CBP-Tuning: Efficient Local Customization for Black-box Large Language Models

Jiaxuan Zhao[1,2*], **Naibin Gu**[1,2*], **Yuchen Feng**[1,2], **Xiyu Liu**[1,2], **Peng Fu**[1,2†], **Zheng Lin**[1,2], **Weiping Wang**[1]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{zhaojiaxuan,gunaibin,fupeng}@iie.ac.cn

Abstract

The high costs of customizing large language models (LLMs) fundamentally limit their adaptability to user-specific needs. Consequently, LLMs are increasingly offered as cloud-based services, a paradigm that introduces critical limitations: providers struggle to support personalized customization at scale, while users face privacy risks when exposing sensitive data. To address this dual challenge, we propose Customized Black-box Prompt Tuning (CBP-Tuning), a novel framework that facilitates efficient local customization while preserving bidirectional privacy. Specifically, we design a two-stage framework: (1) a prompt generator trained on the server-side to capture domain-specific and task-agnostic capabilities, and (2) user-side gradient-free optimization that tailors soft prompts for individual tasks. This approach eliminates the need for users to access model weights or upload private data, requiring only a single customized vector per task while achieving effective adaptation. Furthermore, the evaluation of CBP-Tuning in the commonsense reasoning, medical and financial domain settings demonstrates superior performance compared to baselines, showcasing its advantages in task-agnostic processing and privacy preservation.

1 Introduction

Large language models (LLMs) have demonstrated extremely powerful performance in a wide range of tasks (Brown et al., 2020; Touvron et al., 2023; Yang et al., 2024). However, as these models become larger, the resources required for training and deployment become increasingly expensive (Gu et al., 2024; Feng et al., 2025), making it no longer feasible to fine-tune and deploy a separate model for each downstream task. This limits the ability of users to customize the model according to their specific needs. Parameter-efficient fine-tuning (PEFT) provides a promising solution by fine-tuning very few parameters while keeping most of the model parameters unchanged (Han et al., 2024; Si et al., 2024; Gu et al., 2025a; Yang et al., 2025). The PEFT technique includes adaption-based methods (Houlsby et al., 2019; He et al., 2021), re-parameterization-based methods (Hu et al., 2021; Gu et al., 2025b), and prompt-based methods (Lester et al., 2021; Li and Liang, 2021), which enables a base model to serve multiple users simultaneously.

Despite the efficiency and effectiveness of methods such as Adapter (Houlsby et al., 2019) and LoRA (Hu et al., 2021), these approaches necessitate the server to uphold multiple PEFT modules for downstream tasks. For each batch inference task, a subset of these modules must be selected and

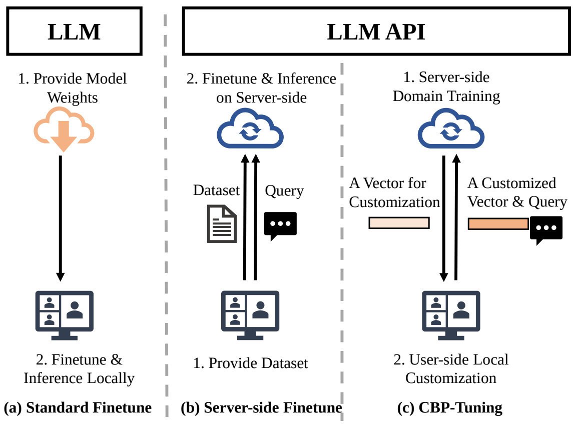
LLM
LLM API
1. Provide Model Weights
2. Finetune & Inference Locally
(a) Standard Finetune
2. Finetune & Inference on Server-side
Dataset
Query
1. Provide Dataset
(b) Server-side Finetune
1. Server-side Domain Training
A Vector for Customization
A Customized Vector & Query
2. User-side Local Customization
(c) CBP-Tuning

Figure 1: Schematic illustration of the comparison among three fine-tuning paradigms. (a) Left: Standard finetune transmits the model weights to the users. (b) Middle: Server-side finetune requires the users to provide the dataset for fine-tuning the model on the server side. (c) Right: Our proposed method, CBP-Tuning, conducts domain training on the server side and allows users to customize a vector locally without transferring data or accessing the model weights.

[*] Equal Contribution.
[†] Corresponding Author.

assembled (Wen and Chaudhuri, 2024), which is inconvenient in the multi-user scenario. In contrast, Prompt Tuning (Lester et al., 2021) is a simpler approach that only requires a certain length of soft prompts to be prepended at the input layer. Before batch processing, users can provide the learned soft prompts and inputs for a specific task to the server.

On the other hand, besides the cost of computation, LLMs also face the issue of privacy protection. Specifically, the server does not want users to access the full model weights, and users do not want to expose their data to the server. Most LLMs are released as services, and users can only access them through a black-box API. Black-Box Tuning (Sun et al., 2022b) describes this scenario as LMasS (Language Model as a Service), allowing users to optimize prompts using local black-box optimization methods. Despite their success, Black-box Tuning methods demonstrate limited versatility across tasks and LLMs (Zheng et al., 2024). In Offsite Tuning (Xiao et al., 2023), users can fine-tune the adapter for downstream data using an emulator sent from the server. In this setting, however, users still need to train and save the adapter parameters for each downstream task and upload them to the server for inference. This presents a challenge:

How to enable users to adapt the model without further retraining at the user side, while avoiding the need for server-side processing of task-specific parameters?

We aim to delegate the computationally intensive challenge of domain learning to server-side training, allowing users to focus only on optimizing customized tasks, thereby achieving efficient local customization. Therefore, we introduce a lightweight and flexible framework, CBP-Tuning (**C**ustomized **B**lack-box **P**rompt Tuning). As illustrated in Figure 1, the server-side conducts domain training and sends a vector for customization to users, where the user can then use a black-box optimization method to adapt this vector to a specific task. Specifically, the server-side trains a prompt generator which is a feed-forward layer with a bottleneck architecture, receiving both instance-specific and task-specific input. With a domain-specific prompt generator in place, users can then fine-tune it in a black-box manner. Following the setting of Black-box Tuning (Sun et al., 2022b), we project high-dimensional task-specific vectors into a smaller subspace. The user-side employs a black-box method to optimize the low-dimensional vector locally, thereby obtaining soft prompts tailored specifically for a given task. The user only needs to locally store a low-dimensional vector corresponding to each downstream task, without returning any additional parameters to the server-side.

Our contributions are summarized as follows:

- We propose a customized and lightweight fine-tuning framework called CBP-Tuning. It is a black-box optimization method based on prompt generators that achieves a balance between server-side and user-side, enabling efficient local customization.
- We design a prompt generator that takes the sum of task-specific vectors and instance-specific vectors as the input to the generator, combining both information.
- We conduct experiments using LLaMA-2-7B, Qwen-2.5-3B and LLaMA-2-13B to evaluate our method on general commonsense reasoning and domain-specific medical and financial tasks. The experimental results demonstrate that our approach outperforms baselines while offering lower costs and ensuring privacy.

2 Preliminary: Black-box Tuning

Black-Box Tuning (BBT) (Sun et al., 2022b) can optimize the continuous prompt prepended to the input via derivative-free optimization. Considering a black-box LLM that predicts a batch of inputs $\mathbf{X}$ and outputs $\mathbf{Y}$, prompt tuning involves training continuous prompts $\mathbf{p} \in \mathbb{R}^{l*d}$ to achieve better performance when the model is fed the optimal prompt vector $\mathbf{p}^*$ together with the input, where l is the length of the continuous prompt and d is the model dimension. The objective of prompt tuning can be formulated as:

$$\mathbf{p}^* = arg \min_{\mathbf{p} \in \mathcal{P}} \mathcal{L}(f(\mathbf{p}; \mathbf{X}), \mathbf{Y}), \tag{1}$$

where $f(\cdot)$ is the black-box LLM inference API, $\mathcal{L}(\cdot)$ is the loss function and $\mathcal{P}$ is a search space of prompts. In standard cases where a model can be accessed, the prompt $\mathbf{p}$ is optimized by gradient-based back-propagation.

BBT leverages the low intrinsic dimensionality of LLMs by optimizing in a reduced subspace $\mathbf{z} \in \mathbb{R}^r$ $(r \ll d)$ via random projection $\mathbf{A} \in \mathbb{R}^{d \times r}$:

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} \mathcal{L}(f(\mathbf{Az}; \mathbf{X}), \mathbf{Y}) \tag{2}$$

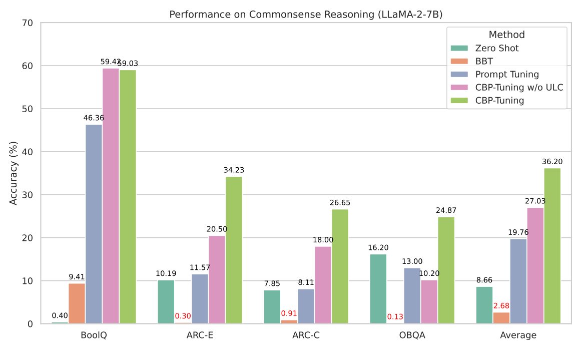

Figure 2: Results for LLaMA-2-7B model in the commonsense reasoning domain. The decrease in accuracy of BBT compared to the Zero Shot is highlighted in red.

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen et al., 2003) maintains a multivariate normal distribution $\mathcal{N}(\mathbf{m}^{(t)}, \sigma^{(t)}\mathbf{C}^{(t)})$ to sample candidate solutions $\mathbf{z}_i^{(t+1)}$. Through iterative population evaluation, the distribution parameters $(\mathbf{m}, \sigma, \mathbf{C})$ evolve toward high-performance regions.

While BBT initially appears suitable for our proposed scenario, enabling local private data training without requiring model weight access, our empirical analysis reveals critical limitations. As shown in Figure 2, direct application of BBT to inject task-specific soft prompts paradoxically degrades performance in commonsense reasoning tasks, achieving only gains on BoolQ (+9.01%) while causing significant average performance drops of 5.98 points across four datasets. This empirical evidence demonstrates that naively applying BBT for transitioning from foundation LLMs to user-customized models proves fundamentally inadequate in practical deployments. The absence of domain-specific and task-agnostic adaptation capabilities in pure BBT settings leads to unstable optimization trajectories and suboptimal task-specific performance. To bridge this gap, CBP-Tuning introduces a two-stage framework, specifically designed to overcome these architectural constraints, we introduce it in detail in the following.

3 Method

3.1 Overview of CBP-Tuning

As shown in Figure 3, the workflow of CBP-Tuning includes Server-side Domain Training and User-side Local Customization. As shown in Figure 3, on the server side, we train a prompt generator on the domain dataset. Users can use the CMA-ES algorithm to optimize the vector without accessing the model weights, thus achieving efficient local customization. Next, we introduce the core component and processes of CBP-Tuning, namely prompt generator, server-side domain training, and user-side local customization.

Algorithm 1 The CBP-Tuning Procedure

```
Input: Black-box LLM $f(\cdot)$, Prompt Generator $\mathcal{G}$,
    Domain dataset $\{\mathcal{X}_D; \mathcal{Y}_D\}$, Training epoches
    $E$, Customization dataset $\{\mathcal{X}_C; \mathcal{Y}_C\}$, Budget
    of API calls $B$, Population size $\lambda$.
 1: for $i = 1$ to $E$ do
 2:     for $(\mathbf{X}, \mathbf{Y}) \in \mathcal{X}_D \times \mathcal{Y}_D$ do
 3:         Get $\mathbf{p}$ using Eq. (4)
 4:         Concatenate input $\mathbf{X}'$ using Eq. (5)
 5:         Optimize $\mathcal{G}^*, \mathbf{z_0}$ using Eq. (6)
 6:     end for
 7: end for
 8: Set $max\_iteration = B/\lambda$ for CMA-ES
 9: for $j = 1$ to $max\_iteration$ do
10:     for $(X, Y) \in \mathcal{X}_C \times \mathcal{Y}_C$ do
11:         Get $\mathbf{p}^*$ using Eq. (7)
12:         Concatenate input $X^*$ using Eq. (8)
13:         Optimize $\mathbf{z}^*$ using Eq. (9)
14:     end for
15: end for
```

3.2 Prompt Generator

LLMs are typically trained on large-scale corpora and lack the ability to handle specific user tasks. To better achieve task customization on the user side, we propose a prompt generator $\mathcal{G}$ to learn prompts that activate the corresponding domain knowledge within the LLM.

The input $I(\mathbf{X}, \mathbf{z})$ of the prompt generator is the sum of two vectors, one is the input instruction and another is a vector for customization:

$$I(\mathbf{X}, \mathbf{z}) = Pooler(Emb(\mathbf{X})) + \mathbf{Az}. \qquad (3)$$

On the one hand, the inputs of the model pass through the embedding layer and obtain a hidden state $\mathbf{h} \in \mathbb{R}^{l \times d}$, where l is the length of the input tokens and d is the model dimension. Subsequently, the hidden state $\mathbf{h}$ is reduced to a d-dimensional vector by mean pooling, which is a part of the input to the prompt generator. On the other hand, another part of the prompt generator's input is a vector of the same size, following BBT, we optimize the vector $\mathbf{z} \in \mathbb{R}^r$ in a smaller subspace($r \ll d$),

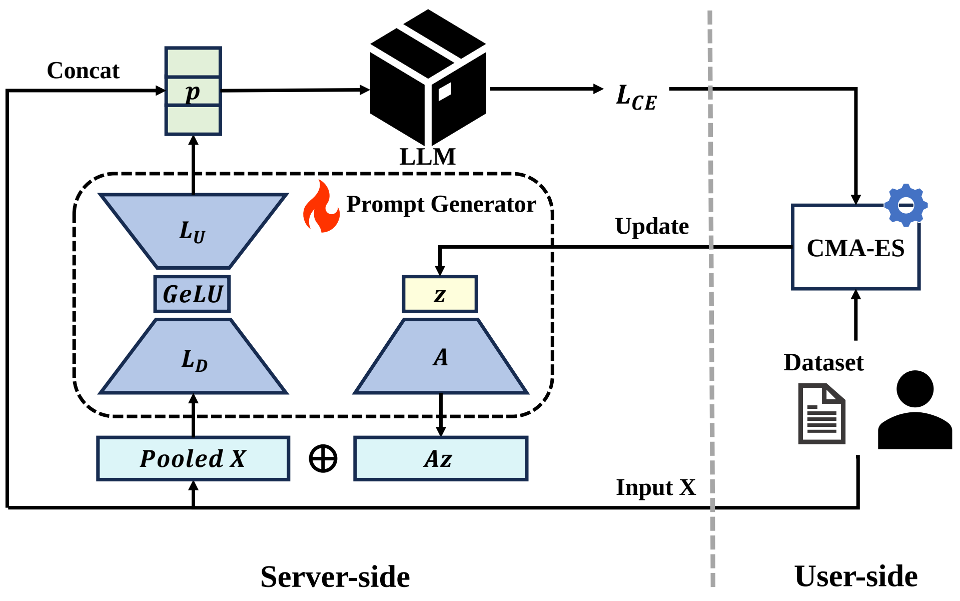

Figure 3: Illustration of Customized Black-box Prompt Tuning (CBP-Tuning). Users can perform efficient customization for each downstream task locally by using a gradient-free optimization algorithm (CMA-ES) to optimize each a low-dimensional vector $\mathbf{z}$, while the server trains a domain-specific prompt generator.

and project $\mathbf{z}$ back to the model dimension space through a projection matrix $A \in \mathbb{R}^{d \times r}$.

The prompt generator is a lightweight module that includes a down-projection layer $\mathbf{L_D} \in \mathbb{R}^{m \times d}$, an activation function $GeLU(\cdot)$, and an up-projection layer $\mathbf{L_U} \in \mathbb{R}^{(t*d) \times m}$, which ultimately outputs a soft prompt that is concatenated with the embedding of the model input, where t is the length of soft prompts. After receiving the input, the prompt generator produces the prompt:

$$\mathbf{p} = \mathbf{L_U}(GeLU(\mathbf{L_D}(I))). \tag{4}$$

Then, the prompt $\mathbf{p}$ will be prepended before the input layer, and the final input $\mathbf{X}^{'}$ is:

$$\mathbf{X}^{'} = Concat(\mathbf{p}; \mathbf{X}). \tag{5}$$

It is important to note that our approach manipulates prompts at the input stage and concatenates them into the input layer. As a result, the prompt generator is called only once during the generation stage, unlike methods such as LoRA, which require loading modules at each layer. For users sharing the same domain, the prompt generator can be shared, supporting efficient batch processing.

3.3 Server-side Domain Training

To better activate the model's domain knowledge without accessing user data, we use domain data on the server side to train the prompt generator by gradient descent, allowing multiple users sharing the same domain to leverage the same prompt generator. We keep the model $f(\cdot)$ frozen and train the prompt generator $\mathcal{G}$ on the dataset from the collected domain. During the server-side domain training, our objective is to optimize:

$$\mathcal{G}^{*}, \mathbf{z_0} = arg\min_{\mathcal{G} \in \mathcal{G}_\theta, \mathbf{z} \in \mathcal{Z}} \mathcal{L}_{CE}(\mathbf{Y}, f(\mathbf{X}^{'})), \tag{6}$$

where the input and output $\mathbf{X} \in \mathcal{X}_D, \mathbf{Y} \in \mathcal{Y}_D$ in Customization dataset $\{\mathcal{X}_D; \mathcal{Y}_D\}$, $\mathcal{G}_\theta$ is the parameter of the prompt generator, $\mathcal{Z}$ is the search space for $\mathbf{z}$, $\mathcal{L}_{CE}$ is the cross-entropy loss function. By conducting server-side domain training, we obtain a parameter generator $\mathcal{G}^{*}$ for a specific domain, along with an initialized low-dimensional vector $\mathbf{z}$ in this domain for subsequent user-side customization.

3.4 User-side Local Customization

At the user side, we aim to customize for the user's tasks, but considering that most users lack the computational resources required for gradient descent typically needed for training, we intend to achieve this in a low-cost manner by black-box tuning. We employ CMA-ES (Hansen et al., 2003) to optimize $\mathbf{z_0}$ and obtain $\mathbf{z}^{*}$ in order to generate task-specific enhanced prompts for improved performance. Formally,

$$\mathbf{p}^* = \mathcal{G}^*(\mathbf{X}, \mathbf{z_0}), \quad (7)$$

$$\mathbf{X}^* = Concat(\mathbf{p}^*; \mathbf{X}), \quad (8)$$

$$\mathbf{z}^* = arg\min_{\mathbf{z}\in\mathcal{Z}} \mathcal{L}_{CE}(\mathbf{Y}, f(\mathbf{X}^*)), \quad (9)$$

where the input and output are $\mathbf{X} \in \mathcal{X}_C$ and $\mathbf{Y} \in \mathcal{Y}_C$ respectively, and $\{\mathcal{X}_C; \mathcal{Y}_C\}$ is the customization dataset.

Similarly, the derivative-free optimization algorithm is also guided by the cross-entropy loss $\mathcal{L}_{CE}$ to optimize the direction. During the local customization stage, users cannot access the model weight, and the model and parameter generator are kept fixed.

So far, users simply require the server to transmit a domain-specific prompt generator to implement local customization for each downstream task. On the server side, we train $\mathbf{L_U}, \mathbf{L_D}, \mathbf{A}, \mathbf{z}$, where the total number of parameters is $m \times d + m \times l \times d + r \times d + r$ required to be stored and sent, while the user side only needs to save $s \times r$ parameters, where s is the number of downstream tasks and r is a low dimension of the vector $\mathbf{z}$.

4 Experiments

4.1 Setup

Models and Datasets. We implement experiments on LLaMA-2-7B, LLaMA-2-13B (Touvron et al., 2023) and Qwen-2.5-3B (Yang et al., 2024) models.

To simulate the user's private data, we perform a complete separation between the domain dataset used for the Server-side Domain Training (SDT) stage and the customization dataset used for the User-side Local Customization (ULC) stage within the same domain. Comprehensive evaluations are conducted on both a relatively general commonsense reasoning domain and a more specialized medical domain.

For the commonsense reasoning domain, the domain dataset in the SDT stage is composed of training sets from PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), and WinoGrande (Sakaguchi et al., 2021). The customization dataset in the ULC stage includes BoolQ (Clark et al., 2019), the challenge set ARC-Challenge and easy set ARC-Easy of ARC (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). For each task in the customization dataset, we randomly select 16 shots from its training set.

For the medical domain, we construct a domain dataset by combining a subset of approximately 6,000 samples from a biomedical instruction-following dataset Medical Meadow (Han et al., 2023), and the first 100,000 samples from Medical Multiple-Choice Question Answering (MedMCQA) (Pal et al., 2022). This domain dataset is used for the SDT stage to enable LLMs to learn how to handle complex medical instructions while also answering medical-related questions. For the customization dataset, we use 6 subsets of MMLU (Hendrycks et al., 2020) that are most relevant to medical knowledge, specifically anatomy, clinical knowledge, college medicine, medical genetics, college biology, and professional medicine. In the ULC phase, we utilize the development sets of these subsets and report their average performance.

For the financial domain, we selected a mixture of the first 10k samples from the sentiment training set in FinGPT's (Yang et al., 2023) instruction-tuning dataset and the first 1k samples from the FiQA_QA (Cheng et al., 2024) dataset as the training set for the SDT stage. For testing, we employed three financial sentiment analysis datasets: Twitter Financial News Sentiment, FiQA_SA (Cheng et al., 2024), and a subset of the Financial Phrase Bank (Malo et al., 2014) containing only samples with 100% annotator confidence. In the ULC stage, we randomly selected 16 samples for optimization.

We conduct experiments under three different random seeds in the three domains. Our evaluation metrics follow the setup of LLM-Adapters (Hu et al., 2023). The detailed setup is provided in Appendix A.

Baselines. We compare our proposed method with the following four baselines. (1) **Zero Shot:** We directly use the alpaca-style (Taori et al., 2023) prompt template to test the model on the given tasks. (2) **Prompt Tuning:** Following (Lester et al., 2021), we freeze the model and only train the prompts prepended to the input embedding. (3) **CBP-Tuning w/o ULC:** We train on a mixed domain dataset, i.e., server-side domain training for CBP-Tuning without further user local customization, then we can implement the complete CBP-Tuning process based on this. (4) **CBP-Tuning:** We use the CMA-ES algorithm to optimize without accessing the model. In the CMA-ES optimization process, we input a batch of training data together and optimize using the average cross-entropy loss for the answer output part. To ensure a fair com-

Model	Setting	Medical Domain							Financial Domain			
		CK	MG	Anatomy	PM	CB	CM	Med Avg	FIQA_SA	TFNS	FPB	Fin Avg
LLaMA-2-7B	Zero Shot	2.26	5.00	0.74	0.00	0.00	1.16	1.53	23.93	13.23	3.75	13.64
	Prompt Tuning	19.62	20.00	11.85	19.12	20.14	16.76	17.92	22.65	11.81	10.51	14.99
	CBP-Tuning w/o ULC	28.30	26.00	25.19	11.76	29.17	**26.59**	24.50	59.83	25.42	43.02	42.76
	CBP-Tuning	**34.21**	**37.00**	**29.63**	**21.20**	**29.40**	21.97	**28.90**	**63.25**	**34.17**	**45.32**	**47.58**
Qwen-2.5-3B	Zero Shot	39.25	49.00	49.63	20.59	36.81	39.31	39.10	66.67	38.65	77.96	61.09
	Prompt Tuning	65.66	71.00	54.81	37.87	67.36	60.12	59.47	58.12	69.26	**83.97**	70.45
	CBP-Tuning w/o ULC	67.92	**74.00**	**59.26**	66.54	**74.31**	61.85	67.31	62.39	69.85	72.44	68.53
	CBP-Tuning	**68.55**	**74.00**	**59.26**	**69.49**	**74.31**	**63.39**	**68.17**	**72.22**	**70.45**	77.80	**73.49**
LLaMA-2-13B	Zero Shot	17.36	13.00	13.33	0.74	15.28	12.14	11.97	68.38	22.82	27.69	39.63
	Prompt Tuning	18.87	12.00	9.63	23.16	14.58	23.12	16.89	44.44	**65.49**	65.81	58.58
	CBP-Tuning w/o ULC	61.13	46.00	34.07	47.06	55.56	**48.55**	48.73	65.38	53.69	63.83	60.96
	CBP-Tuning	**61.51**	**46.00**	**42.96**	**48.28**	**56.25**	47.40	**50.40**	**75.64**	58.04	**71.73**	**68.47**

Table 1: Performance (%) of LLaMA-2-7B, Qwen-2.5-3B, and LLaMA-2-13B models across medical and financial domains. ‘Med Avg‘ and ‘Fin Avg‘ indicate average scores over each domain’s subtasks.

parison, Prompt Tuning, CBP-Tuning, and its w/o ULC baseline are fine-tuned using the same domain dataset within the same domain. The prompt template is presented in Appendix A.1. (5) **LoRA:** A widely-used PEFT method (Hu et al., 2021) that freezes pre-trained model weights and injects trainable, low-rank matrices into the Transformer layers for efficient adaptation. (6) **P-Tuning:** A prompt-based method (Liu et al., 2021b) that keeps the language model frozen and optimizes continuous prompt embeddings via a small prompt encoder. The detailed experimental comparison with these additional baselines is provided in Appendix B.3.

Implementation. For Prompt Tuning, we evaluate each task on the customization dataset after training on the domain dataset. For the SDT stage of CBP-Tuning and its variants, the intermediate dimension m of the prompt generator is set to 256, and the subspace dimension r of vector $\mathbf{z}$ is set to 500. All the above methods keep the model weights frozen during training, and the length of the soft prompt concatenated before the user input embedding is 10. For the ULC stage of CBP-Tuning, the customization dataset size, budget of black-box LLM calls B, population size λ, and σ are set to $\{16, 300, 30, 0.01\}$ in the commonsense reasoning domain, respectively. The hyperparameters during training and the optimization parameters for other domains in the ULC phase are detailed in the Appendix A.2.

4.2 Main Results

We demonstrate the results of baselines on four commonsense reasoning datasets based on the LLaMA-2-7B model and six medical datasets based on all models in Figure 2 and Table 1 respectively.

Comparison on Zero Shot. Across all domains, the untrained zero-shot approach demonstrates the weakest performance among all baselines. Notably, its performance is significantly poorer in the medical and financial domains which require domain-specific knowledge compared to the more general commonsense reasoning domain. While trained methods including Prompt Tuning, CBP-Tuning, and its w/o ULC variant all achieve substantial performance improvements, we observe slight performance degradation for both Prompt Tuning and CBP-Tuning w/o ULC on the OpenBookQA task. This discrepancy may be attributed to potential conflicts between the training data used in the SDT phase and the OpenBookQA task. Importantly, the CBP-Tuning framework achieves comprehensive performance gains across all evaluated datasets.

Comparison on Prompt Tuning. Our experiments reveal that CBP-Tuning and its w/o ULC variant substantially outperform Prompt Tuning, with CBP-Tuning achieving average performance advantages of 16.44 points over Prompt Tuning in commonsense reasoning, and 32.59 points in the financial domain using LLaMA-2-7B. For LLaMA-2-13B, CBP-Tuning boosts the Med Avg from 16.89 (Prompt Tuning) to 50.40 and the Fin Avg from 58.58 to 68.47, showcasing significant gains. Importantly, the full CBP-Tuning framework demonstrates comprehensive performance gains over the Prompt Tuning baseline across all benchmarks. This evidences that through its two-stage optimization, CBP-Tuning not only addresses privacy preservation requirements in client-server deployments but also delivers superior performance compared to directly serving server-side fine-tuned Prompt Tuning models to end-users.

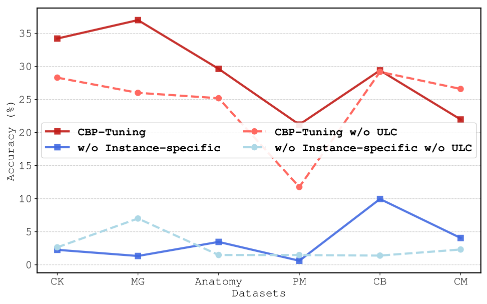

Figure 4: Results for ablation study of instance-specific information in the medical domain using LLaMA-2-7B.

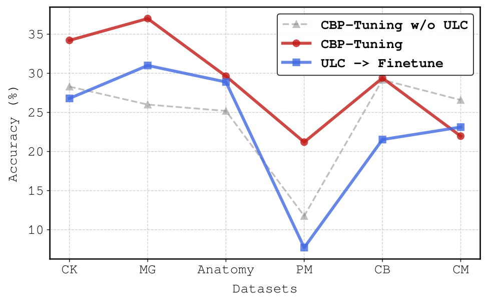

Figure 5: Results for optimization methods analysis in the medical domain using LLaMA-2-7B.

Comparison on w/o User-side Local Customization. Our comparative analysis of CBP-Tuning and its w/o ULC variant illustrates that the ULC phase generally delivers substantial performance improvements, yielding average gains of 9.17 points in the commonsense reasoning domain, 4.40, 0.86 and 1.67 points in the medical domain using LLaMA-2-7B, Qwen-2.5-3B and LLaMA-2-13B respectively. In the financial domain, the ULC phase provides average gains of 4.82 points for LLaMA-2-7B, 4.96 points for Qwen-2.5-3B, and 7.51 points for LLaMA-2-13B. However, we observe localized performance degradation on BoolQ and MMLU Clinical Medicine (CM) tasks with LLaMA-2-7B model. The BoolQ decline may stem from its unique yes/no answer format diverging from other tasks' option format, while MMLU CM's challenges originate from its benchmark characteristics: abbreviated development sets (used for the ULC) coupled with test set questions exceeding 480 words in 8.1% cases. Furthermore, as shown in the Appendix B.1, we examine the final loss of the LLaMA-2-7B model during the ULC phase in the medical domain and found that the loss for the poorly performing CM task was 1–3 orders of magnitude larger than that of other tasks. Crucially, the ULC stage demonstrates task-agnostic effectiveness in boosting model performance, achieving these gains through gradient-free local optimization that preserves privacy constraints.

4.3 Ablation Study of Instance-specific Information

To verify the effectiveness of introducing instance-specific information into the prompt generator, we set up an ablation version without instance-specific information. We compare the setting without mean pooled instance embedding in the medical domain, and all other settings are the same as CBP-Tuning. We can see that the performance of the w/o Instance-specific setting and its w/o ULC variant (represented by the blue solid and dashed lines in Figure 4, respectively) is significantly weaker than the standard CBP-Tuning and its w/o ULC variant (represented by the red solid and dashed lines in Figure 4, respectively). Specifically, the average performance across the six datasets in the medical domain shows that the w/o Instance-specific version drops by 25.29 points, and its w/o ULC variant drops by 21.28 points.

4.4 Different Optimization Methods Analysis

As previously mentioned, implementing privacy protection for both the server and user sides in our framework requires the execution of ULC in the second stage. To investigate the impact of replacing the user local customization phase with fine-tuning, we performed a few-shot fine-tuning version and explored the effects of further fine-tuning using the customization dataset (which, in the medical domain, corresponds to the development set of each task). As shown in the Figure 5, the blue line, red line, and gray dashed line represent the few-shot fine-tuning setting, the standard CBP-Tuning, and its w/o ULC variant, respectively. It is evident that using ULC in the second stage, i.e., gradient-free optimization, consistently outperforms gradient-based fine-tuning. CBP-Tuning performs slightly worse than its few-shot fine-tuning variant only on the CM dataset, with an average improvement of 5.72 points across six medical datasets. In contrast, the few-shot fine-tuning setting shows performance improvements over CBP-Tuning w/o ULC only on the MG and Anatomy datasets, while it leads to a performance decline on the other datasets. We suspect two potential reasons: (1) While gradient-based optimization tends to overfit the training data, CMA-ES is inclined to discover superior solu-

Method	OBQA	BoolQ	ARC-E	ARC-C
CBP-Tuning	85s	54s	101s	109s
Prompt Tuning	*> 10 mins on each dataset*			

Table 2: Comparison of user-side customization time for CBP-Tuning versus full fine-tuning time for Prompt Tuning in the commonsense reasoning domain. All timings were measured on a single NVIDIA A100 40GB GPU using LLaMA-2-7B.

Method	OBQA	BoolQ	ARC-E	ARC-C
CBP-Tuning	16854MB	15668MB	17460MB	17864MB
Prompt Tuning	28572MB	21482MB	31712MB	33544MB

Table 3: GPU memory usage in the commonsense reasoning domain on LLaMA-2-7B.

tions owing to its exploration mechanism. (2) The server-side training instills a structured instruction-following format in the model. However, certain downstream tasks may have unique characteristics that create a mismatch with this pre-trained format. This discrepancy makes the optimization landscape for the vector $\mathbf{z}$ particularly challenging. For tasks where this gap is significant (e.g., MMLU Clinical Medicine), the CMA-ES algorithm may fail to converge to an effective solution, a difficulty empirically evidenced by the high final loss values for such tasks, as detailed in Appendix 8.

4.5 Efficiency Analysis

In Table 2, we show the training time of user-side local customization on four commonsense reasoning datasets using LLaMA-2-7B model. On a single NVIDIA A100 40GB GPU, the optimization can be completed in just 93.5 seconds across these datasets, with the shortest time recorded at 54 seconds on the BoolQ dataset. In contrast, on average, fine-tuning through Prompt Tuning typically demands more than ten minutes. In commonsense reasoning domain's experimental setup, the user-side optimization only requires calling the API 300 times and performing 10 iterations of the CMA-ES algorithm. As shown in Table 3, when using a batch size of 16, our method demonstrates significantly lower GPU memory consumption compared to Prompt tuning. Notably, since CMA-ES optimization does not rely on gradients, the entire optimization process can be performed on the CPU. Overall, our proposed approach enables users to efficiently and locally customize the model in an exceedingly short duration.

5 Related Work

5.1 Prompt-based Learning

Prompt-based learning is a type of method that inserts soft prompts into the input or hidden states of the model. These soft prompts are highly flexible and adaptable during fine-tuning, as they can be optimized in continuous spaces to fit specific tasks. Methods like Prompt Tuning (Lester et al., 2021) and P-tuning (Liu et al., 2021b) incorporate soft prompts into the input layer of the model. Other approaches, such as DePT (Shi and Lipani, 2023), Prefix Tuning (Li and Liang, 2021), and P-tuning v2 (Liu et al., 2021a), add trainable prompts to the keys and values matrices across all layers. These methods focus primarily on task-specific prompt tuning. Additionally, there are methods that integrate instance-specific information. For example, IDPG (Wu et al., 2022) uses a parameterized hypercomplex multiplication prompt generator to produce soft prompts tailored for each instance. LPT (Liu et al., 2022) inserts instance-aware prompts into an intermediate layer of the pre-trained model, rather than the input layer or all layers. We observe that these two types of methods for improving prompt tuning performance decouple task-specific information from instance-specific information and reserve a task vector suitable for local customization at the input of the prompt generator for two-stage optimization.

5.2 Black-box Optimization for LLMs

Black-Box Tuning (Sun et al., 2022b) innovatively uses a derivative-free optimization method to learn prompts in the new scenario called Language-Model-as-a-Service (LMaaS). BBTv2 (Sun et al., 2022a), as an improved version, prepends continuous prompts to every layer of the model and optimizes the prompts at different layers alternately. InstructZero (Chen et al., 2023) employs Bayesian optimization to learn the continuous prompts inserted into open-source LLMs to generate discrete instructions for the black-box LLMs. In addition to using gradient-free optimization algorithms to optimize soft prompts, there are also some recent studies showing that it is possible to optimize black-box large models using white-box models with varying degrees of transparency. Offsite Tuning (Xiao et al., 2023) allows users to adapt to downstream tasks with the help of a lightweight adapter and a compressed emulator without accessing the full model. BBox-Adapter (Sun et al., 2024) adapts black-box

LLMs to specific tasks without accessing internal parameters or output probabilities by training a small adapter model with ranking-based NCE loss and online adaptation. Proxy-tuning (Liu et al., 2024) tunes a smaller LM, applying the difference between the predictions of the small tuned and untuned LMs to shift the direction of tuning for the larger untuned model at the cost of exposing logits. Based on the observation that directly applying black-box optimization to LLMs for certain tasks often results in insufficient performance (Zheng et al., 2024), we only need access and modification after the embedding layer to locally customize black-box LLMs driven by user needs.

6 Conclusion

We present CBP-Tuning, a lightweight and customized fine-tuning framework for black-box large language models. Our approach enables efficient local customization through a black-box optimization method that balances the burden between the server and the user. CBP-Tuning allows users to achieve cost-effective customization for various downstream tasks without compromising privacy. We anticipate that future research can extend CBP-Tuning to a broader range of domains and models.

Limitations

While CBP-Tuning demonstrates promising privacy-preserving capabilities and competitive performance across both target domains, future efforts could focus on two aspects. First, our evaluation was limited to models up to the 13B parameter scale; its efficacy on significantly larger models (e.g., 70B+) remains unexplored due to computational constraints. Second, the user-side customization stage led to performance degradation on specific tasks like BoolQ and MMLU Clinical Medicine. Future efforts could explore adaptive optimization strategies or methods to enhance the robustness of ULC across a more diverse range of task structures.

Ethics Statement

Our work introduces CBP-Tuning, a framework designed for efficient and privacy-preserving local customization of large language models. All datasets used in our study for training and evaluation are publicly available and sourced from established academic benchmarks. No proprietary or sensitive user data was involved in our experiments. Our method is purely algorithmic and is not designed to inherently create or amplify harmful social biases.

Acknowledgement

We thank the anonymous reviewers for their insightful feedback, which greatly improves our paper. This work is supported by the National Natural Science Foundation of China (No. 62472419, 62472420).

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. 2023. Instructzero: Efficient instruction optimization for black-box large language models. *arXiv preprint arXiv:2306.03082*.

Daixuan Cheng, Shaohan Huang, and Furu Wei. 2024. Adapting large language models via reading comprehension. In *ICLR*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Yuchen Feng, Bowen Shen, Naibin Gu, Jiaxuan Zhao, Peng Fu, Zheng Lin, and Weiping Wang. 2025. Dive into moe: Diversity-enhanced reconstruction of large language models from dense into mixture-of-experts. *Preprint*, arXiv:2506.09351.

Naibin Gu, Peng Fu, Xiyu Liu, Ke Ma, Zheng Lin, and Weiping Wang. 2025a. Adapt once, thrive with updates: Transferable parameter-efficient fine-tuning on evolving base models. *Preprint*, arXiv:2506.06844.

Naibin Gu, Peng Fu, Xiyu Liu, Bowen Shen, Zheng Lin, and Weiping Wang. 2024. Light-PEFT: Lightening parameter-efficient fine-tuning via early pruning.

In *Findings of the Association for Computational Linguistics ACL 2024*, pages 7528–7541, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Naibin Gu, Zhenyu Zhang, Xiyu Liu, Peng Fu, Zheng Lin, Shuohuan Wang, Yu Sun, Hua Wu, Weiping Wang, and Haifeng Wang. 2025b. Beamlora: Beam-constraint low-rank adaptation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 11871–11883. Association for Computational Linguistics.

Ashwin Kumar Gururajan, Enrique Lopez-Cuena, Jordi Bayarri-Planas, Adrian Tormos, Daniel Hinjos, Pablo Bernabeu-Perez, Anna Arias-Duart, Pablo Agustin Martin-Torres, Lucia Urcelay-Ganzabal, Marta Gonzalez-Mallo, et al. 2024. Aloe: A family of fine-tuned open healthcare llms. *arXiv preprint arXiv:2405.01886*.

Tianyu Han, Lisa C Adams, Jens-Michalis Papaioannou, Paul Grundmann, Tom Oberhauser, Alexander Löser, Daniel Truhn, and Keno K Bressem. 2023. Medalpaca–an open-source collection of medical conversational ai models and training data. *arXiv preprint arXiv:2304.08247*.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *CoRR*, abs/2110.04366.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

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*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5254–5276.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Xiangyang Liu, Tianxiang Sun, Xuan-Jing Huang, and Xipeng Qiu. 2022. Late prompt tuning: A late prompt could be better than many prompts. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1325–1338.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021a. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473.

Zhengxiang Shi and Aldo Lipani. 2023. Dept: Decomposed prompt tuning for parameter-efficient fine-tuning. *arXiv preprint arXiv:2309.05173*.

Chongjie Si, Xiaokang Yang, and Wei Shen. 2024. See further for parameter efficient fine-tuning by standing on the shoulders of decomposition. *CoRR*, abs/2407.05417.

Haotian Sun, Yuchen Zhuang, Wei Wei, Chao Zhang, and Bo Dai. 2024. Bbox-adapter: Lightweight adapting for black-box large language models. In *International Conference on Machine Learning*, pages 47280–47304. PMLR.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuan-Jing Huang, and Xipeng Qiu. 2022a. Bbtv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3930.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022b. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR.

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. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Yeming Wen and Swarat Chaudhuri. 2024. Batched low-rank adaptation of foundation models. *ICLR 2024*.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, VG Vinod Vydiswaran, and Hao Ma. 2022. Idpg: An instance-dependent prompt generation method. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5507–5521.

Guangxuan Xiao, Ji Lin, and Song Han. 2023. Offsite-tuning: Transfer learning without full model. *arXiv preprint arXiv:2302.04870*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Chenxu Yang, Ruipeng Jia, Naibin Gu, Zheng Lin, Siyuan Chen, Chao Pang, Weichong Yin, Yu Sun, Hua Wu, and Weiping Wang. 2025. Orthogonal fine-tuning for direct preference optimization. *Preprint*, arXiv:2409.14836.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. Fingpt: Open-source financial large language models. *FinLLM at IJCAI*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Yuanhang Zheng, Zhixing Tan, Peng Li, and Yang Liu. 2024. Black-box prompt tuning with subspace learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*.

A Training Details

A.1 Prompt Template

Our prompt template design strictly adheres to Alpaca (Taori et al., 2023), uniformly applied across all datasets through the following structured template:

> *Below is an instruction that describes a task. Write a response that appropriately completes the request.*
> *### Instruction: {instruction}*
> *### Input: {input}*
> *### Response:*

For the setup of the instruction, in the commonsense reasoning domain, we utilize datasets processed by LLM-Adapters (Hu et al., 2023); in the medical domain, we follow the template from Aloe (Gururajan et al., 2024); in the financial domain, we follow the settings of FinGPT (Yang et al., 2023). The Alpaca-style prompt templates to be filled for the three domains are shown in Table 4, where the {subject_name} is the task name of the MMLU subset. For the commonsense reasoning and medical domains, the output format for evaluation and training is 'the correct answer is [ANSWER]' for the LLaMA-2-7B and LLaMA-2-13B, and '[ANSWER]' for the Qwen-2.5-3B. For the financial domain, the output format for evaluation and training is '[ANSWER]' for all models. It is important to note that only in the financial domain do we need to fill both the instruction and the input; in the other two domains, only the instruction is required.

A.2 Hyperparameters

The hyperparameters for Prompt Tuning and CBP-Tuning SDT stage in the three domains are shown in the Table 6.

The hyperparameters for CBP-Tuning ULC stage in the three domains are shown in the Table 5.

A.3 Hardware

We use 2 NVIDIA A100 40G GPUs to conduct our experiments, and the version of transformers library is 4.41.2, while the version of PEFT is 0.3.0. For SDT in the commonsense reasoning domain with LLaMA-2-7B, approximately 12 GPU hours are required.

A.4 Environment of Experiments

The implementation of CBP-Tuning is based on Transformers library, peft library and LLM-Adapters. The data processing and evaluation for two domains also follow LLM-Adapters.

B Additional Results

B.1 Decreased Performance Analysis

The table 8 shows the final loss values to which each dataset converged during the user-side local customization (ULC) stage using the CMA-ES algorithm (lower values indicate more complete optimization). Notably, the final loss for MMLU Clinical Medicine (CM) is 1.361×10^{-2}, three orders of magnitude higher than that for Medical Genetics (MG), which is 2.972×10^{-5}. This indicates that the optimization process for the CM task was much harder to converge, consistent with the performance drop we observed for this task in the main text.

B.2 Bottleneck Dimension Ablation Study

In Table 7 we compare three bottleneck dimensions ($r = 100, 500, 1000$) both without and with the ULC phase using LLaMA-2-7B in the medical domain. The results peak at $r = 500$, where CBP-Tuning achieves an average accuracy of 68.55%, significantly higher than the 58.97% at $r = 100$ and only 15.59% at $r = 1000$. This confirms that 500 provides the best trade-off between representational capacity and optimization stability.

B.3 Performance Comparison with LoRA and P-Tuning

We conducted additional experiments comparing CBP-Tuning with other prominent parameter-efficient fine-tuning (PEFT) methods: LoRA (Hu et al., 2021) and P-Tuning (Liu et al., 2021b). The comparison was performed on the Qwen-2.5-3B model across the medical and financial domains.

As shown in the table 9, CBP-Tuning demonstrates superior average performance compared to both LoRA and P-Tuning. In the medical domain, CBP-Tuning achieves performance improvements of 5.69 points and 2.81 points over LoRA and P-Tuning in average scores, respectively. In the financial domain, the improvements are 1.15 points and 16.82 points, respectively, further validating the effectiveness of our approach.

Furthermore, it is important to note the architectural differences. CBP-Tuning is designed based on Prompt Tuning, as both methods modify only the token embeddings at the input layer. In contrast, LoRA requires adapting and merging low-

Domain	Dataset	Instruction	Input
CR	BoolQ	Please answer the following question with true or false, question: [QUESTION] Answer format: true/false	None
CR	Others	Please choose the correct answer to the question: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer4: [ANSWER_4] Answer format: answer1/answer2/answer3/answer4	None
Medical	All Datasets	The following are multiple choice questions about {subject_name}. Output a single option from the options as the final answer. QUESTION: [QUESTION] Answer1: [ANSWER_1] Answer2: [ANSWER_2] Answer3: [ANSWER_3] Answer4: [ANSWER_4] Answer format: answer1/answer2/answer3/answer4	None
Financial	All Datasets	What is the sentiment of this news? Please choose an answer from {negative/neutral/positive}.	Question Text

Table 4: Prompt Template Configuration for Different Domains. Note that in the Commonsense Reasoning (CR) domain, BoolQ has a distinct instruction template, while Arc-e, Arc-c, and OBQA share a common instruction template.

Domain	Model	Data Size	Budget B	Population λ	Step Size σ
Commonsense	LLaMA-2-7B	16	300	30	0.01
Medical	LLaMA-2-7B	5	1500	30	0.05
	Qwen-2.5-3B	5	300	30	0.01
	LLaMA-2-13B	5	1500	30	0.01
Financial	LLaMA-2-7B	16	300	30	0.05
	Qwen-2.5-3B	16	300	30	0.01
	LLaMA-2-13B	16	300	30	0.05

Table 5: Hyperparameter settings for the ULC stage across domains and models.

Domain	Setting	Learning Rate	Batch Size	Epochs	Cutoff Length
Commonsense	Prompt Tuning	3e-2	16	5	512
	CBP-Tuning	3e-4	16	5	512
Medical	Prompt Tuning	3e-2	16	5	256
	CBP-Tuning	3e-4	16	5	256
Financial	Prompt Tuning	1e-2	16	2	512
	CBP-Tuning	1e-4	16	2	512

Table 6: Hyperparameter settings for Prompt Tuning and CBP-Tuning across different domains.

rank matrices within each layer of the model. This makes the comparison between CBP-Tuning and Prompt Tuning particularly direct, while highlighting CBP-Tuning's potential for higher efficiency in deployment scenarios where modifying internal model weights is more complex or costly.

Method	CK	MG	Anatomy	PM	CB	CM	Avg.
CBP-Tuning (dim=100) w/o ULC	53.21	52.00	54.07	58.82	67.36	49.13	55.77
CBP-Tuning (dim=100, σ=0.01, B=300)	57.86	56.50	58.52	60.05	69.44	51.45	58.97
CBP-Tuning (dim=1000) w/o ULC	4.15	17.00	4.44	9.19	1.39	6.36	7.09
CBP-Tuning (dim=1000, σ=0.05, B=300)	4.03	15.33	3.46	26.72	21.30	22.74	15.59
CBP-Tuning (dim=500) w/o ULC (ours)	74.00	59.26	66.54	74.31	61.85	67.31	67.92
CBP-Tuning (dim=500, σ=0.01, B=300) (ours)	74.00	59.26	69.49	74.31	63.39	68.17	68.55

Table 7: Ablation on bottleneck dimension r for Qwen-2.5-3B in the medical domain.

	CK	MG	Anatomy	PM	CB	CM
Loss	2.078×10^{-4}	2.972×10^{-5}	3.496×10^{-4}	5.906×10^{-3}	2.873×10^{-3}	1.361×10^{-2}

Table 8: Final CMA-ES loss values at the end of the ULC stage for each medical dataset using LLaMA-2-7B.

Model	Setting	Medical Domain							Financial Domain			
		CK	MG	Anatomy	PM	CB	CM	Med Avg	FIQA_SA	TFNS	FPB	Fin Avg
Qwen-2.5-3B	Zero Shot	39.25	49.00	49.63	20.59	36.81	39.31	39.10	66.67	38.65	77.96	61.09
	Prompt Tuning	65.66	71.00	54.81	37.87	67.36	60.12	59.47	58.12	69.26	**83.97**	70.45
	CBP-Tuning w/o ULC	67.92	**74.00**	59.26	66.54	**74.31**	61.85	67.31	62.39	69.85	72.44	68.53
	CBP-Tuning	**68.55**	**74.00**	59.26	**69.49**	**74.31**	**63.39**	**68.17**	72.22	70.45	77.80	**73.49**
	LoRA	64.15	73.00	58.52	59.93	66.67	52.60	62.48	**79.49**	**72.91**	64.62	72.34
	P-Tuning	69.06	71.00	**60.00**	57.35	72.92	61.85	65.36	24.36	69.72	75.93	56.67

Table 9: Performance (%) of Qwen-2.5-3B compared with additional baselines (LoRA and P-Tuning) across medical and financial domains. **Bold** indicates the best performance in each column among all methods.