Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.082658

ARTICLE

An Efficient Federated Learning Optimization Approach Based on Adaptive Hybrid Model Pruning

MengDie Hu[#], Na Wang[*], XueHui Du[#], BaiDong Huang[#] and KaiYuan Wang[#]

Cryptographic Engineering School, Information Engineering University, Zhengzhou, China

*Corresponding Author: Na Wang. Email: twftina_w@126.com

[#]These authors contributed equally to this work

Received: 19 March 2026; Accepted: 20 May 2026; Published: 23 July 2026

ABSTRACT: With the rapid development of the Internet of Things (IoT) and edge intelligence, the volume of data generated by edge devices has grown explosively. Federated learning (FL), characterized by the paradigm of "data remaining local while models are shared," has emerged as a key approach for adapting to the distributed architecture of edge computing, breaking down data silos, and enabling privacy preservation. However, its practical deployment in edge computing environments still faces significant challenges, including limited device resources and pronounced data heterogeneity. Existing pruning strategies for federated learning are predominantly based on static and single-design schemes, making it difficult to achieve a balanced trade-off among training overhead, communication cost, and model accuracy. To address these issues, this paper proposes FedAHP (Federated Learning with Adaptive Hybrid Pruning), an efficiency optimization scheme for federated learning based on adaptive hybrid model pruning. On the client side, an adaptive pruning mechanism driven by training states is designed to dynamically adjust pruning behavior during local training. On the server side, a counter-based heterogeneous aggregation method is adopted to efficiently align updates from clients with different pruning rates, thereby avoiding additional communication overhead. Furthermore, after training becomes stable, a performance-aware periodic structured pruning strategy is introduced to compress the global model scale and reduce subsequent training costs. Experimental results demonstrate that FedAHP maintains high model accuracy on the MNIST, CIFAR-10 and CIFAR-100 datasets while significantly reducing per-round communication overhead and time cost, making it well suited to the resource-constrained requirements of edge computing scenarios.

KEYWORDS: Federated learning; edge computing; model pruning; efficiency optimization

1 Introduction

With the rapid proliferation of Internet of Things (IoT) devices, intelligent terminals, and other edge computing devices, together with the explosive growth of data generated at the network edge, the conventional centralized data processing paradigm is increasingly confronted with excessive computational burden and significantly increased processing latency. As a result, it has become difficult to satisfy the stringent requirements of real-time and data-intensive applications. Edge Computing (EC) addresses this challenge by pushing computation and storage resources downward from centralized cloud data centers to the proximity of data sources, such as base stations, access points, campus micro–data centers, and industrial gateways. By enabling data processing closer to where data are generated, EC can effectively reduce end-to-end latency, alleviate the workload of cloud data centers, and improve the overall efficiency of data processing.

Therefore, it has become a critical infrastructure for supporting real-time and data-intensive intelligent applications [1,2].

Federated Learning (FL) [3], as an emerging distributed machine learning paradigm, enables collaborative model training without requiring participants to share their raw local data. Its core idea is to replace direct data transmission with the exchange of locally trained model gradients or parameters from clients or edge nodes, thereby allowing multiple parties to jointly optimize a global model while preserving data locality. In this manner, FL not only leverages cross-domain data to improve model generalization capability but also avoids, to a large extent, the centralized transmission and storage of raw data. Consequently, FL has been widely regarded as one of the most promising technological paradigms for enabling privacy-preserving distributed intelligence in edge computing environments [4].

Nevertheless, the practical deployment of FL systems still faces several critical challenges. First, edge devices participating in federated training are inherently heterogeneous and resource-constrained. Their limited computation and storage capacities often make it difficult to support complex models and repeated rounds of local training. For models with a large number of parameters, local training incurs substantial computational and memory consumption, which directly increases the time required for each training round. Moreover, some devices may drop out during training due to insufficient memory, excessive energy consumption, or heavy system workload, thereby severely undermining training stability and reducing the convergence efficiency of the global model. Second, local data distributed across edge devices are usually highly heterogeneous. Since different devices serve different user groups, application scenarios, and service tasks, their local datasets often vary significantly in terms of sample size, class distribution, and feature space, exhibiting a typical non-independent and identically distributed (Non-IID) nature. Under such conditions, the model updates uploaded by different clients can differ substantially in both direction and magnitude, which not only increases the difficulty of global aggregation but may also induce severe oscillations during training, slow down convergence, and ultimately degrade both the overall training efficiency and model performance of FL [5].

A systematic review of the literature indicates that existing studies commonly resort to model lightweighting mechanisms to mitigate the above challenges [6,7]. Among these techniques, model pruning is considered one of the most effective solutions. By removing redundant parameters or connections through structured or unstructured pruning, model pruning can reduce the parameter scale and computational complexity of deep learning models, thereby enabling resource-constrained edge devices to participate in federated training with lower resource consumption [8]. However, existing pruning-based federated learning (FL) schemes still suffer from several limitations in strategy design. First, many methods adopt static pruning strategies and lack adaptive mechanisms that can account for device heterogeneity and data distribution characteristics, making it difficult to balance model accuracy and training efficiency. Second, although federated pruning approaches now include structured, unstructured, and hybrid strategies, adaptive coordination across different stages of training remains insufficiently explored. For instance, AutoFLIP [9] demonstrates that hybrid structured-unstructured pruning can improve federated learning efficiency through globally guided pruning cues. Yet, dynamically adjusting pruning according to each client's local training state, while simultaneously supporting efficient heterogeneous aggregation and progressive global model compression, remains an open challenge. Since clients in FL often exhibit diverse data distributions, update magnitudes, and convergence behaviors, a more fine-grained and training-state-aware pruning mechanism may provide additional efficiency benefits. Moreover, once training becomes stable, further compressing the global model can help reduce subsequent communication and local training overhead.

To address the above challenges, this paper proposes FedAHP, an efficiency optimization scheme for federated learning that integrates adaptive hybrid model pruning. Specifically, on the client side, a training-state-aware adaptive unstructured pruning mechanism is designed to reduce the size of uploaded model updates during the early stage of training, thereby alleviating communication overhead. On the server side, a counter-based heterogeneous aggregation method is introduced to effectively align and aggregate model updates generated under different pruning ratios, without incurring additional index transmission overhead. Furthermore, in the later stage of training, when the global model gradually becomes stable, a performance-aware structured pruning strategy is periodically applied by the central server to progressively compress the global model and reduce the subsequent local training burden on resource-constrained edge devices. Through the joint design of adaptive pruning and heterogeneous aggregation, the proposed scheme achieves coordinated optimization of model accuracy, communication efficiency and training efficiency in federated edge learning scenarios.

The main contributions of this paper are summarized as follows:

- We propose a training-state-aware adaptive model update pruning method to reduce the communication overhead of client-side model update transmission. By dynamically adjusting the unstructured pruning ratio according to each client's local training state in the early stage of training, the proposed method effectively compresses uploaded updates while maintaining model performance.
- We develop a counter-based heterogeneous model aggregation method to address the aggregation difficulty caused by inconsistent update lengths under different pruning ratios. Through parameter-wise accumulation and counting, the proposed method can efficiently align heterogeneous client updates without introducing extra index transmission overhead, thereby improving both communication efficiency and system scalability.
- We further design a performance-aware structured pruning method to compensate for the lack of practical hardware acceleration offered by unstructured pruning. Once the global training process becomes stable, the server periodically performs structured pruning with progressively decaying intensity, which gradually reduces the model size and alleviates the subsequent local training burden on edge devices.
- Extensive experiments on the MNIST and CIFAR-10 datasets under both independent and identically distributed (IID) and Non-IID settings demonstrate the effectiveness of the proposed FedAHP scheme. Compared with baseline methods, FedAHP achieves an average accuracy of 87.34%, with improvements ranging from 0.81% to 4.97%. In addition, it reduces per-round communication overhead by 26.19%–64.62% on MNIST and 32.82%–74.19% on CIFAR-10, while reducing per-round time overhead by 58.56%–72.45% on MNIST and 37.41%–61.40% on CIFAR-10. Furthermore, experiments on the CIFAR-100 dataset further demonstrate that the proposed FedAHP scheme exhibits good generalization performance.

2 Related Work

To address the computational, storage, and communication overhead of federated learning (FL) in resource-constrained environments, existing studies have mainly focused on two optimization directions: reducing communication burden and compressing model size. On the one hand, communication overhead during training can be reduced by lowering the frequency of model aggregation, with representative methods including FedAvg [3] and its variants [10]. On the other hand, model compression techniques can be adopted to reduce model size and computational complexity, thereby accelerating convergence and enabling resource-limited edge devices to participate in FL training with lower overhead [11]. At present, the most widely used

model compression techniques mainly include model quantization [12], model pruning [13], and knowledge distillation [14].

According to the studies by Alahmari and Alghamdi [6] and Dantas et al. [7], model pruning is particularly effective for edge devices with limited computational capability and memory resources, compared with model quantization and knowledge distillation. By providing each edge device with a lightweight version of the global model, model pruning can reduce memory footprint during training, local computational overhead, and the communication cost caused by model transmission, thereby improving the scalability of FL systems [8] and making the practical deployment of FL in complex distributed edge environments more feasible [7].

To make model pruning applicable to FL systems, Liu et al. [15] proposed a fixed-rate layer-wise pruning scheme, in which specific network layers are pruned before model parameters are uploaded, thereby compressing the model and reducing communication overhead during training. FedLP [16] is a layer-wise pruning method tailored for FL. In this scheme, each client performs pruning on a per-layer basis using predefined fixed pruning ratios and only retains and uploads the remaining parameters of the pruned layers, while the server aggregates the corresponding layer parameters received from different clients separately. In this way, FedLP can simultaneously reduce local computational overhead on the client side and communication burden during FL training to a certain extent. PruneFL [17] introduces a model masking mechanism into FL and maintains parameter structure consistency during training through a unified global model mask. However, PruneFL is limited to unstructured sparsity, which makes it difficult to obtain direct inference acceleration on hardware platforms in practical deployment.

To better cope with client-side data heterogeneity and differences in local resource conditions, recent studies have shifted their attention toward adaptive model pruning. Such methods typically adjust pruning strategies according to client status, resource constraints, or training dynamics, with the goal of improving the adaptability of model compression in heterogeneous environments.

LotteryFL [18] implements adaptive unstructured pruning based on the Lottery Ticket [19] Hypothesis by dynamically adjusting the pruning ratio according to the importance of parameters in different layers. During training, each client uploads both the local model updates and the corresponding pruning mask to the server, and the server aggregates only the parameters retained after pruning. Through this iterative "pruning–retraining" mechanism, the model can maintain relatively good performance during the sparsification process. Although LotteryFL is effective in reducing the number of transmitted parameters, the transmission of pruning masks incurs additional communication overhead. Hu et al. [20] proposed the CEMP-FL scheme, which dynamically adjusts the pruning ratio of each layer according to inter-layer sensitivity and model sparsity, thereby enabling adaptive pruning. For the sparse models generated after pruning, a parameter matrix compression strategy is further adopted to reduce the actual transmission size of sparse parameter matrices, thus lowering communication overhead during training. Meanwhile, this method alleviates, to some extent, the pruning bias caused by heterogeneous local data distributions across clients. Nevertheless, although unstructured pruning enables fine-grained control over pruning ratios, achieves relatively high compression rates, and causes limited accuracy degradation, the resulting sparse models are usually difficult to translate into direct hardware acceleration on edge devices. In addition, pruning schemes based on the Lottery Ticket Hypothesis often introduce extra computational and storage costs, as well as additional communication overhead due to mask transmission.

To further improve resource adaptability, some studies have begun to explore structured pruning strategies. Diao et al. [21] proposed HeteroFL, an adaptive structured pruning strategy. Under the premise of maintaining a consistent global model structure, it assigns submodels with different complexities to different clients to accommodate device heterogeneity, thereby reducing communication and computational

overhead. A unified global model is then obtained through parameter alignment and weighted aggregation. However, because different clients train different submodels, performance fluctuations may arise during the training process. Lv et al. [22] proposed a device-aware two-stage adaptive pruning method. According to device memory budgets and energy constraints, pruning ratios are allocated in advance. Candidate pruned models are first generated in a coarse-grained manner on high-performance edge nodes and then distributed to clients, after which each client performs fine-grained pruning based on its local data to obtain a model structure better suited to its resource conditions. Although this method improves the adaptability of models to device resources, the differentiated model structures formed across clients increase the complexity of system maintenance and model aggregation. Moreover, performing a second pruning stage on resource-constrained devices may introduce additional local computational and storage overhead. In addition, LP-ALR-HFL [15] adjusts layer-wise pruning ratios according to model accuracy so as to mitigate the impact of Non-IID data distributions on pruning effectiveness and balance model performance against communication overhead. Although these structured pruning methods can effectively reduce communication and computational overhead and provide certain hardware acceleration benefits in practical deployment, they are often accompanied by varying degrees of accuracy degradation.

More recently, hybrid pruning strategies that combine structured and unstructured pruning have also been explored in FL. FedLTH [23] searches for high-accuracy sparse submodels through iterative magnitude pruning (IMP), weight rewinding, and structural transformation. It further allocates the differential privacy budget according to the remaining training rounds, so as to jointly consider model compression and privacy preservation. FedLTH shows certain advantages in terms of model compression and deployment friendliness. However, its uniform privacy budget allocation strategy based on the remaining training rounds cannot finely adjust the privacy budget according to the model training stage, parameter importance, and pruning dynamics. In addition, multiple rounds of rewinding and retraining introduce extra computational overhead. AutoFLIP [9] is a representative example in this direction. It introduces a federated hybrid model pruning framework based on loss landscape exploration, where clients collaboratively characterize optimization-related information to derive global pruning guidance for both structured and unstructured pruning. In this way, AutoFLIP aims to reduce communication and computation costs while improving robustness under heterogeneous data distributions. However, its pruning mechanism is primarily guided by a global exploration stage and globally distilled pruning cues.

Compared with these studies, the proposed FedAHP differs from existing methods in both mechanism and design rationale. Rather than relying on a fixed pruning policy or a primarily global one-shot pruning guidance process, FedAHP performs round-wise adaptive update pruning according to each client's local training state, allowing pruning intensity to change dynamically during training. Moreover, FedAHP combines this client-side adaptive unstructured pruning with a counter-based heterogeneous aggregation mechanism and a performance-aware periodic structured pruning strategy on the server side after training becomes stable. Therefore, although recent work has shown the promise of hybrid pruning in FL, adaptive coordination among client-specific training dynamics, heterogeneous sparse aggregation, and progressive global model compression remains insufficiently studied.

3 Methodology

3.1 Overview of the Proposed FedAHP Scheme

The proposed FedAHP scheme consists of three key components: a *training-state-aware adaptive model update pruning method*, a *counter-based heterogeneous aggregation method*, and a *performance-aware structured model pruning method*. The overall architecture of the proposed FedAHP framework is illustrated in Fig. 1.

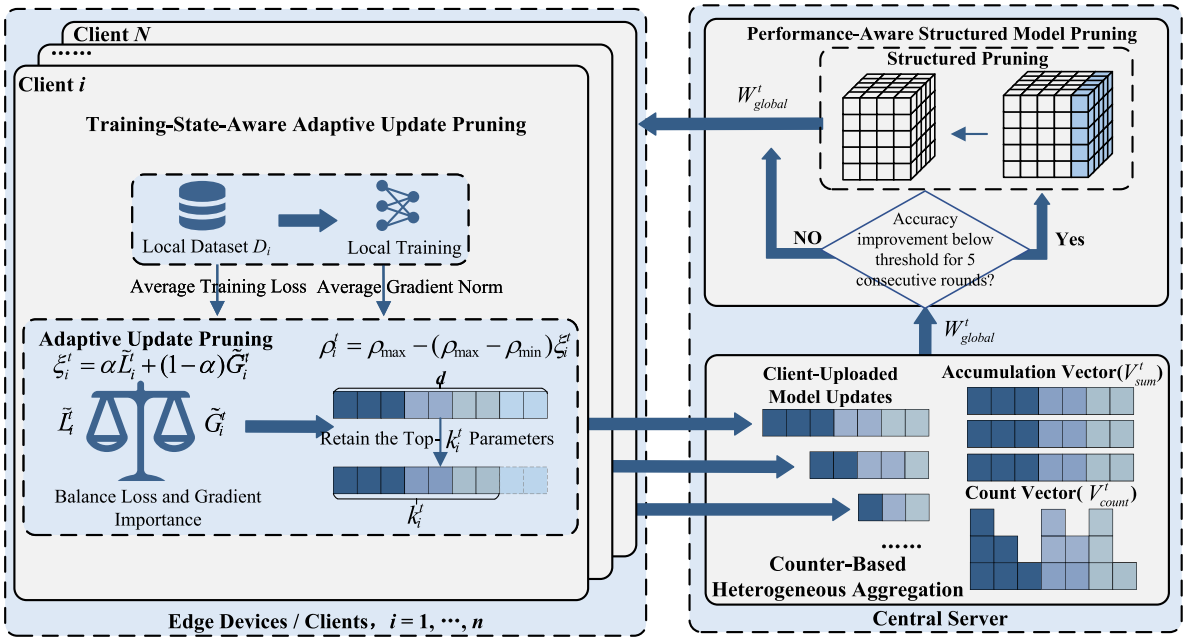


Figure 1: Overall architecture of the proposed FedAHP scheme.

The proposed FedAHP framework does not switch between structured pruning and unstructured pruning as two mutually exclusive alternatives. Instead, the two pruning strategies are applied at different levels and training stages. During regular communication rounds, FedAHP performs client-side adaptive unstructured pruning on local model updates to reduce uplink communication. Each client determines its update pruning ratio according to its local training state. After the global model enters a stable training stage, the server periodically performs performance-aware structured pruning to remove low-importance filters/channels and progressively reduce the global model size. Therefore, unstructured update pruning mainly improves communication efficiency, whereas structured pruning further reduces the model size, computation cost, and subsequent communication overhead.

3.2 Training-State-Aware Adaptive Model Update Pruning Method

In federated learning, unstructured pruning generally cannot provide practical hardware acceleration on client devices. Moreover, due to the inherently Non-IID nature of client-side data, local datasets across different clients often exhibit significant heterogeneity in terms of sample classes, data volume, and feature distributions. Such statistical heterogeneity further leads to noticeable discrepancies in the magnitude of local model updates and the convergence behavior during client-side training. Therefore, conventional federated pruning methods that adopt a fixed pruning ratio based solely on hardware resource constraints (e.g., computational capability, storage capacity, or communication bandwidth) are often unable to accurately match the actual training state of individual clients, and may consequently cause model performance degradation and accuracy loss.

To address the above issue, we first propose a *training-state-aware adaptive pruning method*. In the t-th communication round, the server first distributes the global model $\mathbf{W}_{\text{global}}^t$ to all participating clients. To ensure a consistent parameter priority during update compression across all clients, the server constructs a global importance ranking over model parameters according to their significance in the current global model. Let the total number of model parameters be d, and let the importance score of the j-th parameter

be denoted by s_j^t. To reduce computational complexity, we use the absolute parameter magnitude as a lightweight importance metric, i.e.,

$$s_j^t = \left|\mathbf{W}_{\text{global},j}^t\right|, \quad j = 1, 2, \ldots, d. \tag{1}$$

Then, as shown in (2), the server generates a descending global ranking index π^t according to the importance scores:

$$\pi^t = \text{argsort}\left(\mathbf{s}_j^t\right) \tag{2}$$

where the ranking index π^t is broadcast to all clients, enabling them to prune local updates under a unified global priority criterion. In this way, local independent pruning bias caused by Non-IID data distributions can be mitigated.

After receiving the global model $\mathbf{W}_{\text{global}}^t$, client C_i performs local training on its local dataset D_i and obtains the local model update vector $\mathbf{g}_i^t$ in round t. During local training, client C_i records the training loss and gradient norm of each mini-batch. After completing local training, it computes the average training loss $\bar{L}_i^t$ and the average gradient norm $\bar{G}_i^t$. Here, $\bar{L}_i^t$ characterizes the fitting quality of the current model on the local data distribution of client C_i, while $\bar{G}_i^t$ reflects the intensity of local updates and the current convergence state.

Since the scales and numerical ranges of the training loss and gradient norm are generally different, directly combining them may cause one metric to dominate the pruning decision. Therefore, both quantities are first normalized. Let the minimum and maximum average training losses of all participating clients in round t be denoted by $L_{\min}^t$ and $L_{\max}^t$, respectively, and let the minimum and maximum average gradient norms be denoted by $G_{\min}^t$ and $G_{\max}^t$, respectively. Then, the normalized values $\tilde{L}_i^t$ and $\tilde{G}_i^t$ are computed as

$$\tilde{L}_i^t = \frac{\bar{L}_i^t - L_{\min}^t}{L_{\max}^t - L_{\min}^t + \mu} \tag{3}$$

$$\tilde{G}_i^t = \frac{\bar{G}_i^t - G_{\min}^t}{G_{\max}^t - G_{\min}^t + \mu} \tag{4}$$

where μ is a small positive constant introduced to avoid division by zero. It should be noted that the min–max normalization in this step requires only lightweight scalar statistics from participating clients. Specifically, each selected client reports its average training loss and gradient-norm statistic to the server, and the server computes the corresponding normalization bounds for the current round. This step introduces only a small amount of scalar metadata and can be piggybacked with the regular client–server communication procedure. Therefore, it is not separately included in the reported communication size, whose dominant component is the transmission of model parameters and update tensors.

Based on the normalized loss and gradient statistics, the training state of client C_i in round t is quantified by

$$\xi_i^t = \alpha\tilde{L}_i^t + (1-\alpha)\tilde{G}_i^t \qquad \xi_i^t \in [0,1], \tag{5}$$

where $\alpha \in [0,1]$ is a balancing coefficient used to control the relative contribution of the average training loss and the average gradient norm in training-state estimation. A larger α implies that the pruning decision relies more heavily on the model fitting error, whereas a smaller α places greater emphasis on the local convergence condition.

In general, clients with more stable training states can tolerate stronger compression, while clients with more volatile training dynamics need to preserve more update information for subsequent learning. Accordingly, the pruning ratio of client C_i is adaptively determined according to its training state as

$$\rho_i^t = \rho_{\max} - (\rho_{\max} - \rho_{\min})\,\xi_i^t \tag{6}$$

where $\rho_{\min}$ and $\rho_{\max}$ denote the lower and upper bounds of the pruning ratio, respectively.

From Eq. (6), it can be seen that when the local training process of client C_i is relatively stable, ξ_i^t tends to be smaller and the pruning ratio ρ_i^t approaches the upper bound $\rho_{\max}$. In contrast, when the local training process fluctuates significantly, ξ_i^t becomes larger and ρ_i^t approaches the lower bound $\rho_{\min}$. Therefore, different clients can dynamically adjust their compression strengths according to their own training states.

After determining the pruning ratio for client C_i, the client uses the global ranking index π^t broadcast by the server to retain the top-k_i^t elements of the local update according to the global priority order, thereby obtaining the pruned local update vector:

$$\mathbf{g}_i^t = \mathbf{g}_i^t\left[\pi^t(1{:}k_i^t)\right] \tag{7}$$

$$k_i^t = \left\lfloor d \cdot (1 - \rho_i^t) \right\rfloor \tag{8}$$

Since all clients share the same global ranking index, the upload positions of each client can be uniquely determined by the retained length k_i^t. As a result, in practical communication, clients do not need to explicitly transmit binary masks or nonzero index locations, which avoids the extra communication overhead commonly associated with conventional sparse representations.

3.3 Counter-Based Heterogeneous Aggregation Method

Since different clients adopt different update pruning ratios according to their own training states, the lengths of the uploaded prefix updates, denoted by k_i^t, are generally inconsistent across clients, thereby forming heterogeneous sparse updates. If conventional unstructured sparse aggregation methods are directly applied, the server usually needs to receive explicit index information to identify the nonzero positions contributed by different clients. This introduces additional communication overhead and weakens the practical gains brought by update compression. To address this issue, we propose a *counter-based heterogeneous aggregation method*. By exploiting the globally shared ranking index among all clients, the proposed method maps prefix updates of different lengths into a unified ranking space for aggregation, thereby enabling efficient aggregation of heterogeneous sparse updates without transmitting explicit masks or index information.

Specifically, during model aggregation in the t-th communication round, the server first initializes two global vectors of length d: an accumulation vector $\mathbf{V}_{\text{sum}}^t$ and a counting vector $\mathbf{V}_{\text{count}}^t$. Considering that the local data volumes may differ across clients, a sample-size-weighted statistics mechanism is adopted during aggregation. Let n_i denote the number of local samples owned by client C_i. Then, $\mathbf{V}_{\text{sum}}^t$ is used to accumulate the weighted updates contributed by different clients over the first k_i^t positions in the shared ranking space, while $\mathbf{V}_{\text{count}}^t$ records the total effective sample weight participating in the aggregation at each ranked position. The accumulation process is defined as

$$\mathbf{V}_{\text{sum}}^t[j] \leftarrow \mathbf{V}_{\text{sum}}^t[j] + n_i \cdot \mathbf{g}_i^t[j], \quad j = 1, 2, \ldots, k_i^t \tag{9}$$

$$\mathbf{V}_{\text{count}}^t[j] \leftarrow \mathbf{V}_{\text{count}}^t[j] + n_i, \quad j = 1, 2, \ldots, k_i^t \tag{10}$$

where $g_i^t[j]$ denotes the j-th element of the pruned local update of client C_i in the shared ranking space.

After all client updates have been received, the server performs element-wise normalization according to the accumulated statistics and updates the global model, yielding the global model for round $t+1$, denoted by $\mathbf{W}_{\text{global}}^{t+1}$. For each parameter position $j = 1, 2, \ldots, d$, the update rule is given by

$$\mathbf{W}_{\text{global},j}^{t+1} = \mathbf{W}_{\text{global},j}^{t} + \eta \cdot \frac{\mathbf{V}_{\text{sum}}^{t}[j]}{\mathbf{V}_{\text{count}}^{t}[j] + \mu} \tag{11}$$

where η is the learning rate and μ is a small positive constant introduced to avoid division by zero.

The above aggregation mechanism effectively aligns heterogeneous sparse updates generated under different pruning ratios into a unified parameter ranking space. Therefore, even when clients upload updates with different retained lengths, the server can still perform consistent aggregation without requiring explicit sparse masks or index transmission. This design not only reduces communication overhead, but also improves the scalability of the federated learning system under heterogeneous client conditions.

3.4 Performance-Aware Structured Model Pruning Method

The aforementioned training-state-aware adaptive pruning method mainly operates on the uploaded client updates and can effectively reduce communication overhead. However, it does not directly change the structure of the global model. Therefore, its improvement in the computation and storage costs of subsequent local training on clients remains limited. To further enhance the training and deployment efficiency of federated learning under resource-constrained environments, we design a *performance-aware structured model pruning method*, which is activated after the global model training process becomes sufficiently stable. Specifically, the central server periodically performs structured pruning on the global model, progressively removing structural units with relatively low importance so as to reduce the model size and the local training cost on clients in subsequent rounds.

To avoid performance degradation caused by overly aggressive pruning before model convergence, the proposed method performs periodic structured pruning only when global training has entered a stable stage. Let Acc_t and Acc_{t-1} denote the global model accuracy in rounds t and $t-1$, respectively. The relative accuracy improvement in round t is defined as

$$r_t = \frac{\text{Acc}_t - \text{Acc}_{t-1}}{\text{Acc}_{t-1}} \tag{12}$$

If $r_t < r_{\text{stab}}$ holds for five consecutive rounds, the global training process is regarded as stable, and the server triggers one round of structured pruning, where r_{stab} is a predefined stability threshold.

To alleviate performance fluctuations caused by frequent pruning in the later stage of training, a structured pruning strategy with progressively decaying intensity is adopted. Let ρ_S denote the pruning ratio in the S-th structured pruning operation. Its value decays exponentially with the pruning step index S according to

$$\rho_S = \rho_{\text{init}} \cdot e^{-S} \quad S = 0, 1, 2, \ldots \tag{13}$$

where the initial structured pruning ratio is set as $\rho_{\text{init}} = \frac{\rho_{\text{total}}}{2}$ and the cumulative pruning constraint is imposed as

$$\sum_{s=0}^{S} \rho_S \leq \rho_{\text{total}} \tag{14}$$

where ρ_{total} denotes the upper bound of the cumulative structured pruning ratio. This design prevents excessive model compression and helps preserve model capacity.

When structured pruning is triggered, the server first evaluates the importance of each structural unit in the global model. Let $\Theta_{l,u}$ denote the parameter set corresponding to the u-th structural unit in the l-th layer. The importance score of this structural unit is measured by its ℓ_2 norm:

$$q_{l,u} = \|\Theta_{l,u}\|_2 \tag{15}$$

The server then ranks all structural units in descending order according to their importance scores and retains the most important units based on the current pruning ratio ρ_S. Let $m_l^{(S)}$ denote the number of structural units in layer l before the S-th pruning operation. Then, the number of structural units retained after pruning is determined by

$$\hat{m}_l^{(S)} = \left\lfloor m_l^{(S)}(1-\rho_S) \right\rfloor \tag{16}$$

After pruning, the resulting global model $\mathbf{W}_{\text{global}}^{t+1}$ is redistributed to all clients for subsequent local training. Through this progressive structured pruning process, the model size is gradually reduced while the overall compression ratio remains within a controllable range. As a result, the proposed method improves the lightweight deployment capability of the model and reduces the computational burden on edge devices, while avoiding irreversible performance degradation caused by over-pruning.

3.5 Computational Overhead and Deployment Considerations

The additional computational overhead introduced by FedAHP mainly comes from client-side training-state evaluation, update pruning, global importance ranking, and server-side counter-based aggregation. On the client side, the extra operations only involve computing lightweight statistics, such as the average loss and gradient norm, and selecting retained update entries according to the shared ranking. These operations do not require training an additional pruning network or transmitting explicit sparse masks. On the server side, the counter-based aggregation maintains an accumulation vector and a counting vector, and its cost scales approximately linearly with the number of received sparse update entries. The global importance ranking is computed on the server side. Although a full sorting operation has a complexity of $O(d \log d)$ for a model with d parameters, this cost is incurred at the server rather than on resource-constrained clients and can be amortized by updating the ranking periodically or after structured pruning. After structured pruning is activated, the smaller global model further reduces subsequent downlink communication, uplink update size, and local computation.

4 Experiments and Analysis

4.1 Experimental Settings

All experiments in this paper were conducted under the Ubuntu 22.04 operating system. The hardware platform was equipped with an Intel(R) Xeon(R) Platinum 8358P CPU running at 2.60 GHz and an NVIDIA RTX 4090 GPU with 24 GB memory. The proposed method was implemented using Python 3.12 and PyTorch 2.7.0. The experimental settings were configured as follows: the number of edge clients was set to 100, and the local batch size was set to 32. The client participation rate is set to 30%, meaning that in each training round, 30% of the clients are randomly selected to participate in local training and upload model updates. Stochastic gradient descent (SGD) was adopted as the optimizer, with a learning rate of 0.01 and a momentum coefficient of 0.9. The total number of global communication rounds and local training epochs were set to

200 and 5, respectively. For unstructured update pruning, the lower and upper bounds of the pruning ratio were set to $\rho_{\min} = 0.0$ and $\rho_{\max} = 0.5$, respectively. For structured model pruning, the upper bound of the cumulative pruning ratio was set to $\rho_{\text{total}} = 0.6$, the training-stability threshold was set to $r_{\text{stab}} = 0.002$, and the balancing coefficient in training-state estimation was set to $\alpha = 0.5$.

4.1.1 Datasets

We conduct experiments on MNIST, CIFAR-10, and CIFAR-100 to evaluate the effectiveness and generalization ability of the proposed FedAHP under different task complexities. MNIST is a handwritten digit recognition dataset containing 60,000 training samples and 10,000 test samples from 10 classes. Each sample is a grayscale image with a size of 28×28. CIFAR-10 is a natural image classification dataset containing 50,000 training samples and 10,000 test samples from 10 object categories. Each image is an RGB image with a size of $32 \times 32 \times 3$. CIFAR-100 has the same image size and total number of samples as CIFAR-10, but contains 100 fine-grained classes. It consists of 50,000 training images and 10,000 test images, with 500 training images and 100 test images per class. Due to the larger number of categories and fewer samples per class, CIFAR-100 provides a more challenging benchmark for evaluating the scalability and robustness of federated learning methods.

To simulate statistical heterogeneity across clients, we adopt the Dirichlet-based Non-IID data partition strategy. Specifically, the class distribution of each client is sampled from a Dirichlet distribution with concentration parameter γ. A smaller γ indicates stronger Non-IID heterogeneity among clients, whereas a larger γ leads to a more balanced label distribution. Unless otherwise specified, the Dirichlet concentration parameter is set to $\gamma = 0.5$ by default in our experiments.

4.1.2 Models

To validate the generality and applicability of the proposed scheme, different neural network models were adopted for different datasets.

- ResNet-18: For the CIFAR-10 and CIFAR-100 dataset, ResNet-18 was employed as the backbone model. The output dimension of the final fully connected layer was modified to 10 to match the ten-class classification task.
- Simple CNN: For the MNIST dataset, a lightweight convolutional neural network consistent with that used in the baseline setting [23] was adopted for fair comparison. This model consists of two convolutional layers with 10 and 20 output channels, respectively, both using 5×5 kernels. Each convolutional layer is followed by a ReLU activation function and a max-pooling layer. Finally, two fully connected layers are used for classification.

4.1.3 Evaluation Metrics

To comprehensively evaluate the effectiveness and robustness of the proposed method, both model performance and system efficiency were assessed using the following metrics. Each experiment was repeated five times, and the average results were reported to reduce the influence of randomness and improve the reliability of the conclusions.

- Accuracy (Acc.): The classification accuracy of the global model on the test sets of MNIST and CIFAR-10.
- Training Time (Avg. Time): The average per-round training time of a single client device, measured in seconds.
- FLOPs: To provide a hardware-independent measure of computational cost, we additionally report the number of floating-point operations (FLOPs) of the final global model. FLOPs are measured based on

a single forward pass of the final model after training. For methods with structured pruning, the FLOPs are computed on the structurally compressed model, because removed filters or channels directly reduce the number of arithmetic operations. For methods with only unstructured pruning, dense FLOPs are reported unless specialized sparse acceleration is used, since unstructured sparsity does not necessarily lead to practical computational acceleration on general hardware. The FLOPs compression ratio is computed relative to the FedAvg baseline as FLOPs Comp. $= \frac{\text{FLOPs}_{\text{FedAvg}} - \text{FLOPs}_{\text{method}}}{\text{FLOPs}_{\text{FedAvg}}} \times 100\%$.

- Communication Size (Avg. Comm.): The average total server-side communication traffic per communication round, including both the downlink traffic from the server to the selected clients and the uplink traffic from the selected clients to the server. Since all clients share the same parameter ranking order, explicit sparse indices are not transmitted for each update. Small scalar metadata, such as retained-length indicators and training-state statistics, is not separately counted because its size is negligible compared with model/update tensors. The shared ranking descriptor is also not included in the reported communication size.

4.1.4 Baseline Methods

To evaluate the effectiveness and comparative advantages of the proposed scheme, we compare FedAHP with several representative federated learning and pruning-based baselines. These methods cover classical federated optimization, unstructured pruning, layer-wise pruning, and lottery-ticket-based federated pruning, which are closely related to the communication-efficient federated learning scenario considered in this paper.

- FedAvg [3]: A classical federated learning baseline. In each communication round, the server randomly selects a subset of clients and distributes the latest global model to them. After local training on their private datasets, the selected clients upload their updated model parameters to the server. The server then aggregates the received local models by weighted averaging to obtain the updated global model. Since FedAvg does not adopt any pruning or compression strategy, it serves as the reference method for evaluating the original communication cost and training performance.
- LotteryFL [18]: A federated learning approach inspired by the lottery ticket hypothesis. LotteryFL performs unstructured pruning on edge clients to obtain sparse subnetworks, thereby reducing the number of transmitted parameters and communication overhead. During training, each client uploads the locally updated model parameters together with the corresponding pruning mask, and the server aggregates only the retained parameters after pruning. In our experiments, the unstructured pruning ratio was set to 50%.
- FedLP [16]: A layer-wise pruning method for federated learning. During local training, pruning is performed at the layer level, and each client retains and uploads only a subset of layers. The server then aggregates model parameters layer by layer, thereby reducing both local computation cost and communication overhead. In the experiments, the layer dropout ratio was set to 50%.
- FedLTH [23]: A lottery-ticket-based federated pruning method. This method searches for sparse subnetworks with competitive performance by using iterative magnitude pruning (IMP) and weight rewinding. The obtained sparse subnetwork is then used for federated training to reduce the model size and communication burden. FedLTH aims to identify trainable subnetworks that can maintain model performance under a reduced parameter scale.

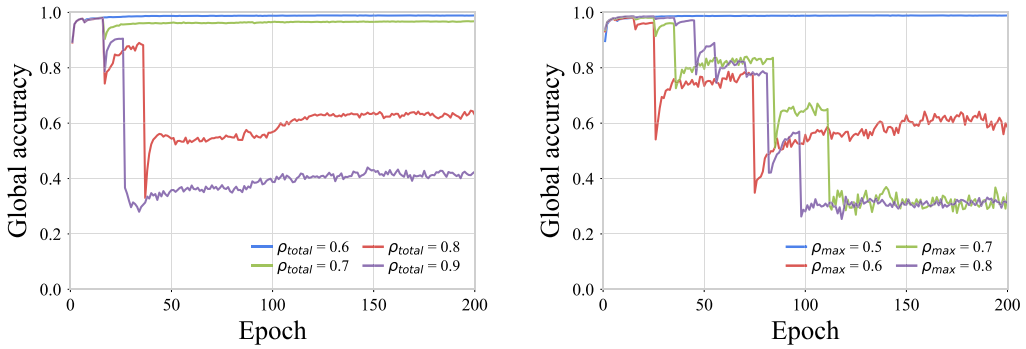
4.2 Parameter Sensitivity Analysis

Since different models and datasets are employed in the experiments, and both the unstructured pruning ratio, the upper bound of the structured pruning ratios affect the training dynamics, the final accuracy

of the global model may vary across different experimental settings. Therefore, this subsection presents a comparative analysis of the global model accuracy achieved by the proposed FedAHP scheme under different experimental scenarios, so as to evaluate the usability and effectiveness of the learned global model.

4.2.1 Impact of Different Pruning Ratios on Global Model Accuracy

To evaluate the impact of the upper bounds of structured and unstructured pruning on model accuracy, we conducted comparative experiments on the MNIST dataset under different values of ρ_{total} and $\rho_{\max}$. The corresponding results are shown in Fig. 2.



(**a**) Effect of ρ_{total} on global model accuracy.

(**b**) Effect of $\rho_{\max}$ on global model accuracy.

Figure 2: Impact of different pruning ratios on global model accuracy.

Fig. 2a illustrates the effect of different values of the upper bound of structured pruning ratio ρ_{total} under a fixed unstructured pruning upper bound $\rho_{\max} = 0.5$. The experimental results show that an appropriate level of structured pruning can compress the model while preserving training stability and model performance, whereas overly aggressive pruning leads to pronounced accuracy degradation. Specifically, when $\rho_{\text{total}} < 0.7$, structured pruning is triggered after the model has already achieved preliminary convergence in the early training stage, and no substantial loss in accuracy is observed. Even if short-term fluctuations occur, the model can quickly recover to its previous performance level, and the final global accuracy remains stable within approximately 98%–99%. This indicates that, within this range, progressive structured pruning can effectively reduce model redundancy without compromising discriminative capability or training stability.

In contrast, when $\rho_{\text{total}} > 0.7$, the model performance exhibits significant fluctuations. More specifically, the global accuracy drops sharply around communication rounds 25–40, followed by a prolonged recovery stage, and the final accuracy stabilizes at only 42%–63%. This suggests that excessively strong structured pruning removes too many critical channels or structural units, causing an abrupt change in the parameter space and driving the optimization trajectory away from its original direction. Although subsequent training can partially compensate for the resulting performance loss, the limited effective model capacity after excessive compression prevents the global model from recovering to the performance level achieved when $\rho_{\text{total}} < 0.7$.

Fig. 2b shows the influence of the upper bound of unstructured pruning ratio $\rho_{\max}$ on global model accuracy under a fixed structured pruning upper bound $\rho_{\text{total}} = 0.6$. The results indicate that when $\rho_{\max} = 0.5$, the model converges rapidly within a small number of communication rounds, with only minor fluctuations during subsequent training, and the final global accuracy remains stable at approximately 98.93%. This demonstrates that when $\rho_{\max} = 0.5$, the proposed adaptive pruning mechanism can effectively

remove redundant parameters while still preserving critical update information, and it also exhibits good compatibility with the dynamic privacy budget allocation mechanism.

By contrast, when $\rho_{\max} = 0.6$, although the model reaches relatively high accuracy in the early training stage, a clear phase-wise degradation and oscillation appear in the middle and later stages of training, and the final accuracy stabilizes at only about 58%–62%. This indicates that the pruning strength has exceeded the tolerable range of the system, resulting in insufficient transmission of effective update information and a larger aggregation bias in the global model.

Therefore, in order to achieve a better trade-off between model accuracy and efficiency optimization, the subsequent experiments in this paper set the upper bound of structured pruning ratio to $\rho_{\text{total}} = 0.6$ and the upper bound of unstructured pruning ratio to $\rho_{\max} = 0.5$.

4.2.2 Effect of Different Balancing Coefficients α on Global Model Accuracy

To evaluate the impact of the balancing coefficient α on the experimental results, we conduct a sensitivity analysis on the MNIST dataset. Specifically, we compare different values of α, i.e., $\alpha \in \{0, 0.25, 0.5, 0.75, 1.0\}$. Among them, $\alpha = 0$ indicates that the training state is determined only by the gradient norm, $\alpha = 1$ indicates that the training state is determined only by the training loss, and $\alpha = 0.5$ indicates that the training loss and the gradient norm are assigned equal importance.

As shown in Fig. 3, FedAHP achieves the best accuracy of 98.93% when $\alpha = 0.5$, indicating that training loss and gradient norm are complementary for estimating the client training state. The former reflects the fitting error, while the latter captures the update intensity and convergence trend, providing a more reliable basis for adaptive pruning.

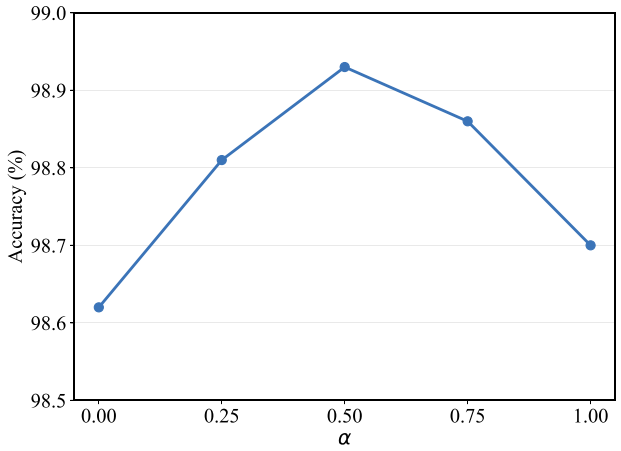


Figure 3: Effect of the balancing coefficient α on global model accuracy.

In contrast, using only the gradient norm ($\alpha = 0$) or only the training loss ($\alpha = 1.0$) reduces the accuracy to 98.62% and 98.70%, respectively, since a single indicator cannot fully characterize heterogeneous local optimization states. The results at $\alpha = 0.25$ and $\alpha = 0.75$ remain high, reaching 98.81% and 98.86%, respectively, showing the robustness of FedAHP. Therefore, $\alpha = 0.5$ is adopted in the subsequent experiments.

4.2.3 Sensitivity Analysis of the Stability Threshold and Detection Window

To verify the rationality of the structured pruning trigger condition, this paper further conducted sensitivity experiments on the stability threshold r_{stab} and the detection window m on the MNIST dataset. The experimental results are shown in Table 1.

Table 1: Sensitivity analysis of the stability threshold r_{stab} and detection window m.

Parameter	Value	Acc. (%)	First Pruning Round	Number of Structured Prunings
r_{stab}	0.001	98.68	34	3
	0.002	98.73	27	4
	0.003	98.66	23	4
	0.005	98.41	18	5
m	3	98.51	24	5
	5	98.73	27	4
	7	98.69	31	4
	10	98.57	39	3

To analyze the sensitivity of the structured pruning trigger, we varied the stability threshold r_{stab} and the detection window length m, as shown in Table 1. First, with $m = 5$ fixed, $r_{\text{stab}} \in \{0.001, 0.002, 0.003, 0.005\}$ was tested. As r_{stab} increases, structured pruning is triggered earlier. Specifically, when $r_{\text{stab}} = 0.001$, the first pruning occurs at round 34 with a final accuracy of 98.68%. When $r_{\text{stab}} = 0.002$, pruning is advanced to round 27 and achieves the highest accuracy of 98.73%. Further increasing r_{stab} to 0.003 and 0.005 triggers pruning too early, reducing the final accuracy to 98.66% and 98.41%, respectively. Thus, $r_{\text{stab}} = 0.002$ provides a better balance between pruning timeliness and training stability.

Next, with $r_{\text{stab}} = 0.002$ fixed, $m \in \{3, 5, 7, 10\}$ was evaluated. A larger m makes the stability criterion more conservative and delays structured pruning. When $m = 3$, pruning starts at round 24 with five pruning operations and 98.51% final accuracy. When $m = 5$, pruning starts at round 27 with four pruning operations and the best accuracy of 98.73%. Increasing m to 7 and 10 further delays the first pruning to rounds 31 and 39, with final accuracies of 98.69% and 98.57%, respectively. These results indicate that a short window is sensitive to transient fluctuations, while an overly long window delays compression and weakens its efficiency benefits.

Overall, r_{stab} and m jointly control the conservativeness of the structured pruning trigger. Aggressive settings may trigger pruning before the optimization direction becomes stable, whereas overly conservative settings delay model compression. Therefore, $r_{\text{stab}} = 0.002$ and $m = 5$ are adopted as the default settings, as they achieve a favorable trade-off among accuracy, stability, and compression efficiency.

4.2.4 Robustness Analysis under Different Degrees of Data Heterogeneity

To further evaluate the robustness of FedAHP under different degrees of data heterogeneity, we extend the original experimental setup by considering four data partition schemes, including IID, Dirichlet Non-IID ($\gamma = 0.5$), Dirichlet Non-IID ($\gamma = 0.1$), and pathological Non-IID (2 classes per client). The global model accuracy of all methods under these scenarios is compared to systematically assess the performance of FedAHP across different heterogeneity levels.

Fig. 4 presents the experimental comparison results under different degrees of data heterogeneity. As the heterogeneity becomes stronger, the accuracy of all methods decreases to different extents. In particular,

FedAvg, LotteryFL, and FedLP are more sensitive to heterogeneous data distributions, and their accuracy drops more significantly as the skewness of client label distributions increases. FedLTH shows a certain level of robustness in most cases, but it still experiences noticeable performance degradation under extremely heterogeneous settings. In contrast, FedAHP consistently maintains relatively high test accuracy under different data partition schemes. This is mainly because FedAHP preserves parameter information that is more critical to the global model while compressing redundant updates, thereby effectively mitigating the aggregation errors caused by biased local updates when client data distributions differ significantly. Consequently, its performance degradation is relatively smaller, demonstrating stronger robustness.

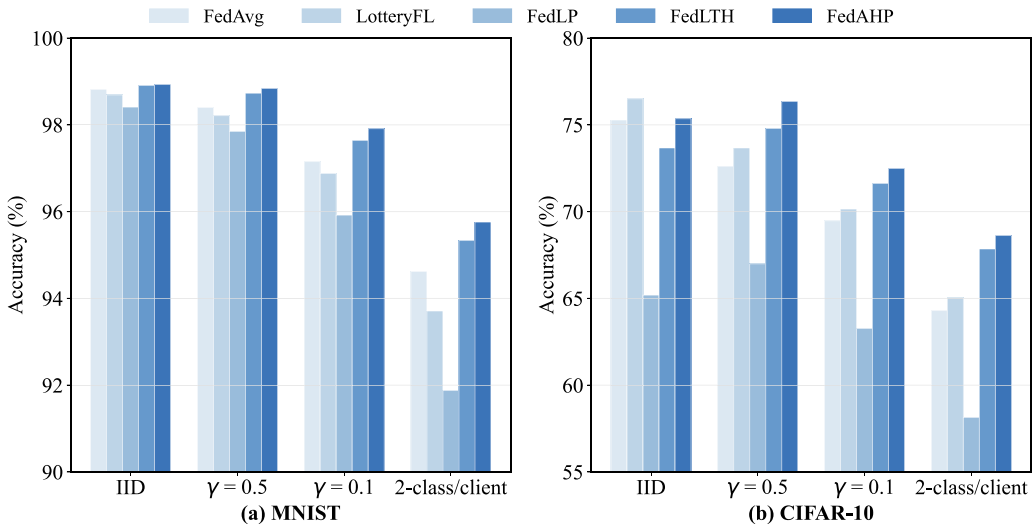


Figure 4: Accuracy comparison of different methods on MNIST and CIFAR-10.

4.2.5 Scalability Analysis with Different Numbers of Clients

To evaluate the scalability of FedAHP under different federated settings, we conduct experiments on MNIST under the Non-IID partition by varying the number of clients from 10 to 100. The setting with 30 clients corresponds to the default configuration used in the main experiments. All other hyperparameters, including local epochs, batch size, optimizer, learning rate, communication rounds, and pruning-related settings, are kept unchanged.

Table 2 summarizes the results. As the number of clients increases, the accuracy of FedAHP decreases only slightly from 98.82% to 98.39%, while remaining at a high level. This indicates that the proposed adaptive pruning and heterogeneous aggregation mechanisms can still maintain stable performance under larger client scales. In addition, the communication cost remains low and nearly unchanged, ranging from 1.22 to 1.27 MB. Although the training time increases as more clients participate, the FLOPs remain stable, showing that the computational overhead is well controlled. Overall, these results demonstrate that FedAHP has good scalability in federated learning scenarios with varying numbers of clients.

Table 2: Scalability analysis of FedAHP with different numbers of clients on MNIST under the Non-IID setting.

Number of Clients	Acc. (%) ↑	Avg. Comm. (MB) ↓	Avg. Time (s) ↓	FLOPs ↓
10	98.82 ± 0.06	1.22 ± 0.02	27.86 ± 0.83	0.49 M
30	98.73 ± 0.07	1.24 ± 0.02	30.69 ± 0.91	0.50 M
50	98.55 ± 0.09	1.25 ± 0.03	38.42 ± 1.12	0.51 M
100	98.39 ± 0.12	1.27 ± 0.03	46.85 ± 1.38	0.52 M

Note: Results are reported as mean ± standard deviation over five independent runs.

4.3 Comparative Experimental Results

This subsection presents a comprehensive comparison between FedAHP and the baseline methods across different models, datasets, and data distribution settings.

To comprehensively evaluate the overall performance of the proposed FedAHP scheme, we compared its global model accuracy with those of the four baseline methods under the same experimental settings. The experimental results are presented in Fig. 5 and Table 3.

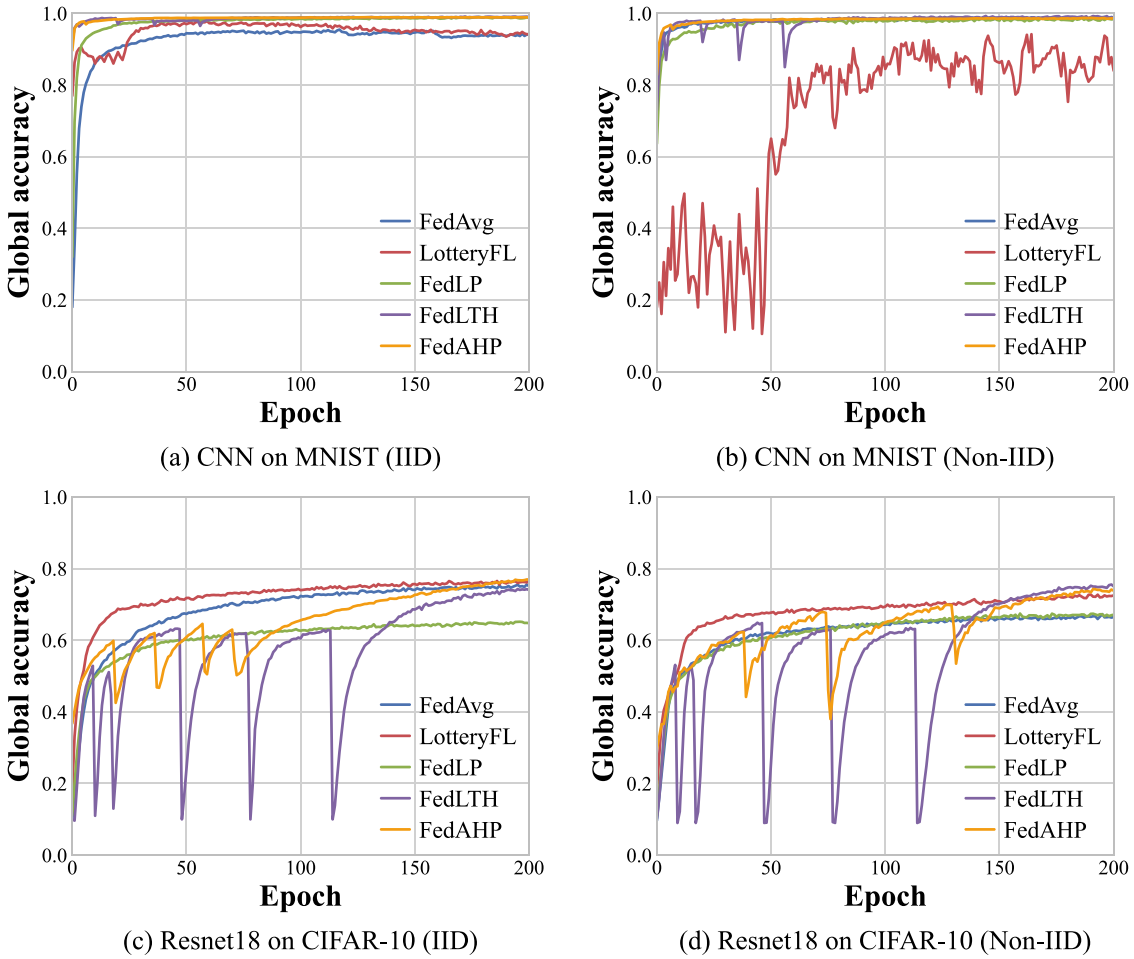


Figure 5: Accuracy comparison of different methods on MNIST and CIFAR-10.

Table 3: Overall performance comparison under IID and Non-IID settings.

Setting	Method	Acc. (%) ↑	Avg. Comm. (MB) ↓	Avg. Time (s) ↓	FLOPs ↓
MNIST (IID)	FedAvg	96.67 ± 0.18	3.42 ± 0.00	85.88 ± 1.26	1.18 M
	LotteryFL	97.15 ± 0.22	2.83 ± 0.03	97.24 ± 1.84	1.18 M
	FedLP	98.91 ± 0.07	2.20 ± 0.04	72.85 ± 1.13	1.18 M
	FedLTH	**98.98 ± 0.05**	1.65 ± 0.02	72.09 ± 1.05	0.64 M
	FedAHP	98.93 ± 0.06	**1.21 ± 0.02**	**26.79 ± 0.84**	**0.49** M
MNIST (Non-IID)	FedAvg	**98.84 ± 0.10**	3.42 ± 0.00	86.21 ± 1.35	1.18 M
	LotteryFL	94.89 ± 0.51	2.86 ± 0.04	106.85 ± 2.06	1.18 M
	FedLP	98.40 ± 0.13	2.20 ± 0.01	83.78 ± 1.44	1.18 M
	FedLTH	98.69 ± 0.08	1.68 ± 0.02	74.06 ± 1.18	0.65 M
	FedAHP	98.73 ± 0.07	**1.24 ± 0.03**	**30.69 ± 0.91**	**0.50** M
CIFAR-10 (IID)	FedAvg	75.28 ± 0.41	1572.25 ± 0.00	176.06 ± 3.12	557.00 M
	LotteryFL	**76.50 ± 0.36**	1218.12 ± 14.36	212.49 ± 4.28	557.00 M
	FedLP	65.19 ± 0.78	1099.90 ± 12.81	180.92 ± 3.55	557.00 M
	FedLTH	73.67 ± 0.52	729.95 ± 8.74	169.99 ± 3.09	310.42 M
	FedAHP	76.35 ± 0.33	**405.76 ± 6.92**	**82.02 ± 2.15**	**232.38** M
CIFAR-10 (Non-IID)	FedAvg	66.08 ± 0.65	1572.25 ± 0.00	183.34 ± 3.46	557.00 M
	LotteryFL	72.62 ± 0.58	1240.35 ± 15.27	210.06 ± 4.61	557.00 M
	FedLP	66.99 ± 0.73	1099.90 ± 11.94	179.70 ± 3.37	557.00 M
	FedLTH	74.78 ± 0.44	727.35 ± 9.16	182.74 ± 3.68	315.76 M
	FedAHP	**75.35 ± 0.39**	**488.62 ± 7.84**	**112.48 ± 2.73**	**248.11** M

Note: Accuracy, communication size, and training time are reported as mean ± standard deviation over five independent runs. FLOPs are measured on the final global model. For unstructured pruning methods, dense FLOPs are reported unless specialized sparse acceleration is used. The best result under each setting is highlighted in bold.

First, as can be observed from Table 3, FedAHP consistently maintains a relatively high performance level under all four experimental settings. Across the four scenarios, namely MNIST (IID), MNIST (Non-IID), CIFAR-10 (IID), and CIFAR-10 (Non-IID), the average accuracy of FedAHP reaches 87.34%, which is higher than that of FedLTH (86.53%), LotteryFL (85.29%), FedAvg (84.22%), and FedLP (82.37%). Among these methods, FedAvg, LotteryFL, and FedLP are more sensitive to data heterogeneity and exhibit more obvious accuracy degradation, whereas FedLTH and FedAHP experience only slight performance fluctuations. This indicates that FedAHP has stronger adaptability to heterogeneous data distributions and can maintain competitive performance under different task complexities and data distribution settings.

Second, the accuracy evolution curves in Fig. 5 show that different methods exhibit clearly different behaviors under IID and Non-IID data distributions, and such differences become more pronounced as the task complexity increases. On the relatively simple MNIST dataset, all methods can eventually achieve high accuracy, but they differ in convergence speed and training stability. On the more challenging CIFAR-10 dataset, the performance gap and fluctuation characteristics among different methods become more evident.

On the MNIST dataset, all methods converge rapidly. Since the MNIST task is relatively simple, the effect of data heterogeneity on the final model accuracy is limited even under the Non-IID setting, although its impact on training stability is still noticeable. As can be seen from Fig. 5a,b, FedAHP exhibits a smoother convergence process and is less affected by data heterogeneity. In contrast, FedAvg converges relatively slowly

in the early training stage. LotteryFL performs personalized pruning at the client side, and the pruning masks differ across clients, which weakens the consistency of global model updates and causes visible accuracy fluctuations during training, especially under the Non-IID setting. FedLP reduces communication by randomly dropping a subset of network layers during transmission; however, this may also discard part of the useful update information, thereby slowing down convergence in the early training stage.

On the CIFAR-10 dataset, due to the higher task complexity, the differences in convergence speed among methods become more pronounced. As shown in Fig. 5c,d, both FedLTH and FedAHP exhibit stage-wise fluctuations during training. This is because the weight rewinding mechanism in FedLTH and the progressive structured pruning strategy in FedAHP both require multiple pruning operations before reaching the target pruning level, which may induce temporary accuracy drops during the training process. Nevertheless, owing to the milder pruning schedule with decaying intensity adopted in FedAHP, its fluctuation amplitude is smaller than that of FedLTH. Moreover, LotteryFL adopts fine-grained unstructured pruning, while FedLP preserves layer-level structural integrity and performs layer-wise aggregation. Therefore, the accuracy fluctuations of these two methods are relatively smaller, and their training curves appear smoother. However, their final accuracies remain lower than that of FedAHP, indicating that smoother training dynamics do not necessarily lead to better ultimate model performance.

In addition to the FLOPs reduction of the final model, we also analyze the additional computational overhead introduced by the pruning strategy. On the client side, FedAHP mainly introduces training-state estimation and important update selection. These operations do not require training an additional pruning network and do not introduce extra mask or sparse index transmission. Therefore, the additional overhead is relatively small compared with local forward and backward propagation. On the server side, the counter-based heterogeneous aggregation mechanism only maintains an accumulation vector and a count vector for update alignment. Its computational cost grows approximately linearly with the number of participating clients and the number of received update parameters, which makes it suitable for large-scale federated learning scenarios. From the perspective of practical deployment, FedAHP is suitable for resource-constrained edge environments. It reduces uplink communication, supports heterogeneous pruning ratios among clients, and further compresses the global model through structured pruning after training becomes stable. As a result, the computation and storage costs of subsequent local training and edge deployment can be reduced.

Overall, the proposed FedAHP achieves the highest average accuracy across the four experimental settings. Compared with the baseline methods, it improves the average accuracy by 0.81%–4.97%. Although FedAHP is not always the best in every single scenario, it consistently maintains competitive accuracy while achieving the lowest communication cost, training time, and FLOPs in all settings. These results demonstrate that the proposed efficiency optimization strategy can effectively reduce system overhead without disrupting the normal training process or significantly degrading model performance.

4.4 Communication Overhead Analysis

As reported in Table 3, FedAHP achieves the lowest communication overhead under all experimental settings. Compared with the baseline methods, the per-round communication size of FedAHP is reduced by 0.44–2.21 MB, 0.44–2.18 MB, 324.19–1166.49 MB, and 238.73–1083.63 MB in the four scenarios considered above, respectively. These results demonstrate that FedAHP exhibits strong communication compression capability under different data distributions and task complexities. In particular, for more challenging tasks and larger-scale models, such as CIFAR-10 with ResNet-18, the communication reduction becomes even more substantial.

Fig. 6 compares the communication compression performance of FedAHP and the baseline methods during federated training. From the overall trends, the communication-size curves of different methods differ significantly throughout the training process. Among them, FedAvg does not employ any communication optimization strategy. Consequently, it needs to transmit the complete global model in both uplink and downlink at every communication round, and its communication size therefore remains almost constant over time.

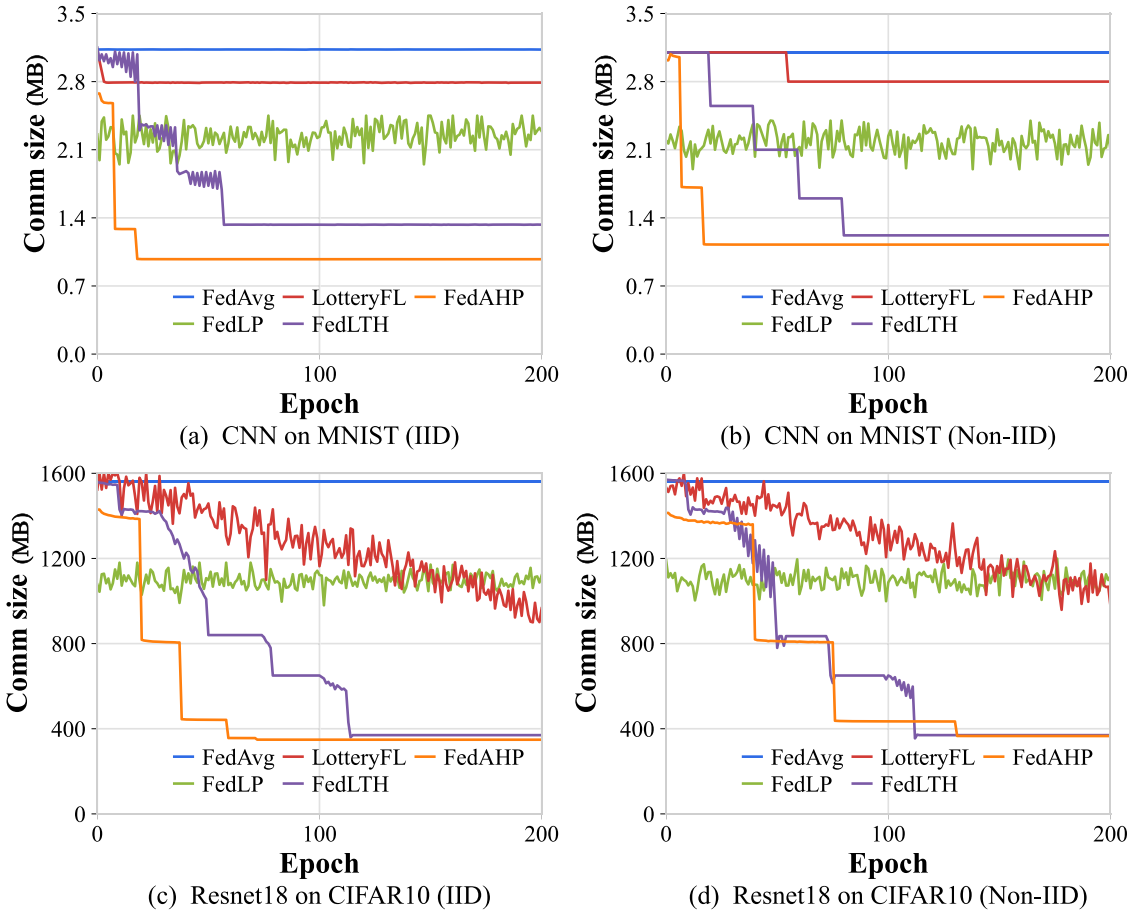


Figure 6: Communication size comparison of different methods on MNIST and CIFAR-10.

FedLP reduces the amount of uploaded parameters by performing random layer-wise aggregation. However, it does not explicitly optimize the communication cost of server-to-client model distribution. As a result, its overall communication reduction is limited, and its communication-size curve exhibits only mild fluctuations. LotteryFL progressively reduces the number of transmitted parameters through unstructured pruning. Nevertheless, because it requires additional transmission of pruning masks to record parameter locations, the actual reduction in communication overhead is smaller than the nominal 50% pruning ratio would suggest.

Although both FedAHP and FedLTH exhibit step-wise descending communication trends, the proposed FedAHP differs fundamentally from FedLTH in its communication mechanism. Unlike the LTH-based iterative pruning and rewinding strategy adopted in FedLTH, FedAHP does not require clients to transmit additional pruning masks or sparse index information when uploading local model updates. Instead, it relies on a globally shared ranking criterion and uploads only the most important update

parameters, thereby significantly reducing client-side uplink communication overhead. Moreover, after the global model gradually converges, FedAHP further performs progressive structured pruning on the server side, which continuously reduces the size of the global model distributed to clients. Therefore, the overall communication overhead of FedAHP is consistently lower than that of FedLTH.

Overall, FedAHP exhibits a clear advantage in reducing communication overhead during federated training. By introducing a hybrid pruning strategy guided by client training states, the proposed method achieves the lowest communication size across different experimental scenarios while preserving competitive model performance.

4.5 Training Time Analysis

To evaluate the time overhead of FedAHP in federated learning systems, we compared it with the baseline methods under the four experimental scenarios. The results are shown in Fig. 7 and Table 3. As can be observed from Table 3, under all experimental settings, FedAHP reduces the per-round training time by 58.56%–72.45% on MNIST and by 37.41%–61.40% on CIFAR-10, compared with the baseline methods.

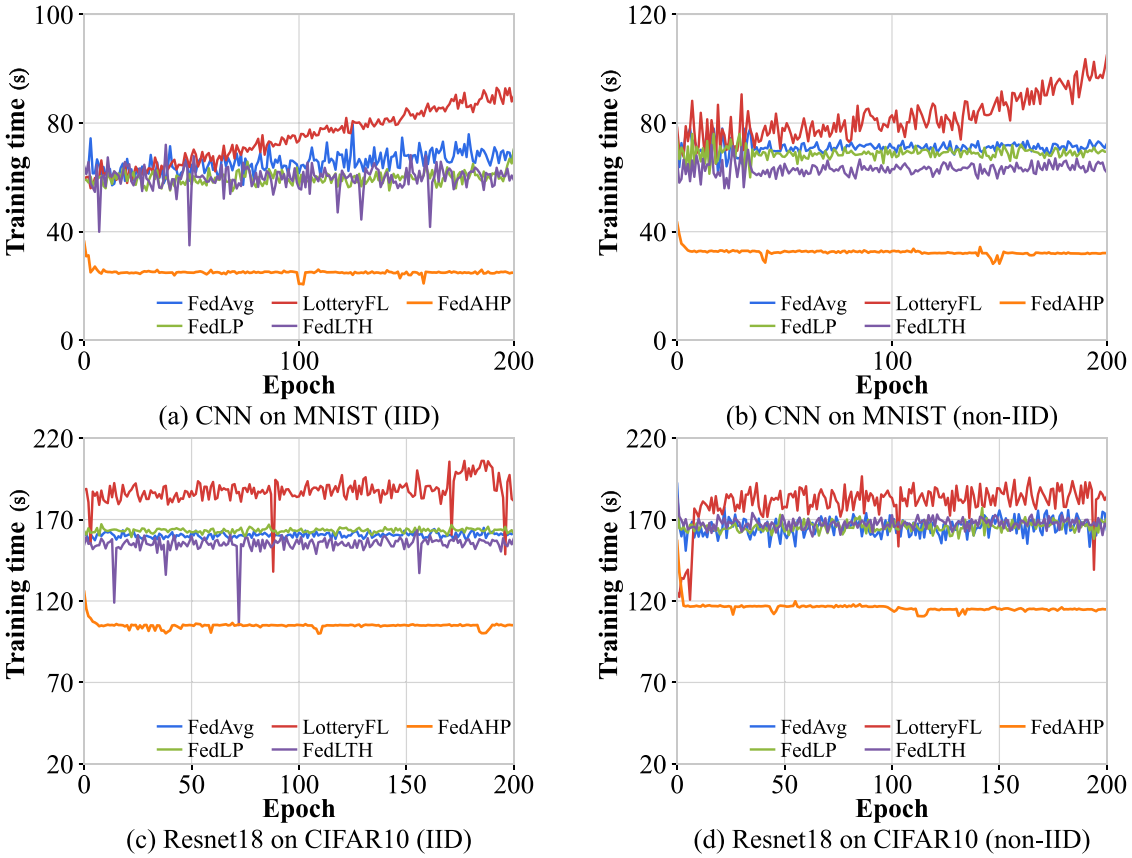


Figure 7: Training time comparison of different methods on MNIST and CIFAR-10.

As shown in Fig. 7, different methods exhibit clear differences in training time during the learning process. These differences mainly arise from the distinct local training procedures, model aggregation mechanisms, and pruning strategies adopted by the compared methods.

FedAvg performs local forward and backward propagation on the complete model in each communication round, and therefore incurs relatively high training time. Although FedLP reduces the number of

uploaded parameters to some extent, each client still updates the complete model during local training. As a result, its training time remains close to that of FedAvg. LotteryFL reduces the number of transmitted parameters through unstructured pruning; however, the resulting irregular sparse structure is difficult to accelerate efficiently on general-purpose hardware and existing deep learning frameworks. In addition, mask-based model aggregation and the maintenance of pruning masks further increase the computational burden, making LotteryFL the most time-consuming method among the compared approaches.

Although FedLTH can significantly reduce communication overhead through structured pruning, its iterative pruning, rewinding, and retraining procedures introduce additional training costs, thereby limiting its advantage in overall time efficiency. In contrast, FedAHP compensates for the limitations of unstructured pruning through progressive structured pruning, while avoiding explicit mask operations during aggregation. Owing to its mask-free aggregation mechanism and gradually compressed model structure, FedAHP is able to substantially shorten the overall training time. After a brief adjustment phase in the early stage of training, its training time quickly decreases and remains stable, with relatively small fluctuations throughout the entire training process.

Overall, FedAHP demonstrates a significant advantage in reducing the training time of federated learning. The proposed method achieves the shortest training time in all four experimental settings, while maintaining favorable stability and efficiency even in complex tasks, Non-IID data distributions, and large-scale model scenarios.

4.6 Extended Experiments

To validate the generalization capability of the proposed method on more complex tasks, we further conduct supplementary experiments on the CIFAR-100 dataset. Compared with CIFAR-10, CIFAR-100 contains more categories and presents a higher classification difficulty, thereby enabling a more comprehensive evaluation of the generalizability of different methods. For a fair comparison, we follow the main experimental settings used on CIFAR-10 and adopt ResNet-18 as the backbone model. The data partitioning strategy remains consistent with the previous experiments, where the Dirichlet distribution is used to simulate non-IID data partitions.

Table 4 summarizes the results on CIFAR-100. FedAHP achieves the highest accuracy under both IID and non-IID settings, reaching 73.08% and 70.41%, respectively. Meanwhile, it reduces the per-round communication cost to 506.37 MB and the training time to 117.31 and 118.92 s under IID and non-IID settings, respectively, outperforming all compared methods in system efficiency.

FedAHP also achieves the highest parameter and FLOPs compression ratios, i.e., 59.24% and 37.18%, respectively. This demonstrates that the proposed hybrid pruning strategy can reduce both transmission and computation costs. Compared with LotteryFL, FedLP, and FedLTH, FedAHP provides a better balance between model compression and accuracy preservation.

These improvements are mainly attributed to the combination of client-side adaptive unstructured update pruning and server-side progressive structured pruning. The former reduces redundant communication while preserving important local updates, and the latter further compresses the model structure after stable global training. Therefore, the CIFAR-100 results further verify the effectiveness, scalability, and practical value of FedAHP in more complex image classification tasks.

Table 4: Experimental results of different methods on the CIFAR-100 dataset.

Method	Acc.	Avg. Comm.	Avg. Time	FLOPs Comp.
		IID		
FedAvg	66.58%	1584.72 MB	189.64 s	0.00%
LotteryFL	70.74%	1261.45 MB	216.83 s	5.92%
FedLP	67.12%	1108.63 MB	184.77 s	12.84%
FedLTH	72.16%	742.88 MB	186.45 s	26.35%
FedAHP	73.08%	506.37 MB	117.31 s	37.18%
		Non-IID		
FedAvg	62.36%	1584.72 MB	191.83 s	0.00%
LotteryFL	67.18%	1261.45 MB	219.27 s	5.92%
FedLP	63.11%	1108.63 MB	186.94 s	12.84%
FedLTH	70.26%	742.88 MB	188.56 s	26.35%
FedAHP	70.41%	506.37 MB	118.92 s	37.18%

4.7 Ablation Study

To further investigate the individual contribution of each component in the proposed FedAHP framework, we conduct an ablation study on the MNIST dataset under both IID and Non-IID settings. All experimental settings, including the optimizer, learning rate, local epochs, and global communication rounds, are kept the same as those used in the main experiments. We compare the full FedAHP with four variants: FedAvg without pruning, FedAHP-S with structured pruning only, FedAHP-U with unstructured pruning only, and FedAHP-Fixed with hybrid pruning but without the adaptive pruning mechanism.

Table 5 reports the ablation results on MNIST. Compared with FedAvg, all pruning-based variants significantly reduce communication cost and training time. The full FedAHP reduces the communication size from 3.42 to 1.21 MB and the training time from 85.88 to 26.79 s under the IID setting, while achieving 98.93% accuracy. Under the Non-IID setting, it reduces the communication size from 3.42 to 1.24 MB and the training time from 86.21 to 30.69 s, while maintaining 98.73% accuracy.

Table 5: Ablation study of FedAHP on the MNIST dataset.

Setting	Method	Structured	Unstructured	Adaptive	Acc. (%) ↑	Comm. (MB) ↓	Time (s) ↓
MNIST (IID)	FedAvg	×	×	×	96.67 ± 0.18	3.42 ± 0.00	85.88 ± 1.26
	FedAHP-S	✓	×	×	98.52 ± 0.09	1.76 ± 0.02	38.64 ± 0.96
	FedAHP-U	×	✓	×	98.71 ± 0.08	1.89 ± 0.03	54.27 ± 1.12
	FedAHP-Fixed	✓	✓	×	98.36 ± 0.11	1.18 ± 0.02	29.83 ± 0.91
	FedAHP	✓	✓	✓	98.93 ± 0.06	1.21 ± 0.02	26.79 ± 0.84
MNIST (Non-IID)	FedAvg	×	×	×	98.84 ± 0.10	3.42 ± 0.00	86.21 ± 1.35
	FedAHP-S	✓	×	×	98.45 ± 0.12	1.79 ± 0.03	41.52 ± 1.03
	FedAHP-U	×	✓	×	98.57 ± 0.10	1.91 ± 0.03	57.38 ± 1.29
	FedAHP-Fixed	✓	✓	×	98.16 ± 0.15	1.20 ± 0.02	33.74 ± 1.06
	FedAHP	✓	✓	✓	98.73 ± 0.07	1.24 ± 0.02	30.69 ± 0.91

Note: FedAHP-S denotes the variant using only structured pruning, FedAHP-U denotes the variant using only unstructured pruning, and FedAHP-Fixed denotes the hybrid pruning variant without the adaptive pruning mechanism. Results are reported as mean ± standard deviation over five independent runs.

FedAHP-S shortens training time through structured pruning but may reduce model representation capability. FedAHP-U maintains competitive accuracy by preserving the model structure, but its irregular sparsity limits hardware acceleration. FedAHP-Fixed lowers communication cost, but its fixed pruning ratios cannot adapt to the training state and may discard useful updates. In contrast, the full FedAHP adaptively adjusts pruning intensity and progressively introduces structured pruning, achieving a better balance among accuracy, communication cost, and training time. These results confirm the complementarity of structured pruning, unstructured pruning, and the adaptive mechanism.

5 Conclusion

This paper proposed FedAHP, an efficiency-oriented federated learning framework based on adaptive hybrid pruning, to address the difficulty of simultaneously balancing training overhead, communication overhead, and model accuracy under conventional static pruning strategies. The proposed method dynamically adjusts the update pruning ratio according to the local training state of each client, thereby jointly improving communication efficiency and training efficiency. To aggregate heterogeneous sparse updates produced under different pruning ratios, a counter-based aggregation mechanism was introduced for update alignment in a shared ranking space. Furthermore, a performance-aware structured pruning strategy was employed after training became stable, so as to further compress the global model and improve deployment efficiency on edge devices.

Experimental results on MNIST and CIFAR-10 show that FedAHP achieves strong overall performance. In comparison with representative baselines, it not only yields superior model accuracy but also substantially reduces per-round communication overhead and training time, demonstrating its effectiveness and practical applicability in edge computing environments.

In future work, we will further conduct detailed system-level analysis, including memory overhead measurements and scalability evaluation in real large-scale federated learning deployments, among others.

Acknowledgement: The authors thank all research members who provided support and assistance in this study.

Funding Statement: This work was supported by the National Natural Science Foundation of China (Grant No. 62102449).

Author Contributions: Conceptualization, Mengdie Hu; Methodology, Mengdie Hu; Data collection: Mengdie Hu; Result analysis and interpretation: Mengdie Hu and Baidong Huang; Manuscript preparation: Mengdie Hu, Na Wang, XueHui Du and Kaiyuan Wang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The dataset used in this paper is the public dataset MNIST dataset, accessed as follows: MNIST, CIFAR-10, CIFAR-100.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. IEEE Internet Things J. 2016;3(5):637–46.
2. Satyanarayanan M. The emergence of edge computing. Computer. 2017;50(1):30–9.
3. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Singh A, Zhu J, editors. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS); 2017 Apr 20–22; Fort Lauderdale, FL, USA. p. 1273–82.

4. Zhou J, Wu N, Wang Y, Lyu F, Shen X. A differentially private federated learning model against poisoning attacks in edge computing. IEEE Trans Dependable Secur Comput. 2023;20(3):1941–58. doi:10.1109/tdsc.2022.3168556.
5. Arouj A, Abdelmoniem AM. Towards energy-aware federated learning via collaborative computing approach. Comput Commun. 2024;221(3):131–41. doi:10.1016/j.comcom.2024.04.012.
6. Alahmari S, Alghamdi I. A comprehensive survey on energy-efficient and privacy-preserving federated learning for edge intelligence and IoT. Results Eng. 2025;28(24):107849. doi:10.1016/j.rineng.2025.107849.
7. Dantas PV, Silva SW Jr, Cordeiro LC, Carvalho CB. A comprehensive review of model compression techniques in machine learning. Appl Intell. 2024;54(22):11804–44. doi:10.1007/s10489-024-05747-w.
8. Xu Y, Liao Y, Xu H, Ma Z, Wang L, Liu J. Adaptive control of local updating and model compression for efficient federated learning. IEEE Trans Mob Comput. 2023;22(10):5675–89. doi:10.1109/tmc.2022.3186936.
9. Internò C, Raponi E, van Stein N, Bäck T, Olhofer M, Jin Y, et al. Federated hybrid model pruning through loss landscape exploration. arXiv:2405.10271. 2025.
10. Wang S, Tuor T, Salonidis T, Leung K, Makaya C, He T, et al. When edge meets learning: adaptive control for resource-constrained distributed machine learning. In: Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications; 2018 Apr 15–19; Honolulu, HI, USA. Piscataway, NJ, USA: IEEE; 2018. p. 63–71.
11. Kim M, Saad W, Mozaffari M, Debbah M, Poor HV. On the tradeoff between energy, precision, and accuracy in federated quantized neural networks. In: Proceedings of the ICC 2022-IEEE International Conference on Communications; 2022 May 16–20; Seoul, Republic of Korea. Piscataway, NJ, USA: IEEE; 2022.
12. Mao Y, Zhao Z, Yan G, Song Y, Chen Y. Communication-efficient federated learning with adaptive quantization. ACM Trans Intell Syst Technol. 2022;13(4):Article 67.
13. Tao B, Chen C, Chen H. Communication efficient federated learning via channel-wise dynamic pruning. In: Proceedings of the 2023 IEEE Wireless Communications and Networking Conference (WCNC); 2023 Mar 26–29; Glasgow, UK. Piscataway, NJ, USA: IEEE; 2023.
14. Wu C, Wu F, Lyu L, Huang Y, Xie X. Communication-efficient federated learning via knowledge distillation. Nat Commun. 2022;13(1):2032. doi:10.1038/s41467-022-29763-x.
15. Liu HT, Wei Z, He RX. Communication-efficient layer-wise pruning algorithm for hierarchical federated learning. Telecommun Eng. 2026;66(2):247–58. (In Chinese). doi:10.20079/j.issn.1001-893x.241106003.
16. Zhu Z, Shi Y, Luo J, Wang F, Peng C, Fan P, et al. FedLP: layer-wise pruning mechanism for communication-computation efficient federated learning. In: Proceedings of the ICC 2023-IEEE International Conference on Communications; 2023 May 28–Jun 1; Rome, Italy. Piscataway, NJ, USA: IEEE; 2023. p. 1250–5.
17. Jiang Y, Wang S, Valls V, Ko BJ, Lee WH, Leung KK, et al. Model pruning enables efficient federated learning on edge devices. IEEE Trans Neural Netw Learn Syst. 2023;34(12):10374–86. doi:10.1109/tnnls.2022.3166101.
18. Li A, Sun J, Wang B, Duan L, Li S, Chen Y, et al. LotteryFL: personalized and communication-efficient federated learning with lottery ticket hypothesis on Non-IID datasets. arXiv:2008.03371. 2020.
19. Frankle J, Carbin M. The lottery ticket hypothesis: finding sparse, trainable neural networks. In: Proceedings of the 7th International Conference on Learning Representations (ICLR); 2019 May 6–9; New Orleans, LA, USA.
20. Hu HF, Zhang X, Zhao HT, Wei Y, Wang R. Communication-efficient model pruning algorithm for federated learning in mobile edge computing. Chin J Internet Things. 2024;8(3):112–26. (In Chinese).
21. Diao E, Ding J, Tarokh V. HeteroFL: computation and communication efficient federated learning for heterogeneous clients. In: Proceedings of the International Conference on Learning Representations (ICLR); 2021 May 3–7; Virtual.
22. Lv M, Zhang H, Wang M, Wang L, Wu P. Democratic learning: a distributed machine learning framework with collaborative voting and model pruning for privacy and security. In: Proceedings of the 2024 International Joint Conference on Neural Networks (IJCNN); 2024 Jun 30–Jul 5; Yokohama, Japan. Piscataway, NJ, USA: IEEE; 2024.
23. Zhang H, Xie Y, Hu S, He M, He P, Zheng J, et al. FedLTH: a privacy-preserving federated learning framework with model pruning on edge clients. In: Proceedings of the 2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS); 2025 Jul 20–23; Glasgow, UK. p. 384–94.