

# FedTPA: Tackling Data Heterogeneity with Adaptive Parameter Allocation in Federated Instruction Tuning

Yixuan Chen, Jinghui Zhang and Ding Ding

School of Computer Science and Engineering, Southeast University, Nanjing, China

{220232353, jhzhang, dingding-1}@seu.edu.cn

**Abstract**—Federated instruction tuning of large language models (LLMs) has recently emerged as a promising research direction for preserving data privacy while enabling collaborative model adaptation. However, due to the heterogeneity of local instruction data across clients in federated settings, assigning the same trainable parameter size to all clients may compromise local learning effectiveness and limit the overall performance of the global model. To address this challenge, we propose FedTPA, a dynamic pruning-based strategy that allocates and adjusts the adapter dimensions of local models based on the distribution of local instruction data and trends in training loss. This allows the trainable parameter size on each client to better align with the complexity and characteristics of its local data. We evaluate FedTPA across multilingual, multi-task, and varying degrees of data heterogeneity scenarios. Experimental results demonstrate that FedTPA outperforms existing federated instruction tuning methods, achieving up to a 3% improvement in Rouge-L scores.

**Index Terms**—Large language models, federated learning, instruction tuning, data heterogeneity

## I. INTRODUCTION

Large language models (LLMs) based on multi-layer Transformer architectures have become a research focus in artificial intelligence [1]. LLMs trained under the pre-training and fine-tuning paradigm [2] have been widely applied in various downstream tasks, such as knowledge question-answering [3], code generation [4], and text translation [5]. Concurrently, with the introduction of instruction tuning [6] and parameter-efficient fine-tuning (PEFT) [7], significant improvements in cross-task performance and fine-tuning efficiency of LLMs have been achieved [8]. Recently, given the increasing scarcity of public data and growing concerns over data privacy and security [9] [10], along with rapid progress in memory capacity and computational capabilities of edge devices, incorporating federated learning into instruction tuning of LLMs has emerged as a novel research direction [11]. To mitigate the communication and storage overhead of transmitting full model parameters, federated instruction tuning commonly adopts a specific PEFT strategy, namely inserting lightweight adapters into the backbone while keeping most parameters frozen [14] [15].

However, since instruction datasets emphasize cross-task and cross-domain characteristics more strongly than traditional datasets, the instruction data owned by different clients may exhibit non-independent and identically distributed (non-IID) characteristics across domains, task types, and complexities. Previous studies in traditional federated learning have demonstrated that non-IID data distributions can significantly impact

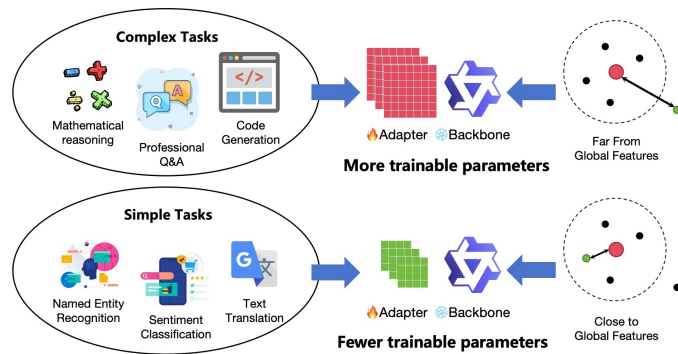


Fig. 1. In federated instruction tuning scenarios, clients facing higher local task complexity (e.g., mathematical reasoning, professional Q&A, code generation) and exhibiting significant divergence between local instruction data features and global distributions require a larger allocation of trainable parameters. In contrast, clients with simpler tasks (e.g., named entity recognition, sentiment classification, text translation) and whose local data are more aligned with global features, can achieve satisfactory adaptation with a smaller parameter budget.

the generalization performance of global models [12] [34]. In existing federated instruction tuning approaches, clients typically use trainable adapter parameters of identical size. This strategy can result in insufficient parameter capacity for clients whose local tasks are more complex or significantly different from global features, preventing adequate fitting of local complex features. Conversely, clients with tasks of lower complexity or similar to global features may face an increased risk of overfitting due to parameter redundancy, ultimately weakening local learning effectiveness [7] [13]. As shown in Figure 1, in an ideal scenario, the parameter capacity should adapt to both task complexity and distributional divergence: clients with harder or more divergent tasks require more parameters, while those with simpler or aligned tasks can be sufficiently served with fewer. This motivates the design of adaptive allocation strategies of local trainable parameter sizes in federated instruction tuning.

Implementing the above strategy faces two main challenges. First, in federated scenarios, data isolation between clients and the server makes it difficult to accurately quantify the discrepancies between each client’s instruction data distribution and the global data characteristics. Such estimation must be done without direct data sharing, relying solely on limited statistics or model updates, which increases the risk of misjudging clients’ true heterogeneity and may lead to suboptimal parameter allocation or client drift. Second, designing an effective mechanism to adapt the trainable parameter size to the local

data’s complexity and divergence from global features is non-trivial. It requires balancing parameter capacity, communication efficiency, and training stability across diverse clients, while ensuring that the resulting updates contribute positively to the global model.

To address these challenges, we propose FedTPA, a novel federated instruction tuning strategy that dynamically adjusts each client’s trainable parameter size based on data distribution and complexity, to improve the global model’s task performance in data heterogeneity scenarios. For the first challenge, FedTPA estimates the divergence between client and global features extracted by the local pre-trained backbone, and allocates larger trainable parameter sizes to highly heterogeneous clients. For the second, it applies adapter pruning with a pruning ratio dynamically tuned to local training performance, aligning parameter capacity with data complexity while preserving stability and efficiency.

Our contributions can be summarized as follows:

- We design a backbone-based strategy for local feature extraction and divergence evaluation without uploading raw data, and allocate trainable parameter sizes to clients according to their divergence from the global features.
- We implement trainable parameter size adjustment among clients through a local adapter pruning algorithm, adapting after each communication round according to trends in local training and validation losses, thus accommodating clients with varying task complexities.
- We compare our approach with existing federated instruction tuning methods, FedIT [14] and FlexLoRA [15], on publicly available real-world instruction tuning datasets, demonstrating our method’s superior task performance across different data distribution scenarios.

## II. RELATED WORK AND MOTIVATION

### A. Parameter-Efficient Fine-Tuning

Currently, mainstream PEFT methods embed additional trainable modules (known as adapters) into the model’s backbone to learn task-specific knowledge, thus reducing the overhead and improving fine-tuning efficiency. Low-Rank Adaptation (LoRA) [13] decomposes changes in each layer’s weight matrices into the product of two low-rank matrices. Prefix Tuning [16] prepends a sequence of learnable task-specific vectors to the input sequence. P-Tuning [17] adds trainable prompt embeddings to the input and optimizes them through a prompt encoder.

We conducted preliminary experiments to compare these three PEFT methods under identical ratios of trainable parameters (0.12%, 0.06% and 0.03%), using no fine-tuning (0%) and full fine-tuning (100%) as baselines. As shown in Table I, LoRA consistently achieves the highest Rouge-L [33] scores across all parameter ratios. Accordingly, we adopt LoRA as the PEFT method for FedTPA.

### B. Heterogeneous Federated Learning

Several optimization strategies have been proposed to address scenarios involving heterogeneous client data. Methods such as

TABLE I  
COMPARISON OF FINE-TUNING METHODS ON ROUGE-L UNDER DIFFERENT RATIOS OF TRAINABLE PARAMETERS. QWEN2.5-3B IS USED AS THE BACKBONE NETWORK, WITH THE DATASET SAMPLED FROM DATABRICKS-DOLLY-15K.

Fine-Tuning Method	0.12%	0.06%	0.03%
LoRA	<b>33.67</b>	<b>33.66</b>	<b>34.05</b>
Prefix Tuning	12.04	15.55	10.97
P-Tuning	32.78	32.33	30.11
No fine-tuning (0%)	12.2		
Full fine-tuning (100%)	22.4		

FedCurv [18] and FedProx [19] introduce regularization terms in the loss function to constrain local models from deviating excessively from the global model. Approaches like Astraea [20] and FedMix [21] employ data augmentation techniques to generate synthetic data, thereby enhancing the diversity of local datasets. Other methods, including FedPer [22] and LG-FedAvg [23], partition the model into globally shared base layers and locally personalized layers, aiming to balance global commonality and local specificity in data distributions. Although these methods have shown promising results on traditional small-scale models, they may encounter challenges such as computational inefficiency, increased risk of hallucination, and incompatibility with PEFT adapters when applied to large language models in federated instruction tuning scenarios.

### C. Heterogeneous Federated Instruction Tuning

Research on heterogeneous federated instruction tuning is still in its early stages. FedIT [14] was the first to introduce instruction tuning for clients with heterogeneous tasks, using a homogeneous LoRA combined with FedAvg to train a unified global model. FlexLoRA [15] leverages heterogeneous-dimensional LoRA and SVD techniques to make full use of each client’s local computational resources. LEGENDS [24] assigns different LoRA module depths and dimensional configurations to clients to improve overall federated fine-tuning efficiency. These studies paid insufficient attention to the learning effectiveness of different trainable parameter sizes on clients with heterogeneous local data characteristics and task complexities, which may undermine the global model’s ability to capture client-specific features. HETLoRA [25] introduces a regularization term to gradually guide each client model to adaptively determine LoRA module dimensions, aiming to match local computational constraints and data complexity. However, its experimental results show that this method converges too slowly, potentially leading to significant computational and communication overhead for large language models with billions of parameters.

## III. METHODS

### A. Preliminaries

LoRA is a high-performance PEFT method that introduces two low-rank adapter matrices  $\mathbf{B} \in \mathbb{R}^{d \times r}$  and  $\mathbf{A} \in \mathbb{R}^{r \times l}$  alongside the weight matrices  $\mathbf{W} \in \mathbb{R}^{d \times l}$  of the pre-trained backbone network  $\mathcal{M}_\phi$ , where  $r \ll \min\{d, l\}$ . It approximates

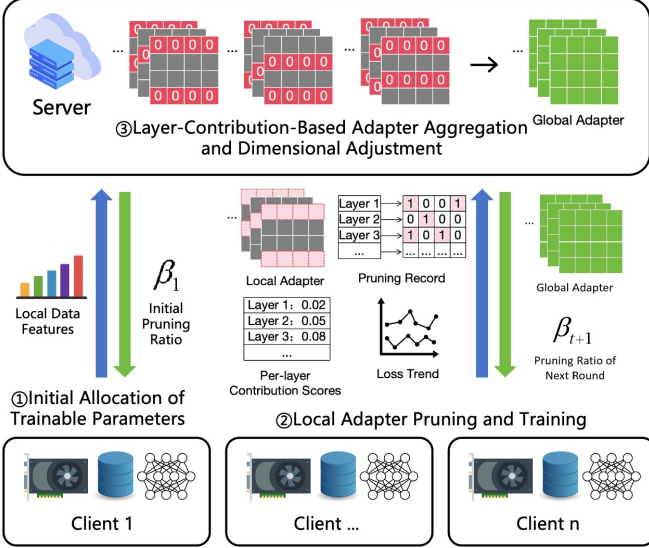


Fig. 2. Overview of FedTPA. The process includes: (1) initial allocation of trainable parameters based on feature divergence between local and global data; (2) local adapter pruning and training, where clients retain higher-contribution dimensions and upload pruning logs with contribution scores; and (3) layer-contribution-based adapter aggregation and dimensional adjustment on the server, which restores heterogeneous adapters, aggregates them with contribution-weighted fusion, and dynamically updates pruning magnitudes according to loss trends.

the changes to the backbone’s weight matrix as the product of two low-rank matrices, i.e.,

$$\Delta W \approx BA. \quad (1)$$

During fine-tuning, the parameters of the pre-trained backbone network are frozen, and only the parameters of the low-rank adapters (referred to as trainable parameters) are updated.

In a federated setting, since the backbone network remains unchanged during training, client  $i$  only needs to upload its local adapter parameters  $M_A^i$  after local training, thereby reducing communication overhead. The server aggregates the adapter parameters uploaded by all clients to obtain a global adapter

$$M_A^g = \text{agg}_{i=1,2,\dots,n} (M_A^i), \quad (2)$$

which is then distributed back to each client for the next round of fine-tuning.

### B. FedTPA Overview

FedTPA proposes a dynamic adjustment strategy for the local trainable parameter size, enabling the allocation of varying trainable parameter capacities to different clients in order to enhance the model’s learning effectiveness on client-specific instruction data. The strategy includes the following steps:

1) *Initial Allocation of Trainable Parameters*: Before training begins, each client extracts and uploads its local data features. The server computes global data features and initializes the trainable parameter size for each client based on the discrepancy between local and global features. Clients with higher data feature divergence are assigned more trainable parameters, allowing the global model to better learn from their instruction data.

2) *Local Adapter Pruning and Training*: The server distributes the global adapter and the configuration for trainable parameter sizes to all clients. Each client then trains on its local dataset while pruning adapter dimensions with lower contributions to convergence, thus realizing heterogeneous trainable parameter sizes. Clients log pruning records and per-layer contribution scores as reference data, which are uploaded to the server along with the local adapter parameters.

3) *Layer-Contribution-Based Adapter Aggregation and Dimensional Adjustment*: After all clients complete local training for the current round, the server first uses the pruning logs to zero-pad the heterogeneous adapters from each client, restoring them to a homogeneous dimension. It then aggregates the local adapter parameters into a global adapter, using the per-layer contribution scores from each client as aggregation weights. Meanwhile, the server monitors local training and validation loss trends across clients, dynamically adjusting the pruning magnitude to ensure that the number of trainable parameters remains well-matched to the complexity of each client’s local data.

### C. Initial Allocation of Trainable Parameters

A key prerequisite for allocating heterogeneous trainable parameter sizes is to quantitatively measure the difference between each client’s local data and the global data features. Thanks to the strong natural language understanding capability of the pre-trained backbone network, it can effectively capture the characteristics of instruction data [26].

Before training begins, client  $i$  uses its local backbone model  $M_\phi^i$  to obtain a feature representation for each instruction sample  $(I, x, y)$  in its local instruction dataset  $\mathcal{D}_i$ . The mean of these feature vectors is then taken as the overall representation of the client’s local data, denoted as:

$$\lambda_i = \frac{1}{|\mathcal{D}_i|} \sum_{(I,x,y) \in \mathcal{D}_i} M_\phi^i(I \oplus x \oplus y), \quad (3)$$

where  $(I, x, y)$  represents a prompt–input–output triplet sample, and  $\oplus$  denotes string concatenation. The client then uploads its extracted local data feature vector  $\lambda_i$  to the server.

Upon receiving all clients’ local feature vectors, the server computes the global data feature vector  $\lambda_g$  using a weighted average, where the weight is proportional to the data volume of each client. Then, the server calculates the cosine distance  $d_i$  between each client’s local feature vector and the global feature vector as a measure of how different the client’s data distribution is from the global one:

$$d_i = 1 - \frac{\lambda_i \cdot \lambda_g}{\|\lambda_i\| \|\lambda_g\|}. \quad (4)$$

Let  $\beta_i^t$  represent the pruning ratio for client  $i$  after local training in global round  $t$ , i.e., the proportion of remaining local adapter parameters relative to the global adapter parameter size. Denote the maximum and minimum values of all clients’ cosine distances as  $d_{\max}$  and  $d_{\min}$ , respectively.

For the client farthest from the global feature (i.e., with  $d_i = d_{\max}$ ), we retain all adapter parameters. For the client closest to the global feature (i.e., with  $d_i = d_{\min}$ ), we set its initial pruning

ratio to a lower bound  $\beta_m$  (a hyperparameter with  $0 < \beta_m < 1$ ). The initial pruning ratio for client  $i$  is then linearly mapped based on its distance:

$$\beta_i^1 = \frac{1 - \beta_m}{d_{\max} - d_{\min}} (d_i - d_{\min}) + \beta_m. \quad (5)$$

In this way, we achieve the goal of assigning an initial number of adapter trainable parameters to each client based on the divergence of its instruction data distribution from the global distribution.

#### D. Local Adapter Pruning and Training

In the local adapter pruning stage, each client prunes a portion of the global adapter’s trainable parameters based on the current global adapter parameters and pruning ratio  $\beta$  received from the server, in order to adapt to the characteristics and complexity of its local instruction data. Specifically, we reduce the number of trainable parameters by removing dimensions in the LoRA adapter that contribute least to convergence — i.e., by pruning certain columns of matrix  $B$  and rows of matrix  $A$ . Prior studies have validated the effectiveness of pruning based on convergence contribution in centralized learning settings [27] [28]. Considering the computational limitations of edge devices, we introduce several simplifications and improvements to environments. We divide the local pruning training process into three stages: warm-up, pruning, and fine-tuning to convergence.

1) *Warm-up Stage*: The local model first performs a fixed number of training steps using standard fine-tuning. During each step, we compute the gradient-weight product for each parameter as a measure of its contribution to convergence. Let the contribution of the  $k$ -th parameter in the  $j$ -th dimension of the  $i$ -th layer’s adapter at training step  $t$  to the loss function  $\mathcal{L}$  be denoted as  $h(w_{i,j,k}^t)$ . To smooth out fluctuations and reduce the influence of noise, we apply Exponential Moving Average (EMA) with decay rate  $\eta$ , resulting in the overall contribution score during the warm-up stage:

$$h(w_{i,j,k}^t) = \eta \cdot h(w_{i,j,k}^{t-1}) + (1 - \eta) |w_{i,j,k}^t \cdot \nabla_{w_{i,j,k}^t} \mathcal{L}|. \quad (6)$$

2) *Pruning Stage*: For each adapter dimension, we calculate the average contribution across all its parameters as its dimension-wise contribution score. We then sort these scores in descending order across all layers and dimensions and retain the top  $\lfloor \beta N r_g \rfloor$  dimensions, where  $N$  is the number of adapter layers and  $r_g$  is the dimension of the global adapter. The structure of each adapter layer is then reconfigured accordingly. At the same time, we maintain a pruning log, a Boolean matrix of shape  $N \times r_g$ , to indicate whether each dimension in each adapter layer was pruned. This log will be used later by the server during aggregation to restore the full adapter structure.

3) *Fine-tuning to Convergence Stage*: After restructuring the adapters, the client continues training for the remaining steps using the pruned adapter.

Through this multi-stage process, each client ends up with an adapter containing a heterogeneous number of trainable parameters, tailored to its own data complexity and feature divergence.

#### E. Layer-wise Contribution-based Aggregation

We aim to assign higher aggregation weights to client models whose local data distributions deviate more from the global features or exhibit higher complexity, encouraging the global model to focus on these clients and thus alleviating client drift under data heterogeneity. Based on the parameter contribution scores obtained during local adapter pruning, and balancing aggregation effectiveness with communication efficiency, we compute contributions at the adapter-layer level and perform aggregation on a per-layer basis.

After completing local training, client  $i$  averages the contribution scores of all retained dimensions within each adapter layer to obtain the layer-wise contribution score:

$$h(\mathbf{M}_{A,i}^j) = \frac{1}{r_i^j} \sum_{k=1}^{r_i^j} h(\mathbf{M}_{A,i}^{j,k}), \quad (7)$$

where  $r_i^j$  denotes the number of dimensions in the  $j$ -th adapter layer of client  $i$ , and  $h(\mathbf{M}_{A,i}^{j,k})$  is the contribution score of the  $k$ -th dimension in that layer.

The server then uses the pruning logs to zero-pad each client’s heterogeneous adapter back to a unified global dimension. Using the normalized layer-wise contribution scores  $\tilde{h}(\mathbf{M}_{A,i}^j)$  from each client as weights, the server performs layer-wise weighted aggregation to construct the global adapter model:

$$\mathbf{M}_{A,g}^j = \sum_{i=1}^n \tilde{h}(\mathbf{M}_{A,i}^j) \mathbf{M}_{A,i}^j. \quad (8)$$

#### F. Dynamic Adjustment of Trainable Parameter Capacity

To enhance the stability of federated instruction tuning, the server dynamically adjusts each client’s pruning ratio  $\beta_i^{t+1}$  for the next round based on the local training or validation loss  $\mathcal{L}_i$  reported in the current round, following the strategy below:

$$\beta_i^{t+1} = \begin{cases} (1 + \delta)\beta_i^t, & \text{if } \mathcal{L}_{i,\text{train}}^t > \mathcal{L}_{i,\text{train}}^{t-1}, \\ (1 - \delta)\beta_i^t, & \text{if } \mathcal{L}_{i,\text{train}}^t \leq \mathcal{L}_{i,\text{train}}^{t-1} \text{ and } \mathcal{L}_{i,\text{eval}}^t > \mathcal{L}_{i,\text{eval}}^{t-1}. \end{cases} \quad (9)$$

On one hand, if the training loss increases, it indicates that the current trainable parameter size is insufficient to effectively fit the client’s data features. Therefore, the pruning ratio is increased moderately in the next round to retain more trainable parameters, thereby enhancing the model’s learning capacity. On the other hand, if the training loss decreases but the validation loss increases, it suggests that the client may be experiencing overfitting. In this case, the pruning ratio is decreased moderately in the next round to reduce model complexity and improve generalization performance. The magnitude of adjustment to the pruning ratio is controlled by the hyperparameter  $\delta$  ( $0 < \delta < 1$ ).

## IV. EXPERIMENTS

### A. Experimental Setup

1) *Model Configuration*: We use Qwen2.5 [29] (3B) as the backbone with 4-bit NF4 quantization [30]. LoRA adapters are inserted into the query and value projections of each Transformer layer for parameter-efficient fine-tuning.

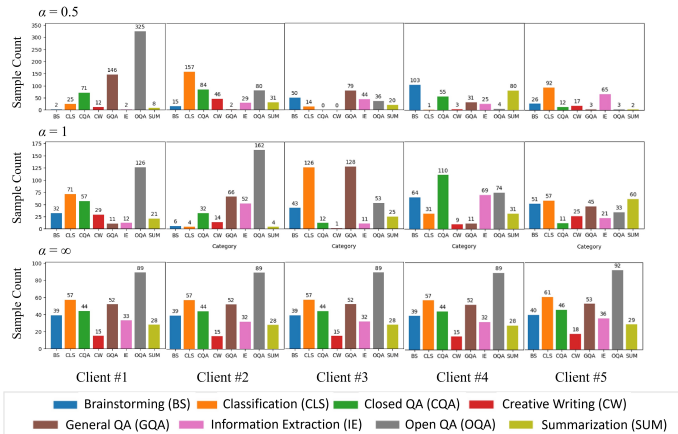


Fig. 3. Task category distribution per client under different data distribution scenarios in Dolly-databricks-15k.

2) *Datasets*: We evaluate on two instruction tuning datasets: Dolly-databricks-15k [31], a general-purpose English dataset covering classification, creative writing, information extraction, and QA; and LawBench [32], a Chinese legal-domain dataset with tasks such as statute memorization, legal consultation, and reading comprehension.

3) *Baseline Methods*: We compare FedTPA with FedIT [14], which employs homogeneous LoRA (rank 16) and FedAvg, and FlexLoRA [15], which supports heterogeneous adapter ranks (8–64) with SVD-based aggregation.

4) *Experimental Setup*: We simulate federated fine-tuning with 5 clients. For each dataset, 2,000 samples are split into 90% training and 10% testing, with training data partitioned under IID and Non-IID Dirichlet settings ( $\infty$ , 1, 0.5). Figure 3 illustrates client-wise task distributions. Local data is split 8:2 for training and validation. FedTPA uses a global adapter rank of 16, runs for 15 communication rounds with one local update per round, and is evaluated using Rouge-L on the global test set.

## B. Results and Analysis

Table II reports the Rouge-L scores of FedTPA compared with FedIT and FlexLoRA on the Dolly-databricks-15k and LawBench datasets. Across all data distribution settings and datasets, FedTPA consistently outperforms the two baselines, achieving improvements of up to approximately 3%. The inferior performance of FedIT and FlexLoRA can be attributed to their limited adaptation strategies: FedIT employs homogeneous adapters, while FlexLoRA allocates adapter dimensions solely according to local computational resources. These strategies often fail to cope with heterogeneous and complex local data distributions, leading to either overfitting or underfitting. In contrast, FedTPA leverages an initial allocation of trainable parameters combined with dynamic adjustment of adapter dimensions during training, effectively mitigating such risks and preserving the global model’s learning capacity.

To further assess the local learning effectiveness of FedTPA, we analyze the evolution of local training and validation loss. As illustrated in Figure 4, the local training loss typically shows

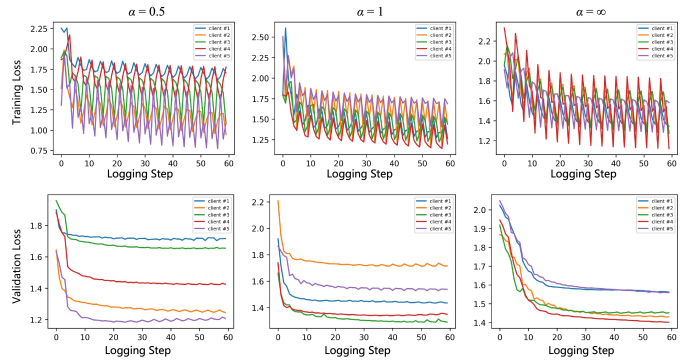


Fig. 4. Trends of local training and validation loss with respect to training steps on each client (each global round contains four logging steps).

TABLE II  
ROUGE-L SCORES OF FEDIT, FLEXLoRA, AND FEDTPA ON DOLLY-DATABRICKS-15K AND LAWBENCH

Dataset	$\alpha$	FedIT	FlexLoRA	FedTPA
Dolly-data-bricks-15k	0.5	33.76	34.91	<b>35.68</b>
	1	33.51	34.95	<b>36.39</b>
	$\infty$	33.97	34.49	<b>35.53</b>
LawBench	0.5	49.45	46.66	<b>50.08</b>
	1	49.62	48.76	<b>50.07</b>
	$\infty$	49.84	48.34	<b>50.18</b>

a brief rebound after each global model update but follows an overall decreasing trend as training progresses. Due to task heterogeneity across clients, the magnitude of local validation loss varies, yet it also steadily decreases and converges. These observations indicate that FedTPA achieves stable convergence across all three data distribution settings.

We also examine the evolution of pruning ratios across global rounds. Initially, the server assigns heterogeneous pruning ratios to clients according to the divergence between local and global data characteristics (Figure 5).

- Under IID distributions ( $\alpha = \infty$ ), the global model already generalizes well on local datasets, and most adapter dimensions require little adjustment.
- Under Non-IID distributions ( $\alpha = 1$  or 0.5), the initial allocation may cause overfitting or underfitting for certain clients as training progresses.

FedTPA mitigates this by monitoring client-specific training and validation loss at every global round and dynamically adjusting adapter dimensions. This adaptive mechanism ensures consistent improvement of the global model on both the global test set and client-side validation sets.

## C. Ablation and Sensitivity Analysis

In this section, we conduct ablation experiments and sensitivity analyses to evaluate the effectiveness of the three core mechanisms in FedTPA: (1) Initial Trainable Parameter Allocation (ITPA), (2) Dynamic Adjustment of Pruning Ratios (DAPR), and (3) Layer Contribution-Weighted Aggregation (LCWA). We first examine the effect of individual hyperparameters  $\beta_m$  and  $\delta$ , then compare the aggregation strategy against FedAvg, and finally investigate the joint contributions of different component combinations.

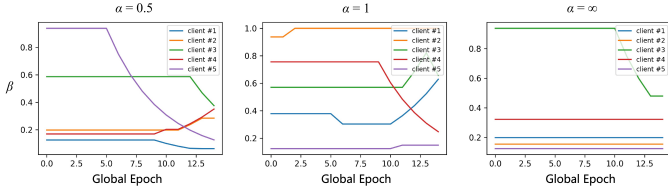


Fig. 5. Evolution of pruning ratio  $\beta$  for each client across global rounds.

TABLE III  
PERFORMANCE COMPARISON BETWEEN FEDAVG AND OUR AGGREGATION METHOD ON EACH TASK

Datasets	Dolly-data-bricks-15k	LawBench
$\alpha = 0.5$	34.27 (-1.41)	48.89 (-1.19)
$\alpha = 1$	34.67 (-1.72)	48.96 (-1.11)
$\alpha = \infty$	35.22 (-0.31)	49.05 (-1.13)

1) *Impact of Initial Allocation ( $\beta_m$ ):* The hyperparameter  $\beta_m$  controls the minimum proportion of trainable parameters initially allocated to each client. We evaluate  $\beta_m \in \{0.125, 0.25, 0.5, 0.75, 1\}$  and report both task performance and the magnitude of pruning ratio adjustments ( $\Delta\beta$ ) during training (Figure 6). When  $\beta_m = 1$ , all clients receive the same allocation, disabling heterogeneous initialization (i.e., ablation of ITPA). This setting leads to degraded convergence, as the server must frequently intervene with large adjustments. In contrast, smaller values ( $<0.5$ ) enable more diverse local adapter sizes, better matching local data complexity and improving stability. This confirms that heterogeneous initialization is crucial, with moderate  $\beta_m$  values achieving the best trade-off.

2) *Impact of Dynamic Adjustment ( $\delta$ ):* The hyperparameter  $\delta$  determines the step size of pruning ratio adjustments. We compare  $\delta \in \{0, 0.1, 0.2, 0.3, 0.5\}$  (Figure 7). Setting  $\delta = 0$  eliminates dynamic adaptation (i.e., ablation of DAPR), which results in inferior performance, since pruning ratios remain fixed. In contrast,  $\delta > 0$  significantly improves results, highlighting the necessity of adaptive control. Sensitivity analysis further reveals that smaller  $\delta$  values (0.1–0.2) achieve the best stability, while larger values introduce oscillations. Thus, fine-grained dynamic control is more effective than aggressive adjustments.

3) *Impact of Contribution-Based Aggregation:* We then evaluate the aggregation strategy (LCWA) against FedAvg, which only weights clients by local dataset size. As shown in Table III, FedAvg consistently underperforms our method across varying heterogeneity levels. This indicates that dataset size alone is not a reliable proxy for contribution to global convergence. By incorporating layer-level contribution signals, LCWA assigns greater weight to influential clients, thereby improving adaptability to diverse instruction data distributions.

4) *Combined Effect of All Mechanisms:* Based on the above analyses, we further examine the joint contributions of the three mechanisms by comparing different component combinations. As summarized in Table IV, the complete model (ITPA+DAPR+LCWA) achieves the best performance (50.08). Removing ITPA (DAPR+LCWA) reduces performance to 49.44, while removing DAPR (ITPA+LCWA) yields 49.58.

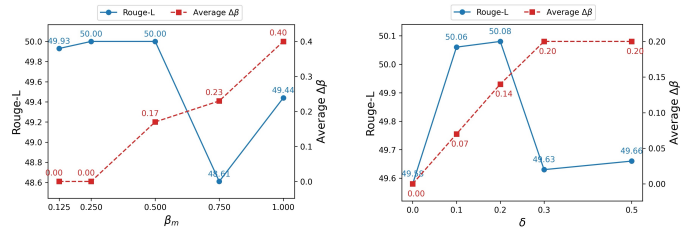


Fig. 6. Impact of different  $\beta_m$  values on the model's performance.

Fig. 7. Impact of different  $\delta$  values on the model's performance.

TABLE IV  
PERFORMANCE COMPARISON OF COMPONENT COMBINATIONS IN FEDTPA

Combination	Performance
ITPA + DAPR + LCWA	50.08
DAPR + LCWA	49.44 (-0.64)
ITPA + LCWA	49.58 (-0.50)
ITPA + DAPR	48.89 (-1.19)

In contrast, eliminating LCWA (ITPA+DAPR) leads to the largest degradation (48.89).

These results demonstrate that all three mechanisms are necessary and complementary: LCWA provides the largest performance boost, while ITPA and DAPR further enhance stability and adaptability. Only their joint synergy (ITPA+DAPR+LCWA) achieves the highest convergence and task performance, confirming the holistic design of FedTPA.

## V. CONCLUSION

In this work, we address the challenge in federated instruction tuning of LLMs, where the use of uniformly sized trainable parameters across clients leads to suboptimal local training performance. To tackle this issue, we propose FedTPA, a strategy that dynamically allocates and adjusts the trainable parameter size in local adapters to enhance the global model's ability to learn from local instruction data.

We validate the effectiveness of our method through experiments on multilingual and multi-task datasets, demonstrating its advantages over existing approaches. Furthermore, we conduct ablation and sensitivity analyses on key components to identify optimal hyperparameter configurations and to verify their effectiveness in enhancing task performance.

While our method focuses on handling data heterogeneity across clients, we acknowledge that heterogeneity in computational resources is also a critical challenge in federated instruction tuning and deserves further investigation in future work.

## VI. ACKNOWLEDGMENT

This work is supported by Science and Technology Major Special Program of Jiangsu Province under Grant No. BG2024028; Jiangsu Provincial Frontier Technology Research and Development Program under Grant No. BF2024070; the Fundamental Research Funds for the Central Universities; National Natural Science Foundation of China under Grant Nos. 62472094, 62572119, 62232004.

## REFERENCES

- [1] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, 30.
- [2] Devlin, J., Chang, M.-W., Lee, K., et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019*, pp. 4171–4186.
- [3] Kamalloo, E., Dziri, N., Clarke, C., et al. (2023). Evaluating Open-Domain Question Answering in the Era of Large Language Models. In *Proceedings of ACL 2023 (Volume 1: Long Papers)*, pp. 5591–5606.
- [4] Nejjar, M., Zacharias, L., Stiehle, F., et al. (2025). LLMs for science: Usage for code generation and data analysis. *Journal of Software: Evolution and Process*, 37(1), e2723.
- [5] Zhu, W., Liu, H., Dong, Q., et al. (2024). Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis. In *Findings of NAACL 2024*, pp. 2765–2781.
- [6] Wei, J., Bosma, M., Zhao, V., et al. (2022). Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations (ICLR)*.
- [7] Houshy, N., Giurghi, A., Jastrzebski, S., et al. (2019). Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the International Conference on Machine Learning (ICML)*.
- [8] Chung, H. W., Hou, L., Longpre, S., et al. (2024). Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research*, 25(70), 1–53.
- [9] Villalobos, P., Ho, A., Sevilla, J., et al. (2024). Will we run out of data? Limits of LLM scaling based on human-generated data. In *Proceedings of ICML 2024*, pp. 49523–49544.
- [10] Nurmi, J., Xu, Y., Boutellier, J., et al. (2023). SPHERE-DNA: Privacy-Preserving Federated Learning for eHealth. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1-6.
- [11] Ye, R., Wang, W., Chai, J., et al. (2024). OpenFedLLM: Training Large Language Models on Decentralized Private Data via Federated Learning. In *Proceedings of KDD 2024*, pp. 6137–6147.
- [12] Zhao, Y., Li, M., Lai, L., et al. (2018). Federated Learning with Non-IID Data. <https://arxiv.org/abs/1806.00582>
- [13] Hu, E. J., Shen, Y., Wallis, P., et al. (2022). LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*.
- [14] Zhang, J., Vahidian, S., Kuo, M., et al. (2024). Towards Building the Federated GPT: Federated Instruction Tuning. In *IEEE ICASSP 2024*.
- [15] Bai, J., Chen, D., Qian, B., et al. (2024). Federated Fine-tuning of Large Language Models under Heterogeneous Tasks and Client Resources. In *NeurIPS 2024*.
- [16] Li, X. L., Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of ACL-IJCNLP 2021*, pp. 4582–4597.
- [17] Liu, X., Zheng, Y., Du, Z., et al. (2024). GPT Understands, Too. *AI Open*, 5, 208–215.
- [18] Shoham, N., Avidor, T., Keren, A., et al. (2019). Overcoming Forgetting in Federated Learning on Non-IID Data. In *NeurIPS 2019*.
- [19] Li, T., Sahu, A. K., Zaheer, M., et al. (2020). Federated Optimization in Heterogeneous Networks. In *Proceedings of MLSys 2020*, Vol. 2.
- [20] Duan, M., Liu, D., Chen, X., et al. (2019). Astraea: Self-balancing Federated Learning for Improving Classification Accuracy of Mobile Deep Learning Applications. In *Proceedings of ICCD 2019*, pp. 246–254.
- [21] Wicaksana, J., Yan, Z., Zhang, D., et al. (2023). FedMix: Mixed Supervised Federated Learning for Medical Image Segmentation. *IEEE Transactions on Medical Imaging*, 42(7), 1955–1968.
- [22] Arivazhagan, M. G., Aggarwal, V., Singh, A. K., et al. (2019). Federated Learning with Personalization Layers. <https://arxiv.org/abs/1912.00818>
- [23] Liang, P. P., Liu, T., Ziyin, L., et al. (2019). Think Locally, Act Globally: Federated Learning with Local and Global Representations. In *NeurIPS 2019*.
- [24] Liu, J., Liao, Y., Xu, H., et al. (2024). Adaptive Parameter-Efficient Federated Fine-Tuning on Heterogeneous Devices. <https://arxiv.org/abs/2412.20004>
- [25] Cho, Y. J., Liu, L., Xu, Z., et al. (2024). Heterogeneous LoRA for Federated Fine-tuning of On-Device Foundation Models. In *Proceedings of EMNLP 2024*, pp. 12903–12913.
- [26] Bang, J., Lee, J., Shim, K., et al. (2024). Crayon: Customized On-Device LLM via Instant Adapter Blending and Edge-Server Hybrid Inference. In *Proceedings of ACL 2024*, pp. 3720–3731.
- [27] Zhang, Q., Zuo, S., Liang, C., et al. (2022). PLATON: Pruning Large Transformer Models with Upper Confidence Bound of Weight Importance. In *Proceedings of ICML 2022*, pp. 26809–26823.
- [28] Zhang, Q., Chen, M., Bukharin, A., et al. (2023). AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *ICLR 2023*.
- [29] Yang, A., Yang, B., Zhang, B., et al. (2024). Qwen2.5 Technical Report. <https://arxiv.org/abs/2412.15115>
- [30] Dettmers, T., Pagnoni, A., Holtzman, A., et al. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. In *NeurIPS 2023*, pp. 10088–10115.
- [31] Conover, M., Hayes, M., Mathur, A., et al. (2023). Free Dolly: Introducing the World’s First Truly Open Instruction-Tuned LLM. <https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm>
- [32] Fei, Z., Shen, X., Zhu, D., et al. (2024). LawBench: Benchmarking Legal Knowledge of Large Language Models. In *Proceedings of EMNLP 2024*, pp. 7933–7962.
- [33] Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pp. 74–81.
- [34] Cai, X., Yu, N., Zhao, M., et al. (2024). Decentralized Federated Learning in Partially Connected Networks with Non-IID Data. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1-6.