

# EFFICIENT HYPER-PARAMETER SEARCH FOR LoRA VIA LANGUAGE-AIDED BAYESIAN OPTIMIZATION

005 **Anonymous authors**

006 Paper under double-blind review

## ABSTRACT

011 Fine-tuning Large Language Models (LLMs) with Low-Rank Adaptation (LoRA)  
 012 enables resource-efficient personalization or specialization, but it comes at the ex-  
 013 pense of additional hyperparameter tuning. Although LoRA makes fine-tuning  
 014 efficient, it is highly sensitive to the choice of hyperparameters, and exhaustive  
 015 hyperparameter search is still computationally very demanding. To address these  
 016 challenges, we propose a framework that integrates the domain knowledge of pre-  
 017 trained LLMs into Bayesian Optimization (BO) to efficiently search for LoRA  
 018 hyperparameters. To leverage the informed knowledge of LLMs, we repurpose  
 019 LLMs as a discrete-to-continuous mapping to link the hyperparameters and their  
 020 domain knowledge with a continuous vector space, where BO is conducted. We  
 021 design and control the mapping by language prompting, where we provide a  
 022 domain-aware textual prompts describing the relationships among hyperparam-  
 023 eters and their respective roles; thereby, we explicitly inject domain knowledge  
 024 about LoRA into the LLM in natural language. Also, we model the residual  
 025 information hard to be linguistically described in the prompt with an additional  
 026 learnable token. This aids BO to sample more high-performing hyperparameters.  
 027 In addition, by leveraging the observation of the strong correlation between the  
 028 respective performance obtained from full and subset training datasets in LoRA  
 029 training regimes, we introduce proxy training and evaluation with a data subset.  
 030 This further increases the efficiency of our method. We demonstrate that our hy-  
 031 perparameter found with only about 30 iterations achieves more than 20% per-  
 032 formance improvement over standard hyperparameters found from about 45,000  
 033 combinations. *Code will be released upon acceptance.*

## 1 INTRODUCTION

036 Large Language Models (LLMs) (Touvron et al., 2023; Team, 2024; Team et al., 2024) have been  
 037 recognized as strong foundation models that can be easily adapted to diverse downstream tasks  
 038 with high performance. However, fully fine-tuning LLMs for specific applications is computa-  
 039 tionally heavy. It requires updating billions of parameters, which demands substantial memory and  
 040 computational resources (Brown et al., 2020; Gururangan et al., 2020). To overcome these limi-  
 041 tations, Parameter-Efficient Fine-Tuning (PEFT) (Houlsby et al., 2019; Ding et al., 2023) methods  
 042 have emerged as effective alternatives, enabling strong task adaptation at significantly reduced cost.  
 043 Among these approaches, Low-Rank Adaptation (LoRA) (Hu et al., 2022) stands out as one of the  
 044 most widely adopted techniques. LoRA freezes the pre-trained weights and introduces lightweight,  
 045 trainable low-rank adapters, allowing models to adapt efficiently to new tasks with only a fraction  
 of the parameters and resources required for full fine-tuning.

046 Despite its effectiveness, identifying an *optimal* hyperparameter configuration for LoRA remains  
 047 challenging, as performance is highly sensitive to hyperparameter choices (Sengupta et al., 2024;  
 048 Biderman et al., 2024; Mao et al., 2025). LoRA involves several key hyperparameters, including the  
 049 rank ( $r$ ), scaling factor ( $\alpha$ ), batch size, learning rate, and dropout rate, which are entangled in com-  
 050 plex ways. Consequently, performance can vary significantly depending on their combinations (Hal-  
 051 fion et al., 2024; Sengupta et al., 2024; Mulakala et al., 2024). Therefore, systematically searching  
 052 for the appropriate configuration is a critical issue. Yet, naïve exploration is infeasible: the hyper-  
 053 parameter search space is combinatorially large, and each evaluation is extremely costly (Valipour  
 et al., 2022; Chavan et al., 2023; Sun et al., 2024; Meo et al., 2024; Bini et al., 2025).

This challenge motivates the use of Bayesian optimization (BO) as a principled framework for Hyperparameter Optimization (HPO). BO has proven highly effective in real-world applications where target function evaluations are expensive, such as drug discovery and materials design (Korovina et al., 2020; Ranković & Schwaller, 2023; 2025). BO relies on a surrogate model to approximate the black-box function defined by hyperparameters and their performance, and uses an acquisition function to select the next configuration by balancing exploration and exploitation. However, directly applying BO to LoRA HPO is non-trivial, since traditional BO methods require the underlying function domain to be continuous while the hyperparameters involve discretes and have no way to integrate the domain knowledge during the optimization process (Yan et al., 2025).

In this work, we propose an efficient BO-based HPO framework tailored to LoRA that incorporates domain knowledge through the LLM. Specifically, hyperparameter configurations are expressed as structured text templates describing each hyperparameter’s name, value, role, and interactions. An LLM processes this template along with a learnable token and converts it into a continuous embedding, where domain knowledge is effectively encapsulated in the learnable token. The learnable token, paired with observed performance, is then used to train a surrogate model, which in turn proposes the hyperparameter candidates that maximize the acquisition function. To further improve efficiency, we introduce a proxy training evaluation that significantly reduces evaluation cost and iteration time, enabling faster and more sample-efficient optimization.

Our framework generalizes beyond LoRA to its variants, including DoRA (Liu et al., 2024a), rsLoRA (Kalajdzievski, 2023), and PiSSA (Meng et al., 2024), and is compatible across different model architectures. Experimental results demonstrate consistent performance improvements when applying our HPO framework across diverse settings. Moreover, our approach proves both more efficient and effective than existing search methods (Oliver & Wang, 2024; Tribes et al., 2024) and alternative optimization strategies (Bergstra & Bengio, 2012; Akiba et al., 2019; Li et al., 2021). Finally, by analyzing the experimental results, we observe that previously unexplored hyperparameter combinations can also deliver strong performance, offering insights for new guidelines.

In summary, our contributions are as follows:

- **The first framework combining an LLM with BO specialized for LoRA HPO.** We propose an efficient BO-based LoRA HPO framework that integrates domain knowledge into the optimization process by leveraging an LLM. This framework enables the selection of appropriate hyperparameters from a vast number of possible combinations.
- **Improving efficiency of the proposed framework.** We introduce a projection layer and a learnable token to accelerate BO process. We also introduce a proxy training evaluation protocol that significantly reduces computational cost, enabling efficient optimization.
- **Empirical validation of efficiency and generalizability.** We demonstrate the generalizability of our framework across LoRA variants and model architectures, showing consistent improvements and offering new insights into effective hyperparameter configurations.

## 2 RELATED WORK

**Low-Rank Adaptation (LoRA) and hyperparameter sensitivity in LoRA.** LoRA (Hu et al., 2022) has become one of the most widely adopted parameter-efficient fine-tuning (PEFT) methods (Houlsby et al., 2019) for Large Language Models (LLMs). By introducing a trainable low-rank adapter into a frozen pre-trained model, LoRA allows efficient task-specific adaptation without updating the full set of model parameters. Building on this idea, various LoRA variants have been proposed to improve stability, convergence, and performance. For example, DoRA (Liu et al., 2024a) decomposed each weight into a fixed magnitude and a learnable low-rank direction and rsLoRA (Kalajdzievski, 2023) rescaled LoRA updates by a factor of  $\alpha/\sqrt{r}$  to improve stability. Meng et al. (2024) suggest PiSSA leveraging the principal singular vectors and values of the original weights to initialize LoRA adapters for faster convergence and performance improvement.

Although several advanced LoRA variants have been proposed, the common issue of sensitivity to hyperparameter selection remains a challenge. In particular, rank ( $r$ ) (Zhang et al., 2024), scaling factor ( $\alpha$ ) (Liu et al., 2025), learning rate (Jin et al., 2023), batch size (Marek et al., 2025), and dropout rate (Lin et al., 2024) identified as key factors that influence final results. This sensitivity often leads to large performance variance and complicates fair comparisons across methods.

Moreover, the “optimal” configuration frequently depends on the dataset and base model in use (Rajabzadeh et al., 2024; Yan et al., 2025). Consequently, systematic approaches for optimizing LoRA hyperparameters remain underexplored, as it is difficult to identify effective configurations while accounting for all these factors. Prior work has explored black-box optimization methods (Inouye et al., 2024; Tribes et al., 2024; Oliver & Wang, 2024; Sengupta et al., 2024) and efficient grid-search strategies (Yan et al., 2025) for LoRA hyperparameter selection. Nevertheless, these approaches commonly suffer from two limitations: (i) domain knowledge is not incorporated into the optimization process, and (ii) evaluation remains costly. Hyperparameter optimization generally requires substantial domain knowledge (Wu et al., 2019; Shawki et al., 2021; Czako et al., 2021; Bowler et al., 2022), and LoRA is no exception due to its adapter-specific properties (Halfon et al., 2024; Yan et al., 2025). To address these limitations, we propose a framework that integrates Bayesian optimization and an LLM. This framework can automatically and effectively identify suitable hyperparameters for LoRA, reducing the need for extensive manual tuning.

**Bayesian optimization for hyperparameter optimization.** Hyperparameter optimization is a critical task that significantly impacts model performance in machine learning. However, evaluating each configuration is often expensive due to the high cost of training. In this context, Bayesian optimization has emerged as a prominent method for HPO, especially in expensive evaluation settings (Snoek et al., 2012; Shahriari et al., 2015). BO uses a surrogate model and acquisition function to efficiently search for high-performing hyperparameters with fewer evaluations.

Although BO is an effective approach, its application in discrete input space such as LoRA is limited (Oh et al., 2019; Deshwal & Doppa, 2021; Chu et al., 2024). To mitigate this, several studies (Zhang et al., 2023; Ramos et al., 2023; Agarwal et al., 2025) have shown that hybrid frameworks combining LLMs with BO represent a promising direction, achieving empirical gains across diverse domains. Such approaches include using LLM agents to propose candidate hyperparameter configurations (Liu et al., 2024b), reformulating BO tasks in natural language to flexibly incorporate search spaces and constraints (Liu et al., 2024c), and enhancing surrogate models with LLM embeddings as input features (Nguyen et al., 2024). These synergies between LLMs and BO extend beyond HPO to other domains, further emphasizing their effectiveness (Ranković & Schwaller, 2023; 2025). Building on this trend, we propose the first framework that integrates BO with LLMs for LoRA HPO. We construct an embedding space tailored to LoRA HPO using an LLM with domain prompting and learnable tokens, and perform BO within this space, improving search efficiency under high-cost evaluation conditions.

### 3 METHOD

We propose a framework that combines a Large Language Model (LLM) with Bayesian optimization (BO) to discover appropriate hyperparameters for LoRA tuning in each task. We obtain continuous embeddings from the LLM and use them as inputs to the surrogate model, enabling a BO process tailored to LoRA Hyperparameter Optimization (HPO). The LLM in our framework not only encodes rich prior knowledge through large-scale pretraining, but also provides a convenient interface for injecting additional knowledge in textual form. Furthermore, to reduce cost, we introduce proxy training evaluation, which estimates the performance of a full-dataset model using a model trained on a subset of the data. With these components, our framework improves not only the sample efficiency of BO, but also the computational efficiency of LoRA hyperparameter optimization as a whole. Section 3.1 introduces the preliminaries of BO, Sec. 3.2 presents the proposed framework and its components, and Sec. 3.3 details our proxy training evaluation.

#### 3.1 PRELIMINARY: BAYESIAN OPTIMIZATION

BO is an efficient approach for optimizing black-box functions, particularly when the evaluation cost is expensive. The goal of BO is to find the optimal input  $\mathbf{x}^*$  from a candidate pool  $\mathcal{X}$  that maximizes a black-box function  $f$ . The objective of BO can be formulated as follows:

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}). \quad (1)$$

Since  $f$  is hard to estimate, *surrogate model*  $\hat{f}$  is used to approximate  $f$ . A common choice for the *surrogate model* is a Gaussian Process (GP), which can be expressed as:  $\hat{f} \sim \mathcal{GP}(\mu, k_\omega)$ , where  $\mu$

162 is mean function and  $k_\omega$  denotes the kernel function with hyperparameter  $\omega$ . For example, in the  
 163 case of the Matérn 5/2 kernel,  $\omega$  includes trainable hyperparameters,  
 164

$$165 \quad k_\omega(\mathbf{x}, \mathbf{x}') = \sigma^2 \left( 1 + \frac{\sqrt{5}d}{\ell} + \frac{5d^2}{3\ell^2} \right) \exp \left( -\frac{\sqrt{5}d}{\ell} \right), \quad (2)$$

166 where  $\ell$  denotes the lengthscale,  $\sigma^2$  denotes covariance and  $d = \|\mathbf{x} - \mathbf{x}'\|_2$ . Given  $n$  observed points  
 167 set  $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ , the surrogate model is tuned on observed points, and an acquisition function  
 168  $\alpha$  is used to determine the next evaluation point  $\tilde{\mathbf{x}}$  based on the posterior from the *surrogate model*:  
 169

$$171 \quad \tilde{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x} | \hat{f}, \mathcal{D}). \quad (3)$$

### 173 3.2 PROPOSED FRAMEWORK

174  
 175 **Overview.** Our framework performs 4 steps in each iteration: (1) Proxy training evaluation  
 176 (*Proxy*), which fine-tunes LoRA on a subset of the dataset and measures its performance; (2)  
 177 Embedding extraction using the LLM; (3) Surrogate model update; and (4) Next evaluation point  
 178 suggestion. For example, in the  $n$ -th iteration, a hyperparameter configuration  $x_n$  is selected, and  
 179 its benchmark performance  $y_n$  is obtained through proxy training evaluation. The configuration  $x_n$   
 180 is then converted into a structured template  $t_n$  via domain-aware prompting. This template, together  
 181 with the learnable token  $\psi$ , is passed into the LLM  $\phi$  and projection layer  $P(\cdot; \theta)$  to produce an  
 182 embedding:  $\mathbf{z}_n = P(\phi(t_n, \psi); \theta)$ . The surrogate model parameterized  $\omega$  is updated by maximizing  
 183 the *marginal log-likelihood* using embedding  $\mathbf{z}_n$  paired with the observed target  $y_n$ , jointly updating  
 184 all trainable parameters  $\omega$ ,  $\theta$ , and  $\psi$ . Finally, the next evaluation point is selected by generating  
 185 embeddings for every hyperparameter configuration  $x_j$  in the candidate pool  $\mathcal{X}_{\text{cand}}$  and evaluating  
 186 the acquisition function  $\alpha$ . Algorithm 1 describes the entire procedure in pseudo-code.

---

#### 187 Algorithm 1 Pseudo code for our framework

---

188  
 189 **Require:** Candidate pool  $\mathcal{X}_{\text{cand}}$ , observed dataset  $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^n$ , budget  $N$ ,  
 190 parameters  $\omega$  (GP),  $\theta$  (Projection layer),  $\psi$  (Learnable token),  
 191 LLM  $\phi$ , acquisition function  $\alpha$ , feature extractor  $g(\cdot; \theta, \psi)$   
 192 **Initialize:** parameters  $\omega, \theta, \psi; \mathcal{D}_0 \leftarrow \emptyset$ ; Choose initial candidate  $x_1 \in \mathcal{X}_{\text{cand}}$   
 193 1: **for**  $n = 1$  to  $N$  **do**  $\triangleright$  **Proxy** means **Proxy training evaluation**  
 194 2:    $y_n \leftarrow \text{Proxy}(x_n)$   
 195 3:    $\mathcal{D}_n \leftarrow \mathcal{D}_{n-1} \cup \{(x_n, y_n)\}$   
 196 4:   Remove  $x_n$  from  $\mathcal{X}_{\text{cand}}$   $\triangleright$  Surrogate model update  
 197 5:   **while** not convergence **do**  
 198 6:     **for all**  $x_i \in \mathcal{D}_n$  **do**  $\triangleright$  **Template** means **Domain-aware prompting**  
 199 7:        $t_i \leftarrow \text{Template}(x_i)$   
 200 8:        $\mathbf{z}_i \leftarrow g(\mathbf{x}_i; \theta, \psi) = P(\phi(t_i, \psi); \theta)$   
 201 9:     **end for**  
 202 10:    Compute *marginal log-likelihood*  $\log p(\mathbf{y} | \mathbf{Z}, \omega, \theta, \psi)$   
 203 11:    Update  $\omega, \theta, \psi$   
 204 12:   **end while**  $\triangleright$  Bayesian optimization  
 205 13:   **for all**  $x_j \in \mathcal{X}_{\text{cand}}$  **do**  
 206 14:      $t_j \leftarrow \text{Template}(x_j)$   
 207 15:      $\mathbf{z}_j \leftarrow g(\mathbf{x}_j; \theta, \psi) = P(\phi(t_j, \psi); \theta)$   
 208 16:     Compute  $\alpha(\mathbf{z}_j; \omega, \theta, \psi)$   
 209 17:   **end for**  
 210 18:    $j' = \arg \max_j \alpha(\mathbf{z}_j; \omega, \theta, \psi)$   
 211 19:   Suggest next evaluation point  $x_{n+1} \leftarrow x_{j'}$   
 212 20: **end for**  
 213 21:  $(x^*, y^*) \leftarrow \arg \max_{(x, y) \in \mathcal{D}} y$   
 214 22: **return**  $x^*$

---

215 **Domain-aware prompting.** We employ domain-aware prompting to explicitly incorporate domain  
 216 knowledge about LoRA hyperparameters into the optimization process. A straightforward text tem-  
 217 plate can be written as  $t = \{\text{name}, \text{value}\}$  (Table A10a). However, this simple format fails to

capture the roles or relationships between hyperparameters. Prior studies have highlighted practical know-how and manual tuning guidelines for hyperparameter tuning (Mohammed & Kora, 2025; He, 2024; Diehl, 2024; unsloth, 2025). For example, Hu et al. (2022) suggest that the scaling factor ( $\alpha$ ) in LoRA behaves similarly to adjusting the learning rate. To better reflect existing know-how and guidelines, we design a structured text template  $t = \{\text{explanation, name, value}\}$  (Table A10b). The explanations emphasize relationship between hyperparameters (e.g. how rank and alpha are commonly set) as well as the training dynamics that arise when their magnitudes vary. This approach goes beyond simple LLM embeddings, enabling the direct insertion of structured, prompt-level information into the embedding space.

**Learnable token and projection layer.** Calibrating the embedding space extracted from the LLM during feature extraction can enhance BO performance compared to using fixed embeddings (Kris-tiadi et al., 2024; Ranković & Schwaller, 2025). Motivated by this insight, we introduce a learnable token  $\psi$  along with a projection layer  $P(\cdot; \theta)$ , parameterized by  $\theta$ , to transform embeddings into a space better suited for BO. We append the learnable token to the domain-aware text template  $t$  and feed both into the LLM to extract embeddings, allowing the token to compactly capture LoRA hyperparameter's knowledge. These embeddings are then passed through the projection layer, producing representations tailored for BO. Throughout this process, the pre-trained LLM remains frozen, while  $\psi$  and  $\theta$  are learnable. The final embedding is obtained via pooling the embedding at last token's position, resulting in the final feature:  $\mathbf{z} = P(\phi(t, \psi); \theta)$ . As a result, the embedding not only explicitly reflects the explanations encoded in the prompt, however also implicitly internalizes LoRA-specific domain knowledge. This improves representational power and enhances BO efficiency with minimal additional parameters.

**Bayesian optimization with LLM.** BO typically employs Gaussian Processes (GPs) as surrogate models, which are effective for modeling distributions over continuous spaces (Beckers, 2021). However, when dealing with complex input spaces that require understanding the relationships among variables, it becomes crucial to use representations capable of capturing such structure (Lee et al., 2025). This is particularly true for the LoRA hyperparameter space, which is inherently discrete and requires domain knowledge. To address this challenge, we integrate an LLM with a learnable token and a projection layer, which inject domain knowledge about LoRA HPO into when extracting embeddings:  $\mathbf{z} = g(\mathbf{x}; \theta, \psi) = P(\phi(t, \psi); \theta)$ . Therefore, we employ LLM-based deep kernel learning to combine the prior knowledge encoded in the LLM with these trainable neural architecture for the GP, thereby transforming standard GP regression into deep kernel learning:

$$k(\mathbf{x}, \mathbf{x}' | \omega) \rightarrow k(g(\mathbf{x}; \theta, \psi), g(\mathbf{x}'; \theta, \psi) | \omega, \theta, \psi). \quad (4)$$

We jointly optimize all trainable parameters,  $\Phi = \{\omega, \theta, \psi\}$ , where  $\omega$ ,  $\theta$ , and  $\psi$  denote the GP kernel, projection layer, and learnable token parameters, respectively. These are optimized by maximizing the marginal log-likelihood:

$$\mathcal{L}(\Phi) = \log p(\mathbf{y} | \mathbf{X}, \Phi) = -\frac{1}{2} \{(\mathbf{y} - \mu\mathbf{1})^\top \mathbf{K}_\Phi^{-1} (\mathbf{y} - \mu\mathbf{1}) + \log |\mathbf{K}_\Phi| + n \log 2\pi\}, \quad (5)$$

$$\Phi^* = \arg \max_{\Phi} \mathcal{L}(\Phi), \quad (6)$$

where  $\mathbf{K}_\Phi$  denotes covariance matrix determined from the covariance kernel of the GP,  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  and  $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$ .

### 3.3 PROXY TRAINING EVALUATION

Previous studies (Klein et al., 2017; Oliver & Wang, 2024) have shown that it is not always necessary to train on the entire dataset at every optimization step because training performance on subset datasets strongly correlates with that of full training. Building on these insights, we introduce a proxy training evaluation strategy to reduce fine-tuning time cost. Specifically, instead of training on the full dataset, we fine-tune the model on a randomly selected subset and measure performance on this smaller training run as a proxy for the true performance. Despite its simplicity, this approach exhibits strong correlation with the true performance, and we find that using only 10% of the data can be sufficient. As a result, we reduce the overall time cost by up to 10x, enabling more optimization iterations within the same computational budget. We further consider data selection strategies such as Liu et al. (2024d), but we observe that our simple random subset achieves comparably high correlation with full-data results relative to these strategies.

270 4 EXPERIMENTS  
271272 4.1 EXPERIMENTAL SETTING  
273274 **LoRA hyperparameters and setup.** We define the candidate pool of hyperparameters as shown in  
275 Table 1. Specifically, we optimize five hyperparameters: rank ( $r$ ), scaling factor ( $\alpha$ ), learning rate,  
276 dropout rate, and batch size—resulting in a search space of more than 45,000 configurations. To  
277 validate our proposed framework, we conduct experiments across multiple LoRA variants, including  
278 rsLoRA (Kalajdzievski, 2023), DoRA (Liu et al., 2024a), and PiSSA (Meng et al., 2024).  
279280 **Tasks.** Following prior work (Meng et al.,  
281 2024), we fine-tune models on the Meta-  
282 MathQA dataset (Yu et al., 2023) and evaluate  
283 performance on GSM8k (Cobbe et al., 2021)  
284 and MATH (Hendrycks et al., 2021), and re-  
285 port Accuracy (%). To test generalization be-  
286 yond mathematical reasoning, we extend ex-  
287 periments to code generation, fine-tuning on  
288 the CodeFeedback dataset (Zheng et al., 2024)  
289 and evaluating on HumanEval (Chen et al.,  
290 2021) and MBPP (Austin et al., 2021), report-  
291 ing Pass@1 which is the probability that the  
292 first generated solution solves the task. Each  
293 training dataset contains 100K samples, with a 10K subset used for proxy training evaluation.294 **Baselines.** We benchmark our framework against several HPO methods: random search (Bergstra  
295 & Bengio, 2012), Optuna (Akiba et al., 2019), standard BO (Oliver & Wang, 2024), latent BO  
296 (LBO) (Li et al., 2021), and NOMAD (Tribes et al., 2024). Details are reported in Appendix B.297 4.2 EXPERIMENTAL RESULTS  
298299 **Hyperparameter optimization for LoRA variants and various models.** Based on previous find-  
300 ings that tasks and architectures demand distinct hyperparameters (Sengupta et al., 2024; He, 2024;  
301 Mohammed & Kora, 2025), we evaluate our framework across diverse LoRA variants and mod-  
302 els. Table 2 shows that adapting our HPO framework enables effective hyperparameter search for  
303 each LoRA variant, consistently improving performance compared to the originally reported re-  
304 sults. Surprisingly, our framework achieves up to 21.46% accuracy improvement, emphasizing the  
305 importance of hyperparameter selection. These results suggest that there is significant room for im-  
306 provement in existing LoRA variants through systematic hyperparameter search. Similarly, Table 3  
307 demonstrates that our approach can identify appropriate model-specific hyperparameters. Across  
308 different backbone Large Language Models (LLMs), adapting our framework consistently achieves  
309 substantial improvements, highlighting its practical utility for fine-tuning newly introduced models.  
310311 **Table 2: Results of applying our framework to LoRA variants.** We set the hyperparameter  
312 configurations suggested by each work, where they were dedicatedly tuned (Kalajdzievski, 2023;  
313 Liu et al., 2024a; Meng et al., 2024; Wang et al., 2024). Using our method, we observe consistent  
314 performance improvements across all variants.315 **Table 1: Hyperparameter search range.** We set  
316 the hyperparameter search ranges based on prior  
317 work (Meng et al., 2024; Wang et al., 2024; In-  
318 ouye et al., 2024; Diehl, 2024; Yan et al., 2025;  
319 unsloth, 2025), resulting in a search space of over  
320 45,000 configurations.

Hyperparameters	Search Range	Count
Rank ( $r$ )	$1 \sim 256 (2^n)$	9
Scaling Factor ( $\alpha$ )	$\frac{r}{2} \sim 128r (2^n r)$	9
Batch Size	$2 \sim 256 (2^n)$	8
Learning Rate	$1e-6 \sim 5e-3$	10
Dropout Rate	$0.0 \sim 0.3 (0.5 \times n)$	7

Strategy	Ours	Accuracy (%)		Pass@1	
		GSM8K	MATH	HumanEval	MBPP
LoRA (Hu et al., 2022)	✗	41.47	5.24	16.31	35.47
	✓	62.93 (+21.46)	12.88 (+7.64)	30.49 (+14.18)	42.59 (+7.12)
rsLoRA (Kalajdzievski, 2023)	✗	41.16	5.46	16.46	35.72
	✓	58.15 (+16.99)	10.76 (+5.3)	29.87 (+13.41)	42.06 (+6.34)
DoRA (Liu et al., 2024a)	✗	40.11	5.36	17.07	36.51
	✓	57.01 (+16.9)	10.78 (+5.42)	30.58 (+13.51)	42.33 (+5.82)
PiSSA (Meng et al., 2024)	✗	52.46	7.34	22.56	40.48
	✓	60.88 (+8.42)	12.06 (+4.72)	31.71 (+9.15)	41.53 (+1.05)

324  
 325  
 326  
 327  
**Table 3: Results of applying our framework across diverse models.** We compare against the hy-  
 328 perparameter settings suggested by PiSSA (Meng et al., 2024), where they were dedicatedly tuned.  
 329 The experiments demonstrate that our method is effective across a wide range of models.  
 330  
 331  
 332  
 333  
 334  
 335

Model	Ours	Accuracy (%)		Pass@1	
		GSM8K	MATH	HumanEval	MBPP
LLaMA2-7B (Touvron et al., 2023)	✗	41.47	5.24	16.31	35.47
	✓	62.93 (+21.46)	12.88 (+7.64)	30.49 (+14.18)	42.59 (+7.12)
Mistral-7B-v0.1 (Jiang et al., 2023)	✗	69.90	19.96	45.73	61.90
	✓	74.07 (+4.17)	23.46 (+3.5)	54.27 (+8.54)	65.08 (+3.18)
Gemma-7B (Team et al., 2024)	✗	75.51	29.44	49.39	63.23
	✓	78.77 (+3.26)	30.24 (+0.8)	53.05 (+3.66)	67.46 (+4.23)

336  
 337  
 338  
 339  
**Table 4: Comparison against existing HPO**  
**methods.** Our method outperforms existing HPO  
 approaches under the same optimization budget.

Search Method	Accuracy (%)		Pass@1	
	GSM8K	MATH	HumanEval	MBPP
Random	59.14	10.51	23.17	36.77
Optuna (TPE)	37.38	4.74	27.44	38.62
BO	57.32	11.42	20.12	35.19
LBO	59.51	11.88	26.83	37.83
Ours	<b>62.93</b>	<b>12.88</b>	<b>30.49</b>	<b>42.59</b>

340  
 341  
 342  
 343  
 344  
 345  
 346  
 347  
 Overall, our framework can be applied as a plug-and-play module that adapts to the LoRA variants  
 and models, giving consistent gains without manual hyperparameter tuning.

348  
 349  
 350  
 351  
 352  
 353  
 354  
 355  
 356  
 357  
 358  
 359  
 We further analyze the hyperparameter combinations discovered in our experiments. As shown in  
 Tables A1 and A2, smaller batch sizes are often preferred, consistent with prior findings (Marek  
 et al., 2025). In addition, applying dropout often leads to better performance. Interestingly, we  
 sometimes observe strong performance when the scaling factor ( $\alpha$ ) is 16 or even 32 times larger  
 than the rank. This observation has not been reported in prior studies, where  $\alpha$  was set to twice  
 the rank according to existing guidelines (Diehl, 2024; unsloth, 2025), or determined based on a  
 rank or fixed  $\alpha$  (Kalajdzievski, 2023; Sun et al., 2024; Liu et al., 2025). This suggest that there may  
 exist settings beyond the commonly chosen rank and  $\alpha$  values that can further improve performance,  
 thereby hinting at the possibility of proposing a new guideline.

360  
 361  
 362  
 363  
 364  
 365  
 366  
 367  
 368  
**Comparison with various HPO methods.** We evaluate the effectiveness of our framework, against  
 widely adopted baselines for HPO. Table 4 summarizes the results of applying each method under  
 the same optimization budget. The results demonstrate that our approach identifies more suitable  
 hyperparameters within a constrained budget. Notably, our method discovers better configurations  
 than other BO-based methods, indicating that leveraging LLMs to provide domain knowledge about  
 the search space can improve both search efficiency and effectiveness. We also compare our frame-  
 work with Tribes et al. (2024), a dedicated approach for LoRA HPO that employs validation loss  
 with the NOMAD algorithm for hyperparameter estimation. As shown in Table 5, our method finds  
 more appropriate configurations in a shorter time. These experiments support that the combination  
 of LLM and BO leads to improvement in both accuracy and efficiency.

369  
 370  
 371  
 372  
 373  
 374  
 375  
 376  
 377  
**Ablation studies.** We conduct a series of experiments to evaluate the effect of each proposed  
 component. Our framework incorporates domain knowledge into the optimization process through  
 three components: domain-aware prompting for explicit knowledge injection, and a projection  
 layer with a learnable token for implicitly encoding domain knowledge. As shown in Table 6,  
 adding each component consistently helps BO to discover better-performing hyperparameter set-

378  
 379  
 380  
 381  
**Table 5: Comparison against existing LoRA**  
**HPO method.** We compare our approach  
 with Tribes et al. (2024), which applies the NO-  
 MAD algorithm specifically for LoRA hyperpa-  
 rameter tuning. Our method is both more time-  
 efficient and more effective, achieving superior  
 performance by a significant margin. Note that  
 H denotes hours.

Method	Time	GSM8K	MATH	HumanEval	MBPP
Tribes et al. (2024)	180 H	52.16	9.12	24.39	37.30
Ours	24 H	62.93	12.88	30.49	42.59

382  
 383  
 384  
 385  
 386  
 387  
**Table 6: Ablation results.** We validate each of  
 our proposed components and find that all con-  
 tribute effectively to LoRA HPO.

Projection Layer	Domain-aware Prompting	Learnable Token	GSM8K	MATH
✗	✗	✗	47.76	8.72
✓	✗	✗	53.98	9.16
✓	✓	✗	61.41	12.46
✓	✓	✓	<b>62.93</b>	<b>12.88</b>

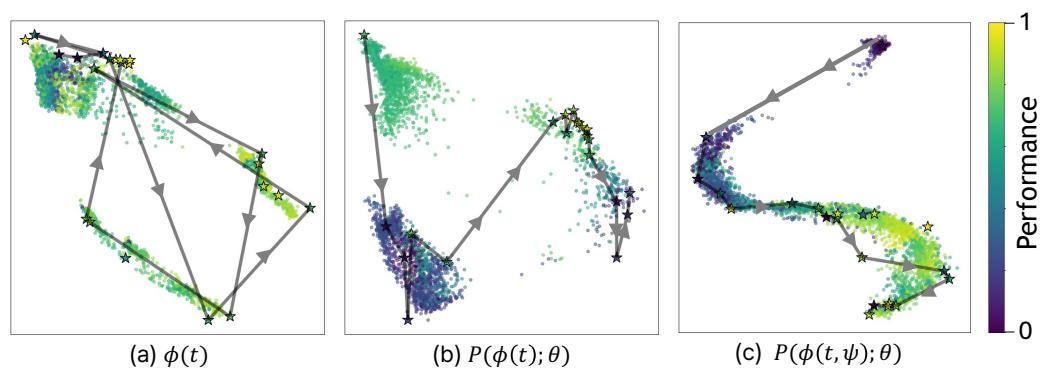


Figure 1: **Qualitative analysis of embedding space evolution using our components.** We illustrate how the embedding space evolves with our proposed components: (a) shows the embedding space from a frozen LLM  $\phi$ ; (b) shows the space when a projection layer  $P(\cdot; \theta)$  is added to the frozen LLM; and (c) shows the space when both the projection layer and the learnable token  $\psi$  are employed. The trajectories in each figure indicate optimization paths across steps, shown in arrow sequence. These results suggest that incorporating the projection layer and learnable token produces a smoother, more structured embedding space suited for BO, thereby enabling efficient optimization.

tings, demonstrating the effectiveness of our framework. Notably, domain-aware prompting plays a crucial role in performance improvement, emphasizing the importance of explicitly injecting domain knowledge at the prompt level. We further analyze the differences in the optimization process introduced by each component. Without any components, BO tends to keep the learning rate nearly fixed, resulting in insufficient exploration of the search space. In contrast, BO with all components explores broadly across the hyperparameter candidate pool. These findings show that our framework enables effective exploration of diverse hyperparameters even with a small number of iterations, allowing BO to operate over a much broader search space.

**Qualitative analysis of the effect of our framework.** We visualize the embedding  $z$  of hyperparameter configurations to illustrate the effect of adding each component of our framework, as shown in Fig. 1. The figure compares three settings: (a) frozen LLM embeddings, (b) embeddings after applying the projection layer, and (c) embeddings with both the projection layer and a learnable token. With frozen LLM embeddings, high- and low-performing hyperparameters configurations remain entangled, leading to an unstable search process. This results indicates that the embedding space does not effectively separate hyperparameter combinations and may hinder the balance between exploration and exploitation. Introducing a projection layer begins to separate the embeddings, revealing clearer structures that distinguish the performance levels. When we additionally incorporate a learnable token, the embeddings exhibit directional organization aligned with performance, enabling more reliable surrogate fitting and a better-organized space overall. Furthermore, we analyze the trajectories of the BO process across different settings and find that optimization using only a frozen LLM proceeds without a clear direction. In contrast, when components for embedding calibration are included, the BO process consistently moves toward the high-performing region. These observations suggest that calibration with a projection layer and a learnable token makes the BO landscape more discriminative and smooth compared to using fixed embeddings, thereby improving search efficiency and final performance.

**Validation of proxy training evaluation.** To reduce the time cost of fine-tuning during HPO, we introduce proxy training evaluation in Sec. 3.3. Using the proposed proxy training evaluation, we estimate the performance by training on a subset of the dataset, treating it as a proxy for full-data performance. To further investigate this, we examine the Pearson correlation between the training performance of subset datasets at various sampling ratios and that of the full dataset. Additionally, we compare with the existing data sampling method,

Table 7: **Correlation between the performance trained on a subset and on the full dataset.** Proxy training evaluation shows comparable correlation to full dataset accuracy, at both random sampling and TSDS (Liu et al., 2024d).

Sampling Method	MATH Reasoning	Code Generation
Random (1%)	0.7031	0.7429
Random (5%)	0.8360	0.9282
Random (10%)	0.8713	<b>0.9427</b>
TSDS (10%) by Test dataset	<b>0.8754</b>	0.9290
TSDS (10%) by Train dataset	0.8649	0.9278

**Table 8: Performance differences across model sizes.** We apply the hyperparameters discovered for each model size of Qwen2.5 to fine-tune all model sizes. “Model” denotes the model being fine-tuned, while “Settings” indicate the size of the model from which the hyperparameters were obtained. The results show that variations in model size do not significantly affect the discovery of effective hyperparameter settings.

Model	Settings	Accuracy (%)		Pass@1	
		GSM8K	MATH	HumanEval	MBPP
Qwen2.5-3B (Qwen et al., 2025)	3B	79.53	43.18	70.12	77.78
	7B	78.54	42.40	68.29	75.13
	14B	78.47	42.94	67.07	77.25
Qwen2.5-7B (Qwen et al., 2025)	3B	84.08	48.90	81.09	78.84
	7B	83.93	48.08	79.88	78.57
	14B	83.09	48.58	81.71	78.31
Qwen2.5-14B (Qwen et al., 2025)	3B	87.41	51.68	82.32	82.80
	7B	86.81	50.62	79.88	81.22
	14B	87.34	51.50	81.71	82.01

TSDS (Liu et al., 2024d), setting the target distribution to the test dataset or to the training dataset. Table 7 demonstrates that proxy training evaluation with a randomly selected 10% subset provides a sufficiently accurate approximation of full dataset performance. These results indicate that our proxy training evaluation serves as an effective and reliable indicator of model performance on the full dataset. Moreover, the correlation obtained from the 10% random subset is comparable to that of TSDS (Liu et al., 2024d) and even achieves the highest correlation in the code generation task. Based on these findings, we adopt 10% random sampling to construct the subset dataset.

**Effect of model size on LoRA HPO.** We investigate the effect of model size on finding suitable LoRA hyperparameters. Specifically, we apply our framework to Qwen2.5 models with 3B, 7B, and 14B parameters, identifying the best hyperparameters for each model size. We then use these configurations to fine-tune models across all sizes. As shown in Table 8, hyperparameter configurations discovered on one model size generally remain effective for other sizes. These results suggest that variations in model size do not significantly affect the discovery of effective hyperparameter settings. In contrast, we find differences between architecture: compared to LLaMA2, Qwen2.5 models tend to prefer smaller ranks and larger batch sizes (see Tables A1 and A2). Moreover, as shown in Table A8, when we cross-apply the configurations found with the Qwen2.5 model to LLaMA2 model and vice versa, we observe substantially larger performance degradation than when each configuration is used within the same model series. These observations indicate that hyperparameters are influenced more by model architecture than scale, which is consistent with findings from prior work (Yan et al., 2025). Since the configurations are largely transferable across scales within the same model series, this implies that tuning costs can be reduced by applying configurations found on smaller models to larger ones, rather than running the framework directly on larger models.

## 5 CONCLUSION

We propose a framework that combines Large Language Models (LLMs) with Bayesian optimization (BO) for LoRA Hyperparameter Optimization (HPO). Domain knowledge about LoRA is explicitly injected into the BO process via domain-aware prompting, while a learnable token and a projection layer transform LLM embeddings into a space better suited for optimization. To further reduce cost, we employ empirically validated proxy training evaluation, which estimates fine-tuning performance using a subset of the training data. As a result, our framework identifies appropriate hyperparameter configurations from a large candidate pool with significantly reduced optimization time. It functions as a plug-and-play module, achieving consistent performance improvements across LoRA variants, model architectures, and model scales. Comparisons with existing HPO methods validate its effectiveness both in terms of cost and performance. Beyond LoRA, we believe this framework can serve as a practical baseline for broader HPO in diverse fine-tuning strategies.

486 6 REPRODUCIBILITY STATEMENT  
487488 We provide comprehensive details of the experimental setup in the Appendix B, D, including  
489 full specifications of the hyperparameters configuration searched, and training procedures, and the  
490 prompt used for LLM feature extraction to ensure clarity and transparency. These materials are  
491 intended to make it straightforward for others to replicate our experiments and verify the reported  
492 results. In addition, we will make the complete codebase publicly available upon acceptance.493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539

## 540 REFERENCES

541

542 Dhruv Agarwal, Manoj Ghuhan, Rajarshi Raj Das, Sandesh Swamy, Sopan Khosla, and Rashmi  
543 Gangadharaiah. Searching for optimal solutions with llms via bayesian optimization. 2025.

544

545 Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna:  
546 A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM  
547 SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

548

549 Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan,  
550 Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language  
551 models. *arXiv preprint arXiv:2108.07732*, 2021.

552

553 Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, An-  
554 drew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo  
555 Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL  
556 <http://arxiv.org/abs/1910.06403>.

557

558 Thomas Beckers. An introduction to gaussian process models. *arXiv preprint arXiv:2102.05497*,  
559 2021.

560

561 James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The journal  
562 of machine learning research*, 13(1):281–305, 2012.

563

564 Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor  
565 Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and  
566 forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

567

568 Massimo Bini, Leander Girrbach, and Zeynep Akata. Delora: Decoupling angles and strength in  
569 low-rank adaptation. *arXiv preprint arXiv:2503.18225*, 2025.

570

571 Alexander L Bowler, Michael P Pound, and Nicholas J Watson. A review of ultrasonic sensing and  
572 machine learning methods to monitor industrial processes. *Ultrasonics*, 124:106776, 2022.

573

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

577

578 Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. One-for-all: Generalized  
579 lora for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*, 2023.

580

581 Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared  
582 Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large  
583 language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

584

585 Jaewon Chu, Jinyoung Park, Seunghun Lee, and Hyunwoo J Kim. Inversion-based latent bayesian  
586 optimization. *Advances in Neural Information Processing Systems*, 37:68258–68286, 2024.

587

588 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,  
589 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to  
590 solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

591

592 Zoltan Czako, Gheorghe Sebestyen, and Anca Hangan. Automaticai—a hybrid approach for auto-  
593 matic artificial intelligence algorithm selection and hyperparameter tuning. *Expert Systems with  
594 Applications*, 182:115225, 2021.

595

596 Aryan Deshwal and Jana Doppa. Combining latent space and structured kernels for bayesian op-  
597 timization over combinatorial spaces. *Advances in neural information processing systems*, 34:  
598 8185–8200, 2021.

599

600 Stephen Diehl. Introduction to fine-tuning large language models. [https://www.stephendiehl.com/posts/training\\_llms/#:~:text=However%20if%20you%20are%20going,increasing%20order%20from%20the%20default](https://www.stephendiehl.com/posts/training_llms/#:~:text=However%20if%20you%20are%20going,increasing%20order%20from%20the%20default),  
601 2024.

594 Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin  
 595 Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained  
 596 language models. *Nature machine intelligence*, 5(3):220–235, 2023.

597

598 Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey,  
 599 and Noah A Smith. Don’t stop pretraining: Adapt language models to domains and tasks. *arXiv*  
 600 *preprint arXiv:2004.10964*, 2020.

601 Alon Halfon, Shai Gretz, Ofir Ariv, Artem Spector, Orith Toledo-Ronen, Yoav Katz, Liat Ein-  
 602 Dor, Michal Shmueli-Scheuer, and Noam Slonim. Stay tuned: An empirical study of the impact  
 603 of hyperparameters on llm tuning in real-world applications. *arXiv preprint arXiv:2407.18990*,  
 604 2024.

605 Pengfei He. Parameter efficient instruction tuning: An empirical study. *arXiv preprint*  
 606 *arXiv:2411.16775*, 2024.

607

608 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,  
 609 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv*  
 610 *preprint arXiv:2103.03874*, 2021.

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

614

615 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang,  
 616 Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

617 Darren Inouye, Lucas Lindo, Robin Lee, and Edmund Allen. Applied auto-tuning on lora hyperpa-  
 618 rameters. 2024.

619

620 Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chap-  
 621 lot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier,  
 622 Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas  
 623 Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*,  
 624 2023.

625 Hongpeng Jin, Wenqi Wei, Xuyu Wang, Wenbin Zhang, and Yanzhao Wu. Rethinking learning  
 626 rate tuning in the era of large language models. In *2023 IEEE 5th International Conference on*  
 627 *Cognitive Machine Intelligence (CogMI)*, pp. 112–121. IEEE, 2023.

628

629 Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint*  
 630 *arXiv:2312.03732*, 2023.

631

632 Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian op-  
 633 timization of machine learning hyperparameters on large datasets. In *Artificial intelligence and*  
 634 *statistics*, pp. 528–536. PMLR, 2017.

635

636 Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff  
 637 Schneider, and Eric Xing. Chembo: Bayesian optimization of small organic molecules with syn-  
 638 thezable recommendations. In *International Conference on Artificial Intelligence and Statistics*,  
 639 pp. 3393–3403. PMLR, 2020.

640

641 Agustinus Kristiadi, Felix Strieth-Kalthoff, Marta Skreta, Pascal Poupart, Alán Aspuru-Guzik, and  
 642 Geoff Pleiss. A sober look at llms for material discovery: Are they actually good for bayesian  
 643 optimization over molecules? *arXiv preprint arXiv:2402.05015*, 2024.

644

645 Seunghun Lee, Jinyoung Park, Jaewon Chu, Minseo Yoon, and Hyunwoo J Kim. Latent bayesian  
 646 optimization via autoregressive normalizing flows. *arXiv preprint arXiv:2504.14889*, 2025.

647

648 Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, and Alistair Shilton. High  
 649 dimensional bayesian optimization using dropout. *arXiv preprint arXiv:1802.05400*, 2021.

650

651 Yang Lin, Xinyu Ma, Xu Chu, Yujie Jin, Zhibang Yang, Yasha Wang, and Hong Mei. Lora dropout  
 652 as a sparsity regularizer for overfitting control. *arXiv preprint arXiv:2404.09610*, 2024.

648 Jun Liu, Zhenglun Kong, Peiyan Dong, Xuan Shen, Pu Zhao, Hao Tang, Geng Yuan, Wei Niu,  
 649 Wenbin Zhang, Xue Lin, et al. Rora: Efficient fine-tuning of llm with reliability optimization for  
 650 rank adaptation. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and*  
 651 *Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

652 Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-  
 653 Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first*  
 654 *International Conference on Machine Learning*, 2024a.

655 Siyi Liu, Chen Gao, and Yong Li. Large language model agent for hyper-parameter optimization.  
 656 *arXiv preprint arXiv:2402.01881*, 2024b.

657 Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models  
 658 to enhance bayesian optimization. *arXiv preprint arXiv:2402.03921*, 2024c.

659 Zifan Liu, Amin Karbasi, and Theodoros Rekatsinas. Tsds: Data selection for task-specific model  
 660 finetuning. *Advances in Neural Information Processing Systems*, 37:10117–10147, 2024d.

661 Yuren Mao, Yuhang Ge, Yijiang Fan, Wenyi Xu, Yu Mi, Zhonghao Hu, and Yunjun Gao. A survey  
 662 on lora of large language models. *Frontiers of Computer Science*, 19(7):197605, 2025.

663 Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Gordon Wilson, and Micah Goldblum.  
 664 Small batch size training for language models: When vanilla sgd works, and why gradient accu-  
 665 mulation is wasteful. *arXiv preprint arXiv:2507.07101*, 2025.

666 Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular  
 667 vectors adaptation of large language models. *Advances in Neural Information Processing Systems*,  
 668 37:121038–121072, 2024.

669 Cristian Meo, Ksenia Sycheva, Anirudh Goyal, and Justin Dauwels. Bayesian-lora: Lora based  
 670 parameter efficient fine-tuning using optimal quantization levels and rank values trough differen-  
 671 tiable bayesian gates. *arXiv preprint arXiv:2406.13046*, 2024.

672 Ammar Mohammed and Rania Kora. A comprehensive overview and analysis of large language  
 673 models: Trends and challenges. *IEEE Access*, 2025.

674 Benarji Mulakala, Madan Lal Saini, Ashirvad Singh, Vamsi Bhukya, and Arnod Mukhopadhyay.  
 675 Adaptive multi-fidelity hyperparameter optimization in large language models. In *2024 8th In-  
 676 ternational Conference on Computational System and Information Technology for Sustainable  
 677 Solutions (CSITSS)*, pp. 1–5. IEEE, 2024.

678 Tung Nguyen, Qiuyi Zhang, Bangding Yang, Chansoo Lee, Jorg Bornschein, Yingjie Miao, Sagi  
 679 Perel, Yutian Chen, and Xingyou Song. Predicting from strings: Language model embeddings  
 680 for bayesian optimization. *arXiv preprint arXiv:2410.10190*, 2024.

681 Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial bayesian opti-  
 682 mization using the graph cartesian product. *Advances in Neural Information Processing Systems*,  
 683 32, 2019.

684 Michael Oliver and Guan Wang. Crafting efficient fine-tuning strategies for large language models.  
 685 *arXiv preprint arXiv:2407.13906*, 2024.

686 Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan  
 687 Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,  
 688 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin  
 689 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,  
 690 Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang,  
 691 Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

692 Hossein Rajabzadeh, Mojtaba Valipour, Tianshu Zhu, Marzieh Tahaei, Hyock Ju Kwon, Ali Ghodsi,  
 693 Boxing Chen, and Mehdi Rezagholizadeh. Qdylora: Quantized dynamic low-rank adaptation for  
 694 efficient large language model tuning. *arXiv preprint arXiv:2402.10462*, 2024.

702 Mayk Caldas Ramos, Shane S Michtavy, Marc D Porosoff, and Andrew D White. Bayesian opti-  
 703 mization of catalysts with in-context learning. *arXiv preprint arXiv:2304.05341*, 2023.  
 704

705 Bojana Ranković and Philippe Schwaller. Bochemian: Large language model embeddings for  
 706 bayesian optimization of chemical reactions. In *NeurIPS 2023 Workshop on Adaptive Experi-  
 707 mental Design and Active Learning in the Real World*, 2023.

708 Bojana Ranković and Philippe Schwaller. Gollum: Gaussian process optimized llms–reframing llm  
 709 finetuning through bayesian optimization. *arXiv preprint arXiv:2504.06265*, 2025.  
 710

711 Bojana Ranković, Ryan-Rhys Griffiths, Henry B Moss, and Philippe Schwaller. Bayesian optimisa-  
 712 tion for additive screening and yield improvements–beyond one-hot encoding. *Digital Discovery*,  
 713 3(4):654–666, 2024.

714 Ayan Sengupta, Vaibhav Seth, Arinjay Pathak, Aastha Verma, Natraj Raman, Sriram Gopalakrish-  
 715 nan, Niladri Chatterjee, and Tanmoy Chakraborty. Robust and efficient fine-tuning of llms with  
 716 bayesian reparameterization of low-rank adaptation. *arXiv preprint arXiv:2411.04358*, 2024.  
 717

718 Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the  
 719 human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):  
 720 148–175, 2015.

721 N Shawki, R Rodriguez Nunez, I Obeid, and J Picone. On automating hyperparameter optimiza-  
 722 tion for deep learning applications. In *2021 IEEE Signal Processing in Medicine and Biology  
 723 Symposium (SPMB)*, pp. 1–7. IEEE, 2021.

724 Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine  
 725 learning algorithms. *Advances in neural information processing systems*, 25, 2012.

726 Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving lora in privacy-preserving federated  
 727 learning. *arXiv preprint arXiv:2403.12313*, 2024.

728 Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashti, Surya Bhupatiraju, Shreya  
 729 Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open  
 730 models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

731 Qwen Team. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

732 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée  
 733 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and  
 734 efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

735 Christophe Tribes, Sacha Benarroch-Lelong, Peng Lu, and Ivan Kobyzev. Hyperparameter opti-  
 736 mization for large language model instruction-tuning, 2024. URL <https://arxiv.org/abs/2312.00949>.  
 737

738 unsloth. Lora hyperparameters guide. <https://docs.unsloth.ai/get-started/fine-tuning-llms-guide/lora-hyperparameters-guide>, 2025. Accessed:  
 739 2025-09-18.  
 740

741 Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter effi-  
 742 cient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint  
 743 arXiv:2210.07558*, 2022.

744 Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Lora-pro: Are low-rank adapters  
 745 properly optimized? *arXiv preprint arXiv:2407.18242*, 2024.

746 Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning.  
 747 In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016.

748 Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter  
 749 optimization for machine learning models based on bayesian optimization. *Journal of Electronic  
 750 Science and Technology*, 17(1):26–40, 2019.

756 Minghao Yan, Zhuang Wang, Zhen Jia, Shivaram Venkataraman, and Yida Wang. Plora: Efficient  
757 lora hyperparameter tuning for large models. *arXiv preprint arXiv:2508.02932*, 2025.  
758

759 Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhen-  
760 guo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions  
761 for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

762 Michael R Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. Using large lan-  
763 guage models for hyperparameter optimization. *arXiv preprint arXiv:2312.04528*, 2023.

764

765 Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. Autolora: Automatically tun-  
766 ing matrix ranks in low-rank adaptation based on meta learning. *arXiv preprint arXiv:2403.09113*,  
767 2024.

768 Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhui Chen, and  
769 Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement.  
770 *arXiv preprint arXiv:2402.14658*, 2024.

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

## APPENDIX

A OPTIMIZING  $\omega, \theta, \psi$  THROUGH MARGINAL LOG-LIKELIHOOD

This part is inspired by Wilson et al. (2016) and Ranković & Schwaller (2025), from which we partially adopt several equations. We formulate *marginal log-likelihood* as follows:

$$\mathcal{L}(\Phi) = \log p(\mathbf{y}|\mathbf{X}, \Phi) = -\frac{1}{2}\{(\mathbf{y} - \mu\mathbf{1})^\top \mathbf{K}_\Phi^{-1}(\mathbf{y} - \mu\mathbf{1}) + \log |\mathbf{K}_\Phi| + n \log 2\pi\}, \quad (7)$$

where  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  and  $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$ .

To maximize *marginal log-likelihood*, gradient-based optimization is used to optimize kernel hyperparameter  $\omega$ , weight of projection layer  $\theta$ , and learnable token  $\psi$ . We define the parameter set  $\Phi = \{\omega, \theta, \psi\}$ . The gradient of the *marginal log-likelihood* can be computed by applying the chain rule with respect to each parameter, resulting in the following decomposition:

$$\frac{\partial \mathcal{L}}{\partial \omega} = \frac{\partial \mathcal{L}}{\partial K_\Phi} \frac{\partial K_\Phi}{\partial \omega}, \quad \frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial K_\Phi} \frac{\partial K_\Phi}{\partial g(\mathbf{x}; \theta, \psi)} \frac{\partial g(\mathbf{x}; \theta, \psi)}{\partial \theta}, \quad \frac{\partial \mathcal{L}}{\partial \psi} = \frac{\partial \mathcal{L}}{\partial K_\Phi} \frac{\partial K_\Phi}{\partial g(\mathbf{x}; \theta, \psi)} \frac{\partial g(\mathbf{x}; \theta, \psi)}{\partial \psi}, \quad (8)$$

$$\frac{\partial \mathcal{L}}{\partial K_\Phi} = \frac{1}{2} K_\Phi^{-1}(\mathbf{y} - \mu\mathbf{1})(\mathbf{y} - \mu\mathbf{1})^\top K_\Phi^{-1} - \frac{1}{2} K_\Phi^{-1}, \quad (9)$$

where  $\frac{\partial K_\Phi}{\partial \omega}$  are the derivatives of the kernel with respect to the kernel hyperparameters,  $\frac{\partial K_\Phi}{\partial g(\mathbf{x}; \theta, \psi)}$  means the implicit derivatives of the kernel with respect to the  $g$ .  $\frac{\partial g(\mathbf{x}; \theta, \psi)}{\partial \theta}$  are the derivatives of the projection layer parameters via backpropagation and  $\frac{\partial g(\mathbf{x}; \theta, \psi)}{\partial \psi}$  are the derivatives of the learnable token parameters via backpropagation. Finally, we can compute the gradient of *marginal log-likelihood* by applying the chain rule.

## B DETAILS OF THE EXPERIMENTAL SETTING

**Implementation details.** Motivated by Ranković & Schwaller (2025), we use Qwen2-7B as the LLM in our framework to extract embeddings for BO, applying last-token pooling. The embedding dimension is set to 3584. We define the projection layer as follows:

$$P(\mathbf{x}; \theta) = \text{ELU}(\text{Dropout}(W\mathbf{x} + b)). \quad (10)$$

We also utilize Matérn-5/2 kernel and Expected Improvement (EI) as acquisition function based on previous studies (Ranković et al., 2024; Ranković & Schwaller, 2025). The backbone LLM used for experiments on LoRA variants is LLaMA2-7B. For cases where our method is not applied, we follow the settings reported in prior work (Meng et al., 2024; Wang et al., 2024). We set the model’s sequence length to 1024 and use a warmup ratio of 0.03 with a cosine learning rate scheduler. To reduce computational cost, all models are trained for a single epoch.

**Details on competing methods.** We conduct our experiments on several HPO methods, random search (Bergstra & Bengio, 2012), Optuna (Akiba et al., 2019), standard BO (Oliver & Wang, 2024), latent BO (LBO) (Li et al., 2021), and NOMAD (Tribes et al., 2024). To ensure fairness, all methods are constrained to 30 optimization iterations. For each method, the top-1 result is obtained by selecting the best hyperparameter configuration on the training subset. We use the BoTorch library (Balandat et al., 2020) for BO and LBO implementations, conducting hyperparameter search with the same design space as ours. For both BO and LBO, each hyperparameter configuration is represented as a 5-dimensional vector and fed into the baselines. For LBO, we adapt the feature extractor proposed by Lee et al. (2025), which consists of two repeated blocks of linear and ReLU layers with a hidden dimension of 64. For Optuna, we use the default TPE setting with categorical hyperparameter candidates. For the method of Tribes et al. (2024), we run the NOMAD algorithm

864  
865  
866  
867 Table A1: **Hyperparameters for math reasoning tasks.** We present the hyperparameter configu-  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

Models	Strategy	Ours	Hyperparameter				
			Rank( $r$ )	Scaling Factor( $\alpha$ )	Dropout	Batch Size	Learning Rate
LLaMA2-7B	LoRA	✗	8	16	0.0	32	2e-05
		✓	256	8192	0.0	4	5e-06
	rsLoRA	✗	8	16	0.0	32	2e-05
		✓	128	1024	0.05	64	5e-05
	DoRA	✗	8	16	0.0	32	2e-05
		✓	16	16	0.3	16	5e-04
PiSSA	✗	128	128	0.0	128	2e-05	
		✓	256	4096	0.0	4	5e-06
LLaMA2-13B	LoRA	✗	8	16	0.0	32	2e-05
		✓	32	512	0.0	2	5e-05
Mistral-7B-v0.1	LoRA	✗	128	128	0.0	128	2e-05
		✓	128	128	0.1	4	3e-05
Gemma-7B	LoRA	✗	128	128	0.0	128	2e-05
		✓	64	256	0.0	2	5e-06
Qwen2.5-3B	LoRA	✓	1	4	0.25	32	5e-05
Qwen2.5-7B	LoRA	✓	32	64	0.25	16	5e-05
Qwen2.5-14B	LoRA	✓	1	4	0.25	32	2e-05

885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

Table A2: **Hyperparameters for code generation tasks.** We present the hyperparameter configu-  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

Models	Strategy	Ours	Hyperparameter				
			Rank( $r$ )	Scaling Factor( $\alpha$ )	Dropout	Batch Size	Learning Rate
LLaMA2-7B	LoRA	✗	8	16	0.0	32	2e-05
		✓	256	128	0.0	4	5e-05
	rsLoRA	✗	8	16	0.0	32	2e-05
		✓	256	128	0.25	4	5e-05
	DoRA	✗	8	16	0.0	32	2e-05
		✓	128	256	0.15	2	3e-05
PiSSA	✗	128	128	0.0	128	2e-05	
		✓	32	1024	0.0	2	3e-05
LLaMA2-13B	LoRA	✗	8	16	0.0	32	2e-05
		✓	256	128	0.25	2	1e-04
Mistral-7B-v0.1	LoRA	✗	128	128	0.0	128	2e-05
		✓	128	256	0.0	2	5e-06
Gemma-7B	LoRA	✗	128	128	0.0	128	2e-05
		✓	256	256	0.25	32	2e-05
Qwen2.5-3B	LoRA	✓	128	64	0.1	128	5e-06
Qwen2.5-7B	LoRA	✓	8	4	0.0	64	2e-05
Qwen2.5-14B	LoRA	✓	128	128	0.15	64	5e-06

907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

by executing our LoRA tuning Python script. All experiments are conducted on two A100-80GB

GPUs.

**Discovered hyperparameters for each experiment.** The hyperparameters discovered after optimiza-  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

918  
 919 **Table A3: Discovered hyperparameters by competing methods for math reasoning tasks.** We  
 920 present the hyperparameter configuration, obtained through the competing method, that is used to  
 921 train MetaMathQA for evaluating on the GSM8K and MATH datasets.

Search Method	Hyperparameter				
	Rank( $r$ )	Scaling Factor( $\alpha$ )	Dropout	Batch Size	Learning Rate
Random	128	1024	0.1	16	5e-05
Optuna (TPE)	256	128	0.0	32	5e-04
BO	16	64	0.25	2	1e-04
LBO	128	4096	0.0	2	5e-06
Tribes et al. (2024)	8	256	0.1	4	1e-04

922  
 923 **Table A4: Discovered hyperparameters by competing methods for code generation tasks.** We  
 924 present the hyperparameter configuration, obtained through the competing method, that is used to  
 925 train CodeFeedback for evaluating on the HumanEval and MBPP datasets.

Search Method	Hyperparameter				
	Rank( $r$ )	Scaling Factor( $\alpha$ )	Dropout	Batch Size	Learning Rate
Random	4	8	0.0	16	5e-05
Optuna (TPE)	256	128	0.2	16	1e-04
BO	16	8	0.15	2	5e-06
LBO	256	128	0.3	256	6e-04
Tribes et al. (2024)	4	64	0.0	4	3e-05

926  
 927 **Table A5: Discovered hyperparameters in ablation studies.** We present the hyperparameter con-  
 928 figuration during our ablation studies, used in Table 6.

Projection Layer	Domain-aware Prompting	Learnable Token	Hyperparameter				
			Rank( $r$ )	Scaling Factor( $\alpha$ )	Dropout	Batch Size	Learning Rate
✗	✗	✗	64	32	0.25	2	4e-04
✓	✗	✗	8	8	0.1	8	1e-04
✓	✓	✗	128	256	0.1	32	3e-04
✓	✓	✓	256	8192	0.0	4	5e-06

929  
 930 **Table A6: Performance across different model sizes.** Adapting our framework to different model  
 931 sizes consistently shows improvements, indicating its effectiveness.

Models	Ours	Accuracy (%)		Pass@1	
		GSM8K	MATH	HumanEval	MBPP
LLaMA2-7B	✗	41.47	5.24	16.31	35.47
	✓	62.93	12.88	30.49	42.59
LLaMA2-13B	✗	55.34	8.68	29.88	46.56
	✓	64.44	14.68	42.07	53.17

932  
 933 **Table A7: Performance differences across model sizes.** We apply the hyperparameters discovered  
 934 for each model size of LLaMA2 to fine-tune all model sizes. “Model” denotes the model being  
 935 fine-tuned, while “Settings” indicate the size of the model from which the hyperparameters were  
 936 obtained.

Model	Settings	Accuracy (%)		Pass@1	
		GSM8K	MATH	HumanEval	MBPP
LLaMA2-7B	7B	62.93	12.88	30.49	42.59
	13B	60.12	10.74	34.15	44.97
LLaMA2-13B	7B	66.57	15.24	42.68	51.59
	13B	64.44	14.68	42.07	53.17

972  
 973 **Table A8: Results of cross-applying hyperparameters across models.** We observe performance  
 974 degradation when hyperparameters discovered for one model series are applied to another. This  
 indicates that our framework effectively searches for hyperparameters suited to each model.  
 975

Model	Settings	GSM8K	MATH
LLaMA2-7B	LLaMA2-7B	62.93	12.88
	Qwen2.5-3B	39.5	5.2
	Qwen2.5-7B	32.68	4.7
	Qwen2.5-14B	52.46	8.06
Qwen2.5-7B	Qwen2.5-7B	83.93	48.08
	LLaMA2-7B	81.12	41.06
	LLaMA2-13B	80.06	40.18

985  
 986 **Table A9: Correlation between performance on a subset and the full dataset.** The percentages  
 987 indicate the sampling ratios from the full dataset. For TSDS (Liu et al., 2024d), we report results  
 988 separately when the target distribution is matched to the test dataset or the training dataset. Pearson  
 989 correlation is used as the evaluation metric.  
 990

Sampling Method	Math Reasoning		Code Generation	
	GSM8K	MATH	HumanEval	MBPP
Random (1%)	0.6879	0.4335	0.8052	0.5469
Random (5%)	0.8197	0.6483	0.8857	0.8879
Random (10%)	0.8566	0.6578	0.8652	0.9286
TSDS (10%) by Test dataset	0.8651	0.7117	0.8589	0.8209
TSDS (10%) by Train dataset	0.8529	0.6602	0.8624	0.9245

## 999 C ADDITIONAL RESULTS

1000 We provide supplementary experiments and analyses in addition to the main results presented in the  
 1001 main paper.  
 1002

1003 **Validation on models of different sizes.** Table A6 summarizes the validity of our framework under  
 1004 model size variations in LLaMA2. Even when the model size increases from 7B to 13B, our method  
 1005 successfully identifies appropriate hyperparameters, demonstrating the framework’s robustness to  
 1006 changes in scale.  
 1007

1008 **Cross-application of hyperparameters within the same model series.** We apply the same pro-  
 1009 cedure as in Table 8 to the LLaMA2 series, transferring hyperparameter settings discovered for one  
 1010 model size to another. The results in Table A7 show that hyperparameter configurations can re-  
 1011 main effective across different scales within the same series. This further suggests the possibility of  
 1012 searching for hyperparameters on smaller models and transferring them to larger ones.  
 1013

1014 **Cross-application of hyperparameters between different model series.** To examine whether  
 1015 hyperparameters identified in one model transfer to another, we conduct experiments applying con-  
 1016 figurations discovered on Qwen2.5 to LLaMA-2, and vice versa, as shown in Table A8. Apply-  
 1017 ing hyperparameters found on Qwen2.5 to LLaMA-2 leads to substantial performance degradation.  
 1018 Similarly, applying those from LLaMA-2 to Qwen2.5 also degrades performance. These results  
 1019 support the claim in Sec. 4.2 that preferred hyperparameter settings vary with model architecture.  
 1020

1021 **Correlation between subset training and full training.** Table A9 shows that correlation between  
 1022 subset training and full training remains consistent across all benchmarks. Notably, randomly sam-  
 1023 pling only 10% of the data still yields high correlation with full-dataset performance. This supports  
 1024 the claim in Sec. 4.2 that random sampling is a reasonable and efficient choice, comparable to more  
 1025 sophisticated data selection methods (Liu et al., 2024d). Thus, instead of tuning on the full dataset,  
 leveraging proxy training evaluation provides a reliable proxy for estimating model performance  
 during hyperparameter search.  
 1026

1026 Table A10: **Prompt templates with and without domain-aware prompting.**  
1027

1028	rank( $r$ )={rank_value}, Scaling factor( $\alpha$ )={alpha_value}, Dropout
1029	Rate={dropout_value}, Batch Size={batchsize_value}, Learning
1030	Rate={lr_value}
1031	(a) Prompt templates without domain-aware prompting
1032	
1033	* <b>Rank (<math>r</math>):</b> Controls adapter capacity by setting the low-rank
1034	dimension, higher $r$ increases expressivity (and memory/compute)
1035	but raises overfitting risk. If you raise $r$ , consider stronger
1036	regularization or a lower learning rate.
1037	* <b>Scaling factor (<math>\alpha</math>):</b> Scales the LoRA update; the effective
1038	update magnitude is $\alpha / r$ , so setting $\alpha \approx r$ keeps
1039	update strength stable. Larger $\alpha$ amplifies adaptation but can
1040	destabilize training if LR is high.
1041	* <b>Dropout:</b> Probability of dropping the adapter path to regularize
1042	training; higher dropout curbs overfitting, especially with large $r$
1043	or small datasets. With higher dropout you can often afford slightly
1044	larger $\alpha$ or LR without instability.
1045	* <b>Batch size:</b> Number of samples per optimizer step larger batches
1046	give smoother gradients and typically permit a proportionally larger
1047	learning rate (linear-scaling rule) at the cost of more memory.
1048	Small batches may need gradient accumulation or a reduced LR.
1049	* <b>Learning rate:</b> Step size for adapter parameters too high can
1050	diverge (especially with large $\alpha/r$ ), too low slows convergence.
1051	Tune in conjunction with batch size and consider schedules (e.g.,
1052	cosine) to balance speed and stability.
1053	* rank( $r$ ): {rank_value}
1054	* Scaling factor( $\alpha$ ): {alpha_value}
1055	* Dropout Rate: {dropout_value}
1056	* Batch Size: {batchsize_value}
1057	* Learning Rate: {lr_value}

1053 (b) Prompt templates with domain-aware prompting

1054 

## D TEMPLATE FOR DOMAIN-AWARE PROMPTING

1055 We provide an example of the template for domain-aware prompting in Table A10. This template  
 1056 focuses on the roles and relationships of each hyperparameter and describes how training dynamics  
 1057 change as their values vary, based on prior studies (Kalajdzievski, 2023; Sun et al., 2024; Diehl,  
 1058 2024; Meng et al., 2024; unsloth, 2025; Liu et al., 2025). Compared to a template without domain-  
 1059 aware prompting, this design captures rich domain knowledge about LoRA hyperparameters, signif-  
 1060 icantly improving the effectiveness of the following Bayesian Optimization. The template can also  
 1061 be modified by users if needed.

1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079