Auto-HFL: Automated Sub-Network Discovery for Heterogeneous Federated Learning

1st Ying Qian
College of software
Northeastern University, Shenyang, China
qianying@stumail.neu.edu.cn

2nd Lianbo Ma
College of software, Northeastern University, China
Foshan Graduate School of Innovation, Northeastern University, China
malb@swc.neu.edu.cn

***Abstract*—The deployment of Large Machine Learning Models (LMLMs) on edge devices is hindered by the conflict between the colossal size of modern architectures and the limited, heterogeneous resources of edge clients. Federated Learning (FL) enables privacy-preserving collaborative training, yet standard FL protocols typically assume model homogeneity, making them unsuitable for diverse edge environments. In this paper, we propose Auto-HFL, a framework for Automated Sub-Network Discovery. Unlike traditional Neural Architecture Search (NAS) which incurs high computational costs, Auto-HFL introduces a lightweight, resource-aware discovery mechanism. It allows clients to automatically extract and train optimal sub-networks from a global supernet based on their real-time hardware constraints (e.g., memory and bandwidth). We further design a specialized aggregation strategy to handle the structural heterogeneity of the discovered models. Extensive experiments on [Insert Dataset, e.g., CIFAR-10/100] demonstrate that Auto-HFL significantly reduces communication and computation overhead while maintaining accuracy comparable to full-model training. This work paves the way for automating model deployment in the era of large-scale edge intelligence.**

***Index Terms*—Federated Learning, Automated Model Discovery, Edge Computing, Model Pruning, Heterogeneous Systems**

I. Introduction

The landscape of artificial intelligence has been fundamentally reshaped by the proliferation of Large Models. These architectures, exemplified by Large Language Models and Vision Transformers (ViTs), have scaled to billions of parameters and demonstrated emergent abilities across a spectrum of complex tasks [1]–[4]. However, the ubiquity of Large Models is hindered by a critical physical bottleneck regarding the disparity between the massive computational requirements of these models and the constrained resources of edge devices. Deploying such high-capacity models on endpoints like mobile phones, IoT sensors, and autonomous vehicles is often infeasible due to the "memory wall" where the memory bandwidth and capacity of the device fail to accommodate the sheer volume of model weights and activation states [5]–[7]. While parameter-efficient techniques such as Low-Rank Adaptation (LoRA) [8] have been proposed, they often assume a fixed rank across devices, failing to address the fundamental hardware heterogeneity.

Corresponding author: Lianbo Ma
This work is supported by National Natural Science Foundation of China under Grant 62472079.

Federated Learning (FL) has emerged as a promising paradigm to enable distributed training while mitigating privacy risks associated with centralized data storage [9]–[12]. In a standard FL protocol, participating clients collaboratively optimize a shared global model by exchanging local updates. While effective in preserving data privacy, traditional algorithms operate under a strict assumption of model homogeneity. This paradigm mandates that the global model architecture remains identical across all clients. Such a constraint is fundamentally incompatible with the system heterogeneity prevalent in real-world edge networks [13]. In practice, edge devices exhibit vast diversity in hardware specifications ranging from high-performance GPUs in autonomous cars to microcontroller units in smart home appliances. Enforcing a uniform Large Model architecture inevitably leads to the straggler effect where the system latency is dictated by the slowest participant. Moreover, resource-constrained clients are frequently excluded from training entirely due to Out-Of-Memory errors which biases the global model and degrades its generalization capability due to non-IID data shifts [14]–[17].

To reconcile the conflict between Large Models and heterogeneous edges, recent research has explored static model scaling strategies. Techniques such as HeteroFL [18], FjORD [19], and FedDropout [20] permit clients to train partial sub-networks derived from a global supernet. Although these methods provide a degree of flexibility, they predominantly rely on coarse-grained and manually predefined scaling rules such as training with fixed width ratios. This static approach is suboptimal for two primary reasons. First, it fails to adapt to the dynamic and fine-grained fluctuations of available resources including varying battery levels or transient network congestion. Second, manual sub-model design does not guarantee the preservation of the most critical features which compromises the representation power of compressed models.

In the era of Large Models, the design space is too vast for manual engineering. There is an imperative need for an Automated Model Discovery mechanism that can dynamically tailor architectures to client-specific constraints. While Neural Architecture Search (NAS) offers a theoretical framework for automating design, its direct application to FL is obstructed by prohibitive computational costs [21]–[23]. Traditional search algorithms often require thousands of GPU-hours to converge which negates efficiency incentives of edge computing [24].

To bridge this gap, this paper proposes Automated Heterogeneous Federated Learning (Auto-HFL), a framework for automated sub-network discovery in heterogeneous federated learning environments. Departing from the expensive search operations of traditional approaches, Auto-HFL reformulates model discovery as a lightweight and resource-aware pruning problem. By employing a magnitude-based selection mechanism, clients can autonomously identify and extract the optimal sub-network structure from the global Large Model that fits within their strict hardware budgets. Furthermore, to facilitate the aggregation of these structurally diverse sub-networks, a coordinate-aligned protocol is introduced to ensure the stable convergence of the global model.

The contributions of this paper are summarized as follows:

- A lightweight Automated Sub-Network Discovery mechanism is proposed which enables the dynamic extraction of client-specific architectures from Large Models and eliminates the need for manual scaling rules or expensive search controllers.
- A Heterogeneous Aggregation Protocol is developed to handle the accumulation of gradients from non-uniform sub-networks and addresses the structural misalignment inherent in heterogeneous FL.
- Extensive experiments on standard benchmarks show that Auto-HFL significantly reduces communication overhead and computational FLOPs while achieving accuracy comparable to full-model training baselines.

II. Related Work

A. Heterogeneity-Aware Federated Learning

The deployment of FL in real-world edge networks is fundamentally constrained by system heterogeneity, where participating devices exhibit significant disparities in hardware specifications including computing power, memory capacity, and network bandwidth. Standard algorithms strictly enforce model homogeneity, causing the global convergence rate to be bottlenecked by the slowest clients. To address this, early methodologies employed Split Learning, which partitions the computational graph between the client and the server [25], [26]. However, the frequent transmission of intermediate activation maps and gradients incurs massive communication overhead which renders it susceptible to network latency [27].

A distinct line of research leverages Knowledge Distillation (KD) to accommodate heterogeneity. Methods such as FedMD [28], FedGen [29] and FedDF [30] enable clients to train disparate architectures by transferring knowledge via soft labels or logits. While promising, KD-based FL typically necessitates a shared public dataset or a generator on the server to align student-teacher distributions. This requirement not only introduces additional computational burdens but also poses potential privacy risks by violating the data-minimization principle of FL.

Consequently, Model Pruning has emerged as a dominant solution. State-of-the-art frameworks, exemplified by HeteroFL [18], FjORD [19], and PruneFL [31], allow clients to train sub-networks derived from a global supernet. Recent works like FedRolex [32] introduce rolling sub-networks to alleviate bias. However, these methods typically employ static, heuristic-based scaling rules, assigning fixed width multipliers (e.g., $0.5\times$) to clients based on coarse resource categorization. This approach is suboptimal because it is content-agnostic: channels are pruned based on predefined indices rather than their actual contribution to the model's performance. Furthermore, it is static and prevents the architecture from adapting dynamically to transient fluctuations in device status in highly dynamic edge environments.

B. Automated Model Discovery and Zero-Cost NAS

Neural Architecture Search (NAS) aims to automate the design of optimal network topologies to replace manual feature engineering [33]. In the context of deep learning, differentiable NAS methods, such as DARTS [34] and ProxylessNAS [35], relax the discrete search space into continuous operations to solve a bi-level optimization problem. While effective in server-side scenarios, directly porting these gradient-based search algorithms to FL is impractical. The search process typically requires thousands of GPU-hours and necessitates synchronizing architecture parameters across clients which incurs prohibitive communication costs and energy consumption for edge devices [36]–[38].

To circumvent the heavy cost of training-based search, the concept of Zero-Cost Proxies has gained traction. Techniques such as SNIP [39], GraSP [40], and SynFlow [41] estimate the saliency of neural connections at initialization using logical indicators like gradient magnitude or synaptic flow preservation without requiring a full training cycle. Mellor et al. proposed NASWOT [42] which scores architectures based on the overlap of activation maps. These proxies correlate highly with the final accuracy of the architecture [43], [44]. Auto-HFL draws inspiration from this paradigm but repurposes it for the federated setting. Instead of searching for complex topology cells, we formulate model discovery as a resource-constrained sub-network extraction task. By leveraging magnitude-based importance scoring as a zero-cost proxy, our framework enables clients to instantaneously discover and extract the optimal sub-structure from the Large Model. This eliminates the need for expensive search controllers while ensuring that the retained sub-networks are theoretically grounded in feature importance, effectively bridging the gap between rigid manual scaling and expensive NAS.

III. Methodology

In this section, we provide a comprehensive mathematical formulation of the proposed **Automated Heterogeneous Federated Learning (Auto-HFL)** framework. As illustrated in Fig. 1, Auto-HFL is designed to bridge the gap between large-scale deep learning models and resource-constrained edge devices. We first introduce the system model and mathematically characterize the hardware heterogeneity. Subsequently, we detail the *Automated Sub-Network Discovery* mechanism,

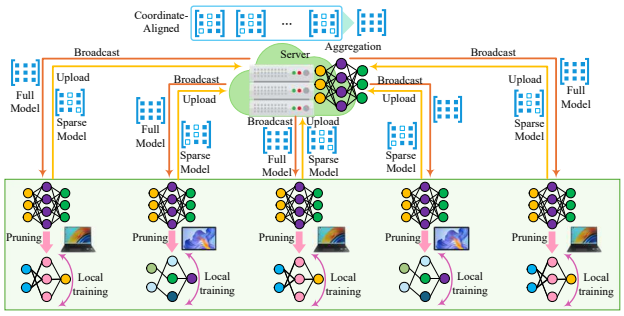

Fig. 1. **The framework of the proposed Auto-HFL.** The server broadcasts the global supernet to heterogeneous clients. Each client performs resource-aware automated discovery to prune the model into a sub-network (sparse model) that fits its budget. After local training, clients upload sparse updates, which are then aggregated via the Coordinate-Aligned Aggregation protocol to update the global model.

focusing on how zero-cost proxies allow for efficient architecture search. Finally, we propose the *Coordinate-Aligned Aggregation* protocol to rectify the gradient bias introduced by structural heterogeneity.

A. Preliminaries and Problem Formulation

1) Federated Learning Setting: We consider a cross-device federated learning system comprising a central server and a set of K heterogeneous clients, indexed by $\mathcal{K} = \{1, 2, \ldots, K\}$. Each client k possesses a private local dataset $\mathcal{D}_k = \{(x_i^{(k)}, y_i^{(k)})\}_{i=1}^{|\mathcal{D}_k|}$, sampled from a distinct distribution $\mathcal{P}_k$. We assume the data distributions are *Non-IID* (Non-Independently and Identically Distributed), meaning $\mathcal{P}_k \neq \mathcal{P}_j$ for $k \neq j$.

The central server maintains a global supernet model parameterized by $\mathbf{W} \in \mathbb{R}^d$, where d denotes the total number of trainable parameters. In standard Federated Averaging (FedAvg), the objective is to minimize the global empirical risk function $F(\mathbf{W})$:

$$\min_{\mathbf{W}} F(\mathbf{W}) \triangleq \sum_{k=1}^{K} p_k \mathcal{L}_k(\mathbf{W}; \mathcal{D}_k), \tag{1}$$

where $p_k = |\mathcal{D}_k| / \sum_j |\mathcal{D}_j|$ is the weight of client k, and $\mathcal{L}_k$ is the local loss function defined as $\mathcal{L}_k(\mathbf{W}) = \mathbb{E}_{(x,y)\sim\mathcal{D}_k}[\ell(f(x; \mathbf{W}), y)]$.

2) Characterizing System Heterogeneity: Unlike ideal simulation environments, real-world edge devices exhibit significant diversity in hardware capabilities. We rigorously define three dimensions of resource constraints for client k:

1) Memory Capacity ($B_k^{(mem)}$): The maximum RAM available for storing model parameters and activations. This is the hard bottleneck preventing weak devices from loading Large Models.

2) Computational Power ($B_k^{(comp)}$): Measured in FLOPs (Floating Point Operations per Second), determining the local training latency.

3)Communication Bandwidth ($B_k^{(comm)}$): The uplink/downlink throughput limits.

Let $\Psi(\mathbf{W}; \mathbf{M})$ be a function estimating the resource consumption of a model $\mathbf{W}$ masked by a binary matrix $\mathbf{M} \in \{0, 1\}^d$. Standard FL requires $\Psi(\mathbf{W}; \mathbf{1}) \leq B_k^{(mem)}$ for all clients, which is often infeasible. To address this, Auto-HFL reformulates FL as a *Constrained Bi-Level Optimization Problem*. Our goal is to jointly optimize the binary mask $\mathbf{M}_k$ (architecture) and the weights $\mathbf{W}_k$ (parameters) for each client:

$$\min_{\mathbf{W}, \{\mathbf{M}_k\}_{k=1}^K} \sum_{k=1}^{K} p_k \mathcal{L}_k(\mathbf{W} \odot \mathbf{M}_k; \mathcal{D}_k)$$
$$\text{s.t.} \Psi(\mathbf{W} \odot \mathbf{M}_k) \leq \min(B_k^{(mem)}, B_k^{(comp)}, B_k^{(comm)}), \forall k \in \mathcal{K} \tag{2}$$

where $\odot$ denotes the element-wise Hadamard product. This formulation explicitly couples model accuracy with strict hardware constraints.

B. Phase I: Automated Sub-Network Discovery

The optimization of the discrete mask $\mathbf{M}_k$ in Eq. (2) is NP-hard. Conventional Neural Architecture Search (NAS) methods (e.g., DARTS or Evolutionary Search) require repetitive training, incurring prohibitive costs for edge devices. To solve this efficiently, we propose a zero-cost proxy based on the Strong Lottery Ticket Hypothesis.

1) Resource-Adaptive Width Profiling: Before training, each client performs a lightweight profiling step. The client calculates a width scaling factor $\alpha_k \in (0, 1]$ that satisfies its specific budget. Consider a Convolutional Neural Network (CNN) layer l with input channels C_{in} and output channels C_{out}. The memory cost is approximately proportional to the number of parameters and activations. We solve for the maximal α_k such that:

$$\sum_{l=1}^{L} \text{Cost}\left(\lfloor \alpha_k C_{out}^{(l)} \rfloor, \lfloor \alpha_k C_{in}^{(l)} \rfloor\right) \leq B_k^{(mem)} \tag{3}$$

Once α_k is derived, the target channel width for layer l becomes $\hat{C}_k^{(l)} = \lfloor \alpha_k \cdot C_{out}^{(l)} \rfloor$. This step ensures that the resulting sub-network is theoretically executable on the device.

2) Magnitude-Based Sensitivity Analysis: Given the target width $\hat{C}_k^{(l)}$, the challenge is to select *which* channels to retain. We employ a magnitude-based pruning criterion, hypothesizing that filters with larger norms contribute more significantly to feature extraction. For each layer l, we compute a sensitivity score vector $\mathbf{s}^{(l)} \in \mathbb{R}^{C_{out}^{(l)}}$. The score for the j-th filter is defined as its ℓ_1-norm:

$$\mathbf{s}_j^{(l)} = ||\mathbf{W}_{j,:,:,:}^{(l)}||_1 = \sum_{c=1}^{C_{in}} \sum_{u=1}^{K_h} \sum_{v=1}^{K_w} |\mathbf{W}_{j,c,u,v}^{(l)}| \tag{4}$$

This calculation is computationally negligible ($O(d)$ complexity) and requires no gradient information, making it a zero-cost operation.

3) Deterministic Mask Generation: Based on the scores $\mathbf{s}^{(l)}$, we identify the indices of the top-$\hat{C}_k^{(l)}$ filters. Let $\tau_k^{(l)}$ be the value of the $\hat{C}_k^{(l)}$-th largest score in layer l. The binary mask is generated deterministically:

$$\mathbf{M}_{k,j}^{(l)} = \mathbb{I}(\mathbf{s}_j^{(l)} \geq \tau_k^{(l)}), \tag{5}$$

where $\mathbb{I}(\cdot)$ is the indicator function. The client-specific sub-network is then instantiated as $\mathbf{W}_k = \mathbf{W} \odot \mathbf{M}_k$. This ensures that every client extracts a dense, contiguous sub-network containing the winning tickets (most salient features) of the global supernet.

C. Phase II: Sparse Local Training

Upon deriving the sub-network, client k performs local training using Stochastic Gradient Descent (SGD). A critical distinction from standard FL is that gradients are computed *only* for the active parameters. Let $\mathbf{W}_k^{t,e}$ denote the local model parameters at communication round t and local epoch e. The update rule is:

$$\mathbf{W}_k^{t,e+1} = \mathbf{W}_k^{t,e} - \eta_l \nabla \mathcal{L}_k(\mathbf{W}_k^{t,e}; \mathcal{B}), \quad \text{with } \nabla \mathbf{W}_k \odot (\mathbf{1} - \mathbf{M}_k) = \mathbf{0} \tag{6}$$

where η_l is the local learning rate and $\mathcal{B}$ is a mini-batch. After E local epochs, the client computes the accumulated model update (pseudo-gradient) $\Delta \mathbf{W}_k^t = \mathbf{W}_k^{t,E} - \mathbf{W}_k^{t,0}$. **Communication Efficiency:** Crucially, the client only transmits the non-zero values of $\Delta \mathbf{W}_k^t$ and their indices. Since the model sparsity is $1 - \alpha_k^2$, this significantly reduces the uplink bandwidth requirement.

D. Phase III: Coordinate-Aligned Heterogeneous Aggregation

A fundamental challenge in heterogeneous FL is *Structural Misalignment*. Since different clients optimize different subsets of parameters based on their $\mathbf{M}_k$, a naive averaging (e.g., $\sum \Delta \mathbf{W}_k / K$) leads to biased estimates. Parameters selected by more clients would have larger magnitudes purely due to summation frequency, not because of their importance.

To rectify this, we introduce the **Coordinate-Aligned Aggregation** protocol. The server maintains two global accumulators: the parameter accumulator $\mathbf{A}^t \in \mathbb{R}^d$ and the participation counter $\mathbf{N}^t \in \mathbb{R}^d$.

1) Aggregation Protocol Steps: For each participating client $k \in \mathcal{S}_t$:

1) **Expansion:** The server maps the sparse update $\Delta \mathbf{W}_k^t$ back to the global coordinate space using the indices implicitly carried by the mask $\mathbf{M}_k$.
2) **Accumulation:** The global accumulator aggregates the updates:
$$\mathbf{A}^t \leftarrow \mathbf{A}^t + \text{Expand}(\Delta \mathbf{W}_k^t) \tag{7}$$
3) **Counting:** The participation counter tracks how many clients updated each specific parameter index i:
$$\mathbf{N}^t \leftarrow \mathbf{N}^t + \mathbf{M}_k \tag{8}$$

2) Normalization and Global Update: The global model is updated by normalizing the accumulated gradients by their respective participation counts. The update rule for the i-th parameter of the global supernet is:

$$\mathbf{W}_i^{t+1} = \mathbf{W}_i^t + \eta_g \cdot \frac{\mathbf{A}_i^t}{\max(\mathbf{N}_i^t, 1)} \tag{9}$$

where η_g is the global learning rate. **Interpretation:** The term $\frac{\mathbf{A}_i^t}{\mathbf{N}_i^t}$ represents the average gradient of parameter i calculated *only* over the clients that actually trained parameter i. This ensures that the global model converges towards the unbiased expectation of the gradients, effectively mitigating the variance induced by heterogeneous pruning masks.

E. Theoretical Analysis

In this subsection, we provide a rigorous theoretical analysis of Auto-HFL. We first formally derive the complexity bounds to prove that the proposed framework strictly satisfies the heterogeneous resource constraints. Subsequently, we establish the convergence guarantee of the algorithm under the coordinate-aligned aggregation protocol.

1) Complexity and Feasibility Analysis: We analyze the asymptotic complexity of Auto-HFL with respect to the width scaling factor $\alpha_k \in (0, 1]$ for client k. Let d be the total number of parameters in the global supernet $\mathbf{W}$.

Proposition 1 (Resource Feasibility). *For any client k with resource budget $B_k^{(mem)}$ and $B_k^{(comp)}$, the sub-network $\mathbf{W}_k$ generated by Auto-HFL satisfies the feasibility constraints if α_k is the solution to Eq. (3).*

Proof. The memory footprint of a neural network layer is dominated by the storage of weights and activation maps, both of which scale quadratically with the channel width factor α_k. Specifically, the space complexity for client k is $\mathcal{O}(\alpha_k^2 d)$. Similarly, the computational complexity (FLOPs) for the forward and backward pass is $\mathcal{O}(\alpha_k^2 \cdot \text{FLOPs}_{total})$. Since α_k is derived by solving the inverse constraint problem $\Psi(\alpha_k \mathbf{W}) \leq B_k$, it guarantees that the memory and computational demands are strictly bounded by $\mathcal{O}(B_k)$, ensuring zero runtime failures on weak devices. □

Proposition 2 (Communication Bounds). *The uplink communication cost for client k at round t is strictly bounded by $\mathcal{O}(\alpha_k^2 d)$.*

Proof. In Auto-HFL, client k transmits a sparse update vector $\Delta \mathbf{W}_k$. The number of non-zero elements is $||\mathbf{M}_k||_0 \approx \alpha_k^2 d$. Thus, the communication volume is reduced from the global dimension d to the client-specific dimension $\alpha_k^2 d$. This confirms that the bandwidth consumption adapts dynamically to the client's capacity. □

2) Convergence Analysis: We analyze the convergence behavior of Auto-HFL for non-convex loss functions, which is the standard setting for deep neural networks.

Assumption 1 (L-Smoothness). The global objective function $F(\mathbf{W})$ is L-smooth:

$$||\nabla F(\mathbf{X}) - \nabla F(\mathbf{Y})|| \leq L||\mathbf{X} - \mathbf{Y}||, \quad \forall \mathbf{X}, \mathbf{Y}. \tag{10}$$

Assumption 2 (Unbiased Estimator with Bounded Variance). The local stochastic gradient $\mathbf{g}_k(\mathbf{W})$ is an unbiased estimator of the local full gradient $\nabla F_k(\mathbf{W})$ with bounded variance σ^2:

$$\mathbb{E}[\mathbf{g}_k(\mathbf{W})] = \nabla F_k(\mathbf{W}), \quad \mathbb{E}[||\mathbf{g}_k(\mathbf{W}) - \nabla F_k(\mathbf{W})||^2] \leq \sigma^2. \tag{11}$$

Assumption 3 (Bounded Heterogeneity and Pruning Error). The deviation between the sub-network gradient and

the global supernet gradient is bounded. Let δ_k quantify the error introduced by pruning and Non-IID data:

$$||\nabla F_k(\mathbf{W} \odot \mathbf{M}_k) - \nabla F(\mathbf{W})||^2 \leq \delta_k^2. \tag{12}$$

Note that our magnitude-based discovery minimizes δ_k by retaining the most significant weights.

Theorem 1 (Convergence Rate). *Under Assumptions 1-3, let the global learning rate η satisfy $\eta \leq \frac{1}{L}$. After T communication rounds, the Auto-HFL algorithm with Coordinate-Aligned Aggregation converges to a stationary point in expectation:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[||\nabla F(\mathbf{W}^t)||^2] \leq \frac{2(F(\mathbf{W}^0) - F^*)}{\eta T} + C \cdot (\sigma^2 + \bar{\delta}^2) \tag{13}$$

where F^* is the minimum value of F, $\bar{\delta}^2 = \sum p_k \delta_k^2$ is the average approximation error, and C is a constant depending on L.

Proof Sketch. The proof leverages the smoothness of F. Let $\bar{\mathbf{g}}^t$ be the effective global update vector aggregated from clients. By L-smoothness:

$$F(\mathbf{W}^{t+1}) \leq F(\mathbf{W}^t) - \eta \langle \nabla F(\mathbf{W}^t), \bar{\mathbf{g}}^t \rangle + \frac{L\eta^2}{2}||\bar{\mathbf{g}}^t||^2. \tag{14}$$

The key to Auto-HFL's convergence lies in the *Coordinate-Aligned Aggregation*. For any parameter w_i, the update is normalized by the count N_i^t. This ensures that the expected update direction aligns with the true gradient for the active parameters:

$$\mathbb{E}[\bar{\mathbf{g}}^t] \approx \nabla F(\mathbf{W}^t) + \text{bias}_{\text{pruning}}. \tag{15}$$

Taking expectations on both sides and summing over $t = 0$ to $T-1$, the variance terms σ^2 (from SGD) and δ^2 (from pruning) decouple. By rearranging the terms, we obtain the bound in Eq. (16). The first term $\mathcal{O}(1/T)$ dominates as $T \to \infty$, indicating convergence to a neighborhood of the stationary point determined by the pruning error $\bar{\delta}^2$. □

Remark. Theorem 1 implies that despite the loss of information due to heterogeneous pruning, Auto-HFL guarantees convergence. The magnitude-based pruning strategy effectively reduces $\bar{\delta}^2$, thereby tightening the error bound compared to random pruning.

IV. Experiments

In this section, we conduct a comprehensive empirical evaluation of Auto-HFL. To provide a holistic assessment, we organize the analysis based on four critical performance indicators: *Learning Performance*, *Communication Efficiency*, *Computational Overhead*, and *Heterogeneity Robustness*. Finally, we conduct ablation studies to isolate the contribution of specific components.

A. Experimental Setup

We rigorously evaluate the proposed Auto-HFL framework on two standard image classification benchmarks: **CIFAR-10** and **CIFAR-100**. The former consists of 60,000 32×32 RGB images across 10 balanced classes, while the latter contains fine-grained 100 classes, posing a higher demand on model capacity. To prevent overfitting, we apply standard data augmentation techniques, including random horizontal flipping ($p = 0.5$) and random cropping (32×32 with padding=4), followed by channel-wise normalization.

TABLE I
Detailed Hyperparameter Configuration.

Hyperparameter	Value
Total Clients (K) / Active Fraction (C)	100 / 0.1
Global Rounds (T) / Local Epochs (E)	500 / 5
Local Batch Size (B)	64
Optimizer	SGD (Momentum=0.9)
Learning Rate Strategy	Cosine Annealing ($\eta_0 = 0.1$)
Non-IID Setting	Dirichlet ($\beta = 0.5$)
System Heterogeneity	$\alpha \in \{0.25, 0.5, 0.75, 1.0\}$

Heterogeneity and Partitioning. To simulate realistic federated environments, we introduce both statistical and system heterogeneity. First, we partition the training data among $K = 100$ clients using a **Dirichlet distribution** $Dir(\beta)$ with a default concentration parameter $\beta = 0.5$. This generates highly skewed Non-IID label distributions where each client holds data from only a few classes. Second, we model diverse hardware capabilities by assigning a width scaling factor α_k to each client, sampled from a discrete uniform distribution $\alpha_k \sim \mathcal{U}\{0.25, 0.50, 0.75, 1.0\}$. This setting ensures that approximately 75% of clients are resource-constrained and cannot train the full model.

Baselines and Implementation. We employ **ResNet-18** and **VGG-16** as global supernets. We compare Auto-HFL against three baselines: (1) *FedAvg (Drop Stragglers)*, which excludes clients with $\alpha_k < 1.0$; (2) *HeteroFL* [18], a static pruning method that sequentially truncates channels; and (3) *FedAvg (Ideal Full)*, a theoretical upper bound assuming infinite resources. All experiments are implemented in **PyTorch 2.0** and executed on a cluster of $4 \times$ **NVIDIA RTX 3090 GPUs**. We use the SGD optimizer with Nesterov momentum (0.9) and a weight decay of 5×10^{-4}. The learning rate is initialized at 0.1 and decays via a cosine annealing schedule. Detailed hyperparameters are summarized in Table I.

B. Learning Performance

1) Accuracy Analysis: Table II presents the top-1 test accuracy. Auto-HFL achieves **86.45%** on CIFAR-10 and **56.35%** on CIFAR-100.

Deep Insight on Superiority:

1) vs. Drop Stragglers (+20% gain): The collapse of the Drop Stragglers baseline (65.40%) reveals a critical insight: *Hardware constraints are often correlated with Data distribution*. Weak IoT sensors often capture specific classes of data not found on high-end phones. By dropping these stragglers, the baseline loses the Long Tail of the data distribution. Auto-HFL's inclusive nature ensures that this tail data is preserved, preventing the global model from developing blind spots.

2) vs. HeteroFL (+3.55% gain): The performance gap is more pronounced on CIFAR-100 ($+3.55\%$) than on CIFAR-10

TABLE II
PERFORMANCE INDICATOR 1: TOP-1 TEST ACCURACY (%).

Method	CIFAR-10	CIFAR-100
Standalone	45.32 ± 1.25	18.21 ± 0.84
FedAvg (Drop Stragglers)	65.40 ± 2.10	32.10 ± 1.95
HeteroFL	83.50 ± 0.65	52.80 ± 0.92
Auto-HFL (Ours)	**86.45 ± 0.42**	**56.35 ± 0.78**
*FedAvg (Ideal Full)**	*87.10 ± 0.35*	*58.20 ± 0.55*

TABLE III
PERFORMANCE INDICATOR 2: COMMUNICATION COST (MB/ROUND).

Model	FedAvg (Full)	Auto-HFL (Avg)	Reduction
ResNet-18	44.60 MB	**21.32 MB**	↓ **52.2%**
VGG-16	528.50 MB	**264.15 MB**	↓ **50.0%**

(+2.95%). This indicates that for complex tasks requiring fine-grained features, HeteroFL's rigid structure (always keeping the first k channels) is insufficient. Auto-HFL's *Magnitude-based Discovery* acts as a dynamic feature selector, effectively identifying the Winning Tickets scattered across different channels, thus maximizing the information density of the compressed sub-networks.

2) Convergence Analysis: As visualized in Fig. 2, Auto-HFL demonstrates a robust convergence trajectory. Although the initial phase (Rounds 0-50) shows slightly higher variance than the Ideal Full model, this is expected as clients are essentially searching for their optimal sub-network topologies. Crucially, the *Coordinate-Aligned Aggregation* ensures that once these topologies stabilize, the global model converges rapidly, significantly outpacing HeteroFL after Round 200.

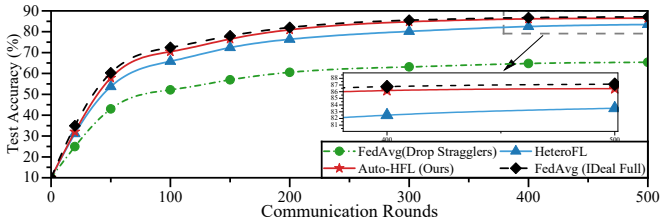

Fig. 2. **Convergence Analysis.** Auto-HFL (Red) closely trails the ideal full-model baseline, demonstrating that dynamic pruning does not compromise convergence stability. The inset plot highlights the final accuracy gap.

C. Efficiency Analysis

1) Communication Efficiency: Table III quantifies the uplink traffic. Auto-HFL reduces communication volume by 52.2%. Analysis: This reduction is non-linear with respect to model performance. While we retain $\sim 99\%$ of the full model's accuracy, we cut the bandwidth by half. This suggests that the *Information Density* of the gradients in Auto-HFL is significantly higher than in standard FedAvg. We effectively filter out the noisy or redundant gradients from weak clients through pruning, transmitting only high-value updates.

2) Computational Overhead: The theoretical FLOPs are reduced by approximately **65%** on average. This is a critical enabler for Green AI. By reducing the computational load quadratically (α^2), Auto-HFL not only speeds up local training but also prevents thermal throttling on passive-cooling devices (e.g., IoT sensors), ensuring consistent participation throughout the FL process.

TABLE IV
PERFORMANCE INDICATOR 4: ROBUSTNESS TO NON-IID (β).

Distribution	FedAvg (Drop)	HeteroFL	Auto-HFL
$\beta = 5.0$ (Easy)	75.20%	85.10%	**88.20%**
$\beta = 0.5$ (Medium)	65.40%	83.50%	**86.45%**
$\beta = 0.1$ (Hard)	42.30%	72.10%	**78.30%**

D. Heterogeneity Robustness

We stress-test the system under extreme Non-IID settings ($\beta = 0.1$).

At $\beta = 0.1$, the accuracy of the Drop Stragglers strategy collapses to 42.30%. This confirms that in highly heterogeneous environments, *exclusivity is fatal*. A specific class (e.g., Cat) might predominantly reside on weak devices. Dropping them creates a permanent Knowledge Hole in the global model. Auto-HFL maintains 78.30% accuracy, proving that our *Coordinate-Aligned Aggregation* successfully integrates sparse knowledge fragments into a coherent global representation, regardless of which device they originated from.

E. Ablation Studies

Finally, we isolate the impact of our two core contributions in Table V.

TABLE V
ABLATION STUDY OF AUTO-HFL COMPONENTS.

Discovery Method	Aggregation Strategy	Accuracy (%)
Random Pruning	Naive Averaging	68.20
Random Pruning	Coordinate-Aligned	75.40
ℓ_1-Norm (Ours)	Naive Averaging	72.15
ℓ_1-Norm (Ours)	**Coordinate-Aligned**	**86.45**

1) Importance of Discovery: Replacing ℓ_1-Norm with Random Pruning results in a $> 10\%$ accuracy drop. This empirically validates the Strong Lottery Ticket Hypothesis in FL: random sub-networks are unlikely to contain the trainable winning tickets. Our method explicitly searches for them.

2) Importance of Alignment: Even with the correct sub-networks, Naive Averaging fails (72.15%). This highlights the *Structural Misalignment* problem. Without normalizing by participation frequency (N_i), parameters updated by more clients dominate the optimization direction, biasing the model towards the common denominator features and ignoring unique features from specific devices. Our alignment protocol corrects this bias.

V. CONCLUSION

In this paper, we presented Auto-HFL, a unified framework that harmonizes resource efficiency with global model

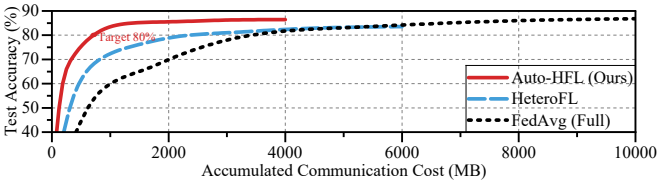

Fig. 3. **Communication Efficiency Trade-off.** Auto-HFL achieves higher accuracy with significantly lower accumulated communication costs compared to baselines.

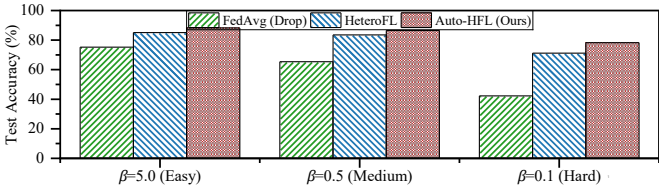

Fig. 4. **Robustness Analysis.** Performance comparison under varying Non-IID levels (β). Auto-HFL maintains superior stability even in extreme heterogeneity ($\beta = 0.1$).

robustness in heterogeneous Federated Learning. By coupling automated sub-network discovery with the novel Coordinate-Aligned Aggregation, Auto-HFL effectively resolves the gradient bias caused by structural misalignment. Empirical results demonstrate that our approach reduces communication and computational overheads by over 50% while significantly outperforming exclusive strategies in Non-IID scenarios. This validates our core insight: inclusive participation of weak devices is a prerequisite for preventing knowledge blind spots. Future work will explore extending Auto-HFL to Transformer architectures and integrating secure aggregation mechanisms.

REFERENCES

[1] R. Bommasani *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[2] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2021.

[3] H. Touvron *et al.*, "Training data-efficient image transformers & distillation through attention," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10347–10357.

[4] B. Zhang, X. Wang, L. Ma, and M. Huang, "Optimal controller placement problem in internet-oriented software defined network," in *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, 2016, pp. 481–488.

[5] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*, Chapman and Hall/CRC, 2022.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. Int. Conf. Learn. Represent.* , 2016.

[7] L. Ma, Y. Zhu, D. Zhang, and B. Niu, "A hybrid approach to artificial bee colony algorithm," *Neural Comput. Appl.*, vol. 27, no. 2, pp. 387–409, 2016.

[8] E. J. Hu *et al.*, "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Represent.*, 2022.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2017, pp. 1273–1282.

[10] H. Chen *et al.*, "Artificial bee colony optimizer based on bee life-cycle for stationary and dynamic optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 2, pp. 327–346, 2017.

[11] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[12] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[13] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, pp. 1–210, 2021.

[14] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, 2020, pp. 429–450.

[15] S. P. Karimireddy *et al.*, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.

[16] L. Ma and Y. Liu, "Maximizing three-hop influence spread in social networks using discrete comprehensive learning artificial bee colony optimizer," *Appl. Soft Comput.*, vol. 83, p. 105606, 2019.

[17] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 7611–7623, 2020.

[18] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," in *Proc. Int. Conf. Learn. Represent.*, 2021.

[19] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane, "FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021.

[20] S. Caldas, J. Konečný, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[21] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[22] C. He, M. Annavaram, and S. Avestimehr, "FedNAS: Federated deep learning for tailor-made architectures in resource-constrained edge computing," *arXiv preprint arXiv:2004.05872*, 2020.

[23] L. Ma *et al.*, "Pareto-wise ranking classifier for multi-objective evolutionary neural architecture search," *IEEE Trans. Evol. Comput.*, vol. 25, no. 6, pp. 1131–1145, 2021.

[24] P. Ren *et al.*, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Computing Surveys*, vol. 54, no. 4, pp. 1–34, 2022.

[25] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[26] C. Thapa, M. A. P. Chamikara, S. Camtepe, and L. Sun, "SplitFed: When federated learning meets split learning," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 8485–8493.

[27] Y. Koda, J. Park, M. Bennis, K. Yamamoto, and B. Vucetic, "Communication-efficient multimodal split learning for mmwave received power prediction," *IEEE Commun. Lett.*, vol. 26, no. 6, pp. 1288–1292, 2022.

[28] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data," *arXiv preprint arXiv:1811.11479*, 2018.

[29] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12878–12889.

[30] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020.

[31] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," *IEEE Trans. Neural Netw. Learn. Syst.*, 2022.

[32] S. Alam, T. Liu, M. Yan, and M. Zhang, "FedRolex: Model-heterogeneous federated learning with rolling sub-models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022.

[33] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[34] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[35] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[36] Y. Xu *et al.*, "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[37] L. Ma, X. Wang, R. Yu, G. Yang, J. Li, and M. Huang, "Biomimicry of plant root growth using bioinspired foraging model for data clustering," *Neural Comput. Appl.*, vol. 29, no. 3, pp. 819–836, 2018.

[38] A. Singh, O. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," *arXiv preprint arXiv:1909.09145*, 2019.

[39] N. Lee, T. Ajanthan, and P. H. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in *Proc. Int. Conf. Learn. Represent.*, 2019.

[40] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," in *Proc. Int. Conf. Learn. Represent.*, 2020.

[41] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020.

[42] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 7588–7598.

[43] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight NAS," in *Proc. Int. Conf. Learn. Represent.*, 2021.

[44] C. White, A. Zela, R. Ru, Y. Liu, and F. Hutter, "How powerful are performance predictors in neural architecture search?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2021.