



# Efficient federated learning on resource-constrained edge devices based on model pruning

Tingting Wu<sup>1,2,3,4</sup> · Chunhe Song<sup>1,2,3</sup>  · Peng Zeng<sup>1,2,3</sup>

Received: 3 January 2023 / Accepted: 16 May 2023 / Published online: 9 June 2023  
© The Author(s) 2023

## Abstract

Federated learning is an effective solution for edge training, but the limited bandwidth and insufficient computing resources of edge devices restrict its deployment. Different from existing methods that only consider communication efficiency such as quantization and sparsification, this paper proposes an efficient federated training framework based on model pruning to simultaneously address the problem of insufficient computing and communication resources. First, the framework dynamically selects neurons or convolution kernels before the global model release, pruning a current optimal subnet and then issues the compressed model to each client for training. Then, we develop a new parameter aggregation update scheme, which provides training opportunities for global model parameters and maintains the complete model structure through model reconstruction and parameter reuse, reducing the error caused by pruning. Finally, extensive experiments show that our proposed framework achieves superior performance on both IID and non-IID datasets, which reduces upstream and downstream communication while maintaining the accuracy of the global model and reducing client computing costs. For example, with accuracy exceeding the baseline, computation is reduced by 72.27% and memory usage is reduced by 72.17% for MNIST/FC; and computation is reduced by 63.39% and memory usage is reduced by 59.78% for CIFAR10/VGG16.

**Keywords** Federated learning · Model pruning · Communication efficient · Edge intelligence

## Introduction

With the rapid growth of the number of intelligent edge devices and the scale of data generated by the Internet of Things [1, 2], distributed training by uploading the data generated by edge devices to a data center for centralized learning will be limited by communication resources and delays. At the same time, the privacy of edge device data and the increased complexity of deep learning models make it difficult for traditional training to rationally use a large amount

of edge data [3]. Edge training based on federated learning [4] provides a feasible solution for overcoming centralized learning. Federated learning cooperates with individual edge devices, trains the same models using local data, and aggregates the parameters of each device to update the global model [5, 6].

A federated learning system typically assumes that clients are equipped with fast processors and sufficient computation capability to perform calculations locally and update model parameters. However, most edge devices, such as mobile phones, wearables, sensors, etc., have limited computing and memory resources, which makes it difficult to support deep learning model training. With the large-scale production of image, video, voice and other data on edge devices, the low-density model can no longer meet the data volume requirements. The high performance and precision of deep neural networks always come at the cost of a larger model size and more computation. The transmission of CNN models with millions of parameters also brings great challenges to network communication. How to ensure efficient federated training of a model on edge devices with weak computa-

✉ Chunhe Song  
songchunhe@sia.cn

<sup>1</sup> State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China

<sup>2</sup> Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China

<sup>3</sup> Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China

<sup>4</sup> University of Chinese Academy of Sciences, Beijing 100049, China

tion capability while communicating efficiently has become a research difficulty.

Model pruning has been widely studied as an important solution in edge inference deployment [7, 8], which indicates that there exists a subnet that can represent the performance of the entire model after training. While the above research prunes pre-trained models in centralized training, our goal is to design pruning strategies during federated training to meet the computational requirements of resource-constrained devices. There has been a lot of researches on reducing communication cost by pruning client models before uploading. However, they only considered the communication cost and did not change the model structure to reduce the computation on the clients. Recently, some research attempts to reduce the computational requirements on the client by modifying the model in training through model pruning. Liu et al. [9] and AdaptCL [10] improve learning efficiency by changing the size of the local training model on the client, but pruning on the client increases additional calculations. PruneFL [11] proposes to use unstructured pruning in training and support corresponding sparse matrix computation by extending the deep learning framework, which will be difficult to generalize. Federated Pruning [12] performs structural pruning on the client model, which is similar to our proposed work, but does not propose corresponding parameter aggregation scheme for lossy pruning.

This paper proposes an efficient federated training framework based on model structure pruning to overcome the above challenges. Different from pruning on the clients, we prune the global model on the computatively powerful server with negligible latency and reduced upstream and downstream communication. We perform structural pruning on the model to change the structure size of the delivered model without relying on dedicated acceleration. Most of the current federated systems adopt synchronization scheme, in which the global model will not converge if the pruned models are directly aggregated. Therefore, we developed a new parameter aggregation update scheme to reduce the error caused by pruning through model reconstruction and parameter reuse. Instead of changing the size of the final global model, the goal is to train the larger model on the server with the smaller models on the clients. Specifically, the framework dynamically selects neurons or convolution kernels for the model, prunes the current optimal sub-net before releasing the global model, and then distributes the compressed model to each client for training. We change the model structure by directly removing redundant convolution kernels or neurons to form a compact model. To summarize, the specific contributions of this article are as follows:

- We propose an efficient federated training framework based on model structure pruning, which greatly reduces the demand for client computing and memory resources by dynamically selecting the optimal sub-model of the current global model for delivery.
- We develop a new parameter aggregation update scheme, which provides training opportunities for global model parameters and maintains the complete model structure through model reconstruction and parameter reuse, reducing the error caused by pruning.
- We conduct a large number of experiments on different data sets and data distributions to verify the effectiveness of the proposed framework, which reduces upstream and downstream communication while maintaining the accuracy of the global model and reducing client computing costs.

## Related work

### Edge inference/training based on model pruning

For resource-limited edge devices, model pruning is proposed to reduce the complexity of a neural network before it is deployed. The earliest research on model pruning was in 1988. Hanson and Pratt [13] proposed an amplitude-based pruning method for shallow fully connected networks. In recent years, [14] combined various methods such as pruning, quantization, and Huffman coding to compress CNNs. Li et al. [15] proposed summing the absolute value of the convolution kernel as a criteria to measure its importance. This was the first time that the convolution kernel was used as the pruning unit to achieve network compression by changing the model structure. The research of [8, 15–22] and others put forward various criteria to determine the importance of convolution kernels on structured pruning, and the redundant convolution kernels are deleted or set to zero. Regarding pruning methods, iterative pruning [23], soft pruning [24] and dynamic pruning [25] were proposed to identify redundant parameters during training. Thus far, how to judge the effectiveness of the parameters in the model and minimize the loss of accuracy is still an unsolved problem.

### Efficient federated learning

To address the bottleneck of communication delay in federated learning, much research has been carried out on gradient compression, including gradient quantization and sparsification. Quantization compresses parameters by changing the number of bits of a floating-point [26]. Bernstein

etal. [27] proposed SIGNSGD, which only transmits the symbols of each small batch of stochastic gradients, and uses majority voting to aggregate the gradient symbols of each client. Sattler et al. [28] and Xu et al. [29] proposed sparse ternary compression (STC) and ternary quantization (FTTQ), respectively, and expressed the model parameters as  $[-1, 0, 1]$ , which greatly reduced the communication overhead. Other variants of the quantization gradient scheme include three-value quantization [30], variance reduction quantization [31], error compensation [32] and gradient difference quantization [33, 34].

Sparsification is equivalent to the client discarding part of the parameters before communication. Strom [35] set a fixed threshold, and the parameters were allowed to upload when the gradient was greater than the threshold. Dryden et al. [36] sparsified the gradient using a fixed ratio, while [37] simplified the gradient update based on a single threshold of the absolute value, with the minimum gradient by removing the absolute value of the R% gradient. AdaComp [38] was based on localized selection of gradient residues and automatically tunes the compression rate depending on local activity. The framework proposed in this paper is completely orthogonal to the above research, greatly reducing edge-side computation while compressing both upstream and downstream communication. In addition, the combination of knowledge distillation and federated learning is gradually used to reduce the amount of computation on the client side. Xing et al. [39] proposed an efficient federated distillation learning system for multitask time series classification. If there is a huge capacity gap between the large teacher model and the small student model, it may be difficult for the student model to learn well, so the optimal design of the server and client models must be determined, and the researches [40–42] designed the model through an optimization problem.

## Approach

As described in [6, 43], in a federated learning system, each edge local model  $W_k$  according to local data, and sends the trained model to the server:

$$W_k(t) = W_k(t-1) - \eta \nabla f_k(W_k(t-1)). \quad (1)$$

The server updates global model  $W_g$  by aggregating the model parameters of each client,

$$W_g = \sum_{k=1}^K p_k W_k, \quad (2)$$

where  $p_k \geq 0$ ,  $\sum_{k=1}^K p_k = 1$ . Therefore, there are two models in the FL system: one is local model, maintained by each

edge device, and the other is global model updated by the central server. If pruning is applied to federated learning system, the first thing to consider is whether to prune the local model on the client or the global model on the server. In addition, how can the model be pruned and how can the parameters of the pruned model be updated?

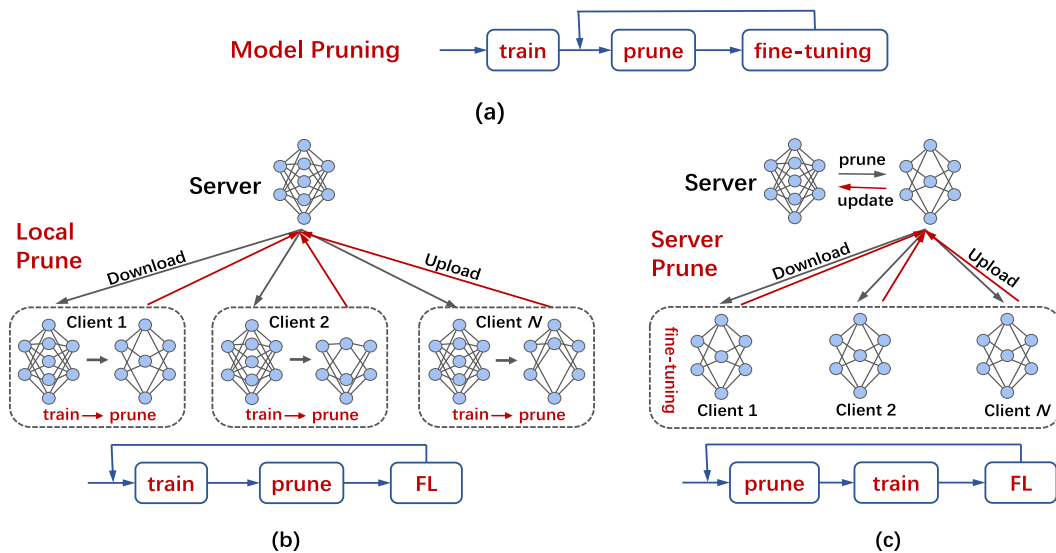
## Where to prune?

Typical model pruning includes three stages [44]: (1) training a large, over-parameterized model (sometimes a pre-trained model), (2) pruning the trained large model according to certain criteria, and (3) fine-tuning the pruned model to restore the lost performance, as shown in Fig. 1a. The processing object of edge inference based on network pruning is the trained model. However, in federated learning, the object of pruning is the model in the training process. Whether to prune the local model on the client or the global model on the server, and how to ensure the convergence of the final model while ensuring the effectiveness of the pruning is the first consideration.

Without changing the structure of the global model, two models are maintained on the server, the global model and the pruned model. Figure 1b shows pruning on the client, (1) global model  $W_g$  is issued, and the client trains several epochs based on local data, (2) pruned model  $W_k^P$  is obtained by pruning trained model  $W_k$ , and (3) each client uploads pruned model  $W_k^P$  and they are aggregated on the server side. Fig. 1c shows pruning on the server, (1) the server prunes global model  $W_g$ , and sends pruned model  $W_g^P$ , (2) the client trains the pruned model for several epochs, and (3) the client uploads the trained model and the global model is aggregated and updated on the server.

All previous sparsity-based studies have trained the full model and then selected the larger or important parameters of the gradient update to upload. Based on current research, local pruning, in which parameters that do not need to be uploaded are directly pruned instead of sparse coding, is the most readily evaluated federated learning pruning scheme. However, we mainly focus on the problem of insufficient computing resources on edge devices, not just the communication problem. We discuss the pruning location from the aspects of fine-tuning method, computation, communication, and model structure:

1. Fine-tuning method: Fine-tuning is an essential step to restore network performance after pruning. In federated learning, the data are only on the client, and fine-tuning can only be performed on the client. As seen in Fig. 1, in local pruning, the client model is first aggregated after pruning, and then the aggregated model is trained (not the pruned model). If fine-tuning is done locally, the amount of computation is greatly increased, and if not, part of



**Fig. 1** Comparison of model pruning on the server and client. **a** Is the typical model pruning process. **b** Is local pruning, where the model uploads the pruned model after the client finishes pruning, and aggregates it on the server. **c** Is server pruning, in which pruning is performed

before the model is issued, only subnet training is performed locally, all training parameters are uploaded, and the model is aggregated and updated on the server

the training information is lost, causing an error in the federated learning. However, in server pruning, pruning is followed by model training. Local training on the client is equivalent to the fine-tuning in model pruning, that is, fine-tuning in the original pruning process is transferred to the client.

2. **Computation:** Model pruning is to solves the problem of insufficient local computation capability. Current pruning methods are mostly data-driven and need to traverse all the parameters of a model. Pruning on the client increases local computation requirements instead.
3. **Communication:** Server pruning can reduce upstream and downstream communication at the same time, while local pruning can only reduce upstream communication without changing the global model.
4. **Model structure:** Due to different local client data distributions, the trained models are different, and a model after pruning on a client is also different (whether the pruning rate is identical or not). For example, a convolution kernel of client  $k$  is pruned, but the convolution kernel of client  $j$  is not pruned. This leads to heterogeneous models during aggregation, which increases the difficulty of updating the parameters of the global model and leads to noise.

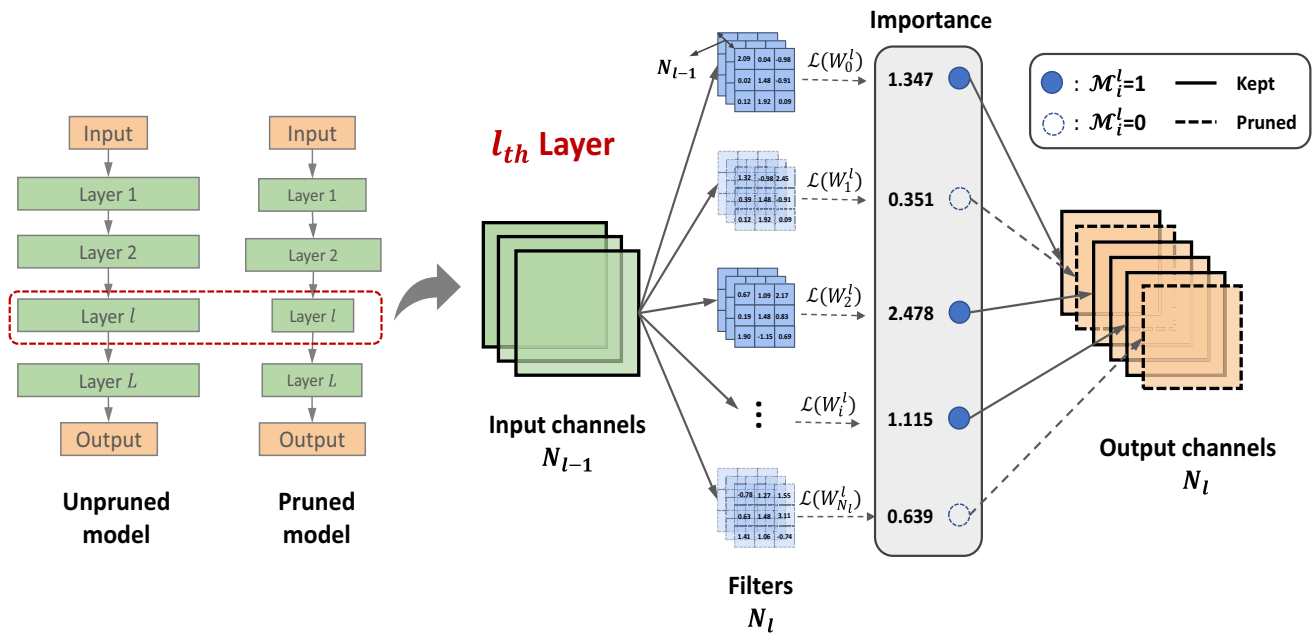
In summary, we choose to perform pruning on the server to reduce upstream and downstream communications without changing the original pruning process, reduce local computation, and avoid the problem of model heterogeneity.

### Model pruning and mask

The basic idea of model pruning is to remove the unimportant parts. Assuming a pretrained CNN model has a set of  $L$  convolutional layers, then the parameters of the  $l_{th}$  layer can be represented as a set of 3D filters  $W_l = \{W_1^l, W_2^l, \dots, W_{N_l}^l\} \in \mathbb{R}^{N_l \times N_{l-1} \times k_l \times k_l}$ , where  $N_l$  represents the number of filters in the  $l_{th}$  layer and  $k_l$  denotes the kernel size. In model pruning,  $W_l$  can be split into two groups, i.e., a subset to be kept  $S_l$  and a subset, with less importance, to be pruned  $P_l$ , where  $m_l$  and  $n_l$  are the number of important and unimportant filters, respectively. Determining which filter needs to be pruned is a combinatorial optimization problem, that can be expressed as follows:

$$\begin{aligned} \min_{\mathcal{M}_l} & \sum_{l=1}^L \sum_{i=1}^{N_l} \mathcal{M}_{li} \mathcal{L}(W_i^l) \\ \text{s.t.} & \sum_{li} \mathcal{M}_{li} = m_l, \end{aligned} \tag{3}$$

where  $\mathcal{M}_{li} \in \{0, 1\}$  is the mask of the filter or neuron which is 1 if  $W_i^l$  is grouped in  $S_l$ , or 0 if  $W_i^l$  is grouped in  $P_l$ .  $\mathcal{L}(\bullet)$  is the criterion for judging the importance of a filter. For the current large network structure, finding a subnet that can be pruned without performance degradation is an NP-hard problem, which is difficult to accurately solve by searching all possible subsets. The current popular pruning method is



**Fig. 2** A round of the model structure pruning process in federated learning. Convolutional kernels with small importance and their corresponding feature maps are directly removed

to determine the importance of parameters based on criteria and delete the parameters with low importance. For example, using the sum of the absolute values of a filter as a criterion, the importance of filter  $\mathcal{L}(W_i^l)$  is,

$$\mathcal{L}(W_i^l) = \sum_{l=1}^{N_{l-1}} \sum |K_l|, \quad K_l \in \mathbb{R}^{k_l \times k_l}. \quad (4)$$

Then,  $\mathcal{L}(W_i^l)$  is sorted, where the  $\text{Top}(m_l)$  is reserved for high importance, and its corresponding  $\mathcal{M}_{li} = 1$ ,  $W_i^l \in S_l$ . The other filters  $P_l$  are pruned, and their corresponding feature maps are also removed at the same time, as shown in Fig. 2.

### Model aggregation and updating

The process of parameter aggregation and updating in a federated learning edge training framework based on model pruning is shown in Fig. 3. The circle in Fig. 3 represents a convolution kernel in a convolutional neural network or a neuron in fully connected neural network. The server maintains two models: the global model  $W_g$  and the pruned model  $W_p$ . The parameter aggregation and update of the  $\tau_{\text{th}}$  round is:

1. Before issuing the global model on the server, the importance of the model parameters is first judged, important neurons are identified, and update the corresponding mask  $\mathcal{M}^\tau$  is updated. For neurons whose  $\mathcal{M}^\tau$  is 0, the neuron is

deleted and the corresponding weight and bias to obtain a compressed small model  $W_p^\tau$ .

2. Pruned model  $W_p^\tau$  is sent to each client, and the client uploads all model parameters after training  $W_k^\tau$ .
3. The server first aggregates the parameters uploaded by each client to obtain  $W_{\text{FL}}^\tau$ .  $W_{\text{FL}}^\tau$  is fused with the model  $W_g^{\tau-1}$  parameters before compression. In  $W_g^\tau$ , the original pruned neuron parameters are consistent with the previous round of  $W_g^{\tau-1}$ , and the unpruned neuron parameters are correspondingly replaced and updated with the parameters in  $W_{\text{FL}}^\tau$  according to the index order of  $\mathcal{M}^{\tau-1} = 1$ . The specific process is shown in Fig. 3. The entire framework of federated learning based on model pruning is shown in Algorithm 1.

$$W_{\text{FL}}^\tau \leftarrow \sum_{k=1}^K p_k W_k^\tau, \quad (5)$$

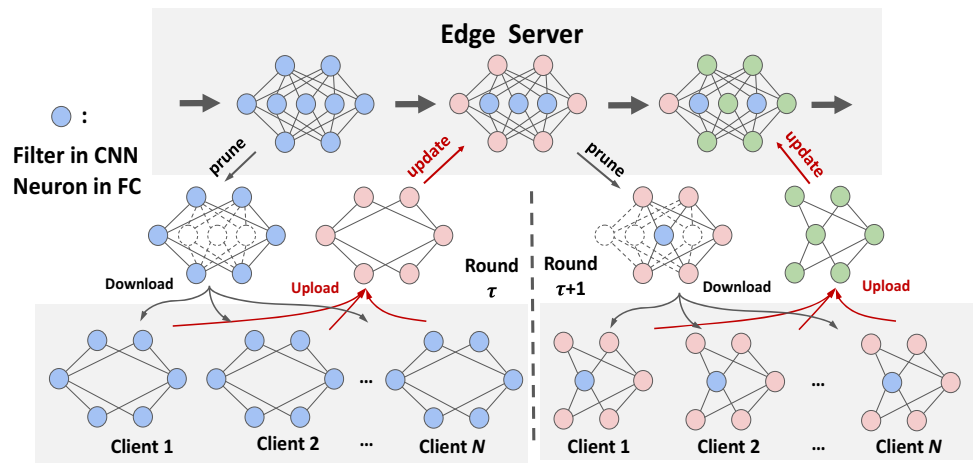
$$W_g^\tau(l, i) = \begin{cases} W_{\text{FL}}^\tau(l, j), & \text{if } W_g^{\tau-1}(l, i) \in S_l^{\tau-1} \\ W_g^{\tau-1}(l, i), & \text{if } W_g^{\tau-1}(l, i) \in P_l^{\tau-1}. \end{cases} \quad (6)$$

### Convergence analysis

We assume that the pruned network parameters obtained from some current importance criteria can represent the performance of the original network, where the parameters retained after pruning contribute greatly to the network. The gradient of each neuron is denoted as  $P(g(W^\tau))$ , then the probability of each neuron being retained in each round is



**Fig. 3** The  $\tau$  and  $\tau + 1$  rounds of the client model and global model aggregation update process



$p_i$ . Therefore, the gradient variance can be reformulated as:

$$P(g) = \left[ \mathcal{M}_1 \frac{g_1}{p_1}, \mathcal{M}_2 \frac{g_2}{p_2}, \dots, \mathcal{M}_d \frac{g_d}{p_d} \right], \tag{7}$$

then the variance of  $P(g(W^\tau))$  can be reformulated as:

$$\begin{aligned} \mathbb{E} \sum_{i=1}^N [P(g)_i^2] &= \sum_{i=1}^N \left[ \frac{g_i^2}{p_i^2} \times p_i + 0 \times (1 - p_i) \right] \\ &= \sum_{i=1}^N \frac{g_i^2}{p_i}. \end{aligned} \tag{8}$$

Therefore, the trade-off between  $p_i$  and the gradient variance can be formulated as the following optimization problem:

$$\min_p \sum_{i=1}^N p_i \quad \text{s.t.} \quad \sum_{i=1}^N \frac{g_i^2}{p_i} \leq (1 + \epsilon) \sum_{i=1}^N g_i^2, \tag{9}$$

where  $0 < p_i \leq 1$  and  $\epsilon$  can control the variance increase of  $g$ . We can get the solution of Eq. (9) by introducing Lagrange multipliers  $\lambda$  and  $\mu_i$ , as the following objective:

$$\begin{aligned} &\min_p \max_{\lambda} \max_{\mu} L(p_i, \lambda, \mu_i) \\ &= \sum_{i=1}^N p_i + \lambda^2 \left( \sum_{i=1}^N \frac{g_i^2}{p_i} - (1 + \epsilon) \sum_{i=1}^N g_i^2 \right) \\ &\quad + \sum_{i=1}^N \mu_i (p_i - 1). \end{aligned} \tag{10}$$

Consider the KKT conditions of the above formulation, we have:

$$1 - \lambda^2 \frac{g_i^2}{p_i^2} + \mu_i = 0. \tag{11}$$

We can get the following connections combined with the complementary relaxation condition of  $\mu_i (p_i - 1) = 0$ :

$$p_i = \begin{cases} 1, & \text{if } \mu_i \neq 0 \\ \lambda |g_i|, & \text{if } \mu_i = 0 \end{cases}. \tag{12}$$

As can be seen that if  $|g_i| \geq |g_j|$  then  $|p_i| \geq |p_j|$ . Therefore, there is a set  $S$  with  $p_j = 1, \forall j \in S$ , and its corresponding  $|g_j|$  has the largest absolute magnitude. Assuming that the size of the set is  $k (0 \leq k \leq N)$  and the elements are ordered by magnitudes, denoted as  $g_{(1)}, g_{(2)}, \dots, g_{(N)}$ , we have

$$p_{(i)} = \begin{cases} 1, & \text{if } i \leq k \\ \lambda |g_{(i)}|, & \text{if } i > k \end{cases}, \tag{13}$$

which further implies

$$\lambda = \frac{\sum_{i=k+1}^N |g_i|}{\epsilon \sum_{i=1}^N g_i^2 + \sum_{i=k+1}^N g_i^2}. \tag{14}$$

And the probability vector  $p$  is

$$p_i = \begin{cases} 1, & \text{if } i \in S \\ \frac{|g_i| (\sum_{i=k+1}^N |g_j|)}{\epsilon \sum_{j=1}^N g_j^2 + \sum_{j=k+1}^N g_j^2}, & \text{if } i \notin S \end{cases}. \tag{15}$$

We can get from  $(\rho, s)$ -approximately sparsity[3] that if there exists a subset  $S$  such that  $|S| = s$  and

$$\|g_{S^c}\|_1 \leq \rho \|g_S\|_1, \tag{16}$$

where  $S^c$  is the complement of  $S$ . Thus, the variance of  $P(g)$  can be bounded by

$$\begin{aligned} \mathbb{E} [\|P(g)\|_0] &= \sum_{i=1}^N p_i = \sum_{i \in S} p_i + \sum_{i \notin S} p_i \\ &= s + \sum_{i \notin S} \frac{|g_i| \left( \sum_{j=k+1}^d |g_j| \right)}{\epsilon \sum_{j=1}^k g_j^2 + (1 + \epsilon) \sum_{j=k+1}^N g_j^2} \\ &= s + \frac{\|g_{S^c}\|_1^2}{\rho \|g_S\|_2^2 + (1 + \rho) \|g_{S^c}\|_2^2} \\ &\leq s + \frac{\rho^2 s \|g_S\|_2^2}{\rho \|g_S\|_2^2 + (1 + \rho) \|g_{S^c}\|_2^2} \\ &\leq (1 + \rho)s. \end{aligned} \tag{17}$$

Convolution kernels or neurons are dynamically selected in each round, and the gradient of the model is bounded by the above formula, thereby ensuring convergence of the model.

In each round, we directly delete all the unimportant neurons and their corresponding weights instead of setting them to 0 by soft training. The client only trains a subnet composed of important neurons, and only updates the selected neuronal parameters during this round. In the model pruning in edge inference, neurons will be permanently removed. However, in federated learning, a fixed subnet structure is determined before the model reaches the ideal performance, which will seriously affect the convergence effect. The proposed framework guarantees synergistic convergence in two aspects: (1) The server maintains two models, and when the global parameters are updated, the pruned neurons still retain the original parameters of the previous round, instead of directly discarding the previously trained parameters. (2) During each round, the neurons that contribute the most to the network performance are always selected for update. The client uploads all training update parameters without losing the learned information, maximizing the use of local training updates, thus ensuring the convergence of training.

## Experiments

### Performance indicators

**Training computation:** In model compression, the computational power required for forward propagation of a model is used to evaluate the complexity of the model. Model training includes forward propagation and back-propagation, where the forward propagation computation is mainly on the feature graph and weight matrix multiplication and the backpropagation computation is on the reverse gradient computation. The

### Algorithm 1 Efficient federated learning based on model pruning

**Require:** global model:  $W_g$ ; client model:  $C_k, k = 1, \dots, K, W_k \leftarrow W_g; \mathcal{M} \leftarrow 0, \mathcal{M} \in \mathbb{R}^{L \times N}, L$  is the number of network layers in  $W_g$ , and  $N$  is the number of neurons in each layer; prune ratio:  $rate$ ;  
**Ensure:**  $W_g^T$   
**1: Global Server:**  
**2: for**  $l = 1, l \leq L; i = 1, i \leq N_l; j = 1, j \leq \|S_l^{\tau-1}\|_0$  **do**  
**3: if**  $W_g^{\tau-1}(l, j) \in S_l^{\tau-1}$  **then**  
**4:** $W_g^\tau(l, i) \leftarrow W_{F_L}^\tau(l, j)$   
**5: else**  
**6:** $W_g^\tau(l, i) \leftarrow W_g^{\tau-1}(l, i)$   
**7: end if**  
**8: end for**  
**9: for**  $l = 1, l \leq L; i = 1, i \leq N_l;$  **do**  
**10:** $\mathcal{L}(l, i) \leftarrow \mathcal{L}(W_g^\tau(l, i))$   
**11: if**  $\mathcal{L}(W_g^\tau(l, i))$  in  $\text{Top}\{\mathcal{L}(l, i) * rate\}$  **then**  
**12:** $W_g^\tau(l, i) \in S_l^\tau$   
**13:** $\mathcal{M}^\tau(l, j) = 1$   
**14: else**  
**15:** $W_p^\tau \leftarrow W_g^\tau - W_g^\tau(l, j)$   
**16:** $W_g^\tau(l, i) \in P_l^\tau$   
**17:** $\mathcal{M}^\tau(l, i) = 0$   
**18: end if**  
**19: end for**  
**20: return**  $W_p^\tau$   
**21: Edge Devices:**  
**22:** $W_k^{\tau-1} \leftarrow W_p^\tau$   
**23:** $W_k^\tau \leftarrow W_k^{\tau-1} - \eta \nabla f_k(W_k^{\tau-1})$   
**24: return**  $W_k^\tau$

MAC operations required by the two are the same. Therefore, the computation required for training the  $l$ th convolutional layer in a convolutional neural network is expressed as:

$$\text{FLOPs}_l = 2 \times 2 \times N_b m_b \times C_{in} \times K^2 \times H \times W \times C_{out}, \tag{18}$$

where  $m_b$  is the minibatch size and  $N_b$  is the total minibatch number.  $K, H, W$  are the size of the convolution kernel and the height and width of the feature map, respectively, and  $C_{in}, C_{out}$  are the number of input and output channels of the convolution layer, respectively. The computation required for training the  $l$ th layer in a fully connected neural network is:

$$\text{FLOPs}_l^{fc} = 2 \times N_b m_b \times (2 \times I - 1) \times O, \tag{19}$$

where  $I$  is the input neuron number and  $O$  is the output neuron number. Therefore, the computation of a single training cycle of the network model can be expressed as:  $\text{FLOPs} = \sum_{l=1}^L \text{FLOPs}_l$ .

**Training memory usage:** We simplify the memory required for training to calculate the weights, gradients and generated feature maps of the network in a single batch (activation for fully connected network). The gradient matrix and the weight

matrix are the same size. Therefore, the memory required for the  $l_{th}$  convolutional layer is:

$$Mem_l = 2 \times (B_f K^2 \times C_{out} \times C_{in}) + B_a H \times W \times C_{out}. \tag{20}$$

For the memory required for training the  $l_{th}$  layer in fully connected neural network:

$$Mem_l^{fc} = 2 \times B_f (I \times O + 1) + 2 \times B_a O, \tag{21}$$

where  $B_f$  and  $B_a$  are data bit values that are usually equal to 32 in an edge device. Therefore, the memory usage of a single training cycle of the model can be expressed as:  $Mem = \sum_{l=1}^L Mem_l$ .

Model parameters: Since the communication time is affected by the bandwidth, we take the parameter of the model as an index to evaluate the communication efficiency. In a convolutional neural network, the parameter quantity of the  $l_{th}$  convolutional layer is:

$$Param_l = K^2 \times C_{in} \times C_{out}. \tag{22}$$

The parameter quantity of the  $l_{th}$  layer in fully connected neural network is:

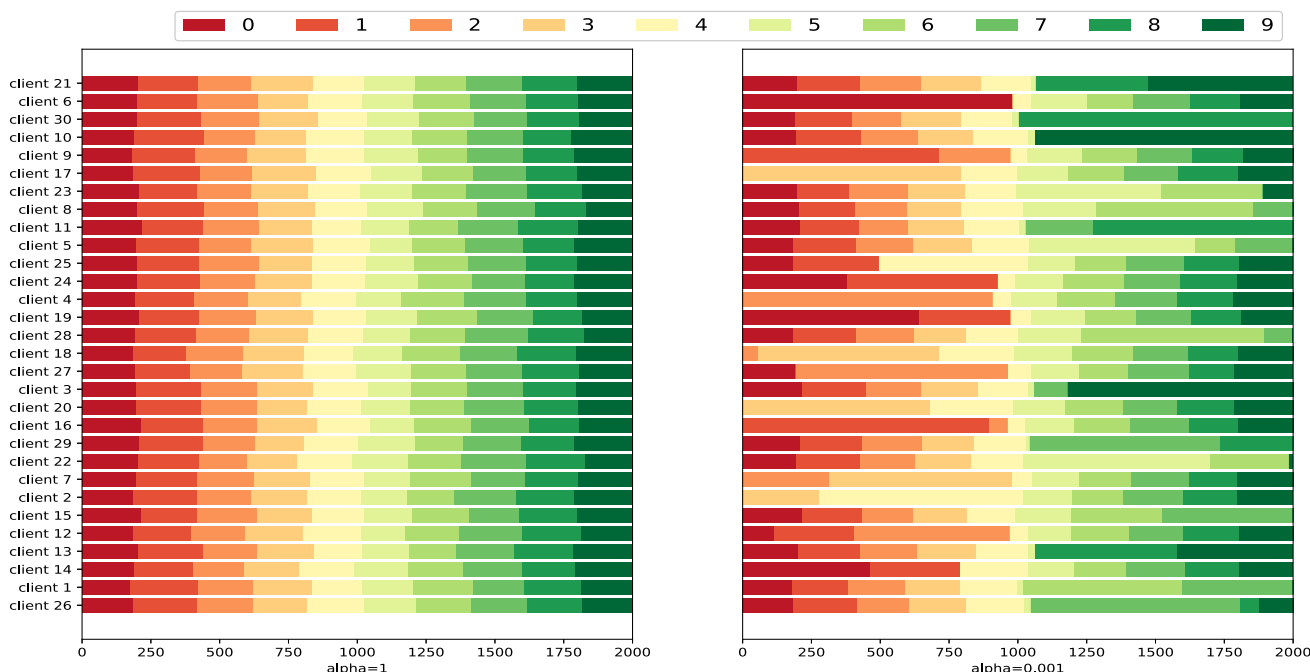
$$Param_l^{fc} = I \times (O + 1). \tag{23}$$

The parameters of the entire model are:  $Param = \sum_{l=1}^L Param_l$ .

### Models and datasets

We evaluate the effectiveness of the proposed framework on two classification tasks: (1) CIFAR10 on VGG16 and (2) MNIST on a 5-layer fully connected network where the number of neurons in each layer is [784,512,512,256, 100,10]. The two models represent the most widely used models at present and are typical tasks in FL. Both models are verified in IID and Non-IID data distribution scenarios. CIFAR-10 dataset contains 60,000 images (50,000 for training, 10,000 for testing) from 10 classes, and MNIST dataset contains 60,000 training and 10,000 test greyscale images of handwritten digits of size  $28 \times 28$ . We assume that the datasets for each client follow a distribution over  $N$  classes parameterized by a vector  $q$  ( $q_i \geq 0, i \in [1, N]$  and  $\|q\|_1 = 1$ ). To obtain a set of clients with different data distributions, we generate  $q \sim Dir(\alpha)$  from Dirichlet distribution, where  $\alpha > 0$  is a concentration parameter controlling the identicalness among clients. For every client, given an  $\alpha$ , we sample  $q$  and assign the client with the corresponding number of images from 10 classes. Fig. 4 illustrates populations drawn from the Dirichlet distribution with  $\alpha = 1$  and 0.001, corresponding to the IID and Non-IID data distribution scenarios in the experiments in this paper, respectively.

On the CIFAR10 dataset, we set the number of clients to 15, and all clients participate in training. The batch size is set to 128, the learning rate is 0.1, SGD is used for training, and the weight decay is  $5e-4$ . On the MNIST dataset, the number



**Fig. 4** Distribution among classes is represented with different colors. The populations in figure generated from Dirichlet distribution with  $\alpha = 1$  and 0.001, respectively, 30 random clients each



of clients is set to 30, the batch size is set to 64, and SGD is used for training one epoch per round with learning rate of 0.01. We use the global model test accuracy obtained by FedAvg [6] and the average loss of each client as the baseline.

### Different pruning rates

We first evaluate the effectiveness of the proposed framework at different pruning rates and the convergence impact on the global model. We set the pruning ratio of the number of neurons in each layer from 30 to 80%, and the calculation amount of the corresponding model decreases by 40–80%. For the MNIST dataset, whether in the IID or Non-IID data distribution, when the pruning rate is within 70%, the increase of the pruning rate has little effect on the convergence speed of the global model, and the final accuracy is still comparable or even higher than full model training, as shown in Fig. 5a–d. However, when the pruning rate reaches 80%, the convergence speed of pruning training is significantly slower, and the shock is more significant. In addition, more rounds are required to achieve the same accuracy as the full model training. When the pruning rate exceeds 80%, the global model begins to diverge, which indicates that the compressed model is too small to fit the data fully.

In VGG, the pruning granularity is a convolution kernel, which is different from the fully connected neural network. We set the pruning ratio of the number of convolution kernels in each layer from 20 to 50%, and the calculation amount of the corresponding model decreases by 35–75%, as shown in Fig. 5e–h. It can be seen from the figure that the convergence speed of the global model slows down with the increasing pruning rate, but the final accuracy is still better than the full model training. However, when the pruning rate increases to 75%, the model convergence speed becomes very slow, and the model performance degrades. In conclusion, our proposed pruning training framework is effective, and the more complex the network is more sensitive it is to pruning training.

### Different parameter selection criteria

In model pruning, the criterion for parameter redundancy is the key to determining the pruning performance. In the proposed framework, the criterion is also an important factor in determining whether the subnetworks represent the current global model. We evaluate the effectiveness of the proposed framework under different criteria. These methods are briefly summarized as follows.

- Random. Parameters are randomly discarded.
- L1 [15]. Using the sum of the absolute values of a filter as a criterion:  $\mathcal{L}(W_i^l) = \sum |\mathcal{W}(i, :, :, :)|$ .
- L2 [15].  $\mathcal{L}(W_i^l) = \sum \|\mathcal{W}(i, :, :, :)\|_2$ .

- BN mask [45]. The  $\gamma$  of  $\hat{z} = \frac{z_{\text{in}} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$ ;  $z_{\text{out}} = \gamma \hat{z} + \beta$  in a BN layer is calculated as the corresponding filter's importance score, where  $z_{\text{in}}$  and  $z_{\text{out}}$  be the input and output,  $\mu_{\mathcal{B}}$  and  $\sigma_{\mathcal{B}}$  are the mean and standard deviation values of input activations over the current minibatch  $\mathcal{B}$ .
- Similarity. Compare the similarity between filters and remove one of them:  $D^{(l)} = \text{dist}(W_j^l, W_k^l), 0 \leq j \leq N_l, j \leq k \leq N_l$

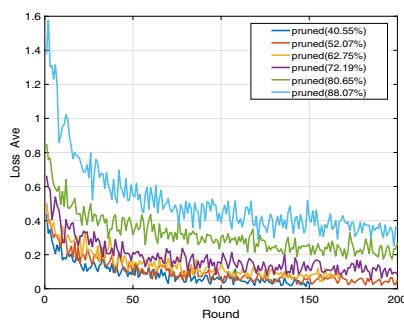
For the MNIST dataset, Fig. 6a and e are the average loss and test accuracy of different pruning criteria under a pruning rate of 40% on the IID data distribution; Fig. 6b and f are the average loss and test accuracy of different pruning criteria on the Non-IID data distribution with a pruning rate of 62%. As we can see, different pruning criteria have significant impacts on the convergence speed and the final obtained global model's performance. The pruning training converges faster than the full model training, and the global model obtained in the same number of rounds has higher accuracy. Among them, random pruning can quickly converge in the proposed framework regardless of IID and Non-IID data distributions, far exceeding other heuristic pruning criteria. Since the above pruning algorithms are proposed based on CNN, the small number of weights of a single neuron in fully connected network can easily lead to partial neuron inactivation (discussed in detail later).

For the CIFAR10 dataset, Fig. 6c and g are the average loss and test accuracy of different pruning methods under 50% pruning of FLOPs and parameters on the IID data distribution; Fig. 6d and h are the average loss and test accuracy of different pruning methods when the FLOPs and parameter pruning rate are 50% on the Non-IID data distribution. The pruning granularity in VGG is a convolution kernel. Except for BN mask and random, different pruning criteria have little effect on the convergence speed and performance, proving that the proposed framework is effective for large network structures. Unlike the fully connected neural network, the global model performance obtained by randomly selected convolutional kernels is not ideal and even diverges when the pruning rate increases. To further explore the relationship between convergence speed and subnet selection, we use the above pruning methods to obtain different subnets at the same time in each round and select the subnet with the best performance for delivery. It is experimentally demonstrated that the adapted method further accelerates the convergence speed, which provides a new idea for us to further accelerate the convergence speed.

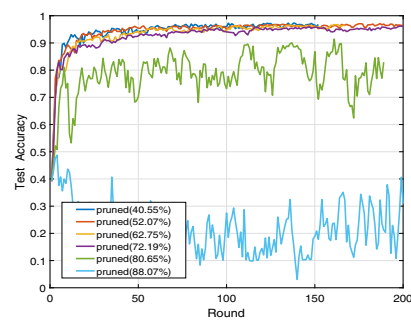
### The efficiency of computation and communication

Our proposed framework reduces computational and memory requirements for edge devices while improving com-

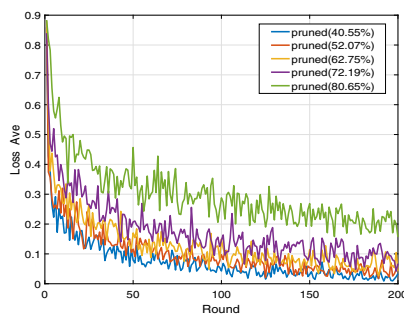
**Fig. 5** **a** and **b** Are the average loss and test accuracy of MNIST/IID under different pruning rates, **c** and **d** are of MNIST/Non-IID, **e** and **f** are of CIFAR10/Non-IID



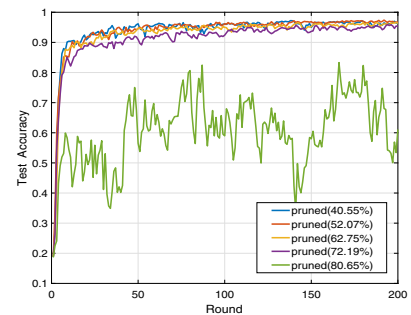
(a) Average loss of different pruning rate on MNIST/IID



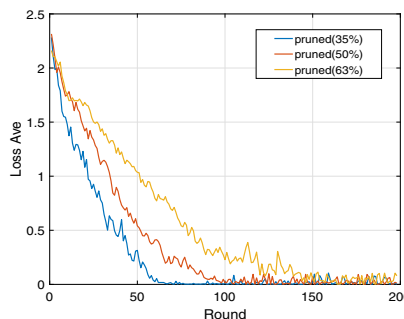
(b) Test accuracy of different pruning rate on MNIST/IID



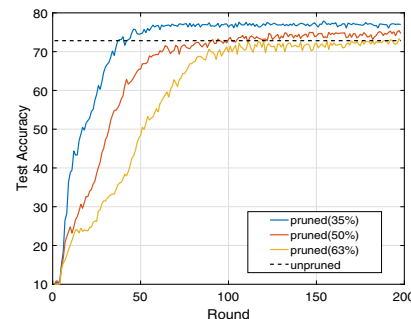
(c) Average loss of different pruning rate on MNIST/Non-IID



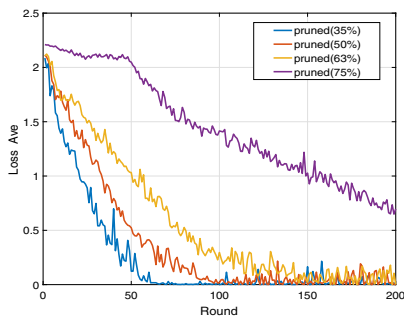
(d) Test accuracy of different pruning rate on MNIST/Non-IID



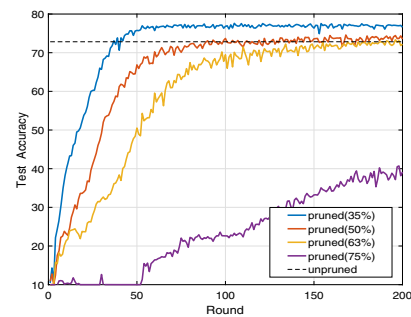
(e) Average loss of different pruning rate on CIFAR10/IID



(f) Test accuracy of different pruning rate on CIFAR10/IID

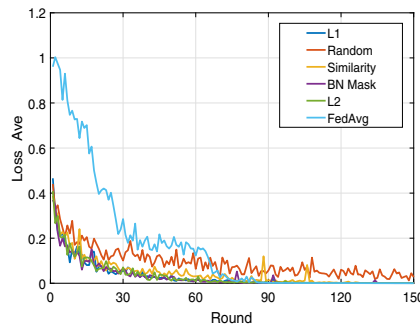


(g) Average loss of different pruning rate on CIFAR10/Non-IID

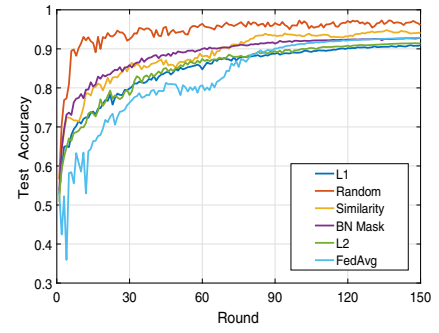


(h) Test accuracy of different pruning rate on CIFAR10/Non-IID

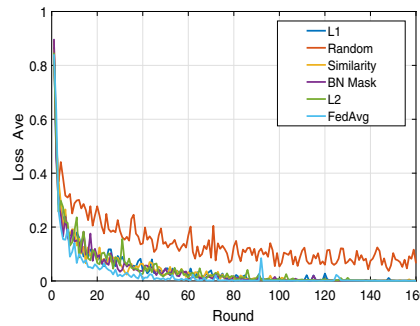
**Fig. 6** **a** and **e** Are the average loss and test accuracy of different pruning criteria under a pruning rate of 40% on MNIST/IID, **b** and **f** are at a pruning rate of 62% on MNIST/Non-IID, **c** and **g** are at a pruning rate of 50% on CIFAR10/IID, **d** and **h** are at a pruning rate of 50% on CIFAR10/Non-IID



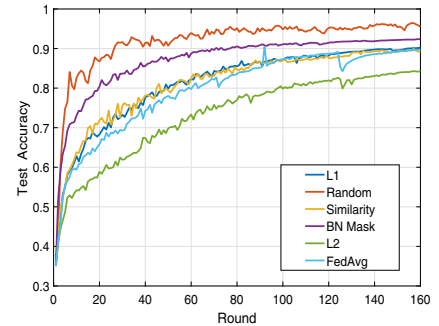
(a) Average loss of MNIST/IID



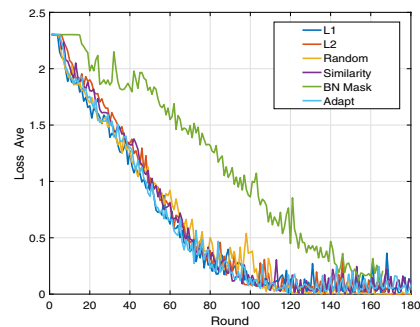
(b) Test acc of MNIST/IID



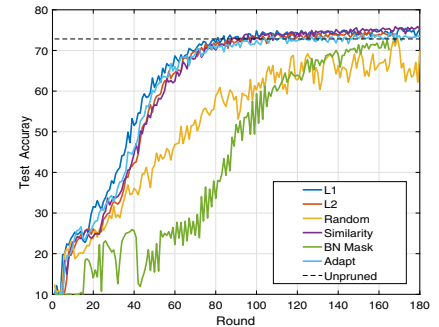
(c) Average loss of MNIST/Non-IID



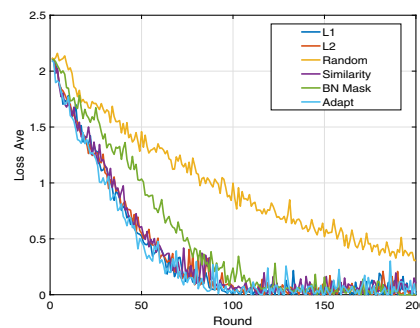
(d) Test acc of MNIST/Non-IID



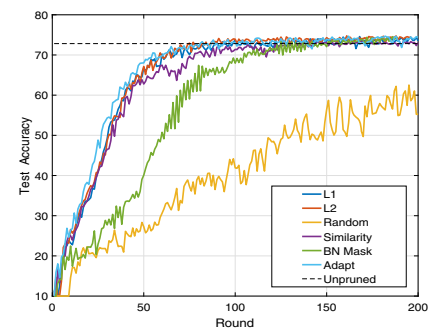
(e) Average loss of CIFAR10/IID



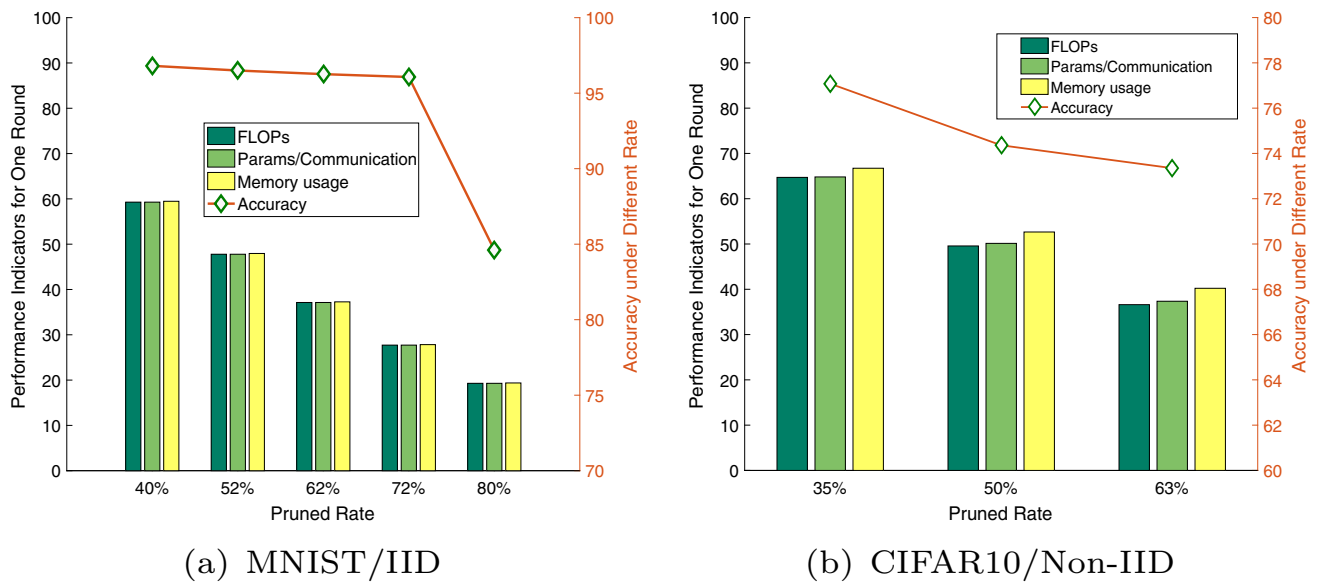
(f) Test acc of CIFAR10/IID



(g) Average loss of CIFAR10/Non-IID



(h) Test acc of CIFAR10/Non-IID



**Fig. 7** The computation, memory usage and communication of the client in a single round under different pruning rates. The line is the accuracy change of the global model. The baseline accuracy of VGG16 using FedAvg training on CIFAR10 is 72.83%, FC/MNIST is 93.39%

**Table 1** Communication cost under different pruning rates

FC/MNIST (Acc = 0.90)			VGG/CIFAR-10 (Acc = 0.70)		
Rate	Com	Speed	Rate	Com	Speed
Baseline	292.17M	1	Baseline	1562.6M	1
0.41	17.09M	17.10×	0.35	623.2M	2.51×
0.52	18.41M	15.87×	0.50	843.64M	1.85×
0.63	16.76M	17.43×	0.63	1369.45M	1.14×
0.72	31.48M	9.28×			
0.81	170.60M	1.71×			

munication efficiency. Finally, we compare the amount of computation and memory on the client under different pruning rates and the communication required to achieve a specific target accuracy. Similarly, to be closer to the actual scenario, we still choose to evaluate on the Non-IID data distribution. The computation, memory usage and communication of the client in a single round are shown in Fig. 7. The total communication amount required for different pruning rates to achieve the same target accuracy is shown in Table 1. Our proposed framework dramatically reduces the amount of computation and memory usage on the client at an accuracy exceeding that of full model training, and reduces the total amount of communication simultaneously. The larger network structure is more sensitive to pruning training, and computational reduction requires more communication rounds to compensate. The smaller network structure still maintains efficient communication while reducing the computation by 80%.

In addition, we compare with the current state-of-the-art federated pruning algorithms such as Federated dropout [46], PruneFL [11], Federated Pruning [12], AdaptCL [10]. Due to the different implementation methods and pruning strategies of each algorithm, AdaptCL performs pruning on the clients, while PruneFL uses an extended framework to support sparse matrix acceleration, so the advantages and disadvantages of the algorithm cannot be measured by the single-round communication time and client model training time. We compare the time taken by different algorithms to achieve the same accuracy at the same pruning rate, and the results are shown in the Table 2. It can be seen from the table that although some methods reduce the resource requirements of the client, they increase the overall training time. Federated dropout has the slowest convergence speed because it randomly prunes the network. PruneFL requires extended library support for fine-grained pruning and has limited acceleration effect. Federated Pruning is unable to rapidly converge because no reasonable parameter aggregation scheme is proposed. AdaptCL performs pruning on the clients, which has a great acceleration effect, but brings additional calculations to the client and increases the delay. The proposed framework significantly reduces the training time of the client and the up-down communication time, and the proposed parameter aggregation scheme ensures the stable convergence of the model and the optimal performance.

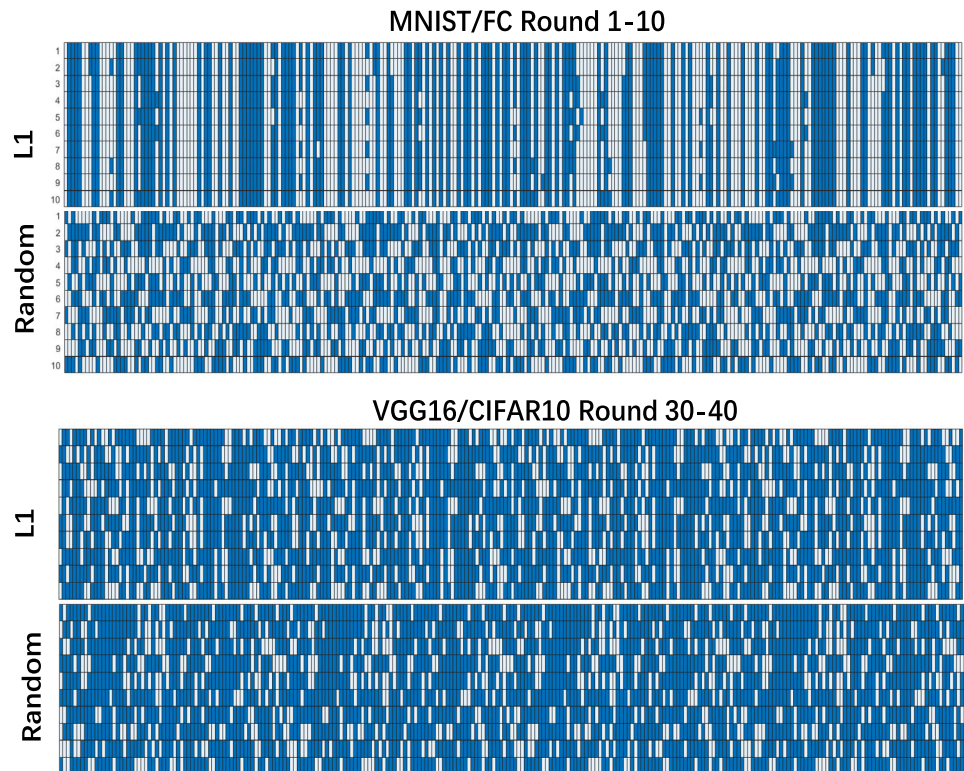
## Ablation study

Since neurons are selected according to their importance, a similar network structure will be selected for similar peri-

**Table 2** Comparison of the time taken by different algorithms to achieve the same accuracy under the same pruning rate

Method	VGG/CIFAR10 (Acc = 70%, Pruned = 50%)		FC/MNIST (Acc = 90%, Pruned = 70%)	
	IID	NonIID	IID	NonIID
FedAvg [6]	425	438	108	144
Fed Dropout [46]	569	432	280	267
PruneFL [11]	378	396	176	127
Fed Pruning [12]	386	419	138	156
AdaptCL [10]	325	317	84	76
Ours	242	210	46	51

**Fig. 8** Visualization of the convolution kernel in the pruned model, where white indicates pruned and blue indicates reserved. The top shows the pruning of the third layer on FC/MNIST. The bottom shows the pruning of the fifth layer on VGG/CIFAR10



ods, resulting in the problem of inactivation of neurons that have not been selected. In the fully connected network experiment, the number of weights of each neuron is small, which leads to errors in judging the importance of neurons based on a data-driven pruning algorithm. Moreover, each round of local training is one epoch, and the parameters vary very little, which makes the pruned network structure similar over several rounds.

To further explore the reasons for neuron inactivation, we visualized the neuron index of each round of pruning when the pruning rate was 50% on MNIST/FC, as shown in Fig. 8. The experiment in Fig. 6 shows that the first 10 rounds converge the fastest, so we analyze the pruning situation of the third fully connected layer (256 neurons) of the first 10 rounds of the pruned model. White indicates the pruned neurons, and blue indicates reserved neurons. It can be seen from the figure

that the model structure obtained by the pruning algorithm of calculating the L1 norm for the weight matrix of each neuron as its importance criterion is very similar. The error of the pruning criterion makes the federated learning fall into a local solution that trains only one subnet, resulting in the inactivation of other neurons in the network. However, random pruning jumps out of the local solution, so it showed better performance in the end.

We also visually analyze the pruning of the fifth layer of the convolution kernel (256 convolution kernels) in each round of pruning in the 30th through 40th rounds of VGG16/CIFAR10. From the experiment in Fig. 6, the L1-based pruning algorithm has the fastest convergence rate and higher accuracy, achieving the expected effect of pruning, which constantly seeks the optimal subnet in the adjustment and distribution of the subnet structure. However, random



pruning can eventually reach convergence after more rounds of training, but the convergence speed is significantly slower. From the visual comparative analysis of the pruning, we can see that the pruning algorithm has a greater impact on the performance of federated learning, and it easily falls into the local solution of a single structure when the pruning standard is not effective. Therefore, identifying the effectiveness of neurons in the training process will be the focus of future research.

## Conclusion

According to the problems of insufficient edge client computing resources and limited communication resources in the actual deployment of federated learning, this paper proposes an efficient federated training framework based on model pruning. We first discuss the problem of pruning position, then analyze the convergence of pruning-based federated learning, and finally explain the detailed process of the aggregation and update of the parameters of the entire framework. This paper applies model pruning to federated learning for the first time, and proposes the corresponding parameter update scheme to ensure the complete training of the model while maintaining the integrity of the learning information of each client. This framework greatly reduces the computational and memory requirements for local training while compressing uplink and downlink communication. Extensive experiments have verified the effectiveness of the framework.

Federated learning is currently in its infancy and many challenges remain. Although we greatly reduce the computing and memory requirements for resource-constrained devices in the federated system, in practical applications, the resource heterogeneity of devices, the withdrawal of participating devices at any time, and the dynamic unknown network environment will still bring about other delays and non-convergence of training. In the future work, we will continue to deeply combine model pruning with reinforcement learning to select reliable participating training devices in dynamic unknown network environment, and customize personalized models for devices with heterogeneous resources.

**Funding** This work were funded by National Key Research and Development Program of China (Grant no. 2017YFA0700300), National Natural Science Foundation of China (Grant no. 61903356) Liaoning Natural Science Foundation (Grant nos. 2021-MS-030, 2022JH6/100100013) and Independent project of State Key Laboratory of Robotics (Grant no. 2022-Z03).

## Declarations

**Conflict of interest** The authors declare no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as

long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Diedrichs AL, Bromberg F, Dujovne D, Brun-Laguna K, Watteyne T (2018) Prediction of frost events using machine learning and iot sensing devices. *IEEE Internet Things J* 5(6):4589–4597
2. Sezer OB, Dogdu E, Ozbayoglu AM (2017) Context-aware computing, learning, and big data in internet of things: a survey. *IEEE Internet Things J* 5(1):1–27
3. Xiao Z, Xu X, Xing H, Song F, Wang X, Zhao B (2021) A federated learning system with enhanced feature extraction for human activity recognition. *Knowl-Based Syst* 229:107338. <https://doi.org/10.1016/j.knosys.2021.107338>
4. Zhu H, Zhang H, Jin Y (2021) From federated learning to federated neural architecture search: a survey. *Compl Intell Syst* 7(2):639–657
5. Lin R, Xiao Y, Yang T-J, Zhao D, Xiong L, Motta G, Beaufays F (2022) Federated pruning: improving neural network efficiency with federated learning, vol 2022. Incheon, Korea, Republic of, pp 1701–1705. In: *Automatic speech recognition; client devices; deep learning; federated learning; federated pruning; large amounts; network efficiency; neural-networks; recognition models; speech data*. <https://doi.org/10.21437/Interspeech.2022-10787>
6. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: *Artificial intelligence and statistics*, PMLR, pp 1273–1282
7. You Z, Yan K, Ye J, Ma M, Wang P (2019) Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks, vol 32. Vancouver, BC, Canada, p. *Baseline models; iterative pruning; pruning algorithms; pruning methods; scaling factors; special operations; state of the art; Taylor expansions*
8. Lin M, Ji R, Wang Y, Zhang Y, Zhang B, Tian Y, Shao L (2020) Hrank: filter pruning using high-rank feature map. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 1529–1538
9. Liu S, Yu G, Yin R, Yuan J, Shen L, Liu C (2022) Joint model pruning and device selection for communication-efficient federated edge learning. *IEEE Trans Commun* 70(1):231–244. <https://doi.org/10.1109/TCOMM.2021.3124961>
10. Zhou, G., Xu, K., Li, Q., Liu, Y., & Zhao, Y. (2021) AdaptCL: Efficient Collaborative Learning with Dynamic and Adaptive Pruning. *arXiv preprint arXiv:2106.14126*
11. Jiang Y, Wang S, Valls V, Ko BJ, Lee W-H, Leung KK, Tassiulas L (2022) Model pruning enables efficient federated learning on edge devices. *IEEE Trans Neural Netw Learn Syst* 2022:1–13. <https://doi.org/10.1109/TNNLS.2022.3166101>
12. Lin R, Xiao Y, Yang T-J, Zhao D, Xiong L, Motta G, Beaufays F (2022) Federated pruning: improving neural network efficiency with federated learning. *arXiv:2209.06359*
13. Hanson S, Pratt L (1988) Comparing biases for minimal network construction with back-propagation. *Adv Neural Inf Process Syst* 1:177–185



14. Han S, Mao H, Dally WJ (2016) Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding, San Juan, Puerto Rico. In: *Complex neural networks; compression methods; dram memory; hardware resources; Huffman coding; loss of accuracy; mobile applications; storage requirements*
15. Li H, Samet H, Kadav A, Durdanovic I, Graf HP (2017) Pruning filters for efficient convnets. Toulon, France
16. Liu B, Wang M, Foroosh H, Tappen M, Pensky M (2015) Sparse convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 806–814
17. Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C (2017) Learning efficient convolutional networks through network slimming. In: *Proceedings of the IEEE international conference on computer vision*, pp 2736–2744
18. He Y, Zhang X, Sun J (2017) Channel pruning for accelerating very deep neural networks. In: *Proceedings of the IEEE international conference on computer vision*, pp 1389–1397
19. Molchanov P, Tyree S, Karras T, Aila T, Kautz J (2017) Pruning convolutional neural networks for resource efficient inference, Toulon, France. In: *Classification tasks; computationally efficient; convolutional kernel; gradient information; Kernel weight; network parameters; resource-efficient; Taylor expansions*
20. Lee N, Ajanthan T, Torr PHS (2019) Snip: Single-shot network pruning based on connection sensitivity, New Orleans, LA, United states. In: *Classification tasks; hyperparameters; iterative optimization; network pruning; new approaches; recurrent networks; reference network; sparse network*
21. He Y, Liu P, Wang Z, Hu Z, Yang Y (2019) Filter pruning via geometric median for deep convolutional neural networks acceleration. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 4340–4349
22. Guo S, Wang Y, Li Q, Yan J (2020) Dmcp: differentiable markov channel pruning for neural networks. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 1539–1547
23. Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. *Adv Neural Inf Process Syst* 28:25
24. He Y, Dong X, Kang G, Fu Y, Yan C, Yang Y (2019) Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE Trans Cybern* 50(8):3594–3604
25. Shen S, Li R, Zhao Z, Zhang H, Zhou Y (2021) Learning to prune in training via dynamic channel propagation. In: *2020 25th international conference on pattern recognition (ICPR)*, pp 939–945, IEEE
26. Tonello N, Gotta A, Nardini FM, Gadler D, Silvestri F (2021) Neural network quantization in federated learning at the edge. *Inf Sci* 575:417–436
27. Bernstein J, Wang Y-X, Azizzadenesheli K, Anandkumar A (2018) signsgd: compressed optimisation for non-convex problems. In: *International conference on machine learning*, pp 560–569, PMLR
28. Sattler F, Wiedemann S, Müller K-R, Samek W (2019) Robust and communication-efficient federated learning from non-iid data. *IEEE Trans Neural Netw Learn Syst* 31(9):3400–3413
29. Xu J, Du W, Jin Y, He W, Cheng R (2020) Ternary compression for communication-efficient federated learning. *IEEE Trans Neural Netw Learn Syst* 2020:2598
30. Wen W, Xu C, Yan F, Wu C, Wang Y, Chen Y, Li H (2017) Terngrad: ternary gradients to reduce communication in distributed deep learning, vol 2017. Long Beach, CA, United states, pp 1510–1520. In: *Accuracy loss; communication time; data parallelism; layer-wise; network communications; performance model; source codes*
31. Zhang H, Li J, Kara K, Alistarh D, Liu J, Zhang C (2017) Zipml: training linear models with end-to-end low precision, and a little bit of deep learning. In: *International conference on machine learning*, pp 4035–4043, PMLR
32. Wu J, Huang W, Huang J, Zhang T (2018) Error compensated quantized sgd and its applications to large-scale distributed optimization. In: *International conference on machine learning*, pp 5325–5333, PMLR
33. Magnússon S, Shokri-Ghadikolaei H, Li N (2020) On maintaining linear convergence of distributed learning and optimization under limited communication. *IEEE Trans Signal Process* 68:6101–6116
34. Mishchenko K, Gorbunov E, Takáč M, Richtárik P (2019) Distributed learning with compressed gradient differences. [arXiv:1901.09269](https://arxiv.org/abs/1901.09269)
35. Strom N (2015) Scalable distributed dnn training using commodity gpu cloud computing. In: *Sixteenth annual conference of the international speech communication association*
36. Dryden N, Moon T, Jacobs SA, Van Essen B (2016) Communication quantization for data-parallel training of deep neural networks. In: *2016 2nd workshop on machine learning in HPC environments (MLHPC)*, pp 1–8, IEEE
37. Aji AF, Heafield K (2017) Sparse communication for distributed gradient descent, Copenhagen, Denmark, pp 440–445. In: *Absolute values; convergence rates; gradient descent; machine translations; positively skewed; sparse matrices; speed up; stochastic gradient descent*. <https://dx.doi.org/10.18653/v1/d17-1045>
38. Chen C-Y, Choi J, Brand D, Agrawal A, Zhang W, Gopalakrishnan K (2018) Adacomp: adaptive residual gradient compression for data-parallel distributed training. In: *Proceedings of the AAAI conference on artificial intelligence*, vol 32
39. Xing H, Xiao Z, Qu R, Zhu Z, Zhao B (2022) An efficient federated distillation learning system for multitask time series classification. *IEEE Trans Instrum Meas* 71:1–12. <https://doi.org/10.1109/TIM.2022.3201203>
40. Zhuang Z, Tao H, Chen Y, Stojanovic V, Paszke W (2022) An optimal iterative learning control approach for linear systems with nonuniform trial lengths under input constraints. *IEEE Trans Syst Man Cybern Syst* 2022:25
41. Zhou C, Tao H, Chen Y, Stojanovic V, Paszke W (2022) Robust point-to-point iterative learning control for constrained systems: a minimum energy approach. *Int J Robust Nonlinear Control* 32(18):10139–10161. <https://doi.org/10.1002/rnc.6354>
42. Stojanovic V, Nedic N (2016) Joint state and parameter robust estimation of stochastic nonlinear systems. *Int J Robust Nonlinear Control* 26(14):3058–3074. <https://doi.org/10.1002/rnc.3490>
43. Bonawitz K, Eichner H, Grieskamp W, Huba D, Ingerman A, Ivanov V, Kiddon C, Konečný J, Mazzocchi S, McMahan HB, et al (2019) Towards federated learning at scale: system design. [arXiv:1902.01046](https://arxiv.org/abs/1902.01046)
44. Liu Z, Sun M, Zhou T, Huang G, Darrell T (2019) Rethinking the value of network pruning, New Orleans, LA, United states. In: *Large models; learning rates; low-resource settings; network pruning; parameterized model; pruning algorithms; pruning methods; state of the art*
45. Wen W, Wu C, Wang Y, Chen Y, Li H (2016) Learning structured sparsity in deep neural networks, Barcelona, Spain, pp 2082–2090. In: *Classification accuracy; compact structures; computation costs; computation resources; high demand; resource constrained devices; structured sparsities*
46. Wen D, Jeon K-J, Huang K (2022) Federated dropout a simple approach for enabling federated learning on resource constrained devices. *IEEE Wirel Commun Lett* 11(5):923–927. <https://doi.org/10.1109/LWC.2022.3149783>