Efficient Hyperparameter Optimization for LLM Reinforcement Learning

Minping Chen[1,*], **Bowen Xiao**[1,*], **Du Liang**[3], **Chuxuan Zeng**[3], **Zeyi Wen**[1,2,†]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]The Hong Kong University of Science and Technology
[3]China United Network Communications Group
[1]{mchen779, bxiao012}@connect.hkust-gz.edu.cn, [1,2]wenzeyi@hkust-gz.edu.cn
[3]{dul28, zengchuxuan}@chinaunicom.cn

Abstract

Hyperparameters are critical to LLM reinforcement learning (RL), but existing hyperparameter optimization (HPO) methods remain inefficient in this area, due to the massive model scale and resource-intensive training cycles. In this paper, we propose Joint Fidelity Hyperparameter Optimization (JF-HPO), which simultaneously adapts both model size and training budget as fidelity. JF-HPO is empowered by: (i) a small proxy model of the target LLM for efficient training and evaluation in each HPO trial; (ii) several carefully designed early-stopping strategies based on training dynamics; (iii) an efficient checkpointing mechanism to eliminate redundant computations. JF-HPO significantly improves the computational efficiency of each trial (up to 14.9×) compared with existing HPO methods, thus achieving better predictive accuracy in most cases under the same time budget. Notably, JF-HPO delivers performance improvements ranging from 5.8% to 111.6% over VeRL Recipe.

1 Introduction

The emergence of large language models (LLMs) has revolutionized a broad spectrum of natural language understanding and generation tasks (Minaee et al., 2024). Recent models such as OpenAI-o1 (OpenAI, 2024) and DeepSeek-R1 (Guo et al., 2025) have demonstrated remarkable reasoning capabilities, largely attributed to the advances in reinforcement learning (RL) techniques. In particular, Reinforcement Learning from Human Feedback (RLHF) (Bai et al., 2022; Ouyang et al., 2022) and Reinforcement Learning with Verifiable Reward (RLVR) (Gao et al., 2024; Guo et al., 2025; Team et al., 2025) have been instrumental in aligning model outputs with human preferences and enhancing logical reasoning through rule-based rewards.

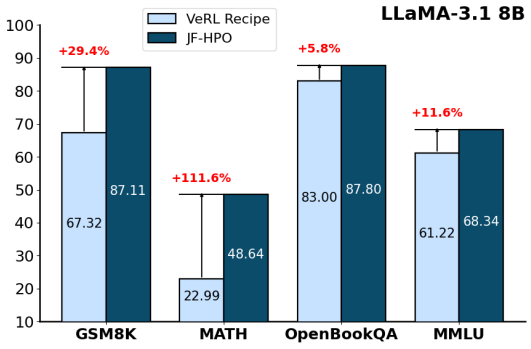


Figure 1: Performance improvements of our JF-HPO method across various tasks.

Despite the success of RLHF and RLVR, RL-based training for LLMs remains sensitive to hyperparameter configurations (Eimer et al., 2023). Small variations in hyperparameters such as learning rate and clipping ratio can lead to significant differences in final model performance and stability (see Figure 3 for more details). Hyperparameter optimization (HPO) thus becomes effective for LLM RL. Figure 1 (and Figure 5) presents the performance comparison between using the recommended hyperparameters from the VeRL Recipe (Sheng et al., 2025) and our HPO method, which clearly demonstrates the effectiveness of HPO. However, HPO is extremely time-consuming for LLM RL, since it involves multiple rounds of training and evaluation. While existing multi-fidelity HPO methods like Successive Halving (Jamieson and Talwalkar, 2016) and BOHB (Falkner et al., 2018) allow early termination of poorly performing configurations, they are still inefficient in the contexts of LLM RL due to the massive model size and high computation cost.

To address the inefficiency issue of existing HPO methods for LLM RL, we propose Joint Fidelity Hyperparameter Optimization (JF-HPO), a novel HPO framework that simultaneously adapts both

[*] Equal contribution
[†] Corresponding author

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27540–27552
July 2-7, 2026

model size and training budget as fidelity within a unified Bayesian optimization procedure. Since training LLM is computation-intensive and HPO involves multiple rounds of model training, JF-HPO leverages a small proxy model of the target large model for efficient training and evaluation. Furthermore, JF-HPO incorporates intelligent early-stopping strategies based on training dynamics, e.g., reward and KL divergence, and introduces an efficient checkpointing mechanism to avoid redundant computation. With these innovations, JF-HPO significantly reduces the computational cost of HPO in the LLM RL scenario, thus outperforming existing HPO methods in most cases under the same time budget. The main contributions of this paper are summarized as follows:

- We propose JF-HPO, a novel HPO method that concurrently adapts both model size and training budget as fidelity. By exploiting a small proxy model for the target large model, JF-HPO reduces remarkable computation cost associated with model training and evaluation in each optimization trial.
- To further accelerate the HPO process, we design several early-stopping strategies that allow timely termination of underperforming trials. Furthermore, we introduce an efficient checkpointing mechanism to avoid redundant computation for multi-fidelity HPO.
- We evaluate JF-HPO across diverse tasks, demonstrating its superior efficiency (up to 14.9× faster per trial) and effectiveness—it outperforms existing HPO methods in most cases (22 out of 24 runs of experiments) or match their performance under the same time budget, and achieves performance gains of 5.8% to 111.6% over VeRL Recipe.

2 Preliminary

In this section, we introduce the preliminaries of reinforcement learning algorithms. In this work, we use Group Relative Policy Optimization (GRPO), which is proposed by DeepSeek (Shao et al., 2024), to demonstrate the effectiveness of our JF-HPO method. Before introducing GRPO, we first present the background of Proximal Policy Optimization (PPO) (Schulman et al., 2017) for completeness.

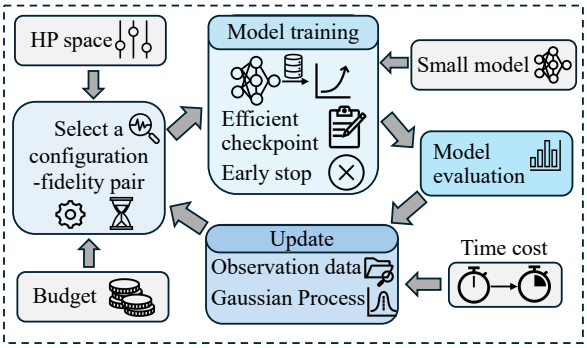


Figure 2: Overview of our joint fidelity hyperparameter optimization (JF-HPO) method.

2.1 Proximal Policy Optimization

Proximal Policy Optimization (PPO) (Schulman et al., 2017) proposes a clipped surrogate objective for optimizing policies. By restricting policy updates to a proximal region of the previous policy through clipping, PPO enhances training stability and boosts sample efficiency of RL. In more detail, PPO maximizes the following objective to update the policy model during training:

$$\mathcal{J}_{\text{PPO}}(\theta) = \mathbb{E}_{(p,o)\sim\mathcal{D},o_{\le t}\sim\pi_{\theta,\text{old}}(\cdot|p)} \left[\min\left(\frac{\pi_\theta(o_t \mid p, o_{<t})}{\pi_{\theta,\text{old}}(o_t \mid p, o_{<t})} \hat{A}_t, \right.\right. \left.\left. \text{clip}\left(\frac{\pi_\theta(o_t \mid p, o_{<t})}{\pi_{\theta,\text{old}}(o_t \mid p, o_{<t})}, 1-\varepsilon, 1+\varepsilon \right) \hat{A}_t \right) \right],$$

where π_θ is the policy model, $\pi_{\theta,\text{old}}$ is the old policy model, (p, o) is a prompt-output pair sampled from dataset $\mathcal{D}$ and old policy $\pi_{\theta,\text{old}}$, ε is the clipping range for stabilizing training, and $\hat{A}_t$ is the advantage estimation at time step t computed based on the generalized advantage estimation (GAE) (Schulman et al., 2015):

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l},$$

$$\delta_t = R_t + \gamma V(s_{t+1}) - V(s_t), 0 \le \gamma, \lambda \le 1,$$

where l is the offset of time step t, γ is a discount factor in range $[0, 1]$, λ is a hyperparameter in GAE, R_t and s_t are the reward and state at time step t, and V is the value function.

2.2 Group Relative Policy Optimization

The value function in PPO is typically a model that has a comparable size to the policy model. This leads to substantial memory consumption and computation overhead. Furthermore, the use of

the value function is not necessary in the LLM context. To address these issues, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) eliminates the value function and computes the advantage based on the average reward of a group of sampled outputs $\{o_i\}_{i=1}^{G}$ for the same prompt p:

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{R_i\}_{i=1}^{G})}{\text{std}(\{R_i\}_{i=1}^{G})},$$

where $\hat{A}_{i,t}$ is the advantage of the i-th output. Then the objective for GRPO is:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(p,o)\sim\mathcal{D},\{o_i\}_{i=1}^{G}\sim\pi_{\theta,\text{old}}(\cdot|p)} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left(\min\Big(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}\big(r_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\big)\hat{A}_{i,t}\Big) - \beta D_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\right)\right],$$

where $r_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t}|p,o_{i,<t})}{\pi_{\theta,\text{old}}(o_{i,t}|p,o_{i,<t})}$, π_{ref} is the reference model, D_{KL} is KL divergence (Kullback and Leibler, 1951) function, and β is a hyperparameter.

3 Methodology

In this section, we first provide the problem formulation and an overview of our method, then we elaborate on the technical details of our method.

3.1 Problem Formulation

Given an RL algorithm f, we use $f(\theta)$ to represent the model performance (e.g., accuracy), where θ denotes the model parameters. Then the objective of HPO for RL can be formulated as solving the following optimization problem, aiming to find the optimal hyperparameter configuration, subject to a total time budget:

$$\phi^* = \arg\max_{\phi\in\Phi} f(\theta,\phi), \text{s.t.} \sum_{i=1}^{N} C(\theta,\phi_i,r_i) < \mathcal{B}$$

where Φ is the hyperparameter space, $C(\theta,\phi_i,m_i)$ denotes the time cost for the i-th HPO trial at the fidelity r_i, N is the total number of HPO trials, $\mathcal{B}$ is a time budget and ϕ^* is the optimal configuration.

Bayesian Optimization (BO) for HPO BO is a classical HPO method, which typically has two key components: a *surrogate model* $\mathcal{M}$ to approximate the objective function $f(\theta)$, i.e., $\mathcal{M}$ is a prior of $f(\theta)$, and an *acquisition function* α to obtain a promising configuration from the hyperparameter space. $\mathcal{M}$ and α are typically a Gaussian Process (GP) and Expected Improvement (EI) (Močkus, 1974), respectively. BO first initializes an observation set $\mathcal{D} = \{(\phi_i, y_i)\}_{i=1}^{t}$, where y_i denotes the model performance, e.g., accuracy, and performs the following interactive steps: (i) maximize the acquisition function to select a configuration ϕ_{t+1}; (ii) evaluate ϕ_{t+1} to obtain y_{t+1} and add (ϕ_{t+1}, y_{t+1}) to the observation set $\mathcal{D}$; (iii) update the surrogate model with $\mathcal{D}$.

3.2 Overview

Before elaborating on the technical details of our Joint Fidelity Hyperparameter Optimization (JF-HPO) method, we provide an overview first. As presented in Figure 2, given the total time budget and hyperparameter space, JF-HPO iteratively performs four key steps: (i) It begins with selecting a configuration-fidelity pair by maximizing the acquisition function. (ii) The selected configuration-fidelity pair is then utilized for model training with a small proxy model, which significantly improves the training efficiency compared with using the target large model. Additionally, a checkpoint mechanism and several early-stopping strategies are designed to further enhance training efficiency. (iii) After model training, evaluation is performed to obtain the model performance, e.g., accuracy. (iv) The observation data and the Gaussian Process are updated. We summarize the overall procedure in Algorithm 1. Next, we introduce JF-HPO in detail.

3.3 Joint Fidelity for HPO

In the vanilla BO, the Expected Improvement (EI) acquisition function is defined as $\alpha = \mathbb{E}[f(\theta,\phi) - f^*(\theta,\phi^+)|\mathcal{D}]$, where $f^*(\theta,\phi^+)$ is the incumbent best model performance evaluated with configuration ϕ^+. In this scenario, the model performs training until reaching the given number of steps/epochs, which is inefficient, especially for large language models. To address this issue, we extend the vanilla BO with a multi-fidelity strategy that is a joint adaptation of model size and training budget. Multi-fidelity optimization is a widely used technique in HPO, offering an efficient and effective approximation by focusing on obtaining performance rankings across different hyperparameters, rather than requiring their exact performance metrics. Model size is one kind of fidelity in our method, by using a small proxy model to efficiently evaluate the rankings of different hyperparameters. To ensure the transfer of hyperparameters, the proxy model is from the same model family and shares the same architecture as the target model.

Algorithm 1 Joint Fidelity HPO for LLM RL

Input: Hyperparameter space Φ, small proxy model $\theta^{'}$, total budget $\mathcal{B}$, max fidelity $r_{\max}$, thresholds τ_1, τ_2
Output: Optimal configuration ϕ^*

Initialize GP with observation data $\mathcal{D} \leftarrow \emptyset$, checkpoint registry $\mathcal{R} \leftarrow \emptyset$, $f^* \leftarrow -\infty$
while $\sum_i C(\theta', \phi_i, r_i) < \mathcal{B}$ **do**
  Select (ϕ_t, r_t) by maximizing the acquisition function (Eq. 1)
  if $(\phi_t, r'_t) \in \mathcal{R}$ for some $r'_t < r_t$ **then**
    Load checkpoint to resume training
  Initialize step counter $s \leftarrow 0$
  while $s < r_t$ **do**
    Train $\theta^{'}$ with configuration ϕ_t
    $s \leftarrow s + 1$
    if $\frac{\Delta \mathcal{L}_{\text{KL}}}{\mathcal{L}_{\text{KL}}} > \tau_1$ **or** $\frac{\Delta R_{\text{train}}}{R_{\text{train}}} > \tau_2$ **or** $R_{\text{train}} = 0$ for k continuous steps **then**
      Early stop
      Continue to next trial
  end while
  Save model checkpoint and update $\mathcal{R}$
  Evaluate $\theta^{'}$ to obtain $f'(\theta', \phi_t, r_t)$
  Update the observation data $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\phi_t, r_t, f'(\theta', \phi_t, r_t), C(\theta', \phi_t, r_t))\}$ (Eq. 2)
  if $f'(\theta', \phi_t, r_t) > f^*$ **then**
    $f^* \leftarrow f'(\theta', \phi_t, r_t)$, $\phi^* \leftarrow \phi_t$
  Update GP (Eq. 3) with $\mathcal{D}$
end while
Return ϕ^*

Let $r \in \{r_1, r_2, ..., r_{\max}\}$ be a set of fidelity, i.e., number of training steps, then our acquisition function α can be defined as:

$$\alpha(\phi_t, r_t) = \frac{\mathbb{E}[f^{'}(\theta^{'}, \phi_t, r_t) - f^*(\theta, \phi^+, r_{\max}) | \mathcal{D}]}{\mathbb{E}[C(\theta^{'}, \phi_t, r_t)]}, \quad (1)$$

where $f^*(\theta, \phi^+, r_{\max})$ denotes the incumbent best model performance evaluated at the highest fidelity $r_{\max}$ using the full model size and hyperparameter configuration ϕ^+, $f^{'}(\theta^{'}, \phi_t, r_t)$ denotes the model performance of a significantly smaller model $f^{'}(\theta^{'})$ at fidelity r_t, and $C(\theta^{'}, \phi_t, r_t)$ denote the time cost. We allocate the fidelity based on the Successive Halving technique (Jamieson and Talwalkar, 2016).

As introduced in Section 3.1, BO needs to update the surrogate model in its interactive optimization process. Generally, we use the newly collected observation data to update the surrogate model. In our method, this is customized as:

$$\mathcal{D} = \mathcal{D} \cup \{(\phi_t, r_t, f^{'}(\theta^{'}, \phi_t, r_t), C(\theta^{'}, \phi_t, r_t)\} \quad (2)$$

The surrogate mode is also a Gaussian Process (GP), defined as:

$$f(\theta, \phi, r) \sim \mathcal{GP}(m(\phi, r), k((\phi, r), (\phi^{'}, r^{'})), \quad (3)$$

where $m(\cdot)$ is the mean function and $k(\cdot)$ is the covariance kernel (e.g., Matérn, RBF).

Table 1: Recommended hyperparameters from the VeRL Recipe and the search space of HPO.

Hyperparameter	VeRL Recipe	Search Space
Learning rate (LR)	1e-6	(5e-8, 1e-5)
LR scheduler type	constant	[constant, cosine]
Actor clip ratio	0.2	(0.05, 0.3)
Gradient clip	1.0	(0.1, 10.0)
KL loss coefficient	0.001	(0.0005, 0.005)
#Rollout	3	[2, 3, 4, 5]

3.4 Early-stopping Strategies

To further improve the HPO efficiency of our method, we design the following effective early-stopping strategies to eliminate the underperforming configurations: (i) We early stop a trial if the increase ratio of the KL divergence exceeds a threshold τ_1, i.e., $\frac{\Delta \mathcal{L}_{\text{KL}}}{\mathcal{L}_{\text{KL}}} > \tau_1$, for k consecutive global training steps, where $\Delta \mathcal{L}_{\text{KL}}$ denotes the increase in KL divergence. The rationale behind this design choice is that the KL divergence helps constrain the policy model, preventing it from deviating excessively from the reference model. A rapidly increasing KL divergence typically indicates that the policy model is changing too quickly between updates. This can result in instability and entrapment in local optima during the training process. (ii) We also early stop a trial if the decrease ratio of the training reward surpasses a threshold τ_2, i.e., $\frac{\Delta R_{\text{train}}}{R_{\text{train}}} > \tau_2$, or the training rewards remain to zero, for k consecutive global training steps, where ΔR_{train} denotes the decrease in training reward. This strategy is based on the rationale that significant decreases in training rewards or stagnant rewards at zero for an extended period suggest that the current configuration is not conducive to learning a successful policy. To prevent promising configurations from being prematurely halted, we set strict thresholds for early stopping (see Section 4.1), resulting in insensitivity and reasonable early stopping.

3.5 Efficient Checkpointing

In addition to the early-stopping strategies, we also introduce an efficient checkpointing mechanism, which substantially reduces the computation costs for HPO. In multi-fidelity HPO methods, e.g., successive halving, the algorithm explores a broad range of configurations at the early stage using a limited budget. Promising configurations are then allocated larger budgets in subsequent rounds. This process requires evaluating the surviving configu-

Table 2: Performance comparison.

Model	Method	GSM8K	MATH	OpenBookQA	MMLU	Average
LLaMA-3.1 8B	VeRL Recipe	67.32	22.99	83.00	61.22	58.63
	Random Search	78.39	30.97	85.60	64.23	64.80
	BOHB	86.66	48.62	85.80	66.08	71.79
	JF-HPO	**87.11**	**48.64**	**87.80**	**68.34**	**72.97**
Qwen-2.5 7B	VeRL Recipe	83.47	63.21	88.20	68.81	75.92
	Random Search	84.99	66.93	89.40	68.98	77.58
	BOHB	81.65	**70.29**	**91.00**	69.58	78.13
	JF-HPO	**88.17**	68.19	**91.00**	**71.23**	**79.65**
Qwen-3 14B	VeRL Recipe	93.03	70.21	90.60	70.92	81.19
	JF-HPO	**94.84**	**71.83**	**92.60**	**72.14**	**82.85**

rations multiple times with different budgets, each involving time-intensive model training. To address this issue, we design a registry-based checkpointing mechanism to minimize redundant computation. Specifically, we maintain a registry table for each trial, which stores its metadata, including hyperparameters, budget, training steps, and the path of the model checkpoints. If a configuration successfully survives the trial with a larger budget, we retrieve its previous model checkpoints from the registry table based on its metadata. Therefore, we do not need to train the model from the start, thereby reducing significant time cost.

3.6 Algorithm Outline

Algorithm 1 outlines our JF-HPO approach. The core optimization procedure iterates while the cumulative computational cost remains below the predefined budget $\mathcal{B}$. Each iteration comprises several distinct phases: The next configuration-fidelity pair (ϕ_t, r_t) is selected by maximizing an acquisition function (Eq. 1) (cf. Line 3). To minimize redundant computation, JF-HPO implements an intelligent checkpointing mechanism. If a partially trained model exists for configuration ϕ_t at a lower fidelity $r'_t < r_t$, training resumes from this checkpoint (Line 4-5). Then the small proxy model θ' undergoes training for $r_t - r'_t$ iterations, and three conditions may trigger early stop (Line 7-13). After training, the model checkpoint is saved, and the registry table $\mathcal{R}$ is updated (Line 14). Meanwhile, the algorithm evaluates the proxy model's performance $f'(\theta', \phi_t, r_t)$, stores the result in the observation set $\mathcal{D}$, updates the best configuration ϕ^* if applicable, and refits the GP surrogate model with the augmented dataset (Line 15-18). After exhausting the budget, the algorithm returns the optimal hyperparameter configuration ϕ^* (Line 20).

4 Experiments

In this section, we present extensive experimental results to validate the effectiveness of our proposed JF-HPO method. We first introduce the experimental setup, and then elaborate on the performance comparison on diverse tasks and in-depth analysis.

4.1 Experimental Setup

Datasets To evaluate the effectiveness of our method, we conduct experiments on various benchmarks: (i) GSM8k (Cobbe et al., 2021) contains 8.5k (7.47k for training and 1.32k for testing) high-quality linguistically diverse grade school math word problems and requires multi-step reasoning. (ii) MATH (Hendrycks et al.) consists of problems from mathematics competitions (7.5k for training and 5k for testing), and each problem has a full step-by-step solution. (iii) OpenBookQA (Mihaylov et al., 2018) is a question-answering task that requires multi-step reasoning, use of additional commonsense knowledge, and rich text comprehension. It contains about 5k training data and 500 testing data. (iv) MMLU (Hendrycks et al., 2021) is a massive multitask test (57 tasks) consisting of multiple-choice questions from various branches of knowledge. We randomly select 10,000 samples from the training set for model learning, and evaluate the model on the test set which has 14,042 samples. For all tasks, we report accuracy as the evaluation metric.

Models and Baselines We carry out experiments on multiple LLMs, including: Qwen-2.5-7B-Instruct (Yang et al., 2025), LLaMA-3.1-8B-Instruct (Touvron et al., 2023) and Qwen-3-14B (Qwen, 2025). We compare our approach with these baselines: **VeRL Recipe** which uses the recommended hyperparameters in the VeRL

Table 3: Ablation Study.

Method	Accuracy
JF-HPO	**88.17**
-w/o proxy model	86.88
-w/o checkpointing	84.84
-w/o early stopping	86.35

recipes for training. **Random Search** (Bergstra and Bengio, 2012) which explores the hyperparameter space by randomly sampling combinations, offering a more efficient alternative to grid search. **BOHB** (Falkner et al., 2018) is a hybrid algorithm that combines Bayesian optimization with the Hyperband resource allocation strategy to efficiently explore hyperparameter configurations.

Implementation Details We implement our approach based on the VeRL (Sheng et al., 2025) and SMAC3 (Lindauer et al., 2022) frameworks. Due to our limited training resources, we can only use a batch size of one or two per GPU in the experiments. Therefore, we do not search the batch size during HPO. Instead, we set the micro batch size to one or two and the global batch size to 256 using the gradient accumulation technique. To maximize the efficiency of HPO, we only set two discrete fidelities of model sizes: the proxy model and the target model. Therefore, BO uses the target large model only at the full fidelity and uses the small proxy model at all other lower fidelity levels. Nonetheless, the range of proxy model sizes can be expanded if needed. The proxy models are from the same series as the target model, and their sizes range from 0.5 billion to 1 billion. We use the best configuration from HPO to train the model for three epochs. Evaluation on the test set is performed after training. τ_1 and τ_2 are set to 15% and 10%, respectively, and k is set to 5. Hyperparameters from the VeRL recipe and the search space are presented in Table 1. We run our method and the HPO baselines under the same time budget, i.e., 48 hours, for fair comparison.

4.2 Main Results

In Table 2, we compare the performance across various tasks and models. Our method demonstrates consistently superior results compared to VeRL Recipe, and achieves better or competitive performance compared to the HPO baselines within the same time budget. For instance, our method

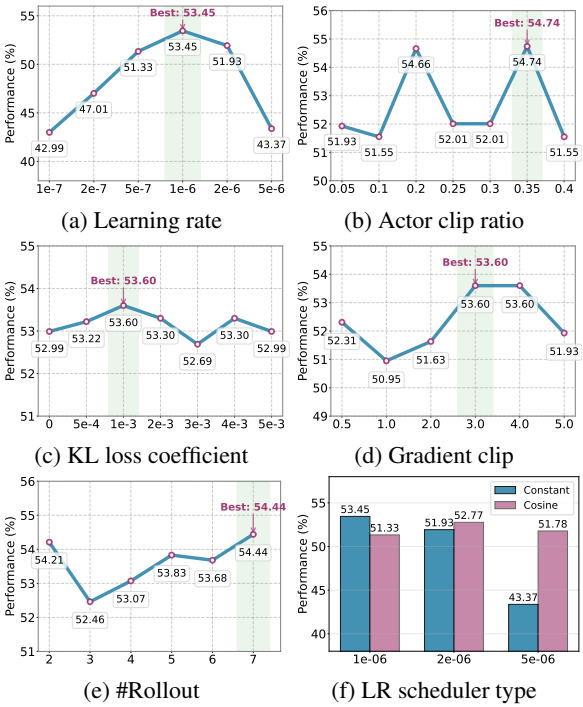


Figure 3: Effect of different hyperparameters on the final performance for the GSM8K task using Qwen-2.5 0.5B as the foundation model.

delivers 3.73%, 2.07%, and 1.52% performance improvements compared to VeRL Recipe, Random Search and BOHB, respectively. This advantage stems from our method's significant enhancement of computational efficiency during individual HPO trials, allowing for a greater number of trials—and consequently, more hyperparameter configurations—to be explored within the allotted time. As a result, our method identifies more effective hyperparameter configurations tailored to different tasks and models.

4.3 Ablation Study

To isolate the effect of each proposed module, we conduct an ablation analysis on the GSM8K task using Qwen-2.5 7B as the backbone. The results are as shown in Table 3, where the following variants are considered: (1) without using the proxy model in the multi-fidelity HPO (-w/o proxy model) (2) without using the efficient checkpointing mechanism (-w/o checkpointing), and (3) without using the early stopping strategies (-w/o early stopping). It is observed that removing any single module leads to a noticeable drop in performance, demonstrating the effectiveness of each module. Additionally, the checkpointing mechanism contributes most to the final performance, because it helps

Table 4: Efficiency comparison on the GSM8K task. BP denotes backward propagation. We use two GPUs for Qwen-2.5 7B and four GPUs for Random Search and BOHB on LLaMA-3.1 8B. Our JF-HPO method uses two GPUs when leveraging the small proxy model for training, and uses four GPUs only for the last trial of training on the large model with the best configuration.

Model	Method	Throughput (Tokens/s)			Avg. Time/Trial	Trial Speedup
		Overall	Rollout	BP & Update		
Qwen-2.5 7B	Random Search	521.6	1528.3	857.1	8.80 h	1.0 ×
	BOHB	521.6	1528.3	857.1	2.20 h	4.0 ×
	JF-HPO	8772.0	15624.0	24103.4	0.59 h	14.9 ×
LLaMA-3.1 8B	Random Search	864.9	1906.5	1712.4	5.38 h	1.0 ×
	BOHB	864.9	1906.5	1712.4	1.80 h	3.0 ×
	JF-HPO	7167.3	15103.8	21647.0	0.59 h	9.1 ×

reduce significant computation cost during HPO, enabling our method to explore more promising hyperparameters within the same time budget.

5 Discussion

5.1 Effect of Different Hyperparameters

To investigate the effect of different hyperparameters on the final model performance, we conduct a parameter sweep over a range of values for each hyperparameter in Table 1. It is important to note that in each experiment, we modify only one hyperparameter while keeping the others consistent with the settings in the VeRL Recipe. The results are shown in Figure 3. It can be observed that the learning rate is the most influential hyperparameter affecting model performance. As the learning rate increases up to 1e-6, the model performance improves significantly. However, when the learning rate exceeds 1e-6, the performance begins to decline. For most of the hyperparameters, we can observe clear performance gaps when varying their values, except for the KL loss coefficient. We also find that the transferability of hyperparameters from the proxy model to the target model depends on the degree of their sensitivity. For example, hyperparameters that have a smaller impact on the model performance, e.g., the KL loss coefficient, are more transferable than the influential hyperparameters, e.g., learning rate and actor clip ratio.

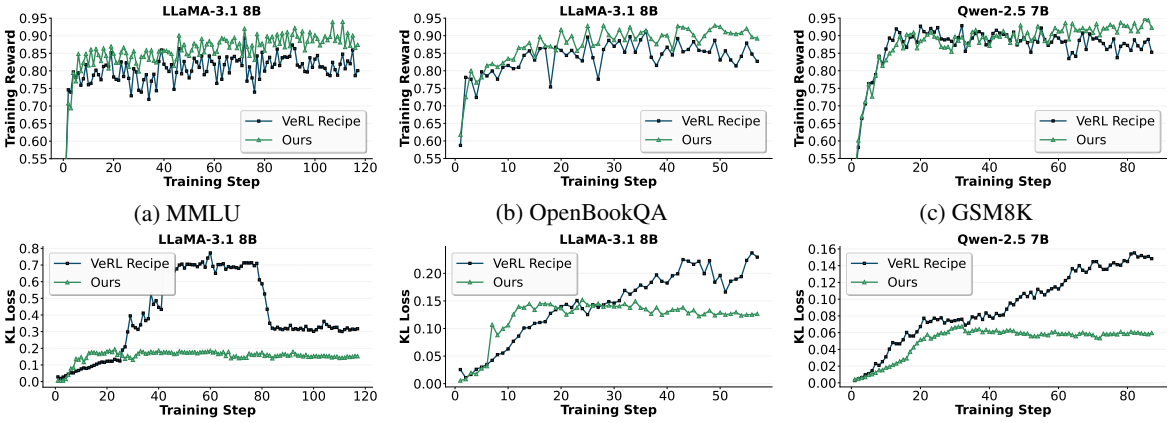
Additionally, the learning rate and its scheduling type jointly affect the model performance. As shown in Figure 3 (f), using a larger learning rate, such as 2e-6 or 5e-6, paired with a cosine scheduler, yields better performance compared to pairing it with a constant scheduler. This is because the cosine scheduler gradually increases then decreases the learning rate, allowing the model to make more refined adjustments as it approaches convergence. By contrast, a constant scheduler maintains the same learning rate throughout the training process, which may lead to overshooting or oscillations in the optimization path.

5.2 Efficiency Comparison

In Table 4, we present the HPO efficiency comparison on the GSM8K task with two different foundation models. Specifically, we compare these efficiency metrics: (i) overall throughput, throughput of rollout, i.e., generation of the outputs, backward propagation, and parameter update in RL; (ii) average time for a trial in HPO; (iii) speedup of a trial. Compared with Random Search and BOHB, our JF-HPO method achieves remarkable improvements in throughput, since it uses a significantly smaller model to search the hyperparameters. Note that the throughputs for Random Search and BOHB are identical, as they both use the target large model for training. JF-HPO also reduces substantial time costs for a trial, and thus it can explore more hyperparameter configurations under the same time budget as the baselines. Consequently, JF-HPO can achieve better final model performance.

5.3 Training Dynamics Comparison

As introduced in Section 3.4, we design several early stop strategies to improve the HPO efficiency based on the change of training reward and KL divergence loss. Here, we compare these two crucial training dynamics in RL to demonstrate why using the configuration obtained from our method performs better. We present the change of these dynamics during training, as illustrated in Figure 4. We employ LLaMA-3.1 8B and Qwen-2.5 7B as the foundational models and select three tasks for this comparison. Our method consistently achieves



(a) MMLU (b) OpenBookQA (c) GSM8K

(d) MMLU (e) OpenBookQA (f) GSM8K

Figure 4: Training reward and KL divergence loss comparison. See Figure 6 for more results.

Table 5: Performance comparison across five difficulty levels of the MATH task on Qwen-2.5 7B.

Method	Level-1	Level-2	Level-3	Level-4	Level-5
VeRL Recipe	91.08	79.98	72.50	58.98	38.80
JF-HPO	91.30	83.56	77.28	65.98	46.07
Δ (%)	**0.24**	**4.48**	**6.59**	**11.87**	**18.74**

overall higher training rewards and exhibits smaller increases in KL divergence loss on different tasks. This suggests that our method demonstrates more stable and effective RL training. Consequently, our method delivers superior evaluation performance across a range of tasks.

5.4 Comparison across Five Difficulty Levels

The MATH dataset contains five subsets of different difficulty levels, where the easiest problems for humans are assigned to "Level 1" and the hardest problems are assigned to "Level 5". Table 2 presents the overall performance of the MATH task. Here, we explore the performance comparison between VeRL Recipe and our method across five difficulty levels, as presented in Table 5. We observe a consistent decline in model performance as the difficulty level increases, with significant performance gaps between different levels. For Level 5, JF-HPO also cannot achieve satisfactory performance, since HPO does not improve the upper bound performance of the given RL algorithm. Nonetheless, JF-HPO consistently outperforms VeRL Recipe, and the performance improvements tend to increase as the difficulty level rises. This indicates that model performance on difficult samples is more sensitive to hyperparameters compared to simpler samples, highlighting that JF-HPO offers significant advantages for addressing challenging tasks.

5.5 Rank Correlation Analysis

In this work, we use model size as one kind of fidelity by exploiting a small proxy model to evaluate the rankings of different hyperparameters. Here, we provide a quantitative analysis of the Spearman/Kendall rank correlation between the proxy model and the target model across different hyperparameter settings. Specifically, we randomly sample five hyperparameter configurations (resulting in a total of 120 different rankings) and use them to train the proxy model and target model, respectively, on the GSM8K task. The results are presented in Table 6, suggesting a good consistency in performance rankings across different configurations between the proxy model and the target model. We also analyze some failure cases and discover that while larger learning rates, such as those exceeding 5e-6, can yield promising results on the small proxy model, they may cause overfitting for the larger target model.

Table 6: Spearman and Kendall rank correlation between the proxy model (Qwen-2.5 0.5B) and the target model (Qwen-2.5 7B) across hyperparameter settings.

#Configurations	Ranking space	Spearman	Kendall
5	120	0.90	0.80

6 Related Work

In this section, we provide an overview of reinforcement learning for large language models (LLMs), and discuss the existing hyperparameter optimization (HPO) methods.

6.1 Reinforcement Learning for LLMs

Reinforcement Learning (RL) (Sutton et al., 1998) plays an essential role in improving the ability of Large Language Models (LLMs) in instruction following and reasoning. Reinforcement Learning from Human Feedback (RLHF) (Bai et al., 2022; Ouyang et al., 2022) is a typical approach to align base models with human preferences using RL algorithms such as Proximal Policy Optimization (PPO) (Schulman et al., 2017). Recently, Large Reasoning Models (LRMs), e.g., DeepSeek-R1 (Guo et al., 2025), have shown the importance of RL in enhancing the reasoning capabilities of LLMs through rule-based rewards, i.e., reinforcement learning with verifiable reward (RLVR). RLVR is commonly used in mathematical reasoning tasks and multiple-choice problems where the reward function outputs a binary score based on whether the model's answer matches the ground truth (Guo et al., 2025; Team et al., 2025; Gao et al., 2024). This avoids the need for reward models, thus improving training efficiency. The success of RLVR is inseparable from advanced RL algorithms, including Group Relative Policy Optimization (GRPO) (Shao et al., 2024), Decouple Clip and Dynamic Sampling Policy Optimization (DAPO) (Yu et al., 2025), and REINFORCE++ (Hu, 2025), etc. Our method is applicable to the most popular LLM reinforcement learning algorithms, which typically adopt either the GRPO or PPO style and integrate with KL divergence loss and reward computation. In this work, we select GRPO, which is a strong RL algorithm and is used in DeepSeek, to demonstrate the effectiveness of JF-HPO for LLM RL.

6.2 Hyperparameter Optimization

Hyperparameter optimization (HPO) is critical in machine learning and deep learning, since hyperparameters have a significant impact on the model performance and convergence rate. Many HPO techniques have been proposed in the past years, including grid search and random search (Bergstra and Bengio, 2012), Bayesian optimization (BO) methods (Snoek et al., 2012; Victoria and Maragatham, 2021; Wu et al., 2019), multi-fidelity strategies (Falkner et al., 2018; Li et al., 2018; Jiang et al., 2024), and gradient-based methods (Maclaurin et al., 2015; Bohdal et al., 2021; Micaelli and Storkey, 2021), etc. For example, Successive Halving (SHA) (Micaelli and Storkey, 2021)begins by testing a group of configurations with limited resources, then advances only the top-performing half to proceed with double the resources. BOHB (Falkner et al., 2018) integrates BO and Hyperband (Li et al., 2018) to effectively manage the exploration-exploitation trade-off while dynamically allocating resources, resulting in enhanced efficiency and robustness compared to using either method alone.

Despite achieving notable advancements, existing HPO methods are suitable for small models. In the context of LLM RL, training is very time-consuming due to the large model size and the requirement of performing both token-by-token generation and backpropagation. As a result, traditional HPO methods involving multiple rounds of training and evaluation become impractical. In this work, we propose to leverage a significantly small proxy model for training and evaluation during HPO, alongside several efficient early-stopping strategies and a checkpointing mechanism to reduce substantial computation cost.

7 Conclusion

In this work, we propose JF-HPO that effectively democratizes the hyperparameter optimization of large language model reinforcement learning by breaking the computational barrier. Unlike existing HPO methods that rely solely on training budget allocation, our approach uniquely integrates proxy model adaptation with registry-based checkpointing and dynamics-aware early stopping. This joint-fidelity mechanism not only accelerates search efficiency by an order of magnitude but also reveals that complex reasoning tasks are particularly sensitive to hyperparameter configurations, necessitating the precise tuning our method provides.

We plan to extend the applicability of JF-HPO to other advanced RL algorithms such as DAPO (Yu et al., 2025). Additionally, we aim to theoretically investigate the correlation bounds between proxy and target models to further refine the fidelity selection process in extremely large-scale scenarios (e.g., 70B model).

Limitations

Our method currently depends on a consistent performance correlation between the proxy and target models to ensure valid hyperparameter transfer. In scenarios involving significant architectural shifts—for instance, transferring patterns from

dense proxies to sparse architectures (e.g., Mixture-of-Experts)—the validity of this correlation remains unexplored. Additionally, although our evaluation covers complex reasoning benchmarks, it does not explicitly account for open-ended generation tasks (e.g., creative writing), where reward signals are inherently more subjective and the optimal hyperparameter landscape may differ.

Acknowledgments

This work is supported by the Research and Development of Heterogeneous Computing Power Center Interconnection and Scheduling Software Stack under Grant No. YF202400000003.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305.

Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. 2021. Evograd: Efficient gradient-based meta-learning and hyperparameter optimization. *Advances in Neural Information Processing Systems*, 34:22234–22246.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Theresa Eimer, Marius Lindauer, and Roberta Raileanu. 2023. Hyperparameters in reinforcement learning and how to tune them. In *International Conference on Machine Learning*, pages 9104–9149. PMLR.

Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pages 1437–1446. PMLR.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. 2024. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Jian Hu. 2025. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*.

Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248. PMLR.

Jiantong Jiang, Zeyi Wen, Atif Mansoor, and Ajmal Mian. 2024. Efficient hyperparameter optimization with adaptive fidelity identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26181–26190.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The Annals of mathematical statistics*, 22(1):79–86.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13(9):9.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. 2022. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. 2015. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122. PMLR.

Paul Micaelli and Amos J Storkey. 2021. Gradient-based hyperparameter optimization over long horizons. *Advances in Neural Information Processing Systems*, 34:10798–10809.

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.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Jonas Močkus. 1974. On bayesian methods for seeking the extremum. In *IFIP Technical Conference on Optimization Techniques*, pages 400–404. Springer.

OpenAI. 2024. Learning to reason with llms.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Qwen. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25.

Richard S Sutton, Andrew G Barto, and 1 others. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *Preprint*, arxiv:2302.13971.

A Helen Victoria and Ganesh Maragatham. 2021. Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 12(1):217–223.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

A AI Usage Disclosure

We utilize generative AI exclusively for minor language refinement tasks, such as automating grammar corrections and enhancing linguistic clarity.

B Notation

In Table 7, we present a comprehensive summary of the notations and their corresponding definitions used throughout this paper, aimed at enhancing clarity and understanding.

C More Experimental Results

Performance Improvements Due to the limited space, we present additional results showcasing the performance improvements of our method across multiple tasks here, as illustrated in Figure 5. The improvements of JF-HPO compared with VeRL Recipe on Qwen-2.5 7B range from 3.2% to 7.9% on the GSM8K, MATH, OpenBookQA and MMLU tasks, which are all statistically significant gains. These results further demonstrate the robustness of our method across tasks of varying complexity.

Table 7: Summary of notations and their definitions in this paper.

Notation	Description
θ, θ'	Parameters of the policy model (θ) and small proxy model (θ')
π_θ, $\pi_{\theta,\text{old}}$, π_{ref}	Current, old, and reference policy models
(p, o), $\mathcal{D}$	Prompt-output pair and dataset
ε	Clipping range for stabilizing policy updates
$\hat{A}_t$, $\hat{A}_{i,t}$	Estimated advantage at timestep t (GAE) and for i-th output in GRPO
γ, λ	Discount factor and GAE hyperparameter
R_t, s_t, V	Reward and state at timestep t, and value function
β	Coefficient for KL divergence penalty
G	Group size in GRPO
ϕ, ϕ^*, Φ	Hyperparameter configuration, optimal configuration, and search space
$\mathcal{B}$, N, r	Total budget, number of trials, and fidelity level
$C(\theta, \phi, r)$, $f(\theta, \phi, r)$, $f'(\theta', \phi, r)$	Time cost, performance of full model, and performance of proxy model
f^*	Incumbent best performance
α, $\mathcal{M}$	Acquisition function and surrogate model (GP) in BO
$\mathcal{L}_{\text{KL}}$, $\Delta\mathcal{L}_{\text{KL}}$	KL divergence loss and its increase
R_{train}, ΔR_{train}	Training reward and its decrease
τ_1, τ_2, k	Early-stopping thresholds and consecutive steps condition
$\mathcal{R}$	Checkpoint registry table

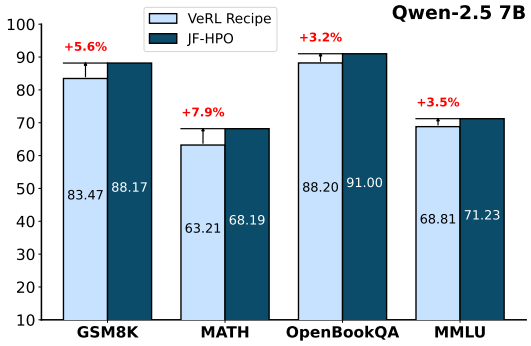


Figure 5: Performance improvements on Qwen-2.5 7B.

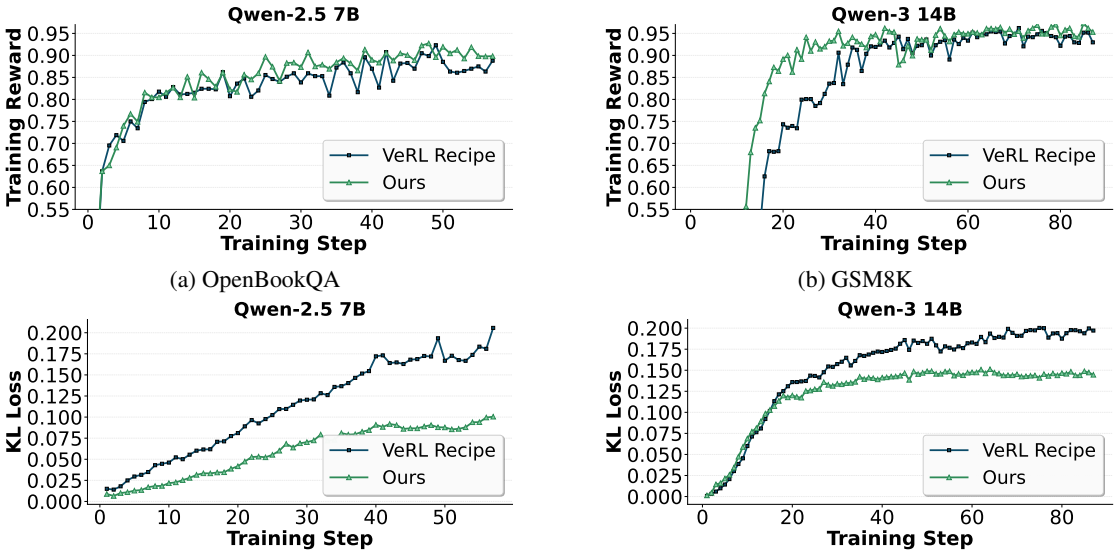
Training Dynamics Comparison Here, we provide additional results of training reward and KL divergence loss comparison on Qwen-2.5 7B and Qwen-3 14B, respectively, as shown in Figure 6. We showcase this comparison on two tasks, i.e., OpenBookQA and GSM8K. Similar to the trends depicted in Figure 4, JF-HPO consistently achieves an overall higher reward and a slower and more controlled increase of KL divergence loss, which is crucial for large language model reinforcement learning. This analysis (Figure 4 and Figure 6) comprehensively reveals the effectiveness and better stability of our method.

Out-of-Distribution Evaluation To investigate the generalization capability of our method, we use the model trained on the MATH task to evaluate on two out-of-distribution (OOD) challenged tasks, i.e., AMC 2023 (Li et al., 2024) and AIME 2025 (Zhang and Math-AI, 2025). The results are presented in Table 8. We conduct this evaluation on the Qwen-2.5 7B model. The AMC 2023 dataset comprises 83 problems, drawn from two rigorous mathematics competitions (AMC 12A and 12B) designed for students in grades 12 and below throughout the United States. The AIME 2025 dataset is a specialized benchmark, featuring 30 problems from the 2025 edition of the American Invitational Mathematics Examination (AIME) II. Note that the AMC 2023 and AIME 2025 tasks only contain the test set, and they lack a training set for reinforcement learning. Therefore, we do not use these two tasks in our main experiments, i.e., Table 2. Instead, we use them for the OOD evaluation.

From Table 8, compared with the baseline, our

Table 8: Out-of-distribution (OOD) performance comparison on Qwen-2.5 7B. The model is trained on the MATH task and evaluated on two OOD tasks.

Method	MATH	AMC 2023	AIME 2025
VeRL Recipe	63.21	27.71	0.0
JF-HPO	68.19	44.58	3.3
Δ (%)	**7.88**	**60.88**	$+\infty$



(a) OpenBookQA (b) GSM8K

(c) OpenBookQA (d) GSM8K

Figure 6: More results of training reward and KL divergence loss comparison.

Table 9: Performance comparison on different MMLU subsets using two foundation models.

Method	Humanities	STEM	Social Sciences	Other
LLaMA-3.1 8B				
VeRL Recipe	55.41	55.66	70.82	69.01
JF-HPO	59.94	58.96	76.08	74.28
Δ (%)	**8.18**	**5.93**	**7.42**	**7.64**
Qwen-2.5 7B				
VeRL Recipe	64.73	61.15	75.14	70.86
JF-HPO	67.30	61.66	79.95	74.60
Δ (%)	**3.97**	**0.83**	**6.40**	**5.28**

JF-HPO method achieves remarkable relative improvements on both the MATH task and the OOD tasks. This indicates our method learns a more robust model with better generalization capability. We also note the poor performance on the AIME 2025 task, since this task is very challenging.

Performance on Different MMLU Subsets The performance of the MMLU task in Table 2 is the average accuracy across 57 subjects. These subjects can be categorized into four groups (Hendrycks et al., 2021): *Humanities*, *STEM*, *Social Sciences*, and *Other*. To investigate whether our method can consistently achieve performance gains on different MMLU subsets, we present the performance comparison on different groups of MMLU subsets, as illustrated in Table 9. It can be observed that our method consistently surpasses VeRL Recipe on each group of MMLU subsets when using LLaMA-3.1 8B and Qwen-2.5 7B as the foundation models. Specifically, for LLaMA-3.1 8B, our method achieves relative performance improvements exceeding 5% across all subset groups. Similarly, for Qwen-2.5 7B, the relative performance improvements of our method are over 5% on the *Social Sciences* and *Other* groups. This analysis demonstrates that our method enhances model performance across various domains, showcasing its robustness and adaptability.