LLM-Flex: Adaptive Heterogeneous Federated Learning Based on LLM in Uncertain Scenarios

Qian Liu
Southeast University
Nanjing, China
qianliu2006@seu.edu.cn

Weilong Wang
Southeast University
Nanjing, China
wang_wl@seu.edu.cn

Borui Li
Southeast University
Nanjing, China
libr@seu.edu.cn

Shuai Huang
Southeast University
Nanjing, China
213233026@seu.edu.cn

Shuai Wang*
Southeast University
Nanjing, China
shuaiwang@seu.edu.cn

***Abstract*—Along with the increasing popularity of Heterogeneous Federated Learning (HFL), existing frameworks are capable of training diverse sub-models on devices with varying resources. However, due to the inherent complexity of balancing system-level trade-offs, these frameworks suffer from a critical usability and configuration problem. Although various HFL methods have been investigated to handle device heterogeneity, there is still a lack of i) an automated mechanism to translate high-level user requirements into optimal configurations, ii) a systematic approach to navigating the accuracy-latency trade-off without expert intervention, and iii) a user-centric paradigm that adapts to specific deployment goals. To address the above issues, this paper introduces LLM-Flex, a novel framework that integrates a Large Language Model (LLM) agent to automate the configuration of HFL systems. By leveraging an LLM to parse natural language intent, LLM-Flex can dynamically select the optimal model architecture and heterogeneity distribution to best fit the user's goals. Meanwhile, our proposed framework utilizes the robust FlexFL algorithm as its execution engine, inheriting its capabilities for adaptive local pruning and self-knowledge distillation. Comprehensive experimental results show that, compared to state-of-the-art baselines with fixed configurations, LLM-Flex can significantly reduce training latency by up to 24.3% in speed-focused scenarios and improve accuracy by up to 37.3% in performance-focused scenarios, successfully aligning system performance with user intent.**

***Index Terms*—Federated Learning, Heterogeneous Systems, Large Language Models, Model Pruning, System Automation, Artificial Intelligence of Things.**

I. Introduction

Federated Learning (FL) has emerged as a key paradigm for privacy-preserving learning in Artificial Intelligence of Things (AIoT) systems [1], [2], [3], [4], [5]. However, its practical deployment is hindered by device heterogeneity [6]. Traditional FL frameworks assume model homogeneity, forcing the system to accommodate the least capable device—a principle often referred to as the Cannikin Law. Consequently, the entire network is restricted to smaller models with limited capacity, stifling the performance of more powerful devices.

To overcome this limitation, Heterogeneous Federated Learning (HFL) [7] offers more flexible solutions, which can be broadly categorized into two branches: completely heterogeneous methods using techniques like Knowledge Distillation (KD) [8], [9], [10] to enable knowledge transfer between models with entirely different architectures. The second, partially heterogeneous methods [11], [12], [13], [14], focuses on generating a spectrum of sub-models from a unified global architecture. A prominent example is HeteroFL [15], which creates custom-fit models for devices by pruning parameters from a larger global model on a layer-by-layer basis.

However, these partially heterogeneous methods, while effective, still rely on pre-defined and often rigid strategies for model generation. The recent and rapid development of Large Language Model (LLM) based Agents presents a new paradigm for tackling this challenge from a higher level of abstraction,extending the evolution from digital twins to autonomous digital twin agents [16], [17], [18]. These agents possess the remarkable ability to directly parse complex, high-level user requirements expressed in natural language. Leveraging their extensive prior knowledge, they can make informed decisions on aspects of the learning process, such as selecting the optimal model architecture and determining the most effective optimization strategy. Crucially, our work considers the explicit user demands for a balance between accuracy and latency, a critical consideration that is overlooked by previous works which typically employ a one-size-fits-all approach. By integrating an LLM agent, we empower the HFL system to autonomously configure itself in response to specific application goals, rather than adhering to a static set of rules.

Nevertheless, even these advanced HFL approaches face hurdles in real-world AIoT deployments, which can be summarized by three primary issues. (1) The suboptimal model performance resulting from indiscriminate pruning techniques that ignore the functional importance of different parameters. (2) Training inefficiencies on-device due to fluctuating resource availability, such as variations in memory or CPU load. (3) The underutilization of powerful large models, which cannot be trained on the majority of resource-constrained clients. Such simplistic pruning strategies limit the potential of the derived heterogeneous models. Furthermore, the dynamic and uncertain nature of device resources can derail local training, leading to degraded performance for the entire system.

This confluence of challenges underscores an urgent need for a more intelligent and adaptive framework capable of generating high-performance heterogeneous models that are resilient to the uncertainties of real-world AIoT environments.

II. Background And Relatedwork

A. Background

Federated Learning. Federated Learning (FL) is a distributed machine learning approach that addresses data privacy protection and decentralization issues.Traditional FL framework usually consists of a server and multiple devices. In FL training, the server maintains a global model and dispatches it to the selected devices for local training in each round. Each device then trains locally on its own data and uploads the trained model to the server after training. Finally, the server aggregates all the received models to generate a new global model, with the optimization objective typically based on the FedAvg algorithm [19].

$$\min_{w} F(w) = \frac{1}{|D|} \sum_{k=1}^{|D|} f_k(w),$$
$$\text{s.t., } f_k(w) = \frac{1}{|D_k|} \sum_{i=1}^{|D_k|} \ell\left(w, \langle x_i, y_i \rangle\right),$$

where $|D|$ is the total number of clients, and w represents the model parameters. The global objective is to minimize the weighted average of the local loss functions $f_k(w)$, where each local loss is computed over a client's private dataset of size $|D_k|$ using a loss function ℓ on samples (x_i, y_i).

Model Pruning. In the field of machine learning, model pruning is a technique to reduce model complexity and computational resource requirements by reducing redundant parameters and connections in neural network models. Common approaches include magnitude-based pruning [30], sensitivity-based pruning [20], and structured pruning [21]. APoZ (Activation Percentage of Zeros) [22][29] is a metric used in model pruning to quantify the sparsity level of neural network activations. It measures the percentage of zero activations in a layer or network after applying a pruning technique.

B. Model Heterogeneous Federated Learning

Model heterogeneous FL [11], [12], [14], [23] has a advantage in solving systemic heterogeneity. Different from traditional FL, model heterogeneous FL maintains some models of different sizes on the cloud server, so as to better deal with the heterogeneous resources of different devices.

For width-wise pruning, Diao et al. proposed HeteroFL [11], tailoring model width to alleviate device system heterogeneity. Similarly, Horvath et al. [34] proposed FjORD using Ordered Dropout. For depth-wise pruning, DepthFL [25] prunes the later parts of deeper networks. In InclusiveFL [24], Liu et al. proposed a layer-wise pruning method with momentum knowledge distillation. For two-dimensional pruning, ScaleFL [14] introduced a method that scales from both width and depth dimensions.

Recently, FlexFL [25] has emerged as a state-of-the-art approach to address resource uncertainty. FlexFL utilizes a flexible pruning strategy based on APoZ scores to generate heterogeneous models. It integrates an adaptive local pruning mechanism and self-KD-based local training, enabling devices to adaptively prune their received model according to their available resources.

However, while FlexFL and aforementioned methods successfully address device-side resource constraints, they overlook the interaction between the system configuration and the user's high-level intent. Most existing works, including FlexFL, rely on static or manually tuned hyperparameters (e.g., the selection of the backbone model architecture) to balance performance. They lack a semantic interface to interpret user requirements regarding the specific trade-off between inference latency and model accuracy.

In real-world deployments, the optimal configuration varies significantly depending on whether the user prioritizes rapid response or high precision. The absence of an intelligent mechanism to map these natural language requirements to low-level system parameters acts as a barrier for non-expert users. To the best of our knowledge, this paper is the first to integrate Large Language Models (LLMs) into the FL framework, aiming to bridge the gap between rigid system configurations and dynamic user requirements through intent-aware parameter optimization.

III. Our Approach

A. Overview

As summarized in the foundational survey of federated learning [26], a standard FL system typically follows a fixed "centralized orchestration, distributed execution" paradigm. While this ensures data privacy, traditional frameworks often suffer from rigid configurations, lacking the flexibility to adapt to dynamic, high-level user goals.

To address this limitation, we propose **LLM-Flex**, a user-centric framework that integrates an intelligent interaction layer atop the traditional FL architecture. As illustrated in Figure 1, the overall workflow of LLM-Flex is streamlined into four sequential phases:

User Requirement Input.Users provide natural language instructions regarding deployment goals.

LLM Intent Analysis. An integrated LLM agent parses the user's input, leveraging a pre-defined reasoning framework encoded in its system prompt to understand the optimization objective.

Configuration Generation. Based on the analyzed intent, the agent automatically generates a tailored system configuration. Specifically, it determines the optimal backbone network architecture and the client heterogeneity device ratio.

Adaptive Execution. Finally, the FlexFL algorithm [25] acts as the execution engine, implementing the LLM's configuration to handle system heterogeneity during training.

B. LLM-driven Configuration Generation

The core innovation of LLM-Flex lies in its single-shot, automated mapping from natural language to a precise, executable configuration. This process is a unified interaction with an LLM, governed by a comprehensive prompt and a structured output schema, accurately reflecting our implementation.

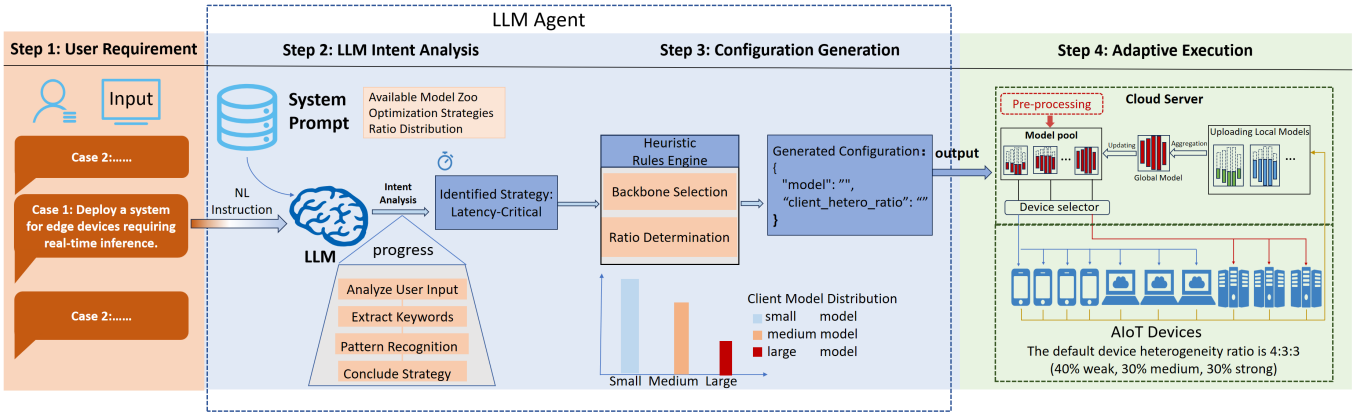

Fig. 1: The overall workflow of our LLM-Flex framework, with Case 1 shown as an example.

As implemented in our system, this is achieved through a single call to an LLM, guided by two key components.

Comprehensive System Prompt. We engineer a detailed system prompt that provides the LLM with all necessary context and reasoning logic[27]. This prompt effectively encodes the domain-specific heuristics for the configuration task, defining: The available Model Zoo and their respective design trade-offs. The three core Optimization Strategies: Latency-Critical, Accuracy-Critical, and Constraint-Aware Balanced, complete with keyword triggers and decision-making rules. (1) Case1: Deploy a system for edge devices requiring real-time inference. (2) Case2: I need the highest possible accuracy for complex image classification. (3) Case3: Balance the model to ensure fast response but keep decent accuracy. The structure of the Heterogeneity Ratio Distribution, including concrete examples for each strategy .

Structured Output via Function Calling. To ensure the LLM's output is reliable and machine-readable, we leverage the model's Function Calling capability. We define a strict Pydantic schema (FLConfigInput) that forces the LLM to return a JSON object containing the model_name and client_hetero_ratio.

This approach transforms the LLM from a simple text generator into a robust configuration engine. For instance, given a nuanced prompt like "maximize accuracy while maintaining low latency," the prompt guides the LLM to adopt the Balanced Strategy, and the function calling schema ensures it outputs a valid configuration. This output directly overwrites the system's default arguments before the training process begins.

C. Workflow of the Execution Engine

Once the configuration is finalized, LLM-Flex executes the training via FlexFL mechanisms across varying data modalities.

a) Dynamic Initialization and Dispatching: The server initializes the specific backbone and heterogeneity ratio determined by the LLM. To generate the heterogeneous sub-models, the engine employs the APoZ-guided pruning strategy from FlexFL [25]. The primary metric, as defined in [25], is the Activation Percentage of Zeros (APoZ)[22] score, A_i, for each layer i:

$$A_i = \frac{1}{|D_p^{test}| \times N} \sum_{k=1}^{|D_p^{test}|} \sum_{j=1}^{N} f(\delta(h_k^i(S_j) = 0)), \quad (1)$$

where $f(\delta)$ is a boolean function, $h_k^i(S_j)$ is the feature map of sample S_j at layer i, and $|D_p^{test}|$ is the size of the proxy dataset. To further refine this process, an adjustment weight is introduced to prioritize larger, more redundant layers [25]:

$$\text{AdjWCal}(l_i, M) = \frac{\log \text{size}(l_i)}{\log \max(\text{size}(l_1), ..., \text{size}(l_{len(M)}))}, \quad (2)$$

where l_i is the i^{th} layer of M, the function $\text{size}(l_i)$ calculates the number of parameters of the i^{th} layer, and $\text{len}(M)$ denotes the number of layers of the model M.

Using these metrics, the server generates the Small, Medium, and Large sub-models and dispatches them to clients according to the specified distribution.

b) Adaptive Local Training: Clients train their assigned sub-models on local private data. To address performance degradation from aggressive pruning, we employ the Self-Knowledge Distillation (Self-KD) strategy from FlexFL [25]. This is achieved by minimizing a combined loss function $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{KL}, \quad (3)$$

Following the formulation in [25], $\mathcal{L}_{CE}$ is the standard cross-entropy loss:

$$\mathcal{L}_{CE} = -\log h(\hat{y}_i)[y_i], \quad (4)$$

where $h(\cdot)$ is the softmax function. The term $\mathcal{L}_{KL}$ represents the knowledge distillation loss[9], calculated as the Kullback-Leibler divergence between the softened outputs of the current model M_i and all smaller sub-models M_j $(j < i)$:

$$\mathcal{L}_{KL} = \frac{1}{i-1} \sum_{j=1}^{i-1} \text{sum}\left(h(\hat{y}_j)/\tau\right) \cdot \tau^2 \log \frac{h(\hat{y}_j)}{h(\hat{y}_i)}, \quad (5)$$

where τ is the temperature parameter. This Self-KD mechanism allows the larger model to learn from the smaller

ones. Additionally, Adaptive Pruning is triggered if a device's resources are insufficient.

c) Heterogeneous Aggregation: The server collects the updated local models S_{upload} from clients. Since all sub-models are pruned from the same global model, the aggregation for each parameter p is calculated as a weighted average over all received models, as formulated in FlexFL [25]:

$$\forall p \in \theta, \quad p = \frac{\sum_{m \in \sigma(p, S_{\text{upload}})} p^m \times d^m}{\sum_{m \in \sigma(p, S_{\text{upload}})} d^m}, \tag{6}$$

where p^m is the parameter from model m, and d^m is its number of local data samples. This synchronized knowledge is then used to update the model pool.

IV. PERFORMANCE EVALUATION

To evaluate the performance of our LLM-Flex framework, we implemented the system using PyTorch, with the FlexFL algorithm serving as the underlying execution engine. For the federated training process, we adopted the same SGD optimizer with a learning rate of 0.01 and a momentum of 0.5 to ensure a fair comparison with the baseline. For local training on client devices, we set the batch size to 50 and the number of local epochs to 5. We assumed a total of $|D| = 100$ simulated AIoT devices were involved, and a fraction of $f = 10\%$ of them were selected in each FL training round. All experiments were conducted on Ubuntu workstation with **Intel i9-13900HX CPU**, **16GB memory**, and **NVIDIA RTX 4060 GPU**.

A. Experimental Settings

1) Device Heterogeneity Settings.: To evaluate the performance of LLM-Flex in uncertain and resource-constrained scenarios, we simulated various devices with different dynamic resources (available memory size), mirroring the setup in FlexFL [25]. Specifically, we employed a Gaussian distribution to define dynamic device resources as follows: $r = r_M - |u|$, where $u \sim \mathcal{N}(0, \sigma^2)$. We adopted three levels of devices, i.e., weak, medium, and strong. We set the ratio of the number of devices at these three levels to 40%, 30%, and 30%, respectively. The distribution of their available memory size across these devices is uncertain, as shown in Table I.

TABLE I: DEVICE UNCERTAINTY SETTINGS

Level	# Device	Maximum Capacity r_M	Variance σ^2
Weak	40%	$r_M = 35$	$\sigma^2 \in [5, 8, 10]$
Medium	30%	$r_M = 60$	$\sigma^2 \in [5, 8, 10]$
Strong	30%	$r_M = 110$	$\sigma^2 \in [5, 8, 10]$

Data Settings. We utilize three well-known datasets: **CIFAR-10**[28]**MNIST**[29], and **TinyImageNet**[30].To simulate non-IID data distributions among clients, which is a common challenge in FL, we employ a Dirichlet distribution Dir(α) to partition the data. A smaller value of α indicates a higher degree of data heterogeneity.

LLM-Flex Configuration and Evaluation Scenarios. The core of our evaluation is to test the LLM agent's decision-making capability. The agent is configured to select from a predefined Model Zoo and generate a heterogeneity ratio distribution.

a) Model Pool Settings: To validate the generality of our approach, the LLM agent can choose from three model architectures with varying complexities: **VGG16** [31], **ResNet34** [32], and **MobileNetV2** [33].

b) Evaluation Scenarios: We evaluate LLM-Flex under three distinct user-intent scenarios, which correspond to the optimization strategies defined in Section III:

- **Latency-Critical:** The user prioritizes minimal training and inference time. We expect the LLM to select a lightweight model and a heterogeneity ratio skewed towards small models .
- **Accuracy-Critical:** The user prioritizes the final global model's performance. We expect the LLM to select a high-capacity model and a ratio skewed towards large models.
- **Constraint-Aware Balanced:** The user requests a trade-off. We expect the LLM to find a optimal balance, for instance by selecting a capable model but with a balanced ratio to manage latency.

For all scenarios, the underlying FlexFL engine uses a target model pruning list of $L_p = [25\%, 50\%, 100\%]$ to generate the Small, Medium, and Large sub-models. The self-KD hyperparameters are kept constant at $\tau = 3$ and $\lambda = 10$.

B. Performance Comparison

In this section, we present a comprehensive performance comparison. We evaluate our proposed LLM-Flex framework against two strong baselines: a fixed configuration of the SOTA **FlexFL**[25] algorithm (referred to as FlexFL-Default) and the **Decoupled**[19] training method. Our analysis focuses on three aspects: (1) the ability to navigate the accuracy-latency trade-off, (2) the impact on training efficiency and convergence, and (3) the generalization capability of our LLM-driven approach.

*1) **Accuracy vs. Latency Trade-off Analysis**:* A core claim of our work is that an LLM can interpret user intent to effectively balance accuracy and latency. To validate this, we evaluated the three user-intent scenarios on CIFAR-10 and MNIST under a challenging Non-IID setting ($\beta = 0.6$). The results, presented in Figure 2, demonstrate LLM-Flex's superior adaptability.

Figure 2 illustrates the dynamic adaptability of LLM-Flex across varying user priorities. In the **latency-critical** scenario (Case 1), the agent correctly generates a latency-skewed configuration on CIFAR-10 (VGG backbone, '7:3:0' ratio) that completes training in just **6,365 s**, achieving a **24.3%** reduction in wall-clock time compared to the FlexFL-Default baseline while maintaining a competitive accuracy of **75.63%**. Conversely, when the objective shifts to **maximizing accuracy** (Case 2), the agent astutely selects high-capacity models and accuracy-skewed ratios. On CIFAR-10, its chosen configuration achieves **81.24%** accuracy, effectively matching the performance of the SOTA FlexFL-Default baseline. On the more complex MNIST task, the agent makes a more aggressive decision, switching to a **ResNet** backbone with a '4:3:3' ratio

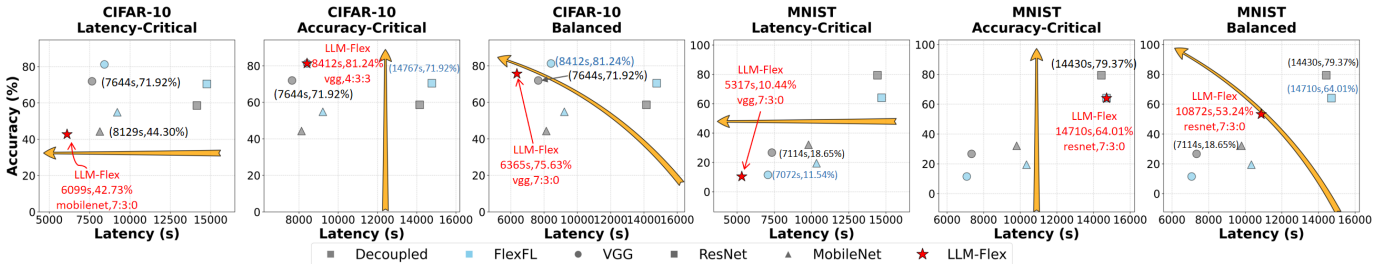

Fig. 2: Performance comparison of LLM-Flex against baselines on Non-IID datasets ($\beta = 0.6$) across three user intents. (**The orange arrow in each subplot indicates the optimization direction.**)

to surpass both baselines and reach **64.01%** accuracy.Finally, the **balanced** scenario (Case 3) highlights the agent's nuanced reasoning in identifying Pareto-optimal points. On MNIST, the agent selects a ResNet model with a '7:3:0' ratio. This configuration achieves a remarkable **53.24%** accuracy—more than double that of the baselines—while completing training in only **10,871 s**, a **26% reduction in time** compared to the accuracy-critical setting. This "best-of-both-worlds" outcome validates that the LLM can generate configurations that are strategically optimized to extract maximum performance with minimal resource expenditure.

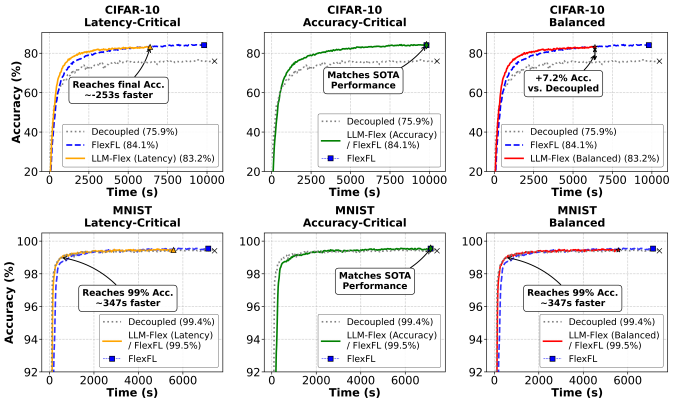

Fig. 3: Training efficiency comparison on IID datasets

*2) **Training Efficiency and Convergence Analysis**:* Beyond the final trade-off point, we analyze the training dynamics on IID data, as shown in Figure 3. On CIFAR-10, the configurations generated by LLM-Flex (Accuracy-Critical and Balanced/Latency-Critical) exhibit significantly faster convergence and achieve approximately **8.1% higher final accuracy** than the Decoupled baseline.

The results on MNIST reveal a deeper insight into the LLM's strategic planning. The zoomed-in plot shows that the **LLM-Flex (Balanced)** strategy reaches over **98%** accuracy approximately **33% faster** than the **LLM-Flex (Accuracy-Critical)** strategy, with a negligible drop in final performance. This proves that the agent's balanced configuration provides a more efficient optimization path, avoiding the diminishing returns of training a full-capacity model for extended periods, and is thus a superior choice for practical applications where rapid deployment to a high-performance state is critical. This demonstrates the LLM's ability to not just optimize for a final state, but for the entire optimization trajectory, making it a superior choice for practical applications where rapid deployment to a high-performance state is critical.

3) Generalization to Other FL Algorithms: To demonstrate that our LLM-driven approach is a generalizable framework rather than a solution coupled to a single algorithm, we migrated it to the **Decoupled** training algorithm on the Tiny-ImageNet dataset. The task was to dynamically configure the model and heterogeneity ratio for the Decoupled engine.

As shown in Figure 4, the LLM's dynamic configurations consistently outperform the fixed default baseline (VGG, 4:3:3) across all scenarios. In Case 1 (Latency-Critical), the LLM's choice of a 'MobileNet, 7:3:0' configuration drastically reduces training time from over 40,000 s to approximately 28,000 s in the IID case, a reduction of nearly 30%. In Case 2 (Accuracy-Critical), by switching to a 'ResNet, 4:3:3' configuration, the LLM boosts accuracy from 25.31% to **34.81%** (IID) and from **26.12%** to **33.88%** (Non-IID). In Case 3 (Balanced), the agent's choice ('ResNet, 7:3:0') achieves a **Pareto improvement** over the default configuration: it is both significantly **faster** and achieves substantially **higher accuracy**. This powerful result confirms that our LLM agent can function as a portable, intelligent front-end, capable of enhancing various FL back-ends by automating the complex task of intent-based hyperparameter optimization.

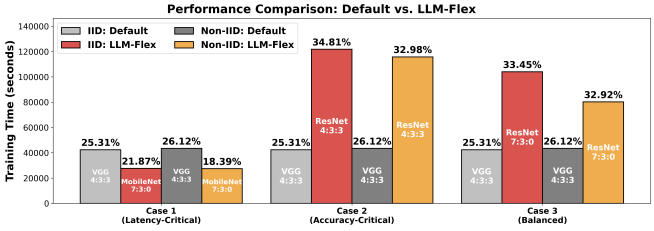

Fig. 4: Performance comparison between the default Decoupled baseline and LLM-Flex across three user intents.

V. Conclusion

Due to the lack of mechanisms to align system configurations with high-level user goals, existing Heterogeneous

FL frameworks suffer from suboptimal performance in real-world deployments, often requiring tedious manual tuning by domain experts. To address this problem, this paper presents a novel FL approach named **LLM-Flex**, which integrates an LLM agent to serve as an intelligent decision-making front-end. Based on our proposed intent-parsing and automated parameter-mapping mechanism, LLM-Flex enables the system to autonomously select the optimal model architecture and heterogeneity distribution to fit various user requirements. Meanwhile, LLM-Flex utilizes the powerful FlexFL framework as its execution back-end, which includes an effective adaptive local pruning mechanism and a knowledge distillation-based local training strategy. Comprehensive experimental results obtained from extensive simulations show that our approach can achieve a superior accuracy-latency trade-off compared with state-of-the-art heterogeneous FL methods, successfully demonstrating its ability to translate user intent into tangible performance gains.

VI. Acknowledgments

This work is partially supported by the National Nat-Science Foundation of China (Grants No. 62272098 and 62302096), the Natural Science Foundation of JiangsuProvince (Grants No. BK20241274).

References

[1] Y. Cai, W. Xi, Y. Shen, C. Sun, S. Wang, W. Gong, and J. Zhao, "Individualized data generation in personalized federated learning," *IEEE Transactions on Mobile Computing*, 2025.

[2] Y. Fan, W. Xi, Y. Shen, and J. Zhao, "Evolutionary cross-client network aggregation for personalized federated learning," *Knowledge-Based Systems*, vol. 309, p. 112866, 2025.

[3] M. Hu, P. Zhou, Z. Yue, Z. Ling, Y. Huang, A. Li, Y. Liu, X. Lian, and M. Chen, "Fedcross: Towards accurate federated learning via multi-model cross-aggregation," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 2137–2150.

[4] X. Ji, Z. Zhu, W. Xi, O. Gadyatskaya, Z. Song, Y. Cai, and Y. Liu, "Fedfixer: Mitigating heterogeneous label noise in federated learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 11, 2024, pp. 12 830–12 838.

[5] W. Wang, Y. Wang, Y. Huang, C. Mu, Z. Sun, X. Tong, and Z. Cai, "Privacy protection federated learning system based on blockchain and edge computing in mobile crowdsourcing," *Computer networks*, vol. 215, p. 109206, 2022.

[6] A. K. Singh, K. R. Basireddy, A. Prakash, G. V. Merrett, and B. M. Al-Hashimi, "Collaborative adaptation for energy-efficient heterogeneous mobile socs," *IEEE Transactions on Computers*, vol. 69, no. 2, pp. 185–197, 2019.

[7] D. Gao, X. Yao, and Q. Yang, "A survey on heterogeneous federated learning," *arXiv preprint arXiv:2210.04505*, 2022.

[8] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International journal of computer vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[10] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3967–3976.

[11] E. Diao, J. Ding, and V. Tarokh, "Heterofl: Computation and communication efficient federated learning for heterogeneous clients," *arXiv preprint arXiv:2010.01264*, 2020.

[12] M. Kim, S. Yu, S. Kim, and S.-M. Moon, "Depthfl: Depthwise federated learning for heterogeneous clients," in *The Eleventh International Conference on Learning Representations*, 2022.

[13] C. Jia, M. Hu, Z. Chen, Y. Yang, X. Xie, Y. Liu, and M. Chen, "Adaptivefl: Adaptive heterogeneous federated learning for resource-constrained aiot systems," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[14] F. Ilhan, G. Su, and L. Liu, "Scalefl: Resource-adaptive federated learning with heterogeneous clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 24 532–24 541.

[15] M. Hu, W. Duan, M. Zhang, T. Wei, and M. Chen, "Quantitative timing analysis for cyber-physical systems using uncertainty-aware scenario-based specifications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 4006–4017, 2020.

[16] J. Yu, J. Zhou, and J. Fu, "From digital twin to digital twin agent," in *2025 1st International Symposium on E-CARGO and Applications (E-CARGO)*. IEEE, 2025, pp. 111–116.

[17] W. Wang, B. Li, S. Wang, and T. He, "Jarvis: Zero-code prototyping of iot applications with composable hardware-software abstractions," *ACM Transactions on Internet of Things*, vol. 6, no. 4, pp. 1–27, 2025.

[18] B. Li, Y. Wang, H. Ma, L. Chen, J. Xiao, and S. Wang, "Mobilora: Accelerating lora-based llm inference on mobile devices via context-aware kv cache optimization," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 23 400–23 410.

[19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[20] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," *Advances in neural information processing systems*, vol. 1, 1988.

[21] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4340–4349.

[22] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016.

[23] R. Liu, M. Hu, Z. Xia, J. Xia, P. Zhang, Y. Huang, Y. Liu, and M. Chen, "Adapterfl: Adaptive heterogeneous federated learning for resource-constrained mobile computing systems," *arXiv preprint arXiv:2311.14037*, 2023.

[24] R. Liu, F. Wu, C. Wu, Y. Wang, L. Lyu, H. Chen, and X. Xie, "No one left behind: Inclusive federated learning over heterogeneous devices," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 3398–3406.

[25] Z. Chen, C. Jia, M. Hu, X. Xie, A. Li, and M. Chen, "Flexfl: Heterogeneous federated learning via apoz-guided flexible pruning in uncertain scenarios," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 11, pp. 4069–4080, 2024.

[26] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[28] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 2002.

[30] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.