Individualized Federated Learning Based on Model Pruning

Yueying Zhou\textsuperscript{1,2}, Gaoxiang Duan\textsuperscript{1,2}, Tianchen Qiu\textsuperscript{1,2}, Li Tian\textsuperscript{1,2}, Xiaoying Zheng\textsuperscript{1,2}, and Yongxin Zhu\textsuperscript{1,2}

\textsuperscript{1}Shanghai Advanced Research Institute, Chinese Academy of Sciences
\textsuperscript{2}University of Chinese Academy of Sciences

December 13, 2023
Individualized Federated Learning Based on Model Pruning

Yueying Zhou\textsuperscript{a,b}, Gaoxiang Duan\textsuperscript{a,b}, Tianchen Qiu\textsuperscript{a,b}, Li Tian\textsuperscript{a,b}, Xiaoying Zheng\textsuperscript{a,b} and Yongxin Zhu\textsuperscript{a,b}

\textsuperscript{a}Shanghai Advanced Research Institute, Chinese Academy of Sciences, , Shanghai, 201210, China
\textsuperscript{b}University of Chinese Academy of Sciences, , Beijing, 100049, China

ARTICLE INFO

Keywords: Individualized Federated Learning
Model Pruning

ABSTRACT

Federated Learning serves as a distributed framework for machine learning. Traditional approaches to federated learning often assume the independence and identical distribution (IID) of client data. However, real-world scenarios frequently feature personalized characteristics in client data, deviating from the IID assumption. Additionally, challenges such as substantial communication overhead and limited resources at edge nodes hinder the practical implementation of federated learning. In response to the challenges in deploying federated learning, including uneven data distribution, communication bottlenecks, and resource limitations at edge nodes, this paper introduces an individualized federated learning framework based on model pruning. This framework effectively adapts the client’s local model to the personalized distribution of local data while meeting the model aggregation requirements on the server. Utilizing sparse operations, the framework achieves personalized model pruning, efficiently compresses model parameters, and reduces computational load on edge nodes. Presently, our approach demonstrates a compression ratio of 3.8% on the non-IID dataset Feminist without compromising final training accuracy, resulting in a 12.3% acceleration in training speed.

1. Introduction

Federated Learning (FL) is a privacy-preserving machine learning approach that involves collaborative model training across multiple independent clients. In this methodology, there is no direct data transmission among the clients, ensuring privacy protection. Federated Learning finds broad applications in various fields such as medical image analysis and personalized recommendation systems. Federated Learning permits individual clients to conduct model training locally. However, local datasets often exhibit substantial variations due to differences in users and geographical locations. A promising research topic is how to enable local models to provide personalized services based on the features of local data without affecting the aggregation of the central server model.

Currently, a widely used aggregation algorithm in federated learning is the FedAvg algorithm proposed by Brendan McMahan (2017). This algorithm requires each client to perform local training for several epochs, after which the parameters of all clients are aggregated on the server. FedAvg is a concise and effective algorithm, its efficacy has been demonstrated in many practical applications. While there exist numerous extensions of the FedAvg algorithm, it is noteworthy that FedAvg performs well only when the data is independently and identically distributed (IID). Its performance degrades when dealing with non-IID data. At the end of each training epoch, the central server distributes the aggregated global model to the clients. The clients then replace their local models with the global model for local training. However, this process leads to the loss of locally specific model parameters, hindering the ability to address the personalized service requirements of individual clients.

Deep neural networks typically have a large number of parameters, requiring significant computational resources during training. Model pruning aims to eliminate redundant parameters and streamline network structures. The use of sparse models and sparse operations can effectively enhance computational efficiency, increase model interpretability, and reduce overfitting (Ru (2019)). Additionally, it can lower memory footprint, making models more adaptable to memory-constrained scenarios and devices. With the widespread adoption of edge devices such as smartphones, deployment of lightweight models gains considerable research significance (Gou Ying (2019)). In the realm of model pruning, numerous studies have demonstrated that a certain degree of pruning does not compromise model accuracy; in fact, it may even lead to improvements (Carreira-Perpiñán and Idelbayev (2018), Zeru Zhang (2021)). Common pruning methods include structured pruning and unstructured pruning. Structured pruning typically involves removing entire filters or entire intermediate layers, while unstructured pruning usually prunes layer weights. Additionally, one-shot pruning methods, often applied before or in the early stages of training, can provide an initial lightweight model (Lee et al. (2018)). However, these methods may result in pruning crucial nodes, leading to a decline in model performance (Yuang Jiang (2022)). Furthermore, they cannot tailor the pruning to better align with local characteristics, thus failing to meet the demands of personalized services.

Conventional weight pruning methods exhibit slow progress in the early stages of training and and lack the dynamic adaptability required to respond to changes in data distribution and task complexity during the training process (Zhao et al. (2018)). If personalized pruning is applied in local learning, challenges arise due to the disparate structures of...
user-side models, exacerbating the complexities associated with model aggregation.

In response to the aforementioned challenges, this paper introduces a novel federated learning approach. This method not only facilitates the realization of personalized, dynamically iterative local models at the individual client level but also yields superior generalization results after aggregation. The primary contributions of this paper are outlined below:

- Introduced a two-step Adaptive Pruning method tailored for local training. This approach allows for the swift identification of the lottery ticket during the initial phase, leading to a substantial reduction in model complexity. Additionally, it enables fine-tuning based on the distinctive characteristics of local data, progressively refining the pruning process. Furthermore, adjustments can be made based on the aggregation effectiveness of the global model.
- Proposed and implemented a federated learning framework based on personalized pruning. This solution takes into account the practical scenario where each client possesses limited data and computational resources with a non-IID distribution. Emphasizing client-level personalized training within the constraints of limited computational resources, this approach provides a local personalized service solution. Moreover, by leveraging client data, global model can achieve superior generalization results under the premise of privacy protection.
- Our approach achieved a model compression ratio significantly lower than the baseline (3.8%) and exhibited a 12.3% improvement in training speed compared to the baseline.

2. Related Works

Currently, there exists a considerable body of work related to personalized federated learning. Among these, clustering-based approaches involve grouping clients with similar data distributions into clusters, leveraging the homogeneity of some client data to mitigate the impact of data heterogeneity on federated learning. Zheng Meiguang (2023) introduced a personalized federated learning method based on mutual information and soft clustering. This method dynamically adjusts clustering results, allowing clients to belong to multiple clusters simultaneously, resulting in a 2.4-3% improvement in the highest average test accuracy. Songlin (2023) transformed non-IID data into multiple IID data subsets, computed data weights for each category on the global server, alleviating the impact of imbalanced overall training data. Apart from achieving a 4.2% accuracy improvement, the method also exhibited robust resistance to poisoning attacks. Saeed Vahidian (2021) proposed an approach for a personalized global model to address the challenges posed by heterogeneous and non-IID data distributions. Zhang Yan (2019) introduced an ensemble learning algorithm based on dynamic sampling probabilities, employing techniques such as resampling to mitigate the impact of imbalanced data on training. Pouya M. Ghari (2022) proposed a personalized federated learning approach based on the random feature (RF) approximation for multi-kernel learning, and mathematically demonstrate its effectiveness in handling heterogeneous data.

Pruning is a potent optimization technique in machine learning, offering advantages such as increased model sparsity, reduced computational resource requirements, enhanced model interpretability, and mitigation of overfitting. In early research on model pruning, it was common to first train a dense neural network and then perform pruning, as training a pruned model directly proved challenging. However, the "lottery ticket hypothesis" proposed by Jonathan Frankle (2018) posits that a randomly initialized dense neural network contains a effective sparse subnetwork, training directly from this subnetwork can yield performance no worse than the original model. This hypothesis provided strong support for the effectiveness of one-shot pruning. Building upon the lottery ticket hypothesis, Zeru Zhang (2021) not only provides theoretical underpinnings but also offers a practical pruning scheme. These approaches primarily focus on non-structural weight pruning. In addition, Mingbao Lin (2020) introduces a pruning method based on high-rank filters in feature maps. This method effectively eliminates less-contributing filters and is particularly suited for handling larger and deeper networks.

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

- 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.
- The inability to effectively protect crucial weights (lottery ticket).

3. Algorithm

In the practical implementation of federated learning, local data often exhibits personalized characteristics, while the aggregated model achieves improved generalization by aggregating client models. To address the personalized requirements of clients, this section introduces an algorithmic framework that strikes a balance between locally trained parameters and those distributed by the central server. This framework aims to enhance training for non-IID data.

3.1. Preliminaries

The parameter symbols and their corresponding meanings used in this system are listed in Table 1.
Individualized Federated Learning Based on Adaptive Pruning

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊙</td>
<td>Hadamard product</td>
</tr>
<tr>
<td>n, N</td>
<td>client index, total number of clients</td>
</tr>
<tr>
<td>k, K</td>
<td>global training round, global epochs</td>
</tr>
<tr>
<td>θₙ</td>
<td>local model for client n</td>
</tr>
<tr>
<td>θₙ(k)</td>
<td>server model to client in k-th iteration</td>
</tr>
<tr>
<td>Dₙ</td>
<td>local dataset for client n</td>
</tr>
<tr>
<td>Dₚₙ(k)</td>
<td>param weight for client n in k-th iteration</td>
</tr>
<tr>
<td>mₚₙ(k)</td>
<td>param mask for client n in k-th iteration</td>
</tr>
<tr>
<td>pₙ</td>
<td>proportion of each client in the aggregation</td>
</tr>
<tr>
<td>𝜃</td>
<td>mask/weight filtering threshold</td>
</tr>
<tr>
<td>gₙ(k)</td>
<td>param gradient for client n in k-th iteration</td>
</tr>
<tr>
<td>post(g)</td>
<td>the index of particular gradient</td>
</tr>
<tr>
<td>density(m, θ)</td>
<td>the function to calculate sparse model density</td>
</tr>
<tr>
<td>loss(·)</td>
<td>loss function</td>
</tr>
</tbody>
</table>

In this context, the client $c_n$ trains a local model $\theta_n$ 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 $m_n(k)$ will be a focal point in this section.

3.2. Theory

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:

$$\theta_s = \arg \min_{m_n, \omega_n} \frac{1}{n} \sum_n p_n \text{loss}(m_n; \omega_n)$$  \hspace{1cm} (1)

The performance of the global model is closely tied to the performance of individual client models. For the minimization of each client’s model, a commonly used method involves Taylor expansion, as described in related works such as Yann LeCun (1989) and Yuang Jiang (2022).

$$\Delta \text{loss} = \frac{\partial \text{loss}}{\partial \theta} \Delta \theta + O(\alpha^2) \approx g_n \Delta \theta$$ \hspace{1cm} (2)

Where $\Delta \theta_n$ can be represented by the following formula:

$$\Delta \theta_n = a_k g_n(k) \odot m_n(k)$$ \hspace{1cm} (3)

Therefore, it can be deduced that $\Delta \text{loss} \approx a_k g_n^2(k) \odot m_n(k)$. From this, the conclusion can be drawn that the rate of loss reduction in the model is primarily associated with gradient information.

3.3. Adaptive Pruning for Local Training

According to the lottery ticket hypothesis Jonathan Franke (2018), each model contains a sparse subnetwork, and training this subnetwork is not only time-efficient but also achieves comparable accuracy to the complete model. In order to quickly identify the lottery tickets in the initial stages, an improved phased pruning method is designed based on PruneFL, a pruning method proposed by Yuang Jiang (2022). In contrast to PruneFL, which randomly selects a user sample for pruning to obtain the lottery ticket network, this enhanced method incorporates more personalized features, initiating the pruning process at the client side. By horizontally comparing the masks uploaded to the server by various clients, not only can more accurate lottery ticket networks be filtered out, but the process is also expedited.

After initializing the clients, the initial pruning is performed on the client side using local data. This step initiates the model by removing redundant structures, compressing the model size. By training from a lightweight model at the outset, this approach effectively reduces the time and computational costs required for training. The process is entirely dependent on user data at the client side, imbuing it with personalized features from the very beginning. Once the initial pruning is completed, each client sends the pruned weights and masks to the server. The server, after deciding on the lottery tickets, sends those tickets back to the clients.

Algorithm 1 Layerwise Pruning

1: terminate = False
2: while not terminate do
3: client send mask $m_n(k)$ to the server
4: for $m_{n,-layer}$ in $m_n(k)$ do
5: if layer is input layer or output layer: then
6: $m_{s,layer} = (\sum_n m_{n,layer}) > \text{thres}_{mask}$
7: else
8: $m_{s,layer} = \bigcap_{n=1}^{N} m_{n,layer}$
9: end if
10: end for
11: density($m_s; \theta_s$) = remained nodes/all nodes
12: terminate = density($m_s; \theta_s$) < 0.05
13: end while
14: return $m_s$

We tried various methods to decide the lottery tickets. One simple decision-making method is to filter out lottery tickets through a threshold. For a given weight, if the number of clients that retain this weight exceeds the threshold, the weight in global model is retained, otherwise not reserved. The calculation method for this approach is expressed in the following formula:

$$m_s = (\sum_n m_n) > \text{thres}_{mask}$$ \hspace{1cm} (4)

In addition to this method, an approach based on the distinctive characteristics of each layer in the model was employed. Different methods were 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.

After the initial pruning, a much smaller model is obtained. In each subsequent pruning round, the model removes or adds back nodes based on their importance. Considering the two most crucial factors in weight pruning are the weights and gradients: nodes with very small weights make minimal contributions to computations. Therefore, in the pruning algorithm, nodes with smaller weights and those already pruned serve as a set for the next round of selection, while nodes with larger weights are protected. Simultaneously, the gradients generated during model training are used as a metric for assessing node importance. During the filtering process, nodes with higher importance are retained, while nodes with lower importance are pruned. In this process, previously pruned nodes may be reinstated. This approach makes it easier to avoid the issue of mistakenly pruning important nodes and affecting model performance. The workflow is primarily illustrated in Algorithm 2.

Algorithm 2 Local Pruning.

1: $\hat{m}_n(k) = m_n(k) + \omega_n(k) < \text{thres}_{\text{weight}}$
2: Set $D = \emptyset$
3: Set $G = \{g_j \mid g_j \in g_n(k) \odot \hat{m}_n(k)\}$
4: for $g_j$ in $G$: do
5: \hspace{1em} $D \cup \max(G)$
6: \hspace{1em} $G \setminus \max(G)$
7: end for
8: $\text{comp} = \text{average}(D) = \frac{1}{|D|} \sum D g_j$
9: for $g_j$ in $D$: do
10: \hspace{1em} if $g_j > \text{comp}: \text{then}$
11: \hspace{2em} $\hat{m}_n(k, \text{pos}) g_j) = 1$
12: \hspace{2em} $D \cup g_j$
13: \hspace{2em} $\text{comp} = \text{average}(D)$
14: \hspace{1em} else
15: \hspace{2em} $\hat{m}_n(k + 1) = m_n(k)$
16: \hspace{1em} end if
17: end for
18: return $\hat{m}_n(k + 1)$

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

4. 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 1.

![Figure 1: Personalized Federated Learning Workflow](image)

4.1. Pruning Structural Evaluation

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.

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:

$$\omega_{\text{local}} = \alpha(k) \odot m_n(k) + \omega_j(k) \odot \hat{m}_n(k)$$

Step 3: Local training is conducted using the merged tensors to obtain new model parameters $\omega_j(k + 1)$. If it is a pruning round, execute the pruning algorithm from Algorithm 3. Multiply the resulting new mask $m_n(k + 1)$ with the model parameters, and send $\omega_j(k + 1) \odot m_n(k + 1)$ to the server.

Algorithm 3 Federated Learning Algorithm.

1: for $n=1$ to $N$ do
2: \hspace{1em} $w_n(k) = \omega_j(k) \odot m_n(k) + \omega_j(k) \odot \hat{m}_n(k)$
3: \hspace{1em} $\Delta \omega_j(k) = w_n(k) - \omega_j(k)$
4: end for
5: $\Delta \sigma = \sum p_n \Delta \omega_j(k)$
6: $\omega_j(k + 1) = \omega_j(k) + \Delta \sigma$
7: $m_n(k + 1) = \left(\sum m_n(k) > \text{thres}_{\text{mask}}\right)$
8: return $\omega_j(k + 1) \odot m_n(k + 1)$

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.

5. Experiments

All experiments were conducted using one central server (Tesla V100-SXM2) and five client devices (GeForce RTX 2080). Both the central server and client CPUs were Intel(R) Xeon(R) Gold 6230 CPUs @ 2.10GHz, 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 (first 4 groups with 38 writers each, and the 5th group with 41 writers), forming a non-IID training set.

We employed a two-layer convolutional network (conv2) for adaptive pruning experiments. The control groups included the PruneFL method provided by Yuang Jiang (2022) 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 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 distribution of client data. Conversely, similar data distributions lead to more analogous masks.

As shown in Figure 2, masks in 2(a) and 2(b) are the results of pruning based on data from two different writers. After the same pruning rounds, 2(a) achieves a 4.7% node retention rate, while 2(b) has only 12.2%. Individual writers exhibit stronger personalized features, resulting in significant differences in the generated masks. On the other hand, 2(c) and 2(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 3. Pruning algorithms based on the average gradient are more likely to prune points with small outlier gradient values. Pruning on a single client, while achieving lower pruning ratios after multiple rounds, does not effectively remove scattered noise points with high gradient values in the mask. These noise points are mostly generated randomly, and although they do not significantly impact overall accuracy, they consume computational resources. Integrating mask information from various clients makes it easier to identify the valid structures in the mask. The masks generated by the

![Figure 2: Comparison of Mask Binary Graph From Different Clients](image)

![Figure 3: Single Client Pruning](image)

![Figure 4: Mask by Layerwise Pruning](image)
Layerwise Pruning method are shown in Figure 4. It can be observed that this method has essentially removed all the noise points in the mask, further reducing model complexity.

### 5.1. 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 5, while Figure 6 and Figure 7 illustrate the changes in model size and training time over the course of epochs, respectively.

From Figure 5, 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 7, 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>rounds</th>
<th>retain pct</th>
<th>time cost</th>
<th>final pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>PruneFL</td>
<td>16</td>
<td>8.6%</td>
<td>306.2s</td>
<td>13.4%</td>
</tr>
<tr>
<td>Threshold Pruning (ours)</td>
<td>1</td>
<td>71.7%</td>
<td>39.7s</td>
<td>19.1%</td>
</tr>
<tr>
<td>Layerwise Pruning (ours)</td>
<td>2</td>
<td>15.7%</td>
<td>72.3s</td>
<td>3.7%</td>
</tr>
</tbody>
</table>

### Figure 5: Model Accuracy Changed with Global Epochs

### Figure 6: Model Size Changed with Global Epochs

### Figure 7: Training Time Changed with Global Epochs

### 6. Conclusions

In federated learning, challenges such as disparate data distributions and weak computational capabilities at edge nodes are prevalent. Addressing issues related to non-IID data distributions, edge node computing power, and communication bottlenecks, this paper proposes an individualized federated learning approach based on model pruning. The federated learning process is divided into multiple stages: initial pruning significantly reduces model complexity, minimizing computational costs during training without compromising accuracy. The aggregation algorithm is designed to aggregate model parameters with different sparse structures, meeting personalized requirements. Importantly, all methods used in this study ensure that user data is not transmitted to other devices, preserving user privacy and security. The proposed techniques are applicable to larger-scale models and more complex scenarios. Future work will explore combining federated learning with personalized pruning methods in various scenarios.

### 7. Acknowledgement

This work was supported by National Natural Science Foundation of China (grant no. U2032125 & grant no. 12373113). Sincere gratitude to all the people who helped me during this period.
References


