pFedGPT: Hierarchically Optimizing LoRA Aggregation Weights for Personalized Federated GPT Models

Zhanming Shen[♠], **TianQi Xu**[♠], **Hao Wang**[♣*], **Jian Li**[♢], **Miao Pan**[♡]
[♠]Zhejiang University, [♣]Stevens Institute of Technology
[♢]Stony Brook University, [♡]University of Houston

Abstract

Federated finetuning of Large Language Models (LLMs) using Low-Rank Adaptation (LoRA) offers computational efficiency and preserves data privacy. However, applying LoRA in federated settings faces significant challenges: standard approaches struggle with data heterogeneity, and existing personalization techniques fail to precisely adapt shared global knowledge to individual client needs. To address these issues, we propose pFedGPT, a framework that leverages Hierarchical Bayesian Optimization (HBO) for fine-grained, personalized LoRA aggregation. pFedGPT intelligently partitions LoRA parameters based on model structure and client information, then employs HBO to hierarchically search for optimal, module-specific weights. This enables a nuanced integration of the downloaded global LoRA state with each client's local model, precisely capturing client-specific requirements. To manage the optimization cost inherent in HBO, pFedGPT incorporates efficient multi-fidelity evaluations and a curriculum learning strategy. Extensive experiments demonstrate that pFedGPT achieves state-of-the-art (SOTA) performance on personalized FL benchmarks, showcasing robustness and scalability while introducing only minimal (approx. 4%) additional optimization overhead. Our results also underscore the limitations of traditional FL methods for LoRA-based LLM personalization, highlighting the need for tailored approaches like pFedGPT.

1 Introduction

The rapid development of Large language models (LLMs) has drawn widespread attention from academia (Devlin et al., 2018; Radford et al., 2019; Raffel et al., 2020; Zhang et al., 2022). To further improve LLM performance on various downstream tasks, the demand for high-quality training data across different domains is growing. However, this creates a conflict with the need to protect data sensitivity and privacy (Balunovic et al., 2022; Gupta et al., 2022; Klymenko et al., 2022).

This challenge has led to increasing interest in fine-tuning LLMs within the framework of federated learning (FL) (Yu et al., 2023; Zhang et al., 2024a), as it allows for decentralized training while preserving data privacy. To reduce communication and computational costs, Parameter-Efficient Fine-Tuning (PEFT) methods like low-rank adaptation (LoRA) (Hu et al., 2021) have been adopted, offering a more efficient way to update models without transmitting large weights. However, applying LoRA in FL presents challenges. As data becomes more heterogeneous across clients, the gap between fully fine-tuning the model and using LoRA widens (Babakniya et al., 2023). Additionally, privacy-preserving techniques, such as gradient noise for differential privacy, can destabilize LoRA's performance (Sun et al., 2024). Moreover, global models may not perform well for specific personalized tasks (Wang et al., 2023).

These problems prompted us to propose a method for achieving model personalization in LLMs. While personalized Federated Learning (pFL) has been well-researched in traditional machine learning, directly applying it to LLM fine-tuning presents challenges. Most existing Personalized Federated Learning (pFL) solutions are designed for fully trained models (Collins et al., 2021; Oh et al., 2021; Zhang et al., 2023b), making them incompatible with Parameter-Efficient Fine-Tuning (PEFT) methods. In addition, some pFL approaches are not specifically optimized for LLMs (Wu et al., 2023; Yi et al., 2023; Zhang et al., 2023a). Although they can theoretically provide personalized solutions, they often fall short in practice due to the complexity of LLMs and the specific needs of PEFT methods. Thus, there is a pressing need for personalized FL approaches tailored to

*Corresponding author (Hao Wang, hwang9@stevens.edu)

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 4766–4778
November 4-9, 2025

LLMs, capable of leveraging global information while enhancing the performance of local models.

Recent efforts to personalize LLMs in FL scenarios still face challenges. FedDPA (Yang et al., 2024) combines local and global LoRA outputs with a single weight, but struggles with managing multiple adapters and fail to capture the personalized information precisely. PerFIT (Zhang et al., 2024b) uses neural architecture search to identify personalized architectures, yet it overlooks that the degree of personalization in the parameter space evolves during training, leading to suboptimal results. In short, these approaches fail to accurately capture the client-specific information in the global model, preventing the local model from fully benefiting from FL. Most importantly, their tailored personalization for LLMs is just superficial and not based on LoRA's own structure.

Thus motivated, to better capture the necessary information in the global model downloaded by each client, dynamically implement optimal personalization of the local model, and tailor the training framework for the LLM, we propose **P**ersonalized **F**ederated **GPT** (*pFedGPT*), a novel pFL method with Hierarchical Bayesian Optimization (HBO) to introduce hierarchical Bayesian optimization based on curriculum learning and multi-fidelity algorithms into the model training. Our contributions are as follows:

- We introduce *pFedGPT*, a method that performs more fine-grained parameter aggregation of local and global model by conducting Bayesian Optimization on a personalized parameter space searched by each client, thus accurately capturing the desired information in the downloaded global LoRA.
- We propose a new LLM-based distribution of data heterogeneity: Task-specific distribution, which we use together with the traditional Dichilet distribution as a benchmark for evaluating the personalization capability of LLMs in the context of FL. Based on this, it proves the inadaptability of the traditional FL method in the PEFT of the LLMs and the necessity of proposing a new personalized method based on LoRA.
- We conducted extensive experiments on three benchmark datasets. The results show that *pFedGPT* performs better than state-of-the-art (SOTA) methods but introduces only 4% additional optimization time.

2 Preliminary

2.1 LoRA

LoRA achieves PEFT by constraining the update of model parameters to maintain a low intrinsic rank. For a pre-trained LLM parameterized by $\theta_{init} \in \mathbb{R}^{d\times k}$, LoRA utilizes a low-rank decomposition AB to represent the update $\Delta\theta$ where $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$ with $r \ll \min(d, k)$. The pre-trained parameter θ remains fixed during the fine-tuning while A and B are optimized. The update of θ_{init} is formed as:

$$\theta_{new} = \theta_{init} + \Delta\theta = \theta_{init} + AB.$$

2.2 Bayesian Optimization (BO)

Bayesian optimization is used to optimize objective functions by modeling the objective function $f(\mathbf{x})$ with a Gaussian process. For a given prior, we have $f(\mathbf{x}) \sim \mathcal{GP}\big(\mu(\mathbf{x}),\, k(\mathbf{x}, \mathbf{x}')\big)$, where $\mu(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$ the covariance function. Given historical data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, the posterior distribution is $p\big(f(\mathbf{x}) \mid \mathcal{D}, \mathbf{x}\big) = \mathcal{N}\big(\mu_n(\mathbf{x}),\, \sigma_n^2(\mathbf{x})\big)$, where $\mu_n(\mathbf{x})$ and $\sigma_n^2(\mathbf{x})$ are the posterior mean and variance. The next sampling point $\mathbf{x}_{n+1}$ is selected by maximizing the acquisition function $\alpha(\mathbf{x})$: $\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}} \alpha(\mathbf{x})$.

2.3 Multi-Fidelity and Curriculum Learning

Multi-fidelity optimization aims to reduce the computational cost of evaluating expensive functions by utilizing cheaper, lower-fidelity approximations. The key idea is to combine information of varying fidelity to efficiently guide the optimization process.

Curriculum learning (Bengio et al., 2009) is a progressive learning strategy with increasing difficulty, which can accelerate the convergence of the training process and improve the generalization ability of the model. CNAS (Guo et al., 2020) extends the concept of curriculum learning from the data level to the generalized model element level. It starts from a small search space to search for neural structure, and uses the learned knowledge to gradually search in a larger space, which significantly improves the search efficiency and enhances the search effect. Our idea is similar to CNAS. The results of a wide-range rough parameter search in a simpler, lower-cost parameter space are used to guide a small-range parameter search in a more complex parameter space.

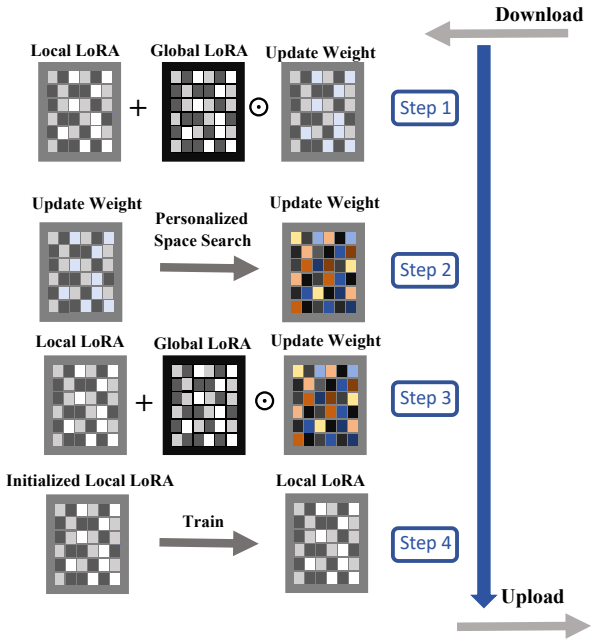

Figure 1: Workflow of the proposed method.

3 Overview

Figure 1 provides an overview of the local learning process on the client side. The client downloads the global LoRA parameters from the server and locally aggregates these parameters with the old local LoRA parameters through Hierarchical Bayesian Optimization for initialization (**Step 1,2,3**), which we refer to as **Algorithm HBO** in the subsequent discussion. Based on this initialization, the client trains the local model, and finally uploads the trained local LoRA parameters to the server (**Step 4**). To initialize next round training, the server aggregate all the LoRA modules from last round and distributes the aggregated LoRA module to clients. The details are as follows:

Step 1: Bayesian optimization based on basic parameter space: The local client segments the basic parameter modules of LoRA based on the different model structures loaded by LoRA (such as $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$). Then, it performs Bayesian Optimization within the basic parameter space defined by these modules to determine the initial optimal update weights $\mathbf{w_Q}$, $\mathbf{w_K}$, and $\mathbf{w_V}$. These weights are used to aggregate the global LoRA parameters with the local LoRA parameters.

Step 2: Personalized parameter space search Mechanism: To enable the local model to better learn the specific parts of the global knowledge at a more fine-grained level, we define a personalized parameter space. Based on the personalized training information, the local client executes the personalized clustering algorithm in each large parameter module divided in the first stage, so as to find the parameter layers with similar personalized degree to form a finer granularity small parameter module like $\mathbf{w_{Q1}}, \mathbf{w_{Q2}}, \ldots, \mathbf{w_{Qk}}$ (where $\mathbf{k}$ is the number of clusters). These smaller parameter modules are the basic units that make up the personalized parameter space, which allows for more detailed optimization.

Step 3: Bayesian Optimization In Personalized Parameter Space Based on Curriculum Learning: The training strategy follows a curriculum learning approach. The results from Bayesian optimization within the basic parameter space, are then used as priors for Bayesian optimization in the searched personalized parameter space. This advanced optimization seeks to determine the optimal way to aggregate the global LoRA parameters with the old local LoRA parameters in a more refined, personalized parameter space.

Step 4: Regular Local Training: Each client performs regular local training using LoRA, which is initialized by optimally combining global knowledge with local knowledge.

The backbone of the LLM is frozen throughout federated training processes. The complete federated training process is described in Appendix A.

4 *pFedGPT*'s Design

4.1 Multi-fidelity Mechanism at the Data Level

To reduce the training costs of Bayesian optimization, we employ a multi-fidelity mechanism at the data level. This mechanism involves selecting a subset of the global dataset that closely resembles the local data distribution for each client as local validation set $\mathbf{V}$, followed by clustering and sampling to create low-fidelity validation datasets $\mathbf{V}_{\text{sampled}}$. The full algorithmic details of subset selection, clustering, and sampling are presented in Appendix B.

The final outcome is the creation of:

1) High-fidelity validation dataset $\mathbf{V}_{\text{high-fidelity}}$: The full validation dataset $\mathbf{V}$ obtained from the global data.

2) Low-fidelity validation dataset $\mathbf{V}_{\text{low-fidelity}}$: A sampled subset $\mathbf{V}_{\text{sampled}}$ of the validation dataset.

4.2 BO based on Basic Parameter Space

Basic Parameter Space Partitioning Based on Model Internal Structure. We classify model parameters according to their roles within the LoRA

layers. Specifically, we focus on value projection (θ_{value}), query projection (θ_{query}), and key projection (θ_{key}). Other projections include output projection, feed-forward network input and output, and word token embeddings (denoted as θ_{output}, $\theta_{\text{fc\_in}}$, $\theta_{\text{fc\_out}}$, θ_{wte}).

Let θ denote the set of all model parameters. We categorize θ into subsets based on their original structural roles. For detailed discussion, we focus on the main attention components. Therefore, in the basic parameter space for the first Bayesian optimization stage, we have the parameter subsets are: $\Theta_{\text{basic}} = \{\theta_{\text{value}}, \theta_{\text{query}}, \theta_{\text{key}}\}$.

Bayesian optimization based on basic parameter space. In the first stage, we optimize the parameters in the basic parameter space. Each parameter subset $\theta_p \in \Theta_{\text{basic}}$ is assigned a hyperparameter $\mathbf{w}_p$ for global optimization. The aggregation of local and global parameters for each parameter subset θ_p is defined as:

$$\theta_{p,\text{agg}} = \mathbf{w}_p \cdot \theta_{p,\text{local}} + (1 - \mathbf{w}_p) \cdot \theta_{p,\text{global}},$$

where $\mathbf{w}_p \in [0, 1]$. The objective function for this optimization is defined as the loss function evaluated on a low-fidelity validation dataset $\mathbf{V}_{\text{low-fidelity}}$. The optimal weights for these parameters are denoted as:

$$\mathbf{w}_{\text{stage-1}} = \arg\min_{\mathbf{w}} L(\mathcal{M}(\mathbf{w}), \mathbf{V}_{\text{low-fidelity}}).$$

Then we use Gaussian process to apply Bayesian optimization to the objective function above. In the end, we get an approximate optimal solution $\mathbf{w}_p^{\text{stage-1}}$ for searching in the basic parameter space based on the low-fidelity dataset, which represents the approximate range of optimal solutions based on which we will conduct further fine-grained searches.

4.3 Personalized Parameter Space Search

To enhance FL by balancing global information with local personalization, we employ a personalized parameter space search mechanism based on the model's internal structure and training information for each client.

Personalized Parameter Space Based on Training Information. After partitioning the basic parameter space, we expand it by clustering parameters based on training information to capture the personalized needs of each client.

For each classified subset $\theta_p \in \{\theta_{\text{value}}, \theta_{\text{query}}, \theta_{\text{key}}\}$, we compute the following metrics for each parameter: 1) Mean squared difference between local and global parameters for LoRA-A and LoRA-B matrices, denoted as $\delta_{A,p,i}$ and $\delta_{B,p,i}$ respectively. 2) The difference in parameter change magnitude between local and global models for LoRA-A and LoRA-B matrices, denoted as $\Delta_{A,p,i}$ and $\Delta_{B,p,i}$ respectively. These metrics are defined as:

$$\delta_{A,p,i} = \frac{1}{n}\sum_{j=1}^{n}(\theta_{p,i}^{l,A,j} - \theta_{p,i}^{g,A,j})^2,$$

$$\delta_{B,p,i} = \frac{1}{n}\sum_{j=1}^{n}(\theta_{p,i}^{l,B,j} - \theta_{p,i}^{g,B,j})^2,$$

$$\Delta_{A,p,i} = \frac{1}{n}\sum_{j=1}^{n}(\Delta\theta_{p,i}^{l,A,j} - \Delta\theta_{p,i}^{g,A,j})^2,$$

$$\Delta_{B,p,i} = \frac{1}{n}\sum_{j=1}^{n}(\Delta\theta_{p,i}^{l,B,j} - \Delta\theta_{p,i}^{g,B,j})^2.$$

Here, $\Delta\theta_{p,i}^{l,A,j}$ and $\Delta\theta_{p,i}^{g,A,j}$ represent the local and global parameter changes for LoRA-A, respectively:

$$\Delta\theta_{p,i}^{l,A,j} = \theta_{p,i}^{(T),A,j} - \theta_{p,i}^{(0),A,j},$$
$$\Delta\theta_{p,i}^{l,B,j} = \theta_{p,i}^{(T),B,j} - \theta_{p,i}^{(0),B,j}.$$

The global parameters $\theta_{p,i}^{g,A}$ and $\theta_{p,i}^{g,B}$, as well as their change magnitudes, are obtained by averaging the local parameters and their changes across all clients:

$$\theta_{p,i}^{g} = \frac{1}{m}\sum_{k=1}^{m}\theta_{p,i}^{k}, \quad \Delta\theta_{p,i}^{g} = \frac{1}{m}\sum_{k=1}^{m}\Delta\theta_{p,i}^{k},$$

where m is the number of clients, $\theta_{p,i}^{k}$ represents the parameters of the k-th client, and $\Delta\theta_{p,i}^{k}$ represents the parameter changes of the k-th client. These metrics form the feature vectors $\mathbf{F}_{p,i}$ for clustering: $\mathbf{F}_{p,i} = [\delta_{A,p,i}, \delta_{B,p,i}, \Delta_{A,p,i}, \Delta_{B,p,i}]$.

Personalized Parameter Partition Result. Finally, we splice the clustering results of all parameter subsets together to get the personalized parameter subset of the client. The subset is defined as: $\Theta_{\text{personalized}} = \{c_{p,1}, c_{p,2}, \ldots, c_{p,n}\}$.

This personalized parameter subset constitutes the local personalized parameter space of the client. Subsequent fine-grained Bayesian optimization will be based on this space.

4.4 BO in Personalized Parameter Space

Our training strategy uses a curriculum learning approach. In the previous steps, we have completed

a simple and inexpensive Bayesian optimization in the basic parameter space and determined the roughly optimal update weights. Now, we want to apply the results of this preliminary phase with the relevant training information as a prior to more complex, costly and high-precision Bayesian optimization. This advanced optimization aims to find the optimal way to aggregate global LoRA parameters with the old local LoRA parameters in a more refined, personalized parameter space.

Curriculum Learning for Personalized BO: In the second stage, each parameter cluster $c_{p,i} \in \Theta_{\text{personalized}}$ is assigned a hyperparameter $\mathbf{w}_{c_{p,i}}$ for global optimization. The aggregation of local and global parameters for each parameter cluster $c_{p,i}$ is defined as: $c_{p,i,\text{agg}} = \mathbf{w}_{c_{p,i}} \cdot c_{p,i,\text{local}} + (1 - \mathbf{w}_{c_{p,i}}) \cdot c_{p,i,\text{global}}$, where $\mathbf{w}_{c_{p,i}} \in [0, 1]$.

Before the second stage bayesian optimization, the curriculum learning initialization incorporates two key components:

1) Initialization using $\mathbf{w}_{\text{stage-1}}$: The results from the first stage are used to initialize the optimization process for each cluster $c_{p,i}$ based on its parameter subset θ_p: $\mathbf{w}_{c_{p,i}}^{(0)} = \mathbf{w}_p^{\text{stage-1}}$.

2) Initialization Incorporating Prior Information: The training information from the basic parameter space optimization serves as the prior for the personalized parameter space optimization. The training process for each cluster $c_{p,i}$ is initialized using the training information from the corresponding parameter subset θ_p. This is achieved by fitting a Gaussian Process (GP) model using the collected prior information: GP $\sim \mathcal{N}(\mathbf{X}_{\text{prior}}, \mathbf{y}_{\text{prior}})$.

The optimization objective for the second stage bayesian optimization is defined as:

$$\mathbf{w}_{\text{stage-2}} = \arg\min_{\mathbf{w}} L(\mathcal{M}(\mathbf{w}), \mathbf{V}_{\text{low-fidelity}}, \mathcal{P}_{\text{prior}}).$$

Selection of Top Results for High-Fidelity Optimization: After performing low-fidelity Bayesian optimization, we select the top k best results and use them to perform high-fidelity optimization on the full validation dataset $\mathbf{V}_{\text{high-fidelity}}$. The final optimal weights are denoted as $\mathbf{w}_{\text{final}}$:

$$\mathbf{w}_{\text{final}} = \arg\min_{\mathbf{w} \in \text{Top-}k} L(\mathcal{M}(\mathbf{w}), \mathbf{V}_{\text{high-fidelity}}).$$

The final parameters are aggregated with the optimal $\theta_{\text{agg}} = \mathbf{w}_{\text{final}} \cdot \theta_{\text{local}} + (1 - \mathbf{w}_{\text{final}}) \cdot \theta_{\text{global}}$.

4.5 Personalized Slow Start Mechanism

In FL, as shown in Appendix D, local fine-tuning may achieve faster initial convergence compared to federated training (FedIT), but it often results in lower final accuracy. Since our method involves the aggregation of locally trained LoRA parameters and globally aggregated LoRA parameters, in order to avoid the local optima caused by the aggregation weight being too biased to the local parameters in the early stage of training, we employ a personalized slow start mechanism. Specifically, we monitor the convergence of the FL process using the relative change in evaluation loss over a sliding window. The process is defined as follows:

Let $\mathbf{L}_t$ and $\mathbf{L}_{t-1}$ be the arrays of training losses in the current and the previous sliding window, respectively, and let $\epsilon > 0$ be a small constant that avoids division by zero. We denote the relative change at round t by Δ_t:

$$\Delta_t = \frac{\left|\text{mean}\left(\mathbf{L}_t\right) - \text{mean}\left(\mathbf{L}_{t-1}\right)\right|}{\max\left(\text{mean}\left(\mathbf{L}_{t-1}\right), \epsilon\right)}.$$

The local client is regarded as having accumulated sufficient global knowledge (the "*Slow-Start*" phase ends) when Δ_t falls below a predefined tolerance $\delta_{\max}$ or when the training epoch index t reaches the upper bound $T_{\max}$:

$$\text{SlowStart}(t) = \begin{cases} \text{True, if } \Delta_t < \delta_{\max} \text{ or } t \geq T_{\max}, \\ \text{False, otherwise.} \end{cases}$$

5 Experiments

5.1 Experimental Settings

Dataset. We conducted our experiments on three datasets from the previous federal learning research: Databricks-dolly-15k (Zhang et al., 2024a), Flan 1 and Flan 2 (Yang et al., 2024). Each dataset has eight different NLP tasks. Details of each task can be found in the original article.

Data Distribution. To emulate the heterogeneous data distribution in local clients, we proposed two data heterogeneity distribution settings based on these datasets. The first is a Dirichlet distribution parameterized by a coefficient β, denoted as Dir(β), with β set to 0.5 throughout the experiments. At the same time, based on the powerful generalization ability of LLMs, we propose a new type of data distribution, which assigns each client a unique task type from the dataset categories, referred to as the *Task-Specific* distribution, as shown in Appendix C. Other training details are documented in Appendix E.

Method	Databricks-dolly-15k		Flan 1		Flan 2	
	Dir(0.5)	Task-Specific	Dir(0.5)	Task-Specific	Dir(0.5)	Task-Specific
FedAvg	72.58	72.59	73.65	71.52	72.76	69.12
FedAvgM	70.26	64.78	65.38	57.99	63.32	55.74
FedAdagrad	72.19	73.38	70.42	70.25	68.78	69.16
FedAdam	70.57	62.03	61.66	66.41	64.39	64.49
FedProx	72.22	72.80	72.98	69.72	75.89	69.02
FedYogi	69.40	63.00	64.43	65.81	64.18	65.35
FedIT	72.30	71.14	70.01	71.50	70.84	70.88
PerFIT	73.49	71.55	73.78	77.91	75.58	70.04
FedDPA	73.30	73.83	72.05	78.69	74.25	72.58
pFedGPT	**73.90**	**74.38**	**74.25**	**79.13**	**76.25**	**72.63**

Table 1: Comparison of our method with traditional and recent FL methods under Dir(0.5) and Task-Specific settings on Databricks-dolly-15k, Flan 1, and Flan 2 datasets.

Method	Dir(0.5)	Task-Specific	Overall
	Mean	Mean	Mean
FedAvg	72.33	71.08	71.71
FedAvgM	66.32	59.50	62.91
FedAdagrad	70.46	70.93	70.70
FedAdam	65.54	64.31	64.93
FedProx	73.03	70.51	71.77
FedYogi	65.34	64.05	64.69
FedIT	71.05	71.17	71.11
PerFIT	74.28	73.17	73.72
FedDPA	73.20	75.03	74.12
pFedGPT	**74.80**	**75.38**	**75.09**

Table 2: Mean performance under Dir(0.5), Task-Specific settings, and overall mean across three datasets.

	Computation	Communication
Method	Total time	Param./iter.
FedAvg	1386 min	$2 \times \Sigma$
FedAvgM	1422 min	$2 \times \Sigma$
FedAdagrad	1424 min	$2 \times \Sigma$
FedAdam	1456 min	$2 \times \Sigma$
FedProx	1506 min	$2 \times \Sigma$
FedYogi	1448 min	$2 \times \Sigma$
FedIT	1369 min	$2 \times \Sigma$
PerFIT	1866 min	$2 \times \Sigma$
FedDPA	2705 min	$2 \times \Sigma$
pFedGPT	1431 min	$2 \times \Sigma$

Table 3: Computing and communication cost on dolly dataset. Σ is the parameter amount in the LoRA.

5.2 Main Results

We compared our method with traditional FL methods compatible with LoRA (FedAvg (McMahan et al., 2017), FedAvgM (Hsu et al., 2019), FedAdagrad (Reddi et al., 2020), FedAdam (Reddi et al., 2020)), FedProx (Li et al., 2020), FedYogi (Reddi et al., 2020), as well as recent works specifically designed for applying LoRA in FL with large models (FedIT (Zhang et al., 2024a), PerFIT (Zhang et al., 2024b), FedDPA (Yang et al., 2024)). Our method was evaluated under the two proposed data distribution settings across the three datasets mentioned above. Following FedIT (Zhang et al., 2024a), we use the GPT-4o score as an evaluation indicator of the effectiveness of our model generation. Other baseline details are documented in Appendix E.2.

The results indicate the effectiveness of our approach across different tasks and data distributions, as shown in Table 1. Our method consistently outperforms traditional FL methods and recent works designed for LLMs with LoRA, highlighting the improvements in local task performance.

The statistical analysis, shown in Table 2, further proves our findings. Under the Dir(0.5), Task-Specific, and combined settings, our method demonstrates higher mean performance compared to traditional FL methods. This consistency across different data distributions and tasks highlights the limitations of traditional methods and emphasizes the necessity for novel pFL approaches.

Above all, our findings are:

1) SOTA Performance: Our method achieves SOTA performance across all tested datasets and methods, demonstrating the robustness and effectiveness of our method in enhancing local task performance.

2) Limitations of Traditional FL Methods: When faced with LLM scenarios that bring new forms of data heterogeneity to distribution, traditional FL methods often exhibit inferior performance under Task-Specific settings compared to Dir(0.5) settings. In contrast, our method shows improved performance under Task-Specific settings, indicating its superior adaptability to new tasks in LLM +FL scenarios. These results underscore the inadequacy of traditional FL methods in handling the complexities of LLMs and diverse data distribu-

Configuration	GPT-4o Avg. Score
Stage-1 BO + low fidelity	72.84
Stage-1 BO + high fidelity	73.56
Stage-2 BO + low fidelity	73.51
Stage-2 BO + high fidelity	73.63
Full pFedGPT (ours)	**73.90**
pFedGPT-slow start removed	73.59
pFedGPT + high fidelity†	**74.01**

Table 4: Ablation on BO stages and validation fidelity levels on Databricks-dolly-15k (Dir(0.5)). Scores are the mean of three independent GPT-4o judgments per output (higher is better). †Always uses the high-fidelity validation set in both stages (higher cost).

tions, thus supporting the need for innovative pFL methods designed for LLMs in the FL context.

5.3 Ablation Study

Effectiveness of *pFedGPT*. As shown in Table 4, we compare *six* settings on Databricks-dolly-15k (Dir(0.5)):
(i) *Stage-1 BO + low fidelity*: optimize only the Basic Parameter Space; validation on the low-fidelity subset;
(ii) *Stage-1 BO + high fidelity*: same as (i) but validation on the full (high-fidelity) set;
(iii) *Stage-2 BO + low fidelity*: each client searches its personalized parameter space; both BO and validation use the low-fidelity subset;
(iv) *Stage-2 BO + high fidelity*: identical to (iii) but validation uses the full set;
(v) *Full pFedGPT (ours)*: curriculum links Stage-1→Stage-2 and switches validation from low→high fidelity, yielding hierarchical BO with multi-fidelity;
(vi) *pFedGPT + high fidelity*: same as (v) but always validates on the full set in both stages (no low-fidelity sampling).
As in the main experiment, each reported number is the average of three independent GPT-4o judgments per output.

Our full *pFedGPT* even outperforms "Stage-2 BO + high fidelity" despite the latter's direct use of the expensive validation set. This suggests that, under the same number of optimization rounds, our hierarchical initialization provides a stronger starting point, allowing faster convergence to the optimum. Moreover, even when using high fidelity at every stage, the improvement over full pFedGPT is marginal (only +0.11), highlighting the efficiency and robustness of our multi-fidelity, curriculum-driven HBO framework.

Effectiveness under Different Learning Rates. To analyze the effectiveness of our method under different learning rates, we first studied the impact of different learning rates using the Databricks-dolly-15k dataset with the Dir(0.5) distribution. We tested three different learning rates: 5×10^{-5}, 1×10^{-4}, and 1.5×10^{-5}. The evaluation loss over communication rounds for each learning rate is illustrated in Figure 2.

Despite the initial slower convergence rate due to the reduced data volume when splitting part of the training set into a global dataset, our method's unique advantages ensure that the model achieves higher final accuracy. Additionally, our approach demonstrates continuous accuracy improvement even when FedIT start to overfit.

Computing and Communication Overhead. We record the total time cost for each method, as shown in Table 3. pFedGPT achieves SOTA performance while introducing only about 4% additional training time and no additional communication cost compared to the baseline methods, placing it among the top performers in terms of efficiency. Moreover, its extra overhead is significantly lower than that of the other two personalization algorithms specifically designed for LLMs, underscoring the superior efficiency of our approach.

Impact of different number of clients. To understand the impact of varying the number of clients on the performance of different FL methods, we conducted experiments with 8, 20, and 50 clients under the Dir(0.5) setting across the above three datasets. We selected FedAvg, FedProx, FedIT, PerFIT, and FedDPA based on their strong performance in the main experiments. The results are shown in Table 5, demonstrating the superior scalability and robustness of pFedGPT in real-world scenarios. In addition, we found that the increase in the number of clients brought more performance gains to the FL approach specifically designed for LLM compared to the traditional FL approach, further supporting our view of the need to customize the FL training approach for LLM.

Impact of different sizes of sampling weights. In order to evaluate the effect of each client's weight sampled from the global dataset on the model performance, we conducted experiments on all the aforementioned datasets using the sample weights {1/8, 1/4, 1/2, 1}. As Figure 3 shows, for a Dirichlet distribution with β set to 0.5, the guided validation set required by each client may need to be more generalized, so when the sample weight of the validation set goes up, there is a slight improvement in model performance, representing higher

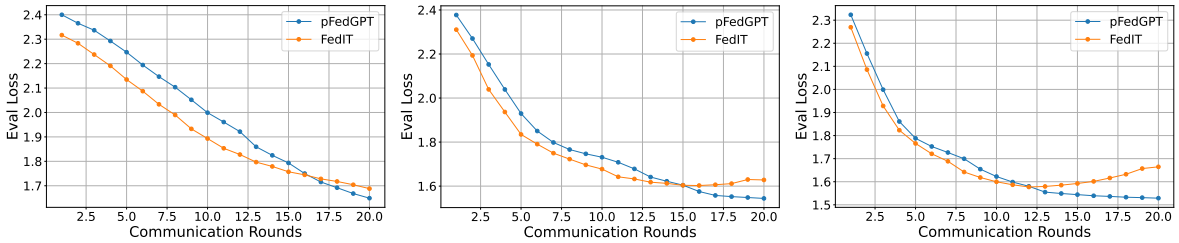

Figure 2: Evaluation loss vs. communication rounds for learning rate 5×10^{-5} / 1×10^{-4} / 1.5×10^{-4}.

Method	Databricks-dolly-15k			Flan 1			Flan 2		
	8 Clients	20 Clients	50 Clients	8 Clients	20 Clients	50 Clients	8 Clients	20 Clients	50 Clients
FedAvg	72.58	71.74	72.51	73.65	73.82	73.87	72.76	70.12	71.07
FedProx	72.22	72.39	74.68	72.98	72.35	73.47	75.89	75.10	74.85
PerFIT	73.49	75.20	76.05	73.78	74.25	75.10	75.58	76.35	76.90
FedDPA	73.30	73.35	74.10	72.05	72.90	72.50	74.25	76.00	76.65
pFedGPT	**73.90**	**75.61**	**76.74**	**74.25**	**74.00**	**75.90**	**76.25**	**77.10**	**77.85**

Table 5: Performance comparison of different methods with varying number of clients under Dir(0.5) setting across Databricks-dolly-15k, Flan 1, and Flan 2 datasets.

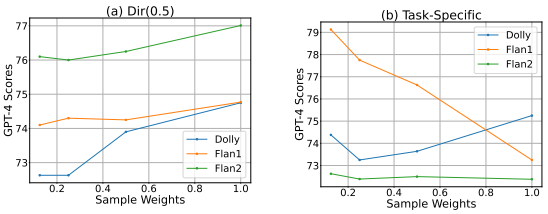

Figure 3: Comparison of model performance with different sizes of sampling weights.

precision of model personalization initialization. However, for the Task-Specific distributed data, the high degree of heterogeneity of the tasks (especially the Flan1/2 dataset also contains heterogeneity of output formats) makes it probably a safer choice for each client to select a smaller validation set that is more accurate. Specifically, in our experimental setup, a sample weight of 1/8 for a task-specific distribution approximates finding only the same Task data in the global dataset as the guided validation set, and thus tends to achieve a good experimental result. However, for our method, even if the worst sampling weights are selected, the model trained by our method still outperforms the vast majority of models on a specific task, and can basically achieve SOTA on the overall performance of all tasks.

6 Related Work

Parameter-Efficient Fine-Tuning (PEFT): In order to further liberate the limitations of FL in the context of LLMs, recent work has focused on integrating PEFT methods with FL Settings, including reducing communication costs (Malaviya et al., 2023; Nguyen et al., 2024; Sun et al., 2024; Xu et al., 2023; Zhang et al., 2023c), protecting differential privacy (Sun et al., 2024; Zhang et al., 2024a), and establishing fine-tuning frameworks (Kuang et al., 2023; Ye et al., 2024; Zhang et al., 2024a). In terms of alleviating data heterogeneity and achieving model personalization, SLoRA (Babakniya et al., 2023) finds a personalized starting point for the model through two-stage training and SVD matrix decomposition. PerFIT (Zhang et al., 2024b) uses neural architecture search to find a personalized architecture for each client. FedDPA (Yang et al., 2024) learns an additional local adapter during training and combines the output of the global and local adapters through an instance-level dynamic weight. Our *pFedGPT* is more accurate than the above methods and does not introduce additional memory costs. Fine-grained adaptive local aggregation based on model internal structure makes it possible to intelligently aggregate global and local models to fit local targets on each client. In addition, because *pFedGPT* modifies only local initialization in FL, it can be applied to existing FL methods to improve their performance without modifying other learning processes.

Bayesian Federated Learning (BFL) (Cao et al., 2023) extends traditional FL by deriving a global posterior distribution that aggregates knowledge from all clients. There are also some existing methods integrating Bayesian optimization (BO) with FL, such as FTS (Dai et al., 2020) and TFP (Zang et al., 2022), focus on improving efficiency through dimensionality reduction or zeroth-order optimization. However, these approaches lack considerations for applying BO to achieve fine-grained personalization and struggle to adapt to the unique challenges of PEFT in LLMs. Our proposed method, pFedGPT, introduces a Hierar-

chical Bayesian Optimization framework tailored for LoRA-based FL, enabling precise integration of global and local information. This approach achieves robust personalization, improved scalability, and state-of-the-art performance, addressing key gaps in existing FL-BO methods.

7 Conclusion

In this paper, we introduced *pFedGPT*, which leverages hierarchical Bayesian optimization to accurately capture the desired information in the downloaded global LoRA and integrates curriculum learning and multi-fidelity algorithms to reduce computational costs while maintaining accuracy. Our experiments show that *pFedGPT* outperforms SOTA methods with minimal extra optimization computational cost as well as maintains scalability and robustness. Additionally, we proposed a task-specific distribution benchmark to evaluate LLM personalization, demonstrating the limitations of traditional pFL methods and the necessity of proposing a new personalized methods based on LLMs.

Limitations

While our approach demonstrates strong performance, it has a few limitations. First, our experiments focus primarily on the QKV projections, which are the most frequently loaded in LoRA-based models, and further exploration is needed for other projections. Second, we did not investigate the similarity of personalization across different projections, which could lead to more efficient optimization by grouping similar projections. Finally, while we introduce a task-specific distribution, further work is needed to develop more advanced methods for measuring heterogeneity between different tasks, which would improve the understanding of LLM personalization in the context of federated learning.

Acknowledgments

The work of Hao Wang was supported in part by NSF 2534286, 2523997, 2315612, and the AWS Cloud Credit for Research program. The work of Jian Li was supported in part by NSF 2315614. The work of Miao Pan was supported in part by NSF 2403249. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. 2023. SLoRA: Federated Parameter Efficient Fine-Tuning of Language Models. *arXiv preprint arXiv:2308.06522*.

Mislav Balunovic, Dimitar Dimitrov, Nikola Jovanović, and Martin Vechev. 2022. Lamp: Extracting Text from Gradients with Language Model Priors. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proc. 26th Annual International Conference on Machine Learning (ICML)*.

Longbing Cao, Hui Chen, Xuhui Fan, Joao Gama, Yew-Soon Ong, and Vipin Kumar. 2023. Bayesian federated learning: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 7233–7242.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. 2021. Exploiting Shared Representations for Personalized Federated Learning. In *Proc. International Conference on Machine Learning (ICML)*.

Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. 2020. Federated bayesian optimization via thompson sampling. *Advances in Neural Information Processing Systems*, 33:9687–9699.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Yong Guo, Yaofo Chen, Yin Zheng, Peilin Zhao, Jian Chen, Junzhou Huang, and Mingkui Tan. 2020. Breaking the Curse of Space Explosion: Towards Efficient NAS with Curriculum Search. In *Proc. International Conference on Machine Learning (ICML)*.

Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. 2022. Recovering Private Text in Federated Learning of Language Models. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. *arXiv preprint arXiv:1909.06335*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685*.

Oleksandra Klymenko, Stephen Meisenbacher, and Florian Matthes. 2022. Differential Privacy in Natural Language Processing: The Story So Far. *arXiv preprint arXiv:2208.08140*.

Weirui Kuang, Bingchen Qian, Zitao Li, Daoyuan Chen, Dawei Gao, Xuchen Pan, Yuexiang Xie, Yaliang Li, Bolin Ding, and Jingren Zhou. 2023. FederatedScope-LLM: A Comprehensive Package for Fine-Tuning Large Language Models in Federated Learning. *arXiv preprint arXiv:2309.00363*.

Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, 37(3):50–60.

Shubham Malaviya, Manish Shukla, and Sachin Lodha. 2023. Reducing Communication Overhead in Federated Learning for Pre-Trained Language Models Using Parameter-Efficient Finetuning. In *Proc. Conference on Lifelong Learning Agents*.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient Learning of Deep Networks from Decentralized Data. In *Proc. International Conference on Artificial Intelligence and Statistics (ICAIS)*.

Duy Phuong Nguyen, J Pablo Munoz, and Ali Jannesari. 2024. Flora: Enhancing Vision-Language Models with Parameter-Efficient Federated Learning. *arXiv preprint arXiv:2404.15182*.

Jaehoon Oh, Sangmook Kim, and Se-Young Yun. 2021. FedBABU: Towards Enhanced Representation for Federated Image Classification. *arXiv preprint arXiv:2106.06042*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. 2020. Adaptive Federated Optimization. *arXiv preprint arXiv:2003.00295*.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. 2024. Improving LoRA in Privacy-Preserving Federated Learning. *arXiv preprint arXiv:2403.12313*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford Alpaca: An Instruction-Following LLaMA Model.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, and 1 others. 2023. How Far Can Camels Go? Exploring the State of Instruction Tuning on Open Resources. In *Proc. Advances in Neural Information Processing Systems (NeurIPS)*.

Xinghao Wu, Xuefeng Liu, Jianwei Niu, Guogang Zhu, and Shaojie Tang. 2023. Bold but Cautious: Unlocking the Potential of Personalized Federated Learning through Cautiously Aggressive Collaboration. In *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*.

Mengwei Xu, Yaozong Wu, Dongqi Cai, Xiang Li, and Shangguang Wang. 2023. Federated Fine-Tuning of Billion-Sized Language Models Across Mobile Devices. *arXiv preprint arXiv:2308.13894*.

Yiyuan Yang, Guodong Long, Tao Shen, Jing Jiang, and Michael Blumenstein. 2024. Dual-Personalizing Adapter for Federated Foundation Models. *arXiv preprint arXiv:2403.19211*.

Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. 2024. OpenFedLLM: Training Large Language Models on Decentralized Private Data via Federated Learning. *arXiv preprint arXiv:2402.06954*.

Liping Yi, Han Yu, Gang Wang, and Xiaoguang Liu. 2023. FedLoRA: Model-Heterogeneous Personalized Federated Learning with LoRA Tuning. *arXiv preprint arXiv:2310.13283*.

Sixing Yu, J Pablo Muñoz, and Ali Jannesari. 2023. Federated Foundation Models: Privacy-Preserving and Collaborative Learning for Large Models. *arXiv preprint arXiv:2305.11414*.

Lu Zang, Yang Qin, and Ruonan Li. 2022. Traffic flow prediction based on federated learning with joint pca compression and bayesian optimization. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3330–3335. IEEE.

Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. 2023a. FedALA: Adaptive Local Aggregation for Personalized Federated Learning. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*.

Jianqing Zhang, Yang Hua, Hao Wang, Tao Song, Zhengui Xue, Ruhui Ma, and Haibing Guan. 2023b. FedCP: Separating Feature Information for Personalized Federated Learning via Conditional Policy. In *Proc. 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. 2024a. Towards Building the FederatedGPT: Federated Instruction Tuning. In *Proc. ICASSP 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Pengyu Zhang, Yingbo Zhou, Ming Hu, Junxian Feng, Jiawen Weng, and Mingsong Chen. 2024b. Personalized Federated Instruction Tuning via Neural Architecture Search. *arXiv preprint arXiv:2402.16919*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, and 1 others. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068*.

Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. 2023c. Fedpetuning: When Federated Learning Meets the Parameter-Efficient Tuning Methods of Pre-Trained Language Models. In *Proc. Annual Meeting of the Association of Computational Linguistics (ACL)*.

Algorithm 1: The *pFedGPT* Framework

Require: N clients, global rounds T, initial global LoRA Θ_0, local learning rate α, slow-start thresholds

Ensure: Local LoRA parameters $\{\hat{\Theta}_i\}_{i=1}^{N}$

Server initializes Θ_0
for round $t = 1 \dots T$ **do**
  Server selects subset of clients I_t and sends Θ_{t-1}
  for client $i \in I_t$ **in parallel do**
    if Slow Start = True **then**
      get $\mathcal{V}_{\text{low-fid}}$ this round (cf. Eq. (1))
      $\theta_{t,i}^{\text{init}} \leftarrow$ Algorithm HBO
    else
      $\theta_{t,i}^{\text{init}} \leftarrow \theta_{\text{global}}$
    end if
    Local training:
    $\Theta_t^i \leftarrow \theta_{t,i}^{\text{init}} - \alpha \nabla_\theta L(\theta_{t,i}^{\text{init}}, \mathcal{D}_i)$
    Send Θ_t^i **to server**
  end for
  Server aggregates
  $\Theta_t \leftarrow \sum_{i \in I_t} \frac{k_i}{\sum_{j \in I_t} k_j} \Theta_t^i$
end for
return $\{\hat{\Theta}_i\}_{i=1}^{N}$

A The pFedGPT Framework

The complete federated training process is described in **Algorithm 1**.

B Details of Validation Subset Construction

B.1 Selection and Clustering of Validation Subsets

Each client selects a subset that is most similar to its local data distribution from the global dataset as a local validation set. This is achieved by calculating the cosine similarity between the local and global dataset embeddings. The similarity scores are sorted to select the top n most similar global data points as the local validation dataset $\mathbf{V}$. We then perform clustering on its normalized embeddings $\mathbf{E}_{\text{V,norm}}$ to identify groups of similar data points. The optimal number of clusters is determined by maximizing the silhouette score, yielding cluster labels $\mathbf{L}$ and cluster sizes $\mathbf{N}_{\text{clusters}}$.

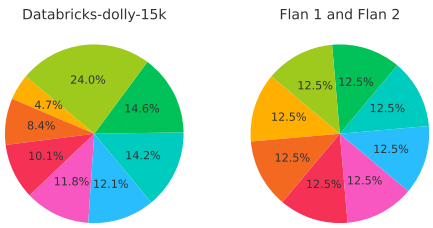

Figure 4: Data distribution of datasets

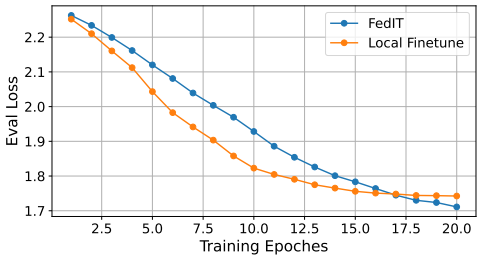

Figure 5: Training evaluation loss comparison between federated learning and local fine-tuning.

B.2 Sampling for Low-fidelity Validation Dataset

After clustering, we perform weighted probability sampling to obtain a low-fidelity validation dataset that best represents the overall data distribution. Let $\mathbf{V}_{\text{sampled}}$ denote the sampled validation dataset, and let α be a hyper-parameter representing the sampling ratio:

$$T = \lfloor \alpha \times |\mathbf{V}| \rfloor .$$

For each cluster, the number of samples to be drawn is proportional to the size of the cluster:

$$\mathbf{n}_c = \left\lfloor T \times \frac{\mathbf{N}_{\text{clusters}}[c]}{|\mathbf{V}|} \right\rfloor . \quad (1)$$

The selected points from each cluster form the final low-fidelity validation dataset $\mathbf{V}_{\text{sampled}}$.

C Data Distribution

We conducted our experiments on three datasets from the previous federal learning research: Databricks-dolly-15k (Zhang et al., 2024a), Flan 1 and Flan 2 (Yang et al., 2024). Each dataset has eight different NLP tasks, and their data distribution is shown in the Figure 4.

D Training Evaluation Loss Comparison

In FL, as shown in Figure 5, local fine-tuning may achieve faster initial convergence compared to federated training (FedIT (Zhang et al., 2024a)), but it often results in lower final accuracy. Since our method involves the aggregation of locally trained LoRA parameters and globally aggregated LoRA parameters, in order to avoid the local optima caused by the aggregation weight being too biased to the local parameters in the early stage of training, we employ a personalized slow start mechanism.

E Training Details

E.1 Dataset Splits

To simulate the scarcity of local data on the client (McMahan et al., 2017; Yang et al., 2024), for the Databricks-dolly-15k dataset, we extracted 20% of the data from each NLP task, ensuring its volume is comparable to the other two datasets. We set 20% of the Databricks-dolly-15k data as the local test set for each client (Zhang et al., 2024b), while for Flan datasets, we followed the original training and testing split. For our method, we extract 40 bars without retracting from the training data for each NLP task class to form a global validation set for our method. Our method will be trained using the segmented data and the other methods will be trained using the original data.

E.2 Classical FL Baselines: Additional Details

To aid general readers, we provide concise descriptions of all classical FL baselines compared in our main results.

- **FedAvg** (McMahan et al., 2017). Clients perform multiple local SGD steps and the server averages model parameters weighted by client data sizes. This sharply reduces communication while preserving global convergence in many practical settings.

- **FedAvgM** (Hsu et al., 2019). Augments FedAvg with *server-side momentum*, smoothing historical update directions and improving convergence stability under non-IID data.

- **FedAdagrad** (Reddi et al., 2020). Maintains each parameter's cumulative squared gradient on the server and scales step sizes with Adagrad, reducing manual tuning and often speeding convergence under heterogeneous data.

- **FedAdam** (Reddi et al., 2020). Incorporates Adam's first- and second-moment estimates at the server, dynamically adapting to gradient magnitude and direction for greater robustness to noisy or sparse gradients.
- **FedYogi** (Reddi et al., 2020). Replaces Adam's second-moment update with Yogi's sign-corrected rule, curbing unbounded moment growth and mitigating learning-rate blow-ups on non-IID data.
- **FedProx** (Li et al., 2020). Adds a proximal term to each client's objective and allows variable local epochs, constraining update drift and handling both statistical and system heterogeneity.

These baselines were designed in the pre-LLM era; their limitations relative to our method highlight the need for LLM-tailored personalized FL (pFL) algorithms. Other baselines specifically designed for applying LoRA in FL with LLMs (e.g., FedIT (Zhang et al., 2024a), PerFIT (Zhang et al., 2024b), FedDPA (Yang et al., 2024)) are discussed in detail in the related-work section (see Section 6).

E.3 Configurations

We used Alpaca-7B (Taori et al., 2023) as our base model, and in the hyperparameters of LoRA and how it was initialized, Optimizer Settings, template of the prompt and other model configurations are completely in accordance with the original FedIT setting (Zhang et al., 2024a), and we follow the original setting for different datasets in their FL research (Yang et al., 2024; Zhang et al., 2024a) in terms of learning rate. We set up 8 clients corresponding to 8 different task data and activated all clients per communication round based on the traditional pFL setup. All the experiments run on 2 × A5000 (24 GB).