Hybrid-LoRA: Bridging Full Fine-Tuning and Low-Rank Adaptation for Post-Training

Chengqian Zhang[1] **Wei Zhu**[2] **Kyumin Lee**[1]
[1]Worcester Polytechnic Institute [2]University of Hong Kong
czhang12@wpi.edu wzhu91@connect.hku.hk kmlee@wpi.edu

Abstract

Post-training has become essential for adapting large language models (LLMs) to complex downstream behaviors, including instruction following, preference alignment, and multi-step reasoning. Reinforcement learning with verifiable rewards (RLVR) has recently emerged as a particularly effective post-training paradigm for improving reasoning capabilities, with critic-free algorithms such as GRPO and GSPO enabling scalable optimization. However, RLVR post-training with full fine-tuning (FFT) requires substantial GPU memory and incurs high training costs. Although parameter-efficient fine-tuning (PEFT) methods, such as Low-Rank Adaptation (LoRA), effectively reduce computational costs, they often suffer from a noticeable performance gap compared to full fine-tuning in post-training for complex reasoning tasks. In this paper, we propose Hybrid-LoRA, an efficient hybrid framework for RLVR post-training that selectively applies full fine-tuning to a small subset of modules less suited to low-rank adaptation, while adapting the remaining components with LoRA. We introduce a novel Hybrid-LoRA Score, computed in a short probing stage, that ranks candidate modules according to their sensitivity to low-rank adaptation; a single ranking yields the fully fine-tuned set for any parameter budget. On reasoning tasks with verifiable rewards, Hybrid-LoRA closely matches full fine-tuning performance under a 10% full fine-tuning module budget, with the remaining candidate modules adapted by LoRA. It outperforms a budget-matched LoRA, random and depth-based selections of the fully fine-tuned modules, and hybrid allocations built on the importance criteria of four existing PEFT methods, achieving improvements of up to 5.65% and on average 4.36% over the best of the latter.

1 Introduction

Large language models (LLMs) have demonstrated remarkable proficiency in complex reasoning tasks through post-training, including supervised fine-tuning (SFT) and reinforcement learning (RL)-based approaches such as reinforcement learning from human feedback (RLHF) and reinforcement learning with verifiable rewards (RLVR) (Ouyang et al., 2022; Shao et al., 2024). Recent critic-free RLVR algorithms, such as Group Relative Policy Optimization (GRPO) and Group Sequence Policy Optimization (GSPO), have become widely used for reasoning-oriented post-training, as they optimize policies using verifiable rewards without requiring a separate critic model. These methods have shown strong performance across downstream tasks, including mathematical reasoning, code generation, and scientific question answering. In the post-training stage of general-purpose large language models, existing RLHF and RLVR pipelines commonly rely on full fine-tuning (FFT), where all model parameters are updated, to adapt pre-trained or mid-trained foundation models for alignment and downstream tasks (Ouyang et al., 2022). However, FFT requires substantial computational resources and is often bottlenecked by GPU memory and training cost. Although parameter-efficient fine-tuning (PEFT) methods such as LoRA substantially reduce the cost of

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

adapting large foundation models (Houlsby et al., 2019; Hu et al., 2022), their low-rank update space can limit model expressiveness, leading to a performance gap compared to full fine-tuning.

Several studies have sought to improve the expressiveness and parameter allocation of LoRA-based adaptation. For example, AdaLoRA (Zhang et al., 2023) decomposes the LoRA update into singular-value components and dynamically allocates the parameter budget by pruning less important components during training. AdaLoRA treats every singular value in the middle matrix with its two corresponding vectors as a triplet, computes an importance score for each of the triplets by multiplying its sensitivity and uncertain scores, and then prunes the ranks with lower importance scores based on the computation budget. Another approach, AutoLoRA (Zhang et al., 2024) propose a bi-level optimization method to find the important rank among evenly distributed ranks in LoRA. AutoLoRA provides a contribution weight for each rank, and then update these weights based on the loss evaluated on validation set after every training step. The high contributing ranks (with higher weights) will be preserved and the model will be retrained when the ranks selection weights converge. Both approaches improve LoRA by allocating its limited rank budget more effectively across modules. However, they still operate entirely within the low-rank adaptation space, leaving a persistent gap from full fine-tuning.

With these insights in mind, we aim to bridge this gap by breaking the binary opposition between full fine-tuning and LoRA: rather than choosing one exclusively, we develop a hybrid fine-tuning framework that combines full-parameter updates with low-rank adaption. Our key hypothesis is that Transformer decoder modules are not equally amenable to low-rank adaptation. Some modules require full-parameter updates to preserve post-training performance, whereas others can be effectively adapted with LoRA. This idea introduces two major technical problems: (1) How can we efficiently estimate each module's suitability for low-rank adaptation and allocate full fine-tuning and PEFT under a fixed parameter budget? (2) How to achieve better performance of RL post-training in the hybrid fine-tuning framework with limited trainable parameters? To address these challenges, we develop a unified framework that integrates suitability estimation with stable RL optimization.

In this paper, our contributions are threefold:

- We propose **Hybrid-LoRA**, a hybrid parameter optimization framework for LLM post-training that selectively assigns Transformer modules to FFT while adapting the remaining modules with LoRA.
- We introduce **Hybrid-LoRA Score**, an efficient module-level evaluation metric based on first-order gradient-weighted sensitivity, to estimate each module's suitability for low-rank adaptation.
- We evaluate our framework across six complex reasoning tasks and various scales, demonstrating its effectiveness on a wide range of downstream benchmarks, including mathematical reasoning, scientific question answering, and code generation tasks. It outperforms hybrid allocations built on the importance criteria of four existing PEFT methods, achieving up to 5.65% and on average 4.36% relative improvement over the strongest of them, and it also outperforms a budget-matched LoRA as well as random and depth-based choices of the fully fine-tuned modules.

2 Related Works

Large Language Model Post-Training. The evolution of LLM post-training has progressed from supervised fine-tuning (SFT) to reinforcement learning from human feedback (RLHF), and more recently to reinforcement learning with verifiable rewards (RLVR). Proximal Policy Optimization (PPO) (Schulman et al., 2017) has been widely used in early RLHF pipelines, where an additional value model is trained to estimate token-level value functions and compute advantages. However, maintaining this critic introduces substantial memory and computational overhead during large-scale LLM post-training. To reduce this overhead, recent RLVR methods have adopted critic-free optimization. Shao et al. (2024) propose Group Relative Policy Optimization (GRPO), which samples multiple responses for each prompt and normalizes rewards within the group to estimate relative advantages without a separate critic. Yu et al. (2025) introduce DAPO, which improves large-scale long-CoT RL training with decoupled clipping, dynamic sampling, and other system-level and algorithmic refinements. Zheng et al. (2025) propose Group Sequence Policy Optimization (GSPO),

which replaces token-level importance ratios with sequence-level importance ratios and performs sequence-level clipping to improve training stability.

Parameter-Efficient Fine-Tuning and LoRA Variants. To reduce the substantial computational and memory burden of full-parameter adaptation, parameter-efficient fine-tuning (PEFT) methods, particularly LoRA (Hu et al., 2022) and its variants, have emerged as a practical alternative for adapting large language models. Several LoRA variants improve efficiency by automatically reallocating the limited rank budget across modules. AdaLoRA (Zhang et al., 2023) introduces a dynamic rank allocation framework that parameterizes LoRA updates with singular-value components and progressively prunes less sensitive components based on a sensitivity-based importance score. SoRA and AutoLoRA (Ding et al., 2023; Zhang et al., 2024) are closely related in their parameterization, as both decompose LoRA updates into sums of rank-one components formed by paired down- and up-projection vectors. SoRA introduces learnable gates over these rank-one components and optimizes them jointly with the LoRA parameters using proximal gradient updates under L_0 regularization, driving some gates exactly to zero so that the corresponding rank components can be pruned. AutoLoRA adopts a bi-level meta-learning framework, where continuous selection variables over rank-one components are optimized according to validation performance, and discrete layer-wise ranks are obtained by thresholding these variables. Different from the approaches above, where LoRA ranks are set to maximum and pruned to meet the target budget, ALoRA (Liu et al., 2024) initializes the rank with target budget uniformly distributed to all modules, calculates importance scores for each rank and reassigns pruned ranks to the modules where no ranks are pruned.

All the above approaches aim to improve LoRA by reallocating its limited rank budget across modules or rank components. However, they remain restricted to the low-rank adaptation space, and thus cannot address cases where the required task-specific update for a module is not well captured by a low-rank parameterization. This limitation is particularly relevant in RLVR post-training, where complex reasoning behaviors may require richer updates in certain Transformer modules. Therefore, rather than asking only how to allocate LoRA capacity, we ask which modules are sufficiently handled by LoRA and which modules require full-rank adaptation. Based on this perspective, we propose Hybrid-LoRA, a hybrid adaptation framework that heterogeneously assigns full fine-tuning or low-rank adaptation to different modules according to their suitability for LoRA adaptation.

3 The Proposed Hybrid-LoRA Framework

We present Hybrid-LoRA, a two-stage framework for efficient LLM post-training that first scores candidate Transformer linear modules through a short LoRA-based probing stage and then allocates a small full fine-tuning budget while adapting the remaining modules with LoRA. As shown in Figure 1, Hybrid-LoRA consists of two stages. In the probing stage, we attach LoRA modules to all candidate linear modules and run a short warm-up training phase. We then compute Hybrid-Score to rank modules by their gradient-weighted sensitivity. In the final training stage, we reinitialize the model, assign low-score modules to full fine-tuning under the parameter budget, and adapt the remaining modules with LoRA. We provide preliminaries on RLVR, GRPO, and GSPO in Appendix B.

3.1 Formulation: Hybrid Adaptation under a Trainable Parameter Budget

Given a Transformer-based LLM (Vaswani et al., 2017) with L layers, we consider the following set of candidate linear module types eligible for adaptation:

$$\mathcal{A} = \{W_q, W_k, W_v, W_o, W_{\text{gate}}, W_{\text{up}}, W_{\text{down}}\}. \tag{1}$$

We define the set of all candidate linear modules as

$$\mathcal{U} = \{1, \ldots, L\} \times \mathcal{A}. \tag{2}$$

Here, $(l, m) \in \mathcal{U}$ denotes the module of type m in layer l.

We aim to partition $\mathcal{U}$ into two disjoint subsets,

$$\mathcal{S}_{\text{fft}} \cup \mathcal{S}_{\text{lora}} = \mathcal{U}, \qquad \mathcal{S}_{\text{fft}} \cap \mathcal{S}_{\text{lora}} = \emptyset. \tag{3}$$

where $\mathcal{S}_{\text{fft}}$ and $\mathcal{S}_{\text{lora}}$ denote the modules assigned to FFT and LoRA adaptation, respectively. Let $P_{l,m}$ denote the number of trainable parameters in module (l, m) under FFT. Given an FFT budget

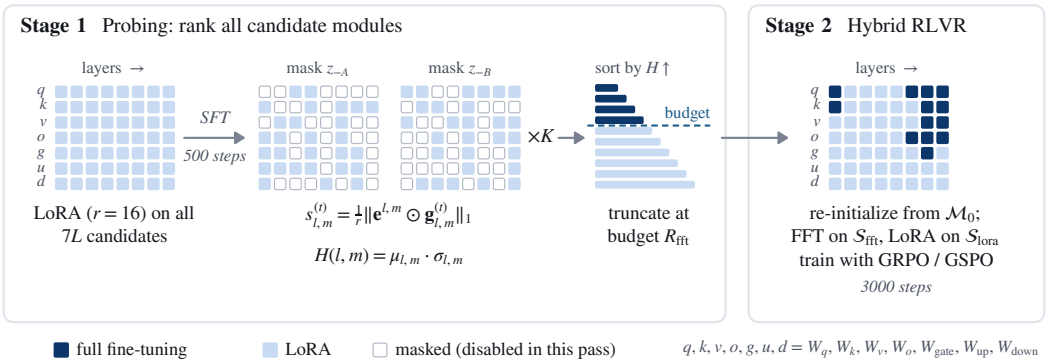


Figure 1: Overview of the Hybrid-LoRA framework. A LoRA-based probing stage, warmed up with a *supervised* (SFT) objective, scores the candidate linear modules. The scores define a budget-independent ranking, and the fully fine-tuned set for a given budget is a prefix of this ranking. The model is then re-initialized; the selected modules are fully fine-tuned, the remaining modules are adapted with LoRA, and this hybrid model is trained with RLVR (GRPO or GSPO).

ratio $R_{\text{fft}} \in (0, 1)$, we impose the constraint

$$\frac{\sum_{(l,m)\in\mathcal{S}_{\text{fft}}} P_{l,m}}{\sum_{(l,m)\in\mathcal{U}} P_{l,m}} \leq R_{\text{fft}}. \tag{4}$$

Our objective is to find a partition $(\mathcal{S}_{\text{fft}}, \mathcal{S}_{\text{lora}})$ that maximizes the post-training performance of the model $\mathcal{M}$:

$$\max_{\mathcal{S}_{\text{fft}},\mathcal{S}_{\text{lora}}} \quad \mathcal{J}\big(\mathcal{M}; \mathcal{S}_{\text{fft}}, \mathcal{S}_{\text{lora}}\big) \quad \text{s.t. Equation 4.} \tag{5}$$

This formulation defines a combinatorial optimization problem over module assignments, which is intractable to solve exactly due to the exponential size of the search space. Therefore, an efficient approximation strategy is required in practice.

3.2 LoRA Re-parameterization and Architecture Parameters

As defined in Equation 1, we consider seven candidate linear module types in each Transformer layer. During the probing stage, we apply LoRA to all candidate modules in each layer. For a given input activation x to module m in layer l, the LoRA-adapted output is written as

$$x' = xW^{l,m} + b^{l,m} + \alpha^{l,m} \cdot xA^{l,m}B^{l,m}. \tag{6}$$

where $W^{l,m}$ and $b^{l,m}$ denote the frozen pretrained weight and bias of the original linear module, respectively. $A^{l,m} \in \mathbb{R}^{d_{\text{model}}\times r}$ and $B^{l,m} \in \mathbb{R}^{r\times d_{\text{out}}}$ are the LoRA low-rank matrices, and $\alpha^{l,m}$ is a module-specific architecture parameter that controls the contribution of the LoRA branch. To facilitate the Hybrid-Score calculation, we follow the SVDLinear parameterization used in AdaLoRA (Zhang et al., 2023) and decompose the LoRA update into a singular-value-style form:

$$x' = xW^{l,m} + b^{l,m} + \alpha^{l,m} \cdot xA^{l,m}E^{l,m}B^{l,m}, \tag{7}$$

where $E^{l,m} = \text{diag}(\boldsymbol{e}^{l,m})$ is the diagonal coefficient matrix in the SVD-style LoRA parameterization, and $\boldsymbol{e}^{l,m} \in \mathbb{R}^r$ contains the singular-value-style coefficients over the r LoRA components.

3.3 Hybrid-Score Calculation

Limitations of existing importance estimators. A straightforward way to estimate module importance is to use the magnitude of the learned architecture parameter as a proxy:

$$I_\alpha(l, m) = |\alpha^{l,m}|. \tag{8}$$

This follows the spirit of differentiable architecture search (Liu et al., 2018) and adaptive LoRA allocation (Zhang et al., 2024). However, learned architecture-parameter magnitudes may not

faithfully reflect the true contribution of an operation (Xiao et al., 2022), making $|\alpha^{l,m}|$ alone insufficient.

A more faithful estimator is perturbation-based scoring, which measures the validation-loss change after removing a module (Ghorbani and Zou, 2019; Xiao et al., 2022):

$$P(l,m) = \mathcal{L}_{\text{val}}(\theta_{\setminus(l,m)}) - \mathcal{L}_{\text{val}}(\theta). \tag{9}$$

However, computing this score for all candidate modules requires $1 + |\mathcal{U}| = 1 + |\mathcal{A}| \cdot L$ validation forward passes per batch, which is prohibitively expensive for LLMs. Therefore, we introduce an attribution-based Hybrid-Score to approximate module contribution efficiently.

Hybrid-Score. We propose Hybrid-Score to estimate how effectively each candidate linear module can be adapted by its LoRA branch after the LoRA-based probing stage. The overall procedure is summarized in Appendix C. Let the validation set be divided into T mini-batches $\{\mathcal{B}_t\}_{t=1}^{T}$. For each validation batch $\mathcal{B}_t$, we randomly partition the candidate module set into two complementary buckets,

$$\mathcal{U} = \mathcal{U}_A^{(t)} \cup \mathcal{U}_B^{(t)}, \qquad \mathcal{U}_A^{(t)} \cap \mathcal{U}_B^{(t)} = \emptyset. \tag{10}$$

We then perform two masked backward passes: one with $\mathcal{U}_A^{(t)}$ disabled and the other with $\mathcal{U}_B^{(t)}$ disabled. Let $z_{-A}^{(t)}$ denote the mask that disables modules in $\mathcal{U}_A^{(t)}$, and $z_{-B}^{(t)}$ denote the mask that disables modules in $\mathcal{U}_B^{(t)}$. For module (l,m), we compute the gradient with respect to the SVD-style diagonal coefficient vector $\boldsymbol{e}^{l,m}$ introduced in Equation 7:

$$\boldsymbol{g}_{l,m}^{(t)} = \nabla_{\boldsymbol{e}^{l,m}} \left[\mathcal{L}_{\text{val}}(\mathcal{B}_t; z_{-A}^{(t)}) + \mathcal{L}_{\text{val}}(\mathcal{B}_t; z_{-B}^{(t)}) \right]. \tag{11}$$

Since module (l,m) is disabled in exactly one of the two passes, only the pass where it remains active contributes a non-zero gradient to $g_{l,m}^{(t)}$.

The batch-level sensitivity score is then computed as

$$s_{l,m}^{(t)} = \frac{1}{r} \left\| \boldsymbol{e}^{l,m} \odot \boldsymbol{g}_{l,m}^{(t)} \right\|_1. \tag{12}$$

Here, $\odot$ denotes element-wise multiplication, so $\boldsymbol{e}^{l,m} \odot \boldsymbol{g}_{l,m}^{(t)}$ measures the dimension-wise first-order contribution of the LoRA coefficients. Across all validation batches, we collect a sensitivity list for each module:

$$\mathcal{S}_{l,m} = \left\{ s_{l,m}^{(t)} \right\}_{t=1}^{T}. \tag{13}$$

The final Hybrid-Score is obtained by aggregating this list:

$$H(l,m) = \mu_{l,m} \cdot \sigma_{l,m}, \tag{14}$$

where

$$\mu_{l,m} = \frac{1}{T} \sum_{t=1}^{T} s_{l,m}^{(t)}, \qquad \sigma_{l,m} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} \left(s_{l,m}^{(t)} - \mu_{l,m} \right)^2}. \tag{15}$$

The random bucket masking is inspired by Shapley-NAS and R-Drop (Xiao et al., 2022; Wu et al., 2021), where importance is estimated under sampled contexts rather than a single fixed architecture. This design accounts for interactions among LoRA modules, since the contribution of one module may depend on which other modules are active. For each validation batch, two complementary masked backward passes ensure that every module is active in one pass and disabled in the other, yielding one batch-level sensitivity value for each module under a sampled co-active context.

Following first-order sensitivity criteria for network pruning (Lee et al., 2018), we define Hybrid-Score as a module-level diagnostic of how effectively each module is adapted by its LoRA branch. Specifically, $\boldsymbol{e}^{l,m} \odot \boldsymbol{g}_{l,m}^{(t)}$ approximates the first-order effect of perturbing the SVD-style diagonal coefficients of module (l,m) on the validation loss. The mean term $\mu_{l,m}$ measures average LoRA-side sensitivity across validation batches, while the standard deviation term $\sigma_{l,m}$ captures variation across sampled co-active contexts, reflecting context-dependent interactions.

Therefore, a high Hybrid-Score indicates that the LoRA branch of module (l, m) has a strong and consistent influence on validation behavior under diverse masking contexts, which we read as LoRA being sufficient for this module. Conversely, a low Hybrid-Score indicates weak LoRA-side adaptation sensitivity. We interpret this as a sign that low-rank adaptation may be insufficient for the module, and we therefore assign low-score modules to full fine-tuning and keep high-score modules under LoRA adaptation. This rank-capacity account is an interpretation, not a demonstrated mechanism: a low score is a priori also consistent with the module being task-irrelevant. The controls in Section 4.5 argue against the irrelevance reading, but do not by themselves establish the rank-capacity account.

3.4 Training Algorithm

The overall training procedure of Hybrid-LoRA is provided in Appendix C. The training process consists of two stages. In the probing stage, we attach LoRA modules to all candidate linear modules in $\mathcal{U}$ from Equation 2 and freeze the pretrained weights. The probing model is warmed up with a *supervised* objective, the next-token loss on reasoning trajectories from the post-training corpus, and $\mathcal{L}_{\text{val}}$ in Equation 11 is the same supervised loss on held-out trajectories. After this short warm-up, the probing model is used to estimate the Hybrid-Score $H(l, m)$ in Equation 14 for each candidate module on the validation set. We probe under the supervised objective for cost: scoring under RLVR would require generating and reward-scoring rollouts for every masking context. This assumes that the module ranking carries over from supervised warm-up to RLVR. We do not verify this transfer directly; what our experiments show is that the partition derived from the supervised probe is effective under RLVR (Section 4.5).

The resulting scores are then used for budget-constrained hybrid module allocation. The probing stage produces a ranking of all $|\mathcal{U}|$ candidate modules that does not depend on the budget. A partition for a given R_{fft} is obtained by truncating this ranking, so a single probing pass serves every budget. Specifically, we sort all candidate modules in ascending order of $H(l, m)$ and greedily add the lowest-score modules into $\mathcal{S}_{\text{fft}}$ until the budget constraint in Equation 4 is reached. The remaining modules are assigned to $\mathcal{S}_{\text{lora}}$ according to the disjoint partition in Equation 3. This provides an efficient approximation to the hybrid allocation objective in Equation 5.

In the formal training stage, we reinitialize the model from the original pretrained checkpoint $\mathcal{M}_0$ and apply the selected hybrid allocation. Modules in $\mathcal{S}_{\text{fft}}$ are fully fine-tuned, while modules in $\mathcal{S}_{\text{lora}}$ are adapted with LoRA. The final model is then optimized with the RLVR objective (GRPO or GSPO; Appendix B) under this mixed parameterization. This design separates module selection from final optimization: the probing stage only determines the adaptation allocation, while the final performance is obtained from a freshly initialized hybrid model.

4 Experiments

In this section, we conduct a series of experiments to evaluate our Hybrid-LoRA method.

4.1 Models

We evaluate Hybrid-LoRA on Qwen-2.5 1.5B, 3B and 7B Instruct. We compare against IO and CoT prompting baselines, full fine-tuning (FFT), a budget-matched LoRA (Hu et al., 2022), and hybrid allocations built on the importance criteria of AdaLoRA (Zhang et al., 2023), AutoLoRA (Zhang et al., 2024), and LoRA-Drop (Zhou et al., 2025) (Section 4.4). The criteria are computed with the methods' open-sourced code.

4.2 Datasets and Evaluation Protocols

Datasets. We evaluate Hybrid-LoRA on six public reasoning benchmarks covering mathematics, knowledge-intensive question answering, and code generation. Specifically, we use Math-500 (Lightman et al., 2023), AIME-24 and AIME-25 (Mathematical Association of America, 2026), MMLU-Pro (Wang et al., 2024), GPQA (Rein et al., 2023), and LeetCodeDataset (Xia et al., 2025). Math-500 and AIME evaluate multi-step mathematical reasoning and competition-level problem solving;

MMLU-Pro and GPQA evaluate challenging academic and professional knowledge reasoning; and LeetCodeDataset measures program synthesis ability.

For post-training, we use Mixture-of-Thoughts (Hugging Face, 2025) as the unified training corpus. It contains approximately 350K verified reasoning trajectories distilled from DeepSeek-R1, providing diverse and structured reasoning supervision for RLVR/GRPO-style post-training. Using a single post-training corpus allows us to compare different adaptation strategies under the same data setting.

Evaluation Protocols. We use task-appropriate metrics for each benchmark. For mathematical reasoning, we report `pass@1` on Math-500 and `pass@4` on AIME-24 and AIME-25, following prior reasoning-oriented evaluation protocols (Yang et al., 2025). Under `pass@k`, each problem is sampled k times and is considered solved if any generated solution matches the reference answer.

For LeetCodeDataset, we report `pass@4`, where each problem is evaluated with four independently generated programs and is counted as correct if any program passes the corresponding test cases. For MMLU-Pro and GPQA, we use a single-response setting and report accuracy based on the standard answer-matching rule of each benchmark.

4.3 Training Details

All experiments are conducted on 8 NVIDIA A800 80GB GPUs. We use the same training hyperparameters for all post-training baselines and our Hybrid-LoRA pipeline unless otherwise specified. The sampling temperature is set to $T = 1.0$. We use a learning rate of 1×10^{-6} for full fine-tuned modules and 2.5×10^{-5} for LoRA modules. For the reinforcement learning stage, we use a group size of 4, a clipping range of ± 0.2, and a KL penalty coefficient of 0.01. For both Hybrid-Score estimation and Hybrid-LoRA training, LoRA modules are initialized with rank $r = 16$. The supervised probing warm-up is trained for 500 steps. We set the full fine-tuning module ratio to $R_{\text{fft}} = 10\%$ and use $K = 20$ random partitions for Hybrid-Score estimation. The RLVR post-training stage runs for 3,000 steps, each sampling a group of 4 rollouts per prompt. During the final training stage, we evaluate the model every 50 steps and keep the checkpoint with the lowest validation loss.

Search space and search cost. Viewed as an architecture search, the allocation problem has one binary decision per candidate matrix. For Qwen-2.5 1.5B, this gives 28 layers $\times$ 7 matrix types $= 196$ decisions, or 2^{196} unconstrained assignments. With the module sizes of this backbone (0.39M parameters for W_k and W_v, 2.36M for W_q and W_o, and 13.76M for each MLP matrix), the number of partitions that satisfy the 10% budget is still on the order of 10^{36}, so exhaustive or leave-one-out search is not an option. Hybrid-LoRA never trains a candidate partition: allocation is the 500-step LoRA warm-up plus $K = 20$ random complementary partitions with two masked forward–backward passes each (40 passes, no generation, no optimizer step). In steps this is about one sixth of the 3,000-step RLVR run, and each step is cheaper than an RLVR step, which must also generate and score rollouts.

4.4 Baselines

We compare Hybrid-LoRA with both inference-only and post-training baselines. In addition to inference-only baselines and the FFT reference, we consider three groups of post-training baselines: criterion-based hybrid allocations, a heuristic hybrid allocation, and controls that isolate the contribution of the Hybrid-Score. For inference-only baselines, **Input–Output prompting (IO)** directly prompts the LLM to generate the final answer without additional training or explicit reasoning instructions, while **Chain-of-Thought prompting (CoT)** asks the model to produce intermediate reasoning steps before giving the final answer, also without any parameter updates.

For post-training baselines, **Full-FT** updates all parameters of the LLM and serves as the upper-bound reference under full fine-tuning.

Criterion-based hybrid allocations. We build hybrid FFT/LoRA allocations from the importance criteria of existing PEFT methods. For **AdaLoRA**, **AutoLoRA**, and **LoRA-Drop**, we rank modules according to their corresponding LoRA-side criteria: allocated rank after adaptive rank allocation, learned architecture weight α, and LoRA output contribution, respectively. Modules with the **lowest scores** under each criterion are treated as less effectively adapted by LoRA and are assigned to full fine-tuning until the 10% budget is reached, while LoRA is applied to the remaining modules. These

rows therefore share our training pipeline and differ from Hybrid-LoRA only in the criterion used to choose $\mathcal{S}_{\text{fft}}$. They are *not* the published methods: those criteria were designed for rank allocation or pruning, and we repurpose them for a task none of them was designed for, so outperforming these rows does not imply outperforming those methods as published. We denote them "Hybrid (*criterion*)" in Table 1.

Heuristic hybrid allocation. LoRA-FFN applies LoRA to all FFN linear layers while fully fine-tuning all attention linear modules.

Controls. To isolate the contribution of the Hybrid-Score, we add three controls (Qwen-2.5 1.5B, GRPO), each with at least as many trainable parameters as Hybrid-LoRA. **Random selection** fills the same budget with modules drawn uniformly at random; we report a single draw trained once, which does not estimate the variability across random selections. **Budget-matched LoRA** applies LoRA to every candidate module with the rank raised to $r = 96$ (112M trainable parameters), so that the entire budget is spent on adapters. **Last-three-layers FFT** fully fine-tunes every matrix in the last three layers and applies rank-16 LoRA to the remaining 25 layers (157M trainable parameters). Since one layer costs 46.8M parameters, whole-layer FFT cannot match 105M exactly, so we use this more generous variant.

4.5 Main Results

The experimental results on the six reasoning benchmarks are presented in Table 1. In the first three columns, we present the post-training approach, parameter efficient approach and number of tunable parameters respectively. Input/output prompting (IO) and Chain-of-Thought (CoT) are included as inference-only baselines without any post-training for reference.

Overall, our proposed **Hybrid-LoRA** consistently outperforms the benchmark-wise best criterion-based hybrid allocation by an average of 4.36% across all model scales and benchmarks, while using only a small fraction of the trainable parameters required by FFT. On Qwen-2.5 1.5B, Hybrid-LoRA attains performance comparable to or better than FFT under both GRPO and GSPO settings, while reducing the number of tunable parameters from 1544M to around 105M. Similar trends are observed on the 3B and 7B models, where Hybrid-LoRA matches or slightly outperforms FFT on most benchmarks, particularly on challenging reasoning tasks such as AIME and GPQA.

Does the Hybrid-Score matter beyond the budget? The three controls on Qwen-2.5 1.5B with GRPO address whether the gain comes from the Hybrid-Score or simply from fully fine-tuning a fixed fraction of the parameters. *The budget alone does not explain it.* Choosing the fully fine-tuned set at random, with a comparable budget and the same pipeline, costs 7.3 points on Math-500, 8.7 on AIME-24, and 6.8 on MMLU-Pro relative to Hybrid-LoRA, despite using 2M more trainable parameters. *Nor does the hybrid structure alone.* Random selection falls below all three criterion-based hybrid allocations, and below the budget-matched LoRA, which spends the same budget entirely on adapters: an uninformative full-fine-tuning allocation can be worse than making no such allocation at all. As this is a single draw, it does not capture the variability across random selections. *Additional low-rank capacity is not a substitute.* Raising the rank of every adapter to 96, six times ours, reaches 74.2 on Math-500 with 112M trainable parameters, 6.0 points below Hybrid-LoRA at 105M. *Nor does a depth heuristic suffice.* Fully fine-tuning the last three layers is the strongest control, but even with roughly 50% more trainable parameters (157M) it trails Hybrid-LoRA by 4.1 points on Math-500 and 4.7 on AIME-24.

Criterion-based hybrid allocations. Hybrid-LoRA also outperforms the hybrid allocations built on the AdaLoRA, AutoLoRA, and LoRA-Drop criteria, with the largest gains on Math-500 and AIME-24/25. Since these rows differ from ours only in the selection criterion, this shows that the Hybrid-Score is the better criterion for choosing which modules to fully fine-tune. Taken together, the results indicate that **how the parameters are allocated across modules matters more than how many are trained**.

4.6 Module Distribution Analysis

Figure 2 visualizes the final module allocation obtained by our Hybrid-Score-based pipeline. The fully fine-tuned modules are concentrated in the attention blocks of the later layers (21–28), covering the

Table 1: Overall reasoning benchmark performance. Full-FT (ref.) is the full-parameter reference. Rows marked Hybrid (*criterion*) share our hybrid FFT/LoRA pipeline and choose the fully fine-tuned set with the importance criterion of the named method; they are not the published methods (Section 4.4). Random selection is a single draw.

Post-training Approach	**Param efficient Approach**	**#Tunable Params**	Math-500 pass@1	AIME-24 pass@4	AIME-25 pass@4	MMLU-Pro Accuracy	GPQA Accuracy	LeetCodeDataset pass@4
Qwen-2.5 1.5B Instruct								
IO	-	0.0	25.4	3.3	0.0	13.1	18.8	0.0
COT	-	0.0	56.8	14.7	10.7	20.6	34.1	14.6
SFT	Full-FT	1544M	68.4	22.7	21.3	46.9	37.9	40.6
GRPO	Full-FT (ref.)	1544M	80.6	40.0	36.7	55.6	37.2	48.3
GRPO	LoRA (r=96, budget-matched)	112M	74.2	30.7	30.0	49.3	32.8	44.6
GRPO	Random selection (single draw)	107M	72.9	29.3	28.7	48.1	31.9	43.8
GRPO	Last-3-layers FFT	157M	76.1	33.3	32.7	51.4	34.5	45.6
GRPO	Hybrid (AdaLoRA crit.)	105M	75.3	32.0	33.3	50.5	33.9	45.4
GRPO	Hybrid (AutoLoRA crit.)	106M	77.4	34.7	31.3	52.1	34.4	45.7
GRPO	Hybrid (LoRA-Drop crit.)	105M	76.8	34.0	32.0	53.2	35.1	46.3
GRPO	Hybrid-LoRA (ours)	105M	**80.2**	**38.0**	**35.3**	**54.9**	**37.3**	**48.0**
GSPO	Full-FT (ref.)	1544M	82.0	42.0	37.3	56.8	38.7	49.1
GSPO	LoRA-FFN	122M	79.2	34.0	30.7	53.1	36.3	44.5
GSPO	Hybrid (AdaLoRA crit.)	106M	79.4	36.0	31.3	53.5	36.6	44.8
GSPO	Hybrid (AutoLoRA crit.)	107M	80.2	35.3	32.7	54.9	37.1	45.2
GSPO	Hybrid (LoRA-Drop crit.)	105M	79.8	40.0	34.0	54.7	36.9	44.7
GSPO	Hybrid-LoRA (ours)	105M	**81.6**	**42.0**	**36.7**	**56.9**	**38.5**	**47.6**
Qwen-2.5 3B Instruct								
CoT	-	0.0	77.2	24.0	20.6	40.4	50.3	40.2
GRPO	Full-FT (ref.)	3086M	91.2	46.0	44.0	71.8	60.7	63.8
GRPO	Hybrid (AutoLoRA crit.)	201M	89.6	40.7	38.7	69.3	58.4	59.6
GRPO	Hybrid-LoRA (ours)	192M	**91.4**	**43.3**	**42.0**	**71.9**	**60.4**	**62.7**
GSPO	Full-FT (ref.)	3086M	92.4	49.3	47.3	73.4	62.1	65.1
GSPO	Hybrid (AutoLoRA crit.)	194M	90.2	44.0	40.7	71.1	60.3	62.6
GSPO	Hybrid-LoRA (ours)	192M	**91.8**	**48.0**	**45.3**	**72.9**	**61.6**	**64.0**
Qwen-2.5 7B Instruct								
GRPO	Hybrid (AutoLoRA crit.)	453M	91.4	52.7	48.7	80.3	73.5	71.5
GRPO	Hybrid-LoRA (ours)	449M	**93.6**	**56.0**	**52.0**	**83.1**	**76.3**	**73.2**
GSPO	Hybrid (AutoLoRA crit.)	453M	92.2	55.3	52.7	81.9	74.8	72.5
GSPO	Hybrid-LoRA (ours)	449M	**94.4**	**60.0**	**55.3**	**83.8**	**77.9**	**74.5**

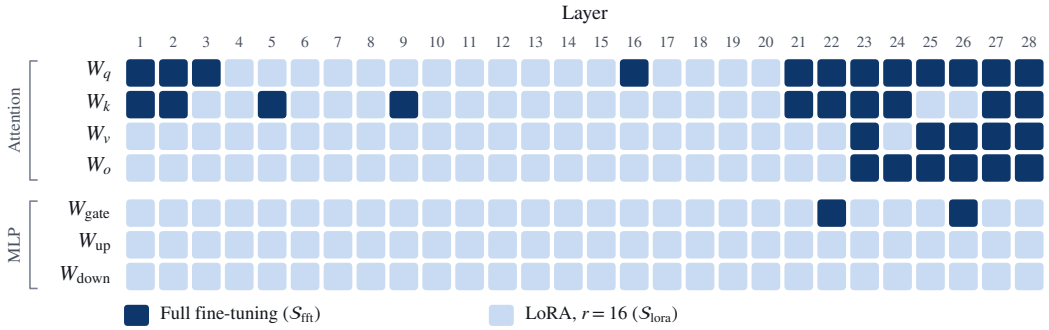
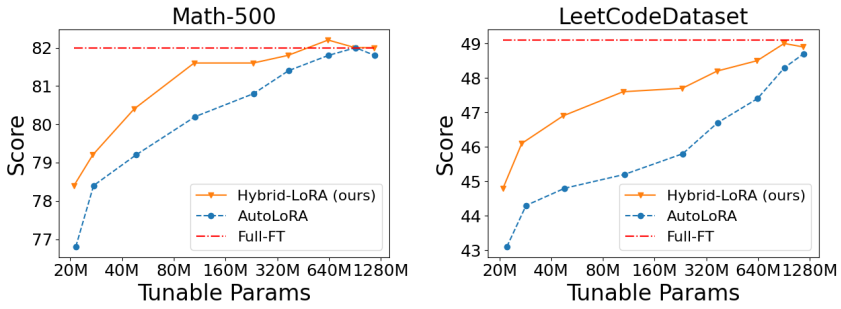


Figure 2: Module allocation selected by Hybrid-LoRA on Qwen-2.5 1.5B (Mixture-of-Thoughts, $R_{\text{fft}} = 10\%$). Dark cells are fully fine-tuned; light cells are adapted with LoRA. Module types differ greatly in size (W_k, W_v: 0.39M; W_q, W_o: 2.36M; each MLP matrix: 13.76M parameters), so the *number* of selected cells is not comparable across module types: one MLP matrix costs as much as 35 key or value projections.

query, key, value, and output projections. The MLP projections are rarely selected: W_{gate} is selected in two layers, and W_{up} and W_{down} are never selected. Since modules are ranked by Hybrid-Score alone and cost enters only through the budget, a low-scoring MLP matrix would be taken early; the absence of W_{up} and W_{down} thus reflects their scores, not their cost, while W_{gate}, at identical cost, is selected twice.

Why depth alone is not enough. Although the selected set is concentrated in the upper layers, it also includes early layers (query and key projections in layers 1–3, 5, 9, and 16). Within each layer it is also specific about *which* projections it takes. A depth rule captures neither property, consistent with the last-three-layers control in Table 1 trailing Hybrid-LoRA despite using roughly 50% more trainable parameters.



(a) Performance on Math-500 (b) Performance on LeetCodeDataset

Figure 3: Ablation studies on the number of tunable parameters.

We read the concentration in later attention projections as a possible sign that rank-16 updates are too restrictive there; this is an interpretation rather than an established mechanism.

4.7 Ablation Studies

Tunable Parameters. We ablate the effect of tunable parameter budgets in Figure 3, using AutoLoRA as the comparison baseline. Hybrid-LoRA matches or exceeds AutoLoRA at every budget on both Math-500 and LeetCodeDataset, and the gap narrows as the budget approaches full fine-tuning. Because the probe yields a budget-independent ranking, the partition for every budget on this curve can be read off a single probing pass.

Hybrid-Score variants. Among alternative aggregation rules that keep the cheap probing stage, the product $\mu_{l,m} \cdot \sigma_{l,m}$ performs best (Appendix D).

5 Conclusion and Limitations

We propose Hybrid-LoRA, which fully fine-tunes the modules least effectively adapted by LoRA and adapts the rest with LoRA, selected by Hybrid-Score, a criterion computed in a short supervised probing stage. It selects a substantially more effective fully fine-tuned set than random, depth-based, or existing importance criteria, and closely matches full fine-tuning across reasoning tasks with verifiable rewards and model scales under a 10% budget.

Limitations. (1) Our evaluation is confined to reasoning tasks with verifiable rewards, the setting RLVR presupposes; agentic and tool-use settings are the primary direction for future work. (2) Due to compute constraints, we scale only up to 7B models in a single model family. (3) The rank-capacity account of why low-scoring modules benefit from full fine-tuning is an interpretation that our controls support but do not prove. (4) Hybrid-LoRA is not a supernet: one probing pass yields partitions for every budget, but each must be post-trained on its own, and sub-networks cannot be deployed without that training. (5) The probe uses a supervised objective as a proxy for RLVR, and we do not directly verify that the module ranking transfers; end-to-end or dynamically updated allocation is a natural next step. (6) Our controls are run on a single backbone (Qwen-2.5 1.5B with GRPO) with a single random draw, and we do not report component ablations that train only one half of the partition, the sensitivity of the partition to the probing hyper-parameters, or a spectral analysis of the learned updates; we leave these to future work.

Acknowledgements and Disclosures

This work was supported by the National Science Foundation under Grant IOS-2430277 and the Paul G. Allen Family Foundation. The authors declare no competing interests.

References

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *Proceedings of the 2023 conference*

on empirical methods in natural language processing, pages 4133–4145, 2023.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pages 2242–2251. PMLR, 2019.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/houlsby19a.html`.

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

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. Alora: Allocating low-rank adaptation for fine-tuning large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 622–641, 2024.

Mathematical Association of America. American invitational mathematics examination (aime). `https://maa.org/maa-invitational-competitions/`, 2026. Accessed: 2026-04-25.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. R-drop: Regularized dropout for neural networks. *Advances in neural information processing systems*, 34: 10890–10905, 2021.

Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. Leetcodedataset: A temporal dataset for robust evaluation and efficient training of code llms. *arXiv preprint arXiv:2504.14655*, 2025.

Han Xiao, Ziwei Wang, Zheng Zhu, Jie Zhou, and Jiwen Lu. Shapley-nas: Discovering operation contribution for neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11892–11901, 2022.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=lq62uWRJjiY`.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5048–5060, 2024.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Hongyun Zhou, Xiangyu Lu, Wang Xu, Conghui Zhu, Tiejun Zhao, and Muyun Yang. Lora-drop: Efficient lora parameter pruning based on output evaluation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5530–5543, 2025.

A Broader Impact

Hybrid-LoRA aims to reduce the computational and memory cost of reasoning-oriented LLM post-training by combining full fine-tuning and LoRA adaptation under a fixed parameter budget. This can make advanced post-training more accessible to researchers and organizations with limited computational resources, and may reduce the cost of adapting LLMs for scientific, educational, and engineering applications. However, more efficient fine-tuning methods may also lower the barrier for adapting LLMs to harmful or misleading uses, such as generating misinformation, spam, or unsafe instruction-following behaviors. In addition, although Hybrid-LoRA improves parameter efficiency, large-scale LLM training and evaluation still incur non-negligible computational and environmental costs. These considerations highlight the need for responsible dataset selection, careful evaluation, and appropriate deployment safeguards when applying efficient post-training methods.

B Preliminaries

B.1 Reinforcement Learning with Verifiable Rewards (RLVR) for Reasoning Post-Training

We consider reinforcement learning with verifiable rewards (RLVR) for reasoning-oriented post-training of large language models. In this setting, a pretrained language model, typically initialized from an SFT checkpoint, receives an input problem x sampled from a dataset $\mathcal{D}$ and generates an output sequence $y \sim \pi_\theta(\cdot \mid x)$. The output may include both an intermediate reasoning process and a final answer. Unlike preference-based reinforcement learning, RLVR uses a rule-based or programmatic verifier that assigns a reward according to whether the generated answer is correct or otherwise verifiable.

Formally, for each input-output pair (x, y), a verifier defines a reward function

$$r(x, y) \in \mathbb{R}. \tag{16}$$

which is typically sparse and depends on the correctness of the final answer, although more structured rewards can also be used when parts of the reasoning process are verifiable. The RLVR objective is to optimize the policy π_θ so as to maximize the expected reward:

$$\max_\theta \ \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[r(x, y)]. \tag{17}$$

B.2 GRPO for RLVR Optimization

Group Relative Policy Optimization (GRPO) is a critic-free policy optimization method widely used for RLVR-based reasoning post-training. Given a prompt x, GRPO samples a group of responses $\{y_i\}_{i=1}^G$ from the current policy and assigns each response a scalar reward r_i. Instead of training a separate value model, GRPO estimates the advantage of each response by normalizing its reward within the sampled group:

$$A_i = \frac{r_i - \mu_r}{\sigma_r}, \qquad \mu_r = \frac{1}{G}\sum_{i=1}^{G} r_i, \qquad \sigma_r = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(r_i - \mu_r)^2 + \epsilon}. \tag{18}$$

The policy is then updated with a PPO-style clipped objective and a KL penalty against a reference policy. In RLVR, the reward is typically constructed from verifiable signals such as answer correctness, format compliance, and length control. Given a prompt x, GRPO samples a group of G responses $\{y_i\}_{i=1}^G$ from the behavior policy $\pi_{\theta_{\text{old}}}$. Each response is assigned a scalar reward r_i, and the group-relative advantage is computed as

$$A_i = \frac{r_i - \mu_r}{\sigma_r}, \qquad \mu_r = \frac{1}{G}\sum_{i=1}^{G} r_i, \qquad \sigma_r = \sqrt{\frac{1}{G}\sum_{i=1}^{G}(r_i - \mu_r)^2 + \epsilon}.$$

For each token $y_{i,t}$, the importance ratio between the updated policy π_θ and the behavior policy $\pi_{\theta_{\text{old}}}$ is

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})}.$$

GRPO optimizes a PPO-style clipped surrogate objective:

$$\ell_{i,t}(\theta) = \min\Big(\rho_{i,t}(\theta)A_i,\ \mathrm{clip}\big(\rho_{i,t}(\theta), 1-\epsilon_{\mathrm{clip}}, 1+\epsilon_{\mathrm{clip}}\big)A_i\Big).$$

The final GRPO objective is

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{x,\{y_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\ell_{i,t}(\theta)\right] - \beta\,\mathrm{KL}(\pi_\theta \,\|\, \pi_{\mathrm{ref}}),$$

where π_{ref} denotes the reference policy and β controls the strength of the KL regularization.

C Algorithms

We provide the detailed procedures of Hybrid-LoRA in Algorithm 1 and Algorithm 2. Algorithm 1 presents the Hybrid-Score calculation, which aggregates gradient-weighted LoRA-side sensitivities over validation batches and randomly sampled masking contexts. Algorithm 2 gives the full Hybrid-LoRA training pipeline: the model is first warmed up with LoRA on all candidate modules, then modules are ranked by their Hybrid-Scores, and finally low-scoring modules are assigned to full fine-tuning while the remaining modules are trained with LoRA.

Algorithm 1 Hybrid-Score Calculation

Require: Probing model $\mathcal{M}_{\mathrm{probe}}$, candidate module set $\mathcal{U}$, validation dataloader $\mathcal{D}_{\mathrm{val}}$
Ensure: Hybrid-Score $H(l,m)$ for each $(l,m) \in \mathcal{U}$

Initialize $\mathcal{S}_{l,m} \leftarrow [\,]$ for each $(l,m) \in \mathcal{U}$
for each validation batch $\mathcal{B}_t \in \mathcal{D}_{\mathrm{val}}$ **do**
  $\mathcal{U}_A^{(t)}, \mathcal{U}_B^{(t)} \leftarrow \textsc{RandomPartition}(\mathcal{U}, p = 0.5)$
  Clear gradients
  Compute $\mathcal{L}_A^{(t)}$ with modules in $\mathcal{U}_A^{(t)}$ disabled
  Backpropagate $\mathcal{L}_A^{(t)}$ and accumulate gradients
  Compute $\mathcal{L}_B^{(t)}$ with modules in $\mathcal{U}_B^{(t)}$ disabled
  Backpropagate $\mathcal{L}_B^{(t)}$ and accumulate gradients
  for each $(l,m) \in \mathcal{U}$ **do**
    $g_{l,m}^{(t)} \leftarrow \nabla_{e^{l,m}}\left(\mathcal{L}_A^{(t)} + \mathcal{L}_B^{(t)}\right)$
    $s_{l,m}^{(t)} \leftarrow \frac{1}{r}\left\| e^{l,m} \odot g_{l,m}^{(t)} \right\|_1$
    Append $s_{l,m}^{(t)}$ to $\mathcal{S}_{l,m}$
  end for
end for
for each $(l,m) \in \mathcal{U}$ **do**
  $H(l,m) \leftarrow \mathrm{Mean}(\mathcal{S}_{l,m}) \cdot \mathrm{Std}(\mathcal{S}_{l,m})$
end for
return H

D Hybrid-Score Variants

We further evaluate several variants of the Hybrid-Score aggregation rule in Table 2. Hybrid-LoRA-1 replaces the product form with a mean-to-variability ratio $\mu_{l,m}/\sigma_{l,m}$, while Hybrid-LoRA-2 uses the coefficient-of-variation-style form $\sigma_{l,m}/\mu_{l,m}$. Hybrid-LoRA-3 adopts a mean-variance-style additive score $\mu_{l,m} + 0.5\sigma_{l,m}$, and Hybrid-LoRA-4 reverses the allocation direction by starting from full fine-tuning and replacing lower-scoring modules with LoRA. The original Hybrid-LoRA and Hybrid-LoRA-4 achieve comparable performance across the two benchmarks. However, Hybrid-LoRA-4 is much less practical because it starts from a full fine-tuning configuration and replaces lower-scoring modules with LoRA, leading to substantially higher allocation cost, running time, and memory usage. In contrast, the original Hybrid-LoRA obtains similar performance through an efficient LoRA-probing stage, making it the more practical design.

Algorithm 2 Hybrid-LoRA Training

Require: Pretrained LLM $\mathcal{M}_0$, candidate module set $\mathcal{U} = \mathcal{A} \times \{1, \ldots, L\}$, warmup steps T_{warm}, validation dataloader $\mathcal{D}_{\text{val}}$, FFT budget ratio R_{fft}

Ensure: Hybrid fine-tuned model $\mathcal{M}_{\text{hybrid}}$

Initialize probing model $\mathcal{M}_{\text{probe}} \leftarrow \mathcal{M}_0$

Attach LoRA modules to all candidate modules $(l, m) \in \mathcal{U}$ on probing model $\mathcal{M}_{\text{probe}}$

Freeze pretrained weights and train only LoRA parameters

Warm up $\mathcal{M}_{\text{probe}}$ for T_{warm} steps with the supervised (SFT) objective

Compute Hybrid-Scores for all modules:

$$H(l, m) \leftarrow \textsc{HybridScore}(\mathcal{M}_{\text{probe}}, \mathcal{U}, \mathcal{D}_{\text{val}})$$

Rank all modules in $\mathcal{U}$ by $H(l, m)$ in ascending order

Select the lowest-scoring modules under the FFT budget R_{fft} as S_{fft} ▷ a prefix of the ranking

Define the remaining modules as

$$S_{\text{LoRA}} = \mathcal{U} \setminus S_{\text{fft}}$$

Reinitialize hybrid model $\mathcal{M}_{\text{hybrid}} \leftarrow \mathcal{M}_0$

Unfreeze full parameters of modules in S_{fft}

Attach LoRA modules to modules in S_{LoRA}

Train $\mathcal{M}_{\text{hybrid}}$ with the RLVR objective (GRPO or GSPO)

return $\mathcal{M}_{\text{hybrid}}$

Table 2: Ablation study of different Hybrid-Score variants on reasoning benchmarks.

Method	Math-500 pass@1	LeetCodeDataset pass@4
Hybrid-LoRA	**81.6**	47.6
Hybrid-LoRA-1 ($\mu_{l,m}/\sigma_{l,m}$)	80.8	46.9
Hybrid-LoRA-2 ($\sigma_{l,m}/\mu_{l,m}$)	79.4	45.3
Hybrid-LoRA-3 ($\mu_{l,m} + 0.5 \times \sigma_{l,m}$)	81.2	46.4
Hybrid-LoRA-4 (pruning from Full-FT)	81.4	**47.7**

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Both of our abstract and introduction introduce Hybrid-LoRA, which reflects our major contribution throughout the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitation of our paper in the last section "Conclusion and Limitations".

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: Our paper proposes empirical results, where theoretical proofs are not applicable.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Though we decide not to open-source our code, we provide a detailed guideline of how our proposed method is implemented and applied on models, with complete training details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We decide not to release our source code.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide in-detailed explanation on dataset selection, training process, and hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Each RLVR post-training run is expensive, so most configurations are trained once. Error bars and significance tests are therefore not reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specified our computational resources in Section "Training Details".

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. We use public benchmark datasets and pretrained models for evaluating fine-tuning strategies, do not collect human-subject data, and do not conduct experiments involving private or sensitive personal information.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed the broader impact after the conclusion section of the paper.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release new model weights or datasets with high misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cite the creators of all datasets, pretrained models, and baseline methods used in the paper. All assets are used for research purposes in accordance with their publicly available licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce new datasets, model checkpoints, benchmarks, or other reusable assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or human-subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research or study participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are a core component of this research. The paper studies efficient post-training of LLMs and describes the pretrained backbone models, fine-tuning methods, training objectives, datasets, evaluation benchmarks, and implementation details used in the experiments.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.