k,j})$ captures the error of the model on the data pair $\{x_{k,j}, y_{k,j}\}$.

While the data is generally non-IID in FL [Kairouz *et al.*, 2021], we exploit the Jensen–Shannon (JS) divergence [Fuglede and Topsoe, 2004] to represent the non-IID degree of the data on a device and the server as shown in Formula 2:

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

where $P_m = \frac{1}{2}(P_k + \overline{P})$, $\overline{P} = \frac{\sum_{k=1}^{N} n_k P_k}{\sum_{k=1}^{N} n_k}$, $P_k = \{P_k(y)|y \in \mathcal{Y}\}$ with $P_k(y)$ representing the probability that a data sample corresponds to Label y in Device k or the server ($k = 0$), and $\mathcal{D}_{KL}(\cdot||\cdot)$ is the Kullback-Leibler (KL) divergence [Kullback, 1997] defined in Formula 3:

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

A higher non-IID degree indicates that the data distribution of a specific device or the server differs from the global distribution more significantly. While the data cannot be transferred between the server and devices, we assume that the statistical meta information, i.e., P_k and n_k, can be shared between the server and devices during the training process, which incurs much less privacy concern [Lai *et al.*, 2021].

As shown in Figure 1, the training process of FedDUAP consists of multiple rounds, and each round is composed of 6 steps. In Step ①, the server randomly selects a fraction ($\mathscr{D}^t$ with t representing the round) of devices ($\mathscr{D}$) to train a global model and distributes the global model to each device. Then, the model is updated using the local data of each device in Step ②. The updated models are uploaded to the server in Step ③, and aggregated using FedAvg [McMahan *et al.*, 2017] in Step ④. Afterward, the model is updated (server update) using the server data in Step ⑤, which is detailed in Section 3.2. Finally, in a specific round, the model is pruned using the statistic information of both the server data and the device data, which is explained in Section 3.3. Steps ① - ④ are similar to those of conventional FL steps, while we propose using the server data to dynamically update the global model (Step ⑤) so as to improve the accuracy and an adaptive model pruning method based on both the server data and the device data (Step ⑥) to accelerate the training of FL.

3.2 Server Update

In this section, we propose our FL algorithm, i.e., FedDU, which takes advantage of both server data and device data to update the global model. FedDU exploits the server data to dynamically determine the optimal steps of the server update while considering the non-IID degrees of the server data and the device data in order to improve the convergence and accuracy of the global model.

While the size of server data may be much bigger than that within a single device, the naive update of the aggregated model with the stochastic gradients generated from the server data may suffer from objective inconsistency [Wang *et al.*, 2020]. In order to reduce the objective inconsistency, we normalize the stochastic gradients calculated based on the server data, inspired by [Wang *et al.*, 2020]. Then, the model update in FedDU is defined in Formula 4.

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

where w^t represents the weights of the global model at Round t, $w^{t-\frac{1}{2}}$ represents the weights of the global model after aggregating the models from devices as defined in Formula 5 [McMahan *et al.*, 2017], τ_{eff}^{t-1} is the effective step size for the server update defined in Formula 7, η is the learning rate, $\overline{g}_0^t(\cdot)$ is the normalized stochastic gradient on the server at Round t and defined in Formula 6.

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

where $n' = \sum_{k\in\mathscr{D}^t} n_k$ is the total number of samples in the selected devices, $g_k^{t-1}(\cdot)$ is the gradient calculated based on the local data in Device k.

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

where $w^{t-\frac{1}{2},i}$ is the parameters of the updated aggregated model, after i iterations within the batch of the server update at Round t, $g_0^{(t-1)}(\cdot)$ represents the corresponding stochastic gradients on the server, and $\tau = \lceil\frac{|n_0|E}{B}\rceil$ is the number of iterations performed within the server update, with E representing the number of local epochs and B representing the batch size. Within each round, the model is updated with multiple iterations by sampling a mini-batch of server data.

As τ_{eff}^t is critical to the training process, we propose to dynamically determine the effective step size based on the accuracy of the aggregated model, the non-IID degrees of the data, and the number of rounds, as defined in Formula 7.

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

Algorithm 1 Federated Dynamic Server Update (FedDU)

```
Input:
    $\mathscr{D}^t$: The set of selected devices at Round $t$
    $\mathcal{D}_k$: The dataset on Device $k$ with 0 representing that on the server
    $w^{t-1}$: The global model at 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
    $\eta$: The learning rate
Output:
    $w^t$: The global model at 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 $w^{t-1}, \eta$ according to Formula 5
 5: Calculate $w^t$ using $w^{t-\frac{1}{2}}, decay, E, B, P, \eta$ according to Formula 4
```

where acc^t is the accuracy (evaluated using the insensitive server data) of the aggregated model at Round t, i.e., $w^{t-\frac{1}{2}}$ defined in Formula 5, n_0 is the number of samples in the server data, $\mathcal{D}(\cdot)$ is defined in Formula 2, $\overline{P'}^t = \frac{\sum_{k \in \mathscr{D}^t} n_k P_k}{\sum_{k \in \mathscr{D}^t} n_k}$ denotes the distribution of all the data in the selected devices at Round t, P_0 represents the distribution of the server data, $decay \in (0, 1)$ is used to ensure that the final trained model converges to the solution of Formula 1, and $\mathcal{C}$ is a hyper-parameter. $f'(acc)$ is a function based on acc. At the beginning of the training, acc is relatively small, and the value corresponding to $f'(acc)$ is prominent to exploit the server data and the high performance of the server to update the model. At the end of the training, in order to achieve the objective defined in Formula 1, $f'(acc)$ becomes small to reduce the influence of the server data.

FedDU is shown in Algorithm 1. The local model update is performed in parallel in each selected device (Lines 1 - 3). Then, the aggregated model is calculated using Formula 5 (Line 4). Afterward, the new global model is updated based on the server data using Formula 4 (Line 5).

3.3 Adaptive Pruning

We propose a layer-adaptive pruning method, i.e., FedAP, to improve the efficiency of the FL training process. FedAP performs unique pruning operations on the server, which are adapted to the different dimensions and importance of multiple layers, based on the non-IID degrees of the server data and the device data. Please note that the pruning process is carried out at a specific round during the training process.

FedAP is shown in Algorithm 2. For each device and the server (Lines 2 - 4), we calculate the expected pruning rate using the server data and the device data (Line 3). In a device k or the server, given a neural network with initial parameters W_k, after T rounds of training, we denote the updated parameters by W'_k and the difference by $\Delta_k = W_k - W'_k$. Then, we calculate the Hessian matrix of the loss function, i.e., $H(W'_k)$, and sort its eigenvalues in ascending order, i.e., $\{\lambda_k^m | m \in (1, d_k)\}$ with d_k representing the rank of the Hessian matrix

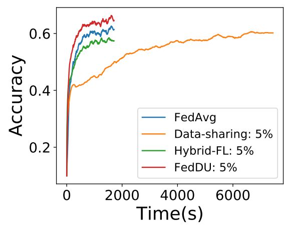
Algorithm 2 Federated Adaptive Structured Pruning (FedAP)

```
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 at Round $t$
    $W = [v_1, v_2, \cdots, v_m]$: The list of parameters in the model $w^*$ with $m$ representing the number of parameters
Output:
    $w'$: The pruned model at 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 8
 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 $l$-th layer of $w'$ with $w'_l$
16: end for
```

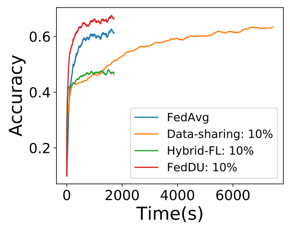
and m representing the index of an eigenvalue. We define a base function $B_k(\Delta_k) = H(W'_k) - \nabla L(\Delta_k + W'_k)$ with $\nabla L(\cdot)$ representing the gradient of the loss function, and we denote its Lipschitz constant as $\mathscr{L}_k$. Inspired by [Zhang *et al.*, 2021], we find the first m_k that satisfies $\lambda_{m_{k+1}} - \lambda_{m_k} > 4\mathscr{L}_k$ to avoid accuracy reduction, and we calculate the expected pruning rate by $p_k^* = \frac{m_k}{d_k}$, i.e., the ratio between the number of pruned eigenvalues and that of all the eigenvalues. As the expected pruning rate in each device is of much difference because of the non-IID distribution of data, we use Formula 8 to calculate an aggregated expected pruning rate for the entire model on the server (Line 5).

$$p^* = \sum_{k=0}^{n} \frac{\frac{n_k}{\mathcal{D}(P_k)+\epsilon}}{\sum_{k'=0}^{n} \frac{n_{k'}}{\mathcal{D}(P_{k'})+\epsilon}} * p_k^*, \tag{8}$$

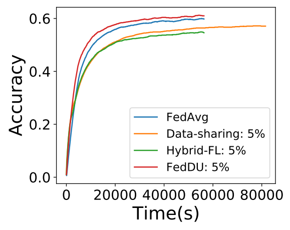
where ϵ is a small value to avoid division by zero. We calculate a global threshold value ($\mathscr{V}$), which is used to calculate the pruning rate of each layer. The global threshold value equals the absolute value of the $\lfloor R * p^* \rfloor$-th parameter (Line 7) in all the parameters in ascending order (Line 6). Then, for each convolutional layer (Line 8), we calculate the pruning rate of the layer by dividing the number of parameters having inferior absolute values than the threshold value by the total number (Lines 9 - 11). Inspired by [Lin *et al.*, 2020a], we prune the model based on the ranks of feature maps (the output of filters) (Lines 12 - 15). Let us denote the ranks of feature maps of Layer l by $R_l = \{r_l^j | j \in (1, d_l)\}$, where d_l represents the number of filters in the l-th layer. While feature maps are almost unchanged for a given model [Lin *et al.*, 2020a], we assume that the ranks on the server are similar



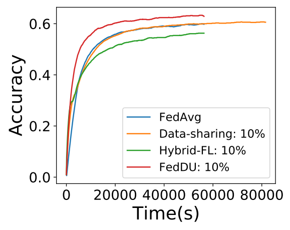
(a) CNN



(b) CNN



(c) ResNet



(d) ResNet

Figure 2: The accuracy and training time with diverse model update methods corresponding to FedDU with $p = 5\%$ and $p = 10\%$.

to those in each device. Thus, we prune the model with the feature maps on the server. We calculate (Line 12) and sort (Line 13) the ranks of the feature maps in R_l. We preserve the filters corresponding to the last $d_l - \lfloor p_l^* * d_l \rfloor$ ranks in the sorted R^l (in ascending order), in order to achieve the highest pruning rate $p_l \leq p_l^*$ (Line 14). Finally, we replace the layer of the original model with the preserved filters (Line 15). The process can be applied without the server data by carrying out the execution of Line 12 at a device and transferring the ranks R_l back to the server.

4 Experiments

In this section, we present the experimental results to show the advantages of FedDUAP by comparing it with state-of-the-art baselines, i.e., FedAvg [McMahan *et al.*, 2017], Data-sharing [Zhao *et al.*, 2018], Hybrid-FL [Yoshida *et al.*, 2020], HRank [Lin *et al.*, 2020a], IMC [Zhang *et al.*, 2021], and PruneFL [Jiang *et al.*, 2019].

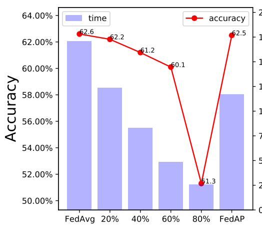
4.1 Experimental Setup

We set up an FL system composed of a parameter server and 100 devices. In each round, we randomly select 10 devices to participate in the training. We consider the datasets of CIFAR-10 and CIFAR-100 [Krizhevsky *et al.*, 2009] for comparison. We consider four models, i.e., a simple synthetic CNN network (CNN), ResNet18 (ResNet) [He *et al.*, 2016], VGG11 (VGG) [Simonyan and Zisserman, 2015], and LeNet in our experimentation.

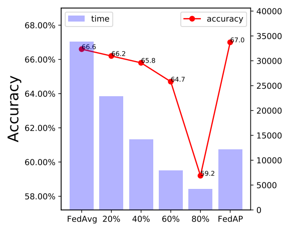
We report the results of CNN based on CIFAR10 and those of ResNet based on CIFAR100, while the other experimentation results reveal consistent findings. In Figure 3, we report the results of CNN and VGG based on CIFAR10 for the interest of training time. p represents the ratio between the size of data in the server and that in devices.

4.2 Evaluation with non-IID Data

In this section, we first compare FedDU with FedAvg, Data-sharing, and Hybrid-FL in terms of model accuracy. Then,



(a) CNN



(b) VGG

Figure 3: The accuracy and the training time with FedAvg, HRank of diverse pruning rates, and FedAP.

we compare the adaptive pruning method, i.e., FedAP, with FedAvg, IMC, HRank, and PruneFL, in terms of model efficiency. Finally, we compare FedDUAP, which contains both FedDU and FedAP, with the six state-of-the-art baselines.

Evaluation on FedDU

Data-sharing sends the server data to devices so that devices can mix the local data with the server data to form a dataset with a smaller non-IID degree to increase the accuracy of the global model. Hybrid-FL treats the server as an ordinary device using the FedAvg algorithm.

As shown in Figure 2, FedDU leads to a higher accuracy compared with FedAvg (up to 4.9%), Data-sharing (up to 4.1%), and Hybrid-FL (up to 19.5%) for both CNN and ResNet when $p = 5\%$ and 10%. Compared with FedDU, Data-sharing needs to transfer the server data to devices, which may incur a significant communication cost and privacy problems. In addition, Data-sharing needs a much longer training time to achieve the accuracy of 0.6 (up to 15.7 times slower) than FedDU on CNN.

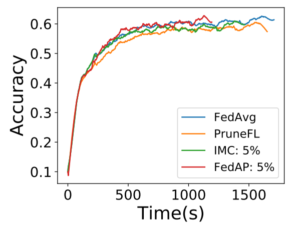
We also evaluate the performance of FedDU with diverse settings of τ_{eff} and server data. The dynamic adjustment of τ_{eff} significantly outperforms a static value (up to 1.7%) in terms of accuracy. In addition, with various distributions of the server data, we find that when the non-IID degree of the server data is smaller, the global model can get faster performance improvement (up to 6.1 times faster) when comparing the non-IID degree of 0.31 with that of $9.0 * 10^{-6}$.

Evaluation on FedAP

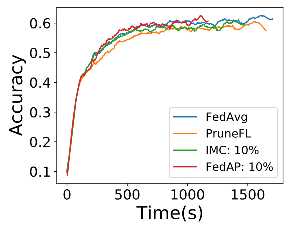
In this section, we compare FedAP with two adapted methods, i.e., HRank [Lin *et al.*, 2020a], and IMC [Zhang *et al.*, 2021], and a state-of-the-art pruning method in FL, i.e., PruneFL [Jiang *et al.*, 2019]. We exploit HRank with the server data during the training process of FL with diverse pruning rates, e.g., 0.2, 0.4, 0.6, and 0.8, of each layer. Similarly, we adapt the IMC method using the server data within the training process of FL. Among the three baseline approaches, HRank is structured, while IMC and PruneFL are unstructured. As shown in Figure 3, FedAP can achieve an appropriate pruning rate (37.8% on average for CNN and 72.0% on average for VGG), which corresponds to a faster training speed (up to 1.8 times faster), while ensuring an excellent accuracy of the model (up to 0.1% accuracy reduction compared with that of unpruned FedAvg). The training time and the accuracy decrease simultaneously when the pruning rate increases with HRank. It is evident that VGG prefers a pruning rate of 40%, while CNN favors a pruning

Table 1: **The final accuracy, training time, and computational cost with FedAvg, IMC, PruneFL, and FedAP.** “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 0.55 for ResNet. “Million FLoating-Point Operations (MFLOPs)” represents the computational cost in devices.

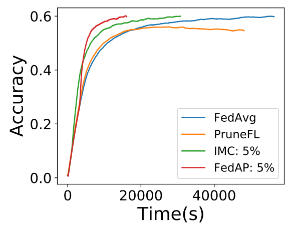
Model	Methods	Accuracy	Time	MFLOPs
CNN	FedAvg	**0.626**	838	4.6
	IMC	0.608	1316	4.6
	PruneFL	0.605	1536	4.6
	FedAP	**0.625**	**696**	**3.1**
ResNet	FedAvg	0.600	18059	557.3
	IMC	0.601	10141	557.3
	PruneFL	0.560	17212	557.3
	FedAP	**0.603**	**5262**	**246.9**



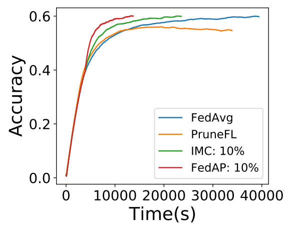
(a) CNN



(b) CNN



(c) ResNet



(d) ResNet

Figure 4: The accuracy and the training time with FedAvg, IMC, PruneFL, and FedAP with $p = 5\%$ and $p = 10\%$.

rate of 20% to achieve similar performance as FedAP, which reveals that different models correspond to different appropriate pruning rates. Our proposed method, i.e., FedAP, can generate adaptive pruning rates for each layer while achieving accuracy comparable with FedAvg. In addition, the accuracy of the pruned model of VGG can be even higher (0.4%) than the original one, while the training time is one time faster to achieve the accuracy of 0.6. We then compare FedAP with FedAvg, IMC, and PruneFL. As shown in Figure 4, the training process of FedAP is significantly faster than baselines. As shown in Table 1, FedAP achieves the highest accuracy among the pruning methods and corresponds to negligible accuracy reduction (up to 0.1% compared with unpruned FedAvg), while the training time is significantly shorter to achieve the accuracy of 0.55 (2.4 times faster than FedAvg, up to 92.7% faster than IMC, and 2.3 times faster than PruneFL). As unstructured pruning methods cannot exploit general-purpose hardware to speed up the calculation, the computational cost remains unchanged. In contrast, FedAP leads to a much smaller computational cost (up to 55.7% reduction).

Table 2: **The final accuracy, training time, and computational cost with diverse approaches.** “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 0.55 for ResNet. “MFLOPs” represents the computational cost in devices. “NaN” represents that the accuracy does not achieve the required accuracy. “D-S” represents Data-sharing.

Model	Method	Accuracy	Time	MFLOPs
CNN	FedAvg	0.626	838	4.6
	D-S	0.634	4447	4.6
	Hybrid-FL	0.480	NaN	4.6
	IMC	0.608	1316	4.6
	PruneFL	0.605	1536	4.6
	FedDUAP	**0.662**	**278**	**2.9**
ResNet	FedAvg	0.600	18059	557.3
	D-S	0.607	19114	557.3
	Hybrid-FL	0.563	41422	557.3
	IMC	0.601	10141	557.3
	PruneFL	0.560	17212	557.3
	FedDUAP	**0.635**	**4718**	**250.8**

Evaluation on FedDUAP

We compare FedDUAP, consisting of both FedDU and FedAP, with the baseline approaches. As shown in Table 2, FedDUAP achieves higher accuracy compared with FedAvg (4.8%), Data-sharing (12.0%), Hybrid (18.4%), IMC (6.9%), and PruneFL (6.3%). In addition, FedDUAP corresponds to a shorter training time to achieve the target accuracy (up to 2.8 times faster than FedAvg, 15.0 times faster than Data-sharing, 8.8 times faster than Hybrid-FL, 3.7 times faster than IMC, and 4.5 times faster than PruneFL) and a much smaller computational cost (up to 61.9% reduction for FedAvg, Data-sharing, Hybrid, IMC, and PruneFL).

5 Conclusion

In this work, we propose an efficient FL framework based on shared insensitive data on the server, i.e., FedDUAP. FedDUAP consists of a dynamic update FL algorithm, i.e., FedDU, and an adaptive pruning method, i.e., FedAP. We exploit both the server data and the device data to train the global model within FL while considering the non-IID degrees of the data. We carry out extensive experimentation with diverse models and real-life datasets. The experimental results demonstrate that FedDUAP significantly outperforms the state-of-the-art baseline approaches in terms of accuracy (up to 4.8% higher), efficiency (up to 2.8 times faster), and computational cost (up to 61.9% smaller).

Acknowledgments

This work was partially (for J. Jia) supported by the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[Amazon, 2021] Amazon. Amazon cloud. https://aws.amazon.com, 2021. Online; accessed 30/12/2021.

[CCP, 2018] California consumer privacy act home page. https://www.caprivacy.org/, 2018. Online; accessed 14/02/2021.

[Fuglede and Topsoe, 2004] Bent Fuglede and Flemming Topsoe. Jensen-shannon divergence and Hilbert space embedding. In *Int. Symposium on Information Theory (ISIT)*, page 31, 2004.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[He *et al.*, 2020] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[Jeong *et al.*, 2018] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[Jiang *et al.*, 2019] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *arXiv preprint arXiv:1909.12326*, 2019.

[Kairouz *et al.*, 2021] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, and Mehdi Bennis et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1), 2021.

[Konečnỳ *et al.*, 2016] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[Kullback, 1997] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

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

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

[Li *et al.*, 2021] Qinbin Li, Bingsheng He, and Dawn Song. Practical one-shot federated learning for cross-silo setting. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2021.

[Lin *et al.*, 2020a] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1529–1538, 2020.

[Lin *et al.*, 2020b] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

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

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

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

[Official Journal of the European Union, 2016] Official Journal of the European Union. General data protection regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679, 2016. Online; accessed 12/02/2021.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. on Learning Representations (ICLR)*, 2015.

[Wang *et al.*, 2020] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[Yoshida *et al.*, 2020] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani. Hybrid-FL for wireless networks: Cooperative learning mechanism using non-iid data. In *IEEE Int. Conf. on Communications (ICC)*, pages 1–7, 2020.

[Zhang *et al.*, 2021] Zeru Zhang, Jiayin Jin, Zijie Zhang, Yang Zhou, Xin Zhao, Jiaxiang Ren, Ji Liu, Lingfei Wu, Ruoming Jin, and Dejing Dou. Validating the lottery ticket hypothesis with inertial manifold theory. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[Zhao *et al.*, 2018] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[Zhou *et al.*, 2022] Chendi Zhou, Ji Liu, Juncheng Jia, Jingbo Zhou, Yang Zhou, Huaiyu Dai, and Dejing Dou. Efficient device scheduling with multi-job federated learning. *AAAI Conf. on Artificial Intelligence*, 2022. To appear.