electronics

MDPI

Article

Personalized Federated Learning Incorporating Adaptive Model Pruning at the Edge

Yueying Zhou [1,2], Gaoxiang Duan [1,2], Tianchen Qiu [1,2], Lin Zhang [3], Li Tian [1,2,*], Xiaoying Zheng [1,2] and Yongxin Zhu [1,2,*]

[1] Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai 201210, China; zhouyueying@sari.ac.cn (Y.Z.); duangx@sari.ac.cn (G.D.); qiutc@sari.ac.cn (T.Q.); zhengxy@sari.ac.cn (X.Z.)
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] Shanghai Nuclear Engineering Research & Design Institute Co., Ltd., Shanghai 200030, China; zhangl@snerdi.com.cn
* Correspondence: tianl@sari.ac.cn (L.T.); zhuyongxin@sari.ac.cn (Y.Z.)

Abstract: Edge devices employing federated learning encounter several obstacles, including (1) the non-independent and identically distributed (Non-IID) nature of client data, (2) limitations due to communication bottlenecks, and (3) constraints on computational resources. To surmount the Non-IID data challenge, personalized federated learning has been introduced, which involves training tailored networks at the edge; nevertheless, these methods often exhibit inconsistency in performance. In response to these concerns, a novel framework for personalized federated learning that incorporates adaptive pruning of edge-side data is proposed in this paper. This approach, through a two-staged pruning process, creates customized models while ensuring strong generalization capabilities. Concurrently, by utilizing sparse models, it significantly condenses the model parameters, markedly diminishing both the computational burden and communication overhead on edge nodes. This method achieves a remarkable compression ratio of 3.7% on the Non-IID dataset FEMNIST, with the training accuracy remaining nearly unaffected. Furthermore, the total training duration is reduced by 46.4% when compared with the standard baseline method.

Keywords: federated learning; personalized models; adaptive model pruning; Non-IID; sparse models

Citation: Zhou, Y.; Duan, G.; Qiu, T.; Zhang, L.; Tian, L.; Zheng, X.; Zhu, Y. Personalized Federated Learning Incorporating Adaptive Model Pruning at the Edge. *Electronics* **2024**, *13*, 1738. https://doi.org/10.3390/electronics13091738

Academic Editors: Shahab Tayeb, Jimmy Ming-Tai Wu and Matin Pirouz

Received: 5 March 2024
Revised: 23 April 2024
Accepted: 25 April 2024
Published: 1 May 2024

1. Introduction

Privacy protection, shortened user query times, and personalized services are driving the deployment of neural networks from central servers to edge devices. Federated learning (FL) is a distributed machine learning method that operates under the premise of privacy protection [1]. However, conventional federated learning performs well only under the assumption of independent and identically distributed (IID) data, showing poor performance for Non-IID data [2]. Furthermore, the increasing complexity of deep neural network structures poses bottlenecks for device computing power and communication, making it challenging to migrate to edge computing nodes [3]. One-shot pruning, which significantly reduces model parameters during initial training, allows the model to be suitable for resource-constrained computing nodes. The combination of model pruning and personalized learning streamlines network structures, improves computational efficiency through sparse operations, reduces memory usage, and better caters to the lightweight deployment needs of edge devices like smartphones.

While there is considerable research on federated learning and model pruning individually, the integration of the two is limited. Current methods primarily focus on pruning the global model without addressing Non-IID issues in practical applications. Existing personalized learning approaches, such as clustering [4–6], knowledge distillation [7–9], and meta-learning [10,11], incur additional computational costs and still exhibit unstable

performance in resolving Non-IID problems. Existing pruning methods [12–14] need training to identify critical structures, providing no advantage in swiftly discovering winning lottery tickets [15].

To address Non-IID, computational, and communication bottleneck challenges, this paper proposes a personalized federated learning method with efficient adaptive pruning based on edge-side data. The model's training and pruning are divided into two stages: initialization and adaptive pruning. The key innovations and contributions in this paper include the following:

- Proposing and implementing a federated learning framework based on personalized pruning. Existing research combining federated learning and model pruning tends to focus on pruning the global model on the server, with insufficient discussion on the heterogeneity of client-side models. This framework uses heterogeneous models to address Non-IID problems while achieving good generalization effects;
- Introducing a mask decision algorithm based on internal layer characteristics, enabling the fastest discovery of winning lottery tickets and reducing the time cost of one-shot pruning in the initialization stage to one-tenth of the baseline method [16];
- Utilizing sparse models for effective parameter compression, significantly reducing the computation and communication overhead of edge nodes. With almost unchanged model accuracy, our method achieves a model compression ratio far below the baseline method (3.8%) and a 46.4% improvement in training speed compared to the baseline method [16].

2. Related Works

2.1. Personalized Federated Learning

FedAvg, proposed by McMahan et al. [17] in 2017, has been the prevailing approach in federated learning. However, experimental findings suggest that data heterogeneity notably hampers the convergence speed and accuracy of the FedAvg model, particularly in the case of highly imbalanced Non-IID datasets [18]. Non-IID distributions may involve label skew, feature skew, or quality skew [19]. Currently, there has been considerable research on addressing this heterogeneity in training data, as discussed in papers [4–11,20–22]. Ma et al. [2] outlined various methods for tackling the Non-IID problem, including data-based methods [20] such as data sharing, enhancement, and selection; model-based methods [21,22] such as redesigning model update and aggregation as well as various optimization methods; algorithm-based methods [10,11] such as meta-learning, multi-task learning, and lifelong learning; and framework-based methods [4–9] such as clustering learning, knowledge distillation, and Base + Personalization layers.

To solve the Non-IID problem, Chen et al. [20] proposed representation augmentation algorithm FRAug to perform client-personalized augmentation in the embedding space to improve the training robustness against feature distribution shift. Vahidian et al. [23] proposed an approach for a personalized global model to address the challenges posed by heterogeneous and Non-IID data distributions. Ghari et al. [24] proposed a personalized federated learning approach based on the random feature (RF) approximation for multi-kernel learning and mathematically demonstrated its effectiveness in handling heterogeneous data.

However, personalized federated learning still faces challenges such as performance instability [6] (most of the time, personalized model performance still lags behind the global model) and higher computational complexity.

2.2. Model Pruning

The lottery ticket hypothesis (LTH) proposed by Jonathan et al. [15] states that a randomly initialized dense neural network contains a highly sparse subnetwork, known as winning tickets, capable of achieving superior performance when trained independently. These winning tickets can be unearthed through iterative or one-shot pruning methods, where parameters with the smallest magnitudes are systematically removed [25]. Since its

inception, LTH has garnered significant attention and been extensively studied [12,25–27]. Zhang et al. [12] not only employed dynamical systems theory and inertial manifold theory to theoretically substantiate the efficacy of the lottery ticket hypothesis, but also proposed a practical, lossless pruning solution. In order to reduce the time consumption of the iterative train–prune–retrain process associated with LTH, researchers have explored methods to identify winning tickets early in the training process. You et al. [26] introduced Early-Bird (EB) tickets in DNNs, which emerge very early in training, and used mask distance to draw EB tickets. Building upon the EB strategy, Kim et al. [27] introduced Early-Time (ET) tickets in SNNs and established a stopping criterion for training based on the evaluation of KL divergence between class prediction distributions at various time intervals.

There are also filter pruning [13] and one-shot pruning methods [28]. As for methods that combine model pruning and federated learning, Jiang et al. [16] introduced PruneFL, a two-stage pruning framework combining with federated learning to achieve efficient training. Huang et al. [29] proposed FedTiny, which incorporates adaptive batch normalization (BN) selection and progressive pruning to mitigate bias in coarse pruning and to reduce memory consumption in finer pruning. This method selects a less biased initial coarse-pruned model by comparing BN measurements updated from clients, while our approach achieves global mask determination with minimal computational cost. Jiang et al. [30] introduced the Complement Sparsification (CS) algorithm, involving collaborative pruning at both server and clients. While similar to our approach in Section 3.3, we utilize local complementary aggregation for personalized model generation. Additionally, Deng et al. presented TailorFL [31], tailoring personalized submodels considering resource capability and local data distribution to enhance personalization performance amid system and data heterogeneity.

This method focuses more on achieving performance on heterogeneous devices, while our approach emphasizes balancing model accuracy and training time.

In summary, the current approach can also be optimized in the following aspects:

- Pruning methods designed to accelerate training speed often sacrifice significant accuracy, failing to strike a balance between speed and precision;
- Simulations based on a single-machine approach to federated learning may not accurately reflect the actual performance, disregarding communication costs in distributed execution environments. This discrepancy between simulated results and real-world performance could be substantial;
- Conventional pruning methods find it challenging to protect crucial weights (winning ticket), while LTH-based pruning methods incur significant computational costs, failing to strike a balance between computational costs and pruning effectiveness.

3. Personalized Federated Learning Algorithms with Edge-Side Pruning

To address the aforementioned challenges in practical federated learning, this section introduces a personalized federated learning framework with edge-side pruning. Through pruning on the edge, local model structures that align with the features of the local data are obtained, thereby meeting the personalized requirements. Additionally, decision-making from personalized models assists the central server in filtering out more important features extracted from convolutional networks, thereby addressing the performance degradation issue caused by Non-IID data.

3.1. Preliminaries

In the overall algorithm flow, federated learning is primarily divided into two stages: the initialization stage and the adaptive pruning stage. This two-stage pruning framework is derived from the work of Jiang et al. [16]. The process begins with the initialization phase, where clients perform one-shot pruning based on local feature data. The server then uses the pruning results to decide on potential winning lottery tickets. Subsequently, in the adaptive pruning stage, local clients adjust the pruning based on training outcomes,

either removing or adding back some weights until a certain level of balance is achieved. The main operational process of the system is illustrated in Figure 1.

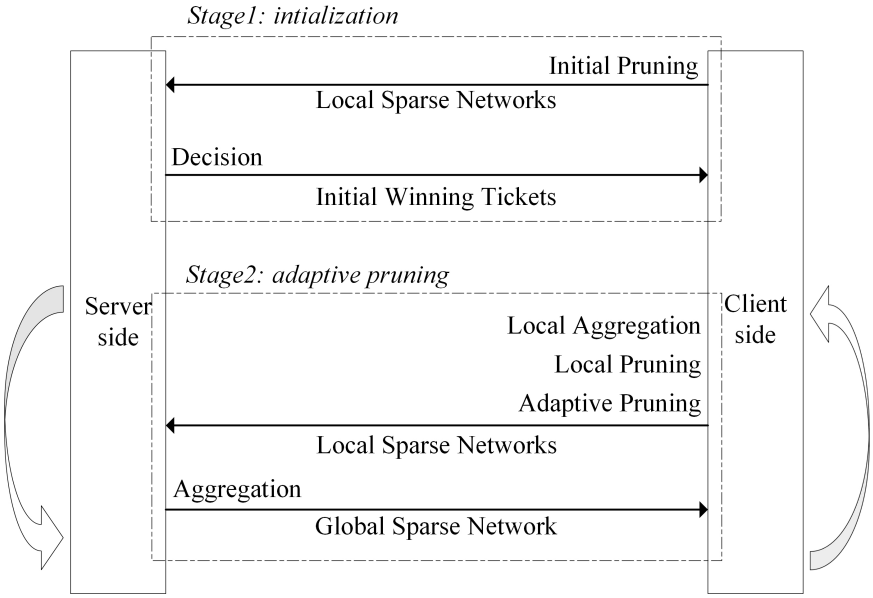

Figure 1. Overall workflow.

The parameter symbols and their corresponding meanings used in this system are listed in Table 1. In this context, the client c_n trains a local model θ_n on the local dataset D_n, and its model parameters can be equivalently expressed as $\omega_n(k) \odot m_n(k)$. Here, $m_n(k)$ represents a boolean tensor with the same shape as the weights in a given layer. The acquisition and utilization of $m_n(k)$ is a focal point in this section.

Table 1. Preliminaries.

Notation	Definition
$\odot$	Hadamard product
$\oplus$	binary OR operation
n, N	client index, total number of clients
k, K	global training round, global epochs
θ_n	local model for client n
$\theta_s(k)$	server model to client in k-th iteration
D_n	local dataset for client n
$\omega_n(k)$	param. weight for client n in k-th iteration
$m_n(k)$	param. mask for client n in k-th iteration
p_n	proportion of each client in the aggregation
$thres$	mask/weight filtering threshold
$g_n(k)$	param. gradient for client n in k-th iteration
$pos(g)$	index of a particular gradient
$density(m, \theta)$	function to calculate sparse model density
$L(\omega)$	loss function

3.2. Adaptive Pruning for Local Training

In the federated learning process, the global model is aggregated from individual client models, and the computation of the global model follows the principles outlined by the following formula:

$$\theta_s = \arg\min_{m_n,\omega_n} \frac{1}{n} \sum_n p_n L(\omega_n) \tag{1}$$

The performance of the global model is closely related to the performance of client models. According to Equation (1), it can be assumed that, when the loss of each client decreases, the aggregated global model loss will also decrease. Model pruning can be employed to facilitate the rapid convergence of the model, necessitating the identification of a suitable criterion for model pruning. One commonly used method involves performing a Taylor expansion on the loss function $L(\omega)$:

$$\begin{aligned} \delta L &= L(\omega_n(k+1)) - L(\omega_n(k)) \\ &= \sum_i g_i \delta\omega_i + \sum_{i,j} \frac{1}{2} H_{i,j} \delta\omega_i \delta\omega_j + O(\omega^3) \end{aligned} \tag{2}$$

$$g_i = \frac{\partial L}{\partial \omega_i}, H_{i,j} = \frac{\partial^2 L}{\partial \omega_i \partial \omega_j} \tag{3}$$

In deep learning, it is impractical to compute the Hessian matrix $H_{i,j}$ in each training cycle due to the computational requirements ($O(N_\omega^2)$ complexity, where N refers to parameter size). To simplify computations, the focus is primarily on the first term in Equation (2). During model pruning, $\delta\omega_i$ represents all the weights in the model and can be expressed using the following formula:

$$\delta\omega_i = \alpha_k g_n(k+1) \odot m_n(k+1) \tag{4}$$

Therefore, it can be deduced that $\delta L \approx \sum_i g_i \delta\omega_i = \alpha_k \sum_i g_i^2(k)$. Therefore, we can conclude that the rate of loss reduction in the model is primarily correlated with the gradient information g_i, and, the larger the retained g_i, the more quickly the loss decreases. This establishes the criteria in model pruning, determining whether a particular weight should be pruned in the current pruning iteration.

According to the lottery ticket hypothesis [15], there exists a sparse subnetwork for every model, and training this subnetwork does not compromise on time and accuracy compared to the complete model. To quickly identify the winning lottery ticket in the initialization stage, initial pruning is performed on the client side. By horizontally comparing the masks uploaded by various clients to the server, not only can the "winning tickets" be more accurately filtered out, but the process is also faster. The initialization stage workflow is shown in Figure 2.

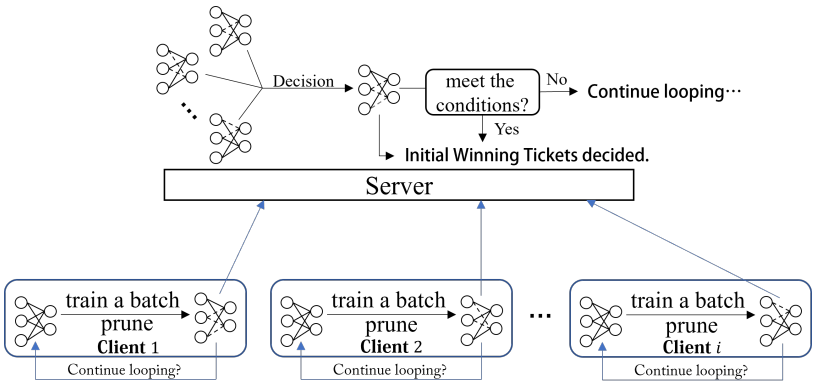

Figure 2. Initialization stage.

During the initialization phase, clients perform initial pruning on their local data. This step immediately removes some redundant structures from the model, compressing the model size. Starting training with a lightweight model from the beginning can effectively reduce the time and computational costs of training. This process relies entirely on user data on the client side, making it personalized from the start. After initial pruning, each client sends the pruned weights and masks to the server. The server then decides the lottery ticket network and sends it back to the clients.

Various methods were employed to decide the global sparse network. One simple decision method is to set thresholds for parameters. For a particular weight, if the number of clients retaining that weight exceeds the threshold, the weight is retained. The calculation method is shown in the following formula:

$$m_s = (\sum_n m_n) > thres_{mask} \tag{5}$$

In addition to this method, an approach based on the distinctive characteristics of each layer in the model is employed. Different methods are designed for selecting the lottery ticket network at each layer. The process is illustrated in Algorithm 1. This method not only safeguards specific weights, such as those in the input and output layers, but also accelerates the identification of the lottery tickets, typically within three pruning rounds. The Layerwise Pruning algorithm prunes really quickly: if the algorithm loops for over five iterations, the model will become too sparse to train, so it is appropriate to set $thres_{density}$ close to the density of the model after the algorithm loops for two or three iterations.

Algorithm 1 Layerwise Pruning

$terminate = False$
while not terminate **do**
  clients do local pruning using Algorithm 2
  clients send mask $m_n(k)$ to the server
  for $m_{n,layer}$ in $m_n(k)$ **do**
    if *layer* is input layer or output layer: **then**
      $m_{s,layer} = (\sum_n m_{n,layer}) > thres_{mask}$
    else
      $m_{s,layer} = m_{1,layer}$
      for n=2 to N **do**
        $m_{s,layer} = m_{s,layer} \odot m_{n,layer}$
      end for
    end if
  end for
  $density(m_s; \theta_s)$ =remained nodes/all nodes
  $terminate = density(m_s; \theta_s) < thres_{density}$
end while
return m_s

Upon entering the adaptive pruning stage, a small sparse model is obtained on the client side. For clarity, in the pruning process, we refer to the finest granularity of pruning as "nodes." Here, the deployed local pruning method is a combination of magnitude pruning and first-order pruning. From the previous derivation (Equation (1)–(4)), it can be deduced that $\delta L \approx \alpha_k \sum_i g_i^2(k)$. The sum of squared gradient value g_i^2 is a direct influencing factor for rapidly reducing loss. Therefore, a three-step local pruning method is introduced, as is shown in Figure 3: Firstly, the nodes with absolute values smaller than $thres_{weight}$ are pruned, while protection is initially applied to nodes with larger absolute weight values; Secondly, the formerly pruned nodes and the newly pruned nodes together make the selection set; Thirdly, recover those nodes with relatively high gradients in the selection set. By doing this, a balance has been achieved between considering magnitude and

squared gradients. The local adaptive pruning algorithm workflow is primarily outlined in Algorithm 2.

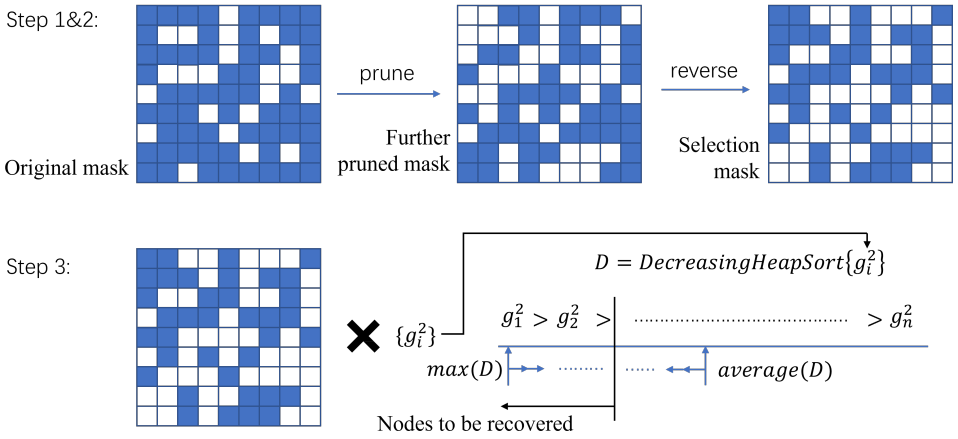

Figure 3. Local pruning algorithm.

Algorithm 2 Local Pruning

$\hat{m}_n(k) = \bar{m}_n(k) \oplus (\omega_n(k) < thres_{weight})$
$m_n(k) = \omega_n(k) \geq thres_{weight}$
Selection Set $\mathcal{G} = \{g_j^2 \mid g_j \text{ in } g_n(k) \odot \hat{m}_n(k)\}$
$\mathcal{D} = DecreasingHeapSort(\mathcal{G})$
$comp = average(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} g_j^2$
$\mathcal{G} = \mathcal{D}$
for g_j^2 in $\mathcal{G}$: **do**
  if $g_j^2 > comp$: **then**
    $\hat{m}_n(k, pos(g_j^2)) = 1$
    $\mathcal{D} \cup g_j^2$
    $comp = average(\mathcal{D})$
  else
    $\hat{m}_n(k, pos(g_j^2)) = m_n(k, pos(g_j^2))$
  end if
end for
$m_n(k+1) = \hat{m}_n(k)$
return $m_n(k+1)$

Therefore, in each subsequent pruning iteration, the adaptive pruning algorithm will either remove or reintroduce some nodes based on their importance. The selection is made among nodes with small absolute weights and nodes that have already been pruned. In this process, to avoid mistakenly pruning important nodes, pruned nodes with higher g_i^2 may be reintroduced during the iteration. This approach makes it easier to recover winning tickets when they are mistakenly pruned during the process, thus mitigating any potential impact on model performance.

Because the gradient distribution of the model is unpredictable, it is difficult to decide on a threshold for filtering squared gradients g_i^2 beforehand. Therefore, employing an algorithm that can autonomously determine a suitable threshold for squared gradients is advantageous. In the function for local pruning, the set $\mathcal{D}$ initially equals the union of all the squared gradients of the entire selection set. As $max(\mathcal{D})$ is repeatedly included back into $\mathcal{D}$, the average of $\mathcal{D}$ becomes larger. Meanwhile, $max(\mathcal{D})$ gradually decreases in descending order until they converge at a critical point. Through this approach, there is no need to set a threshold manually, as the function will automatically determine the threshold.

By combining the initial pruning described above with subsequent adaptive pruning methods, the model's size can be significantly reduced, its training speed can be accelerated, and a dynamic balance in the pruning process can be achieved after several rounds.

3.3. Federated Learning Method based on Adaptive Pruning

In traditional federated learning approaches, both server and client models share the same model structure. While this facilitates model aggregation, it compromises personalized features. To address the need for personalized models and cope with diverse client data, individualized pruning is applied at each client during the federated learning process. Each client submits personalized sparse parameters to the server, as illustrated in the workflow in Figure 4.

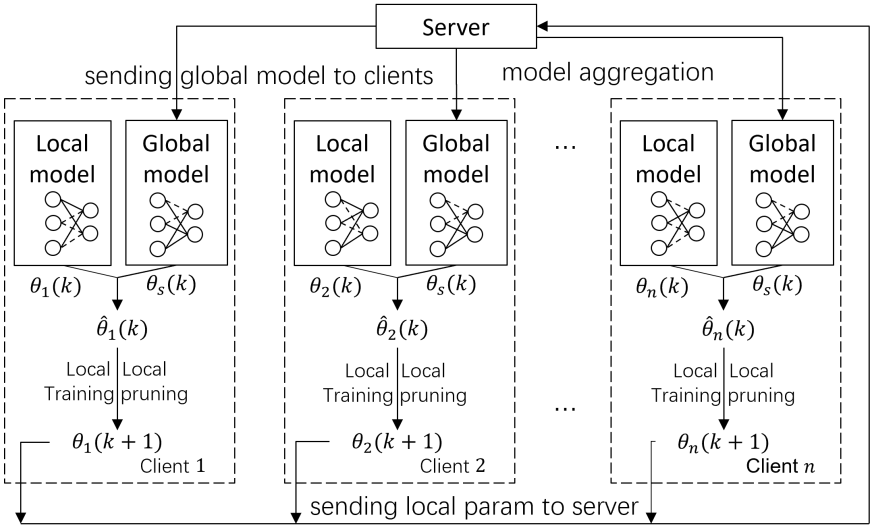

Figure 4. Personalized federated learning workflow.

The method involves the following steps:

Step 1: After receiving models uploaded by clients, the server aggregates personalized parameters from each model and sends the aggregated result, the global model, to all clients. The aggregation process can be represented by Algorithm 3. Generally, $thres_{mask}$ is chosen to be around N/2;

Step 2: Clients receive the global model and load it into local caches. Due to the adoption of personalized pruning strategies, the sparse structure of local models differs from the global model. To address the issue of merging parameters with different sparse structures, a set of dense tensors is cached locally. These tensors are used to receive and merge local parameters with global model parameters, as illustrated in the following formula:

$$\omega_{local} = \omega_s(k) \odot m_s(k) + \omega_n(k) \odot \bar{m}_s(k) \tag{6}$$

With this arrangement, the client would use the updated global model $\omega_s(k)$ for the weights included in both the global and local masks (i.e., in $m_s(k) \odot m_n(k)$), while retaining the previous local model for the remaining of the locally significant weights (in $\bar{m}_s(k) \odot m_n(k)$);

Step 3: After receiving global parameters from the server, local training is conducted, using the merged tensors to obtain new model parameters, which can be represented as $\omega_n(k+1) = LocalTrain(\omega_{local} \odot m_n(k))$. In a pruning round, the pruning algorithm from Algorithm 3 is executed after local training. Usually, the pruning frequency is set to once per 50 or 100 local training epochs. After obtaining the resulting new mask $m_n(k+1)$, multiply it with the model parameters to obtain $\omega_n(k+1) \odot m_n(k+1)$, which is then sent to the server.

Algorithm 3 Global Merge Algorithm

for n = 1 to N **do**
  $w_n(k) = \omega_n(k) \odot m_n(k) + \omega_s(k) \odot \bar{m}_n(k)$
  $\Delta\omega_n(k) = w_n(k) - \omega_s(k)$
end for
$\Delta\varpi = \sum_n p_n \Delta\omega_n(k)$
$\omega_s(k+1) = \omega_s(k) + \Delta\varpi$
$m_s(k+1) = (\sum_n m_n) > thres_{mask}$
return $\omega_s(k+1) \odot m_s(k+1)$

This approach enables personalized pruning federated learning with different sparse model structures on each client. While achieving the flexibility to aggregate various model parameter shapes, this method also increases the computational requirements on local devices.

4. Experimental Results

All experiments were conducted using one central server (NVIDIA Tesla V100-SXM2 GPU) and five client devices (NVIDIA GeForce RTX 2080 GPU). Both the central server and client CPUs were Intel(R) Xeon(R) Gold 6230 CPUs @ 2.10 GHz, with 80 cores. The experiments utilized the FEMNIST Non-IID dataset for practical testing in federated learning. The dataset consists of 193 different handwriting fonts, each from a different writer, with 28×28 images per writer categorized into 62 classes. The 193 writers were divided into 5 groups based on the number of clients (the first 4 groups with 38 writers each, and the 5th group with 41 writers), forming a Non-IID training set. $thres_{mask}$ is set to 4 in initial pruning and 3 in adaptive pruning; $thres_{weight}$ is set to prune 30% of the nodes.

We employed a two-layer convolutional network (conv2) for adaptive pruning experiments. The control groups included the PruneFL Prototype method provided by Jiang et al, which is part of the work in paper [16], and the conv2 model with only initial pruning. Due to the current lack of robust support for sparse computation on GPUs, the entire pruning process was primarily conducted in a CPU environment. To simplify, we later refer to the PruneFL prototype algorithm as simply 'PruneFL'.

4.1. Pruning Structural Evaluation

To assess the preservation of pruned nodes across various client models, we take a specific layer of the model as an example. We visualize the model's mask by rearranging the parameters into a two-dimensional representation, creating a binary image based on their 0–1 distribution. In the visualizations, we observe significant differences in the masks formed after pruning on different clients due to the diverse distributions of client data. Conversely, similar data distributions lead to more analogous masks.

As shown in Figure 5, masks in Figure 5a,b are the results of pruning based on data from two different writers. After the same pruning rounds, Figure 5a achieves a 4.7% node retention rate, while Figure 5b has only 12.2%. Individual writers exhibit stronger personalized features, resulting in significant differences in the generated masks. On the other hand, Figure 5c,d are pruned based on two sets of non-overlapping data from multiple writers. As the personalized features are less distinct, the generated masks are more similar (node retention rates of 8.4% and 8.6%, respectively).

Furthermore, although pruning based on a single client (PruneFL) has some effect, it is not effective in filtering out the most important lottery tickets, as shown in Figure 6. The pruning algorithm based on the mean of gradients is more prone to cutting off points with outlier small gradient values. Pruning on a single client can achieve relatively low pruning ratios after multiple iterations, but it does not effectively eliminate scattered noise points with high gradient values in the mask. These noise points are mostly generated by randomness and correspond to unimportant textures. These weights not only interfere with the model's primary tasks but also consume computational resources. By integrating mask information from various clients, it becomes easier to discern the model's effective

structure. The masks generated by the Layerwise Pruning method are shown in Figure 7. It can be observed that this method has essentially removed all the noise points in the mask, further reducing model complexity.

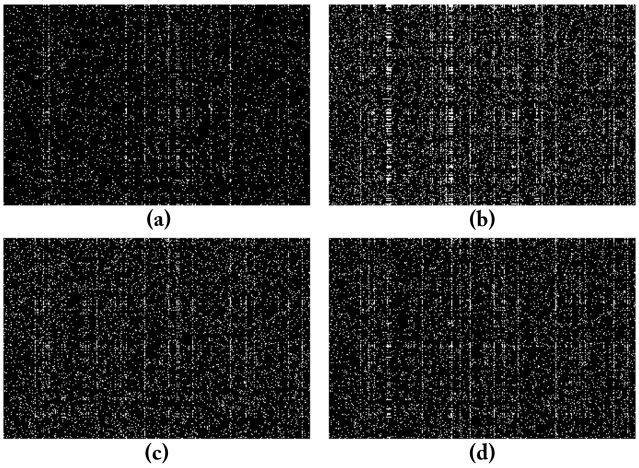

Figure 5. Comparison of mask binary graph from different clients. (**a**,**b**) are pruned based on data from two different writers. (**c**,**d**) are pruned based on two sets of non-overlapping data from multiple writers.

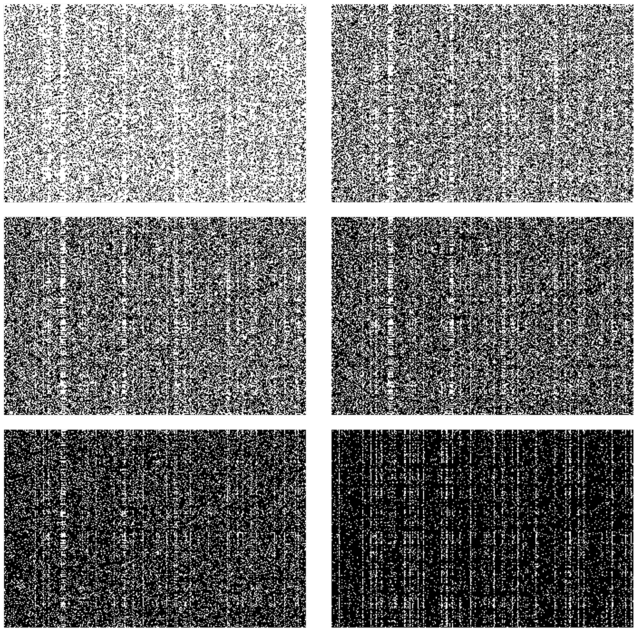

Figure 6. Ineffective filtering with pruning on a single client.

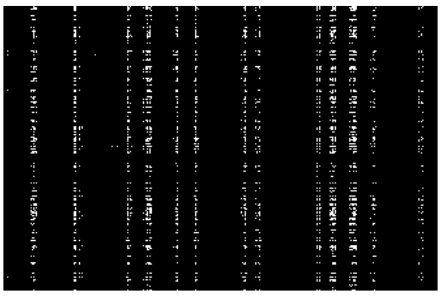

Figure 7. Mask by Layerwise Pruning.

4.2. Pruning Performance Evaluation

The model sizes and initial pruning times resulting from different pruning schemes are presented in Table 2. The variation in model training accuracy across training epochs is depicted in Figure 8, while Figures 9 and 10 illustrate the changes in model size and training time over the course of epochs, respectively. "Conventional" here represents a method that undergoes only initial pruning without subsequent iterative pruning. Table 3 provides a performance comparison of various federated learning algorithms that combine model pruning, while Figure 11 contrasts the stability of performance among three adaptive pruning methods.

Table 2. Results of various pruning schemes.

Method	Rounds	Retain Pct.	Time Cost	Final Pct.
PruneFL [16]	16	8.6%	306.2 s	13.4%
Threshold Pruning (ours)	1	71.7%	39.7 s	19.1%
Layerwise Pruning (ours)	2	15.7%	72.3 s	3.7%

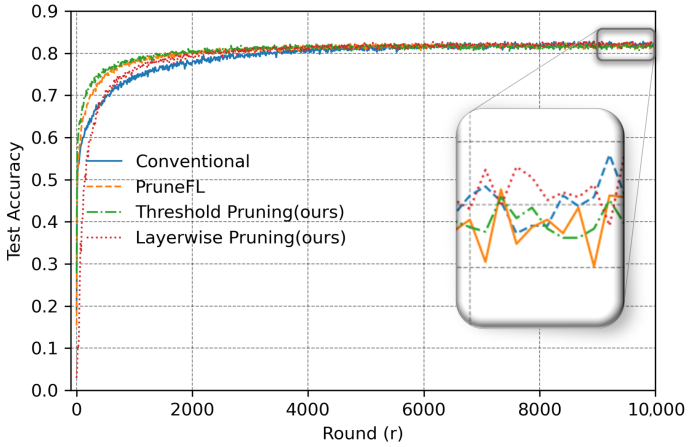

Figure 8. Model accuracy vs. global epochs.

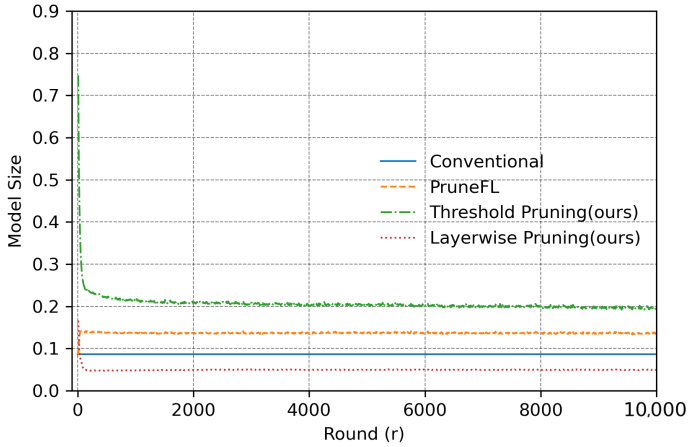

Figure 9. Model size vs. global epochs.

From Figure 8, it can be observed that, even with different initial pruning strategies, the final models achieve similar levels of accuracy. The Threshold Pruning method has the highest proportion of initially pruned and retained nodes (71.7% and 19.1%, respectively), exhibiting the fastest increase in model accuracy across epochs. However, as indicated in Figure 10, it also has the slowest training speed. While Layerwise Pruning retains a slightly higher proportion of nodes in the initial pruning phase compared to the PruneFL

method (15.7% vs. 8.6%), it demonstrates the highest efficiency in initial pruning. Layerwise Pruning achieves 15.7% in only 2 pruning rounds, as opposed to PruneFL, which requires 16 rounds to reach 8.6%. Moreover, the final proportion of retained nodes in Layerwise Pruning is significantly lower than in PruneFL (3.7% vs. 13.4%). Furthermore, Layerwise Pruning exhibits the second-fastest increase in model accuracy across epochs and the fastest training speed among the three adaptive pruning methods.

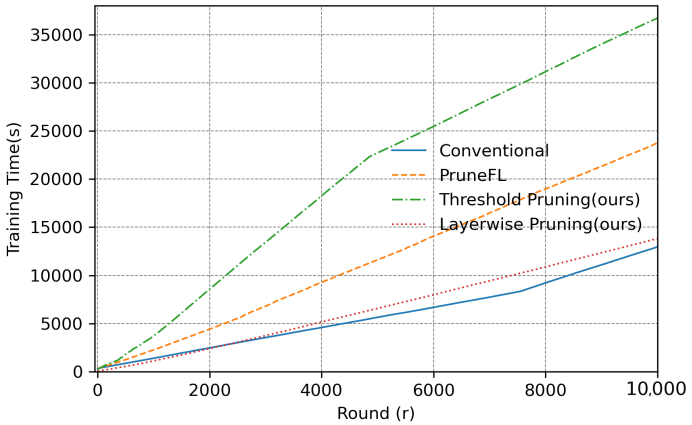

Figure 10. Training time vs. global epochs.

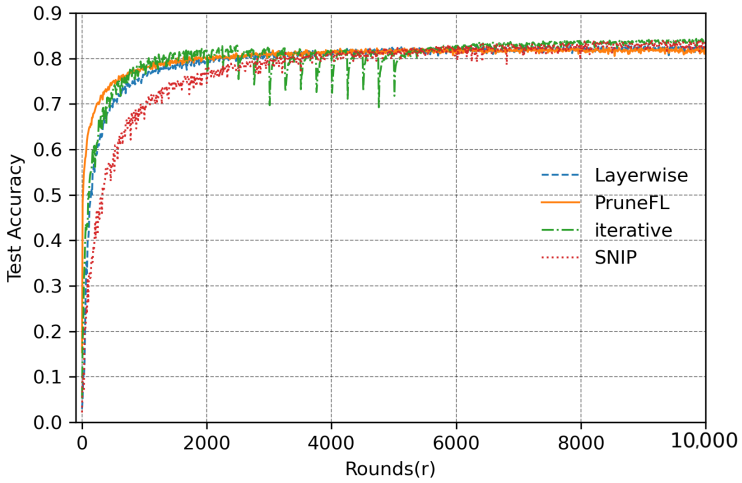

Figure 11. Stability of several adaptive pruning methods.

In Table 3, various federated learning methods incorporating pruning are compared. Figure 11 shows the accuracy vs. training epochs of some of the algorithms. Although the PruneFL algorithm reaches 70% and 80% accuracy in fewer training epochs, the time to achieve 70% and 80% accuracy is actually longer due to the longer time consumed by the initial pruning algorithm. Our method also surpasses iterative pruning in training speed. Additionally, our method demonstrates a significantly higher level of stability compared to Iterative Pruning, as illustrated in Figure 11, where Iterative Pruning shows irregular performance drops. Contrasted with the one-shot pruning method SNIP, under the same pruning ratio control of 3.7%, our method achieves a substantial improvement in convergence speed, and its performance stability also surpasses SNIP. Compared to the Conventional FL method without adaptive pruning, our approach introduces only a 6.6% increase in training time while incorporating an adaptive pruning process. Our approach achieves client-side pruning at minimal cost while incorporating personalized features, allowing the model to adapt to various application scenarios and meet the requirements for edge services.

Table 3. Performance comparison of several FL Methods combined with pruning.

Methods	acc	Time for 70% acc/s	Time for 80% acc/s	Convergence Time/s	Time for 10,000 r/s	Model Retention Ratio	Model Size	Client Memory Usage	GPU Memory Usage
Layerwise	83.0%	**549.97**	**2210.78**	**5161.72**	**13,816.69**	**3.7%**	1760 KB	3.083 GB	1117 MB
PruneFL [16]	82.5%	747.79	3396.00	9277.19	23,766.16	8.6%	3024 KB	3.121 GB	1323 MB
SNIP [28]	84.0%	1419.31	4468.36	9654.30	15,099.10	3.7%	1760 KB	\	7619 MB
Iterative	84.3%	617.71	2036.17	9171.22	15,332.01	3.7%	1760 KB	\	7075 MB
Conventional	82.8%	909.75	3521.44	4586.73	12,959.22	8.6%	3024 KB	2.936 GB	1117 MB

We also conducted experiments to compare with the latest related work. In Jiang et al.'s work "Complement Sparsification" [30] (referred to as CS below), the authors trained with 10 random users in each communication cycle of FL, simulating a scenario where only a subset of users participate in training due to resource limitations and network fluctuations in real FL processes. We replicated this experimental setup. The FEMNIST dataset consists of data from 193 users, posing experimental challenges with 193 device setups. Therefore, this experiment was conducted in a simulated environment. Additionally, initial pruning was stopped when the model sparsity reached 55%, differing from the experimental setup described earlier. Note that the code for the CS work has not been made publicly available, and, although both the IC model used in CS and the conv2 model used in this paper are CNNs, they are not identical. To facilitate better comparison, we designed a vanilla FL control group using the same model (conv2) as in this paper, similar to CS. The IC model in CS converged in 500 rounds. Hence, we first compared the performance of our work and vanilla FL after the first 500 rounds of training. The accuracy vs. communication rounds plot obtained from our experiments is shown in Figure 12. CS achieved a maximum accuracy of 74.3% in 500 rounds, slightly lower than the 76.9% of the IC model's vanilla FL. Meanwhile, our method achieved an accuracy of 74.6% in 500 rounds, compared to 73.5% for conv2 vanilla FL, demonstrating better performance than vanilla FL in the first 500 rounds.

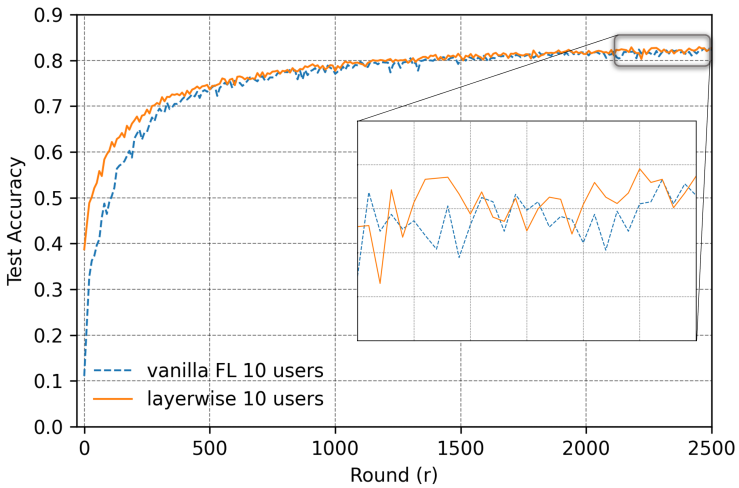

Figure 12. Model accuracy vs. communication rounds for conv2 trained with 10 random users in each round.

In addition, although the IC model reached a maximum accuracy of 74.3%. When the model sparsity exceeded 70%, the accuracy dropped to around 65%. At 80% sparsity, the accuracy dropped to below 60%. In contrast, as is shown in Figure 13, our method achieved slightly better performance than vanilla FL, even at 84.6% sparsity after 500 rounds, and the sparsity gradually increased to nearly 90% in subsequent training processes without com-

promising model accuracy compared to vanilla FL, demonstrating superior performance to CS in this aspect.

Other recent works mentioned in this paper, such as FedTiny and TailorFL, are not compared experimentally due to significant differences in experimental settings (such as dataset, model, and pruning strategies) and the lack of publicly available code.

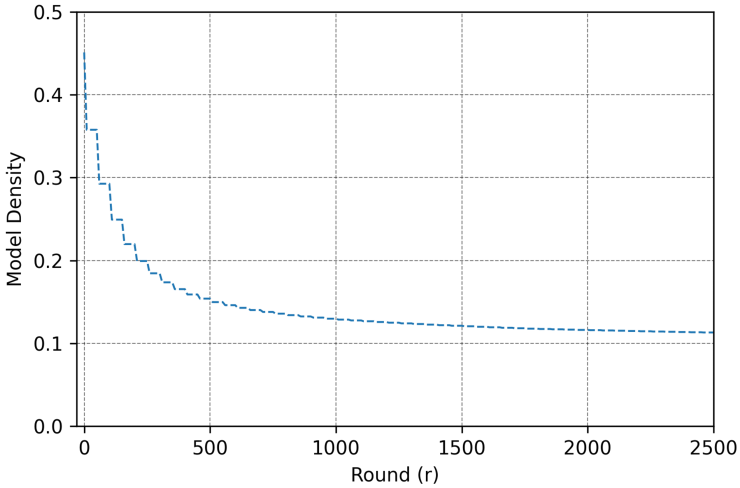

Figure 13. Model density vs. communication rounds.

5. Conclusions

In the practical application of federated learning, issues such as performance degradation due to Non-IID data distribution, high communication costs at edge nodes, and insufficient edge computing power have become increasingly prominent. To address these challenges, this paper proposes a personalized federated learning approach based on edge-side adaptive model pruning. The federated learning framework contributes in two ways: first, a novel initial one-step pruning is proposed to significantly reduce the model's complexity at the outset, enabling adaptation to the constrained conditions of edge nodes; second, fine-tuning and gradual pruning based on the personalized distribution of local data are realized. Additionally, we employ a specially designed aggregation algorithm to combine model parameters with different sparse structures, ensuring that the global model maintains good generalization performance. All methods used in this study involve no operations that transmit user data to other devices, thereby safeguarding user data privacy. The proposed algorithm achieves a compression of 3.7% of the model size, thereby reducing communication overhead. The overall training speed is increased by 46.4% compared to the baseline method, with minimal impact on training accuracy. The methods presented in this paper can be extended to larger-scale models and more complex scenarios. In future work, we will investigate the combination of federated learning with personalized pruning methods in various scenarios.

Author Contributions: Methodology, Y.Z. (Yueying Zhou); Software, Y.Z. (Yueying Zhou); Validation, Y.Z. (Yueying Zhou); Formal analysis, G.D. and T.Q.; Writing—original draft, Y.Z. (Yueying Zhou); Writing—review & editing, X.Z. and Y.Z. (Yongxin Zhu); Supervision, L.Z., X.Z. and Y.Z. (Yongxin Zhu); Funding acquisition, L.T., X.Z. and Y.Z. (Yongxin Zhu). All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (Grant No. U2032125, and Grant No. 12373113).

Data Availability Statement: The dataset used in this study is publicaly available. The FEMNIST dataset can be obtained from the following link: https://github.com/TalwalkarLab/leaf/tree/master/data/femnist (accessed on 4 March 2024).

Conflicts of Interest: Author Lin Zhang was employed by the company Shanghai Nuclear Engineering Research & Design Institute Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Wan, C.; Wang, Y.; Xu, J.; Wu, J.; Zhang, T.; Wang, Y. Research on Privacy Protection in Federated Learning Combining Distillation Defense and Blockchain. *Electronics* **2024**, *13*, 679. [CrossRef]
2. Ma, X.; Zhu, J.; Lin, Z.; Chen, S.; Qin, Y. A state-of-the-art survey on solving non-IID data in Federated Learning. *Future Gener. Comput. Syst.* **2022**, *135*, 244–258. [CrossRef]
3. Skianis, K.; Giannopoulos, A.; Gkonis, P.; Trakadas, P. Data Aging Matters: Federated Learning-Based Consumption Prediction in Smart Homes via Age-Based Model Weighting. *Electronics* **2023**, *12*, 3054. [CrossRef]
4. Zheng, M.; Yang, Y. Personalized federated learning algorithm based on mutual information and soft clustering. *Comput. Eng.* **2023**, *49*, 20–28.
5. Qiu, T.; Zheng, X.; Zhu, Y.; Feng, S. Federated learning architecture for non-IID data. *Comput. Eng.* **2023**, *49*, 110–117.
6. Yu, M.; Zheng, Z.; Li, Q.; Wu, F.; Zheng, J. A Comprehensive Study on Personalized Federated Learning with Non-IID Data. In Proceedings of the2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/ SocialCom/SustainCom), Melbourne, Australia, 17–19 December 2022; pp. 40–49. [CrossRef]
7. Zhu, Z.; Hong, J.; Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. In Proceedings of the 38th International Conference on Machine Learning, Virtual Event, 18–24 July 2021.
8. Zhang, L.; Shen, L.; Ding, L.; Tao, D.; Duan, L.Y. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022.
9. Zhang, J.; Shi, Y. A Personalized Federated Learning Method Based on Clustering and Knowledge Distillation. *Electronics* **2024**, *13*, 857. [CrossRef]
10. Zhan, Z.; Zhang, X. Computation-Effective Personalized Federated Learning: A Meta Learning Approach. In Proceedings of the 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), Hong Kong, China, 18–21 July 2023 ; pp. 957–958. [CrossRef]
11. Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M.A.; Dally, W.J. Retrospective: EIE: Efficient Inference Engine on Sparse and Compressed Neural Network. *arXiv* **2023**, arXiv:2306.09552.
12. Zhang, Z.; Jin, J.; Zhang, Z.; Zhou, Y.; Zhao, X.; Ren, J.; Liu, J.; Wu, L.; Jin, R.; Dou, D. Validating the lottery ticket hypothesis with inertial manifold theory. *Adv. Neural Inf. Process. Syst. (Neurips)* **2021**, *34*, 30196–30210.
13. Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; Shao, L. Hrank: Filter pruning using high-rank feature map. In Proceedings of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 1529–1538.
14. Hayou, S.; Ton, J.-F.; Doucet, A.; Teh, Y.W. Robust pruning at initialization. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Online, 4–7 May 2021. Conference Track Proceedings, 2021.
15. Frankle, J.; Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv* **2018**, arXiv:1803.03635.
16. Jiang, Y.; Wang, S.; Valls, V.; Ko, B.J.; Lee, W.H.; Leung, K.K.; Tassiulas, L. Model pruning enables efficient federated learning on edge devices. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 34, 10374–10386. [CrossRef] [PubMed]
17. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
18. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated learning with non-iid data. *arXiv* **2018**, arXiv:1806.00582.
19. Ye, M.; Fang, X.; Du, B.; Yuen, P.C.; Tao, D. Heterogeneous federated learning: State-of-the-art and research challenges. *Acm Comput. Surv.* **2023**, *56*, 1–44. [CrossRef]
20. Chen, H.; Frikha, A.; Krompass, D.; Gu, J.; Tresp, V. FRAug: Tackling federated learning with Non-IID features via representation augmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2023, Paris, France, 2–6 October 2023; pp. 4849–4859.
21. Zhang, H.; Liu, J.; Jia, J.; Zhou, Y.; Dai, H.; Dou, D. Fedduap: Federated learning with dynamic update and adaptive pruning using shared data on the server. *arXiv* **2022**, arXiv:2204.11536.
22. Jiang, Z.; Xu, Y.; Xu, H.; Wang, Z.; Qiao, C.; Zhao, Y. Fedmp: Federated learning through adaptive model pruning in heterogeneous edge computing. In Proceedings of the 2022 IEEE 38th International Conference on Data Engineering (ICDE), Kuala Lumpur, Malaysia , 9–12 May 2022; pp. 767–779.
23. Vahidian, S.; Morafah, M.; Lin, B. Personalized federated learning by structured and unstructured pruning under data heterogeneity. In Proceedings of the 2021 IEEE 41st international conference on distributed computing systems workshops (ICDCSW), Washington, DC, USA, 7–10 July 2021; pp. 27–34.
24. M Ghari, P.; Shen, Y. Personalized Online Federated Learning with Multiple Kernels. *Adv. Neural Inf. Process. Syst. (Neurips)* **2022**, *35*, 33316–33329.

25. Liu, B.; Zhang, Z.; He, P.; Wang, Z.; Xiao, Y.; Ye, R.; Zhou, Y.; Ku, W.S.; Hui, B. A survey of lottery ticket hypothesis. *arXiv* **2024**, arXiv:2403.04861.
26. You, H.; Li, C.; Xu, P.; Fu, Y.; Wang, Y.; Chen, X.; Baraniuk, R.G.; Wang, Z.; Lin, Y. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv* **2019**, arXiv:1909.11957.
27. Kim, Y.; Li, Y.; Park, H.; Venkatesha, Y.; Yin, R.; Panda, P. Exploring lottery ticket hypothesis in spiking neural networks. In *European Conference on Computer Vision*; Springer Nature: Cham, Switzerland, 2022; pp. 102–120.
28. Lee, N.; Ajanthan, T.; Torr, P.H. Snip: Single-shot network pruning based on connection sensitivity. *arXiv* **2018**, arXiv:1810.02340.
29. Huang, H.; Zhang, L.; Sun, C.; Fang, R.; Yuan, X.; Wu, D. Distributed pruning towards tiny neural networks in federated learning. In Proceedings of the 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), Hong Kong, China, 18–21 July 2023; pp. 190–201.
30. Jiang, X.; Borcea, C. Complement sparsification: Low-overhead model pruning for federated learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2023; Volume 37, pp. 8087–8095.
31. Deng, Y.; Chen, W.; Ren, J.; Lyu, F.; Liu, Y.; Liu, Y.; Zhang, Y. Tailorfl: Dual-personalized federated learning under system and data heterogeneity. In Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems, Boston, MA, USA, 6–9 November 2022; pp. 592–606.