Fast-RL: Accelerating Reinforcement Learning for Large Reasoning Models

Sitong Wu[1] **Haoru Tan**[2] **Bin Xia**[1] **Bei Yu**[1] **Xiaojuan Qi**[2] **Jiaya Jia**[1,3]*

[1]The Chinese University of Hong Kong [2]The University of Hong Kong
[3]The Hong Kong University of Science and Technology

Abstract

Large Reasoning Models (LRMs) have made remarkable progress, driven by a paradigm shift from fast thinking to slow thinking. While slow-thinking unlocks superior reasoning ability, its long-form nature imposes a prohibitive computational burden on Reinforcement Learning (RL) for LRMs. In this paper, we propose **Fast-RL**, a novel framework that accelerates RL training for slow-thinking LRMs. Our acceleration principle stems from a foundational insight: *fast-thinking sampling serves as an efficient and powerful proxy for slow-thinking sampling in enhancing reasoning, since the core reasoning skills are shared across thinking modes and can be improved through either trajectory type.* Fast-RL consists of two stages. **Stage 1** establishes reliable prompt-based thinking-mode control, so that a slow-thinking-oriented model triggers fast-thinking under a specific prompt while preserving its native slow-thinking under the standard prompt. **Stage 2** then enhances reasoning ability through RL with *exclusively* fast-thinking sampling for acceleration. A lightweight LoRA adapter is introduced as a dedicated mode controller: it is trained in Stage 1 and *frozen* in Stage 2, decoupling mode-control parameters from reasoning parameters and protecting the model's native slow-thinking from being disturbed by fast-thinking RL. After Stage 2, the LoRA acts as a disposable training scaffold that can be either merged into the main model or simply discarded. Experiments on diverse multimodal and text-only LRMs show that Fast-RL accelerates training by about $10\times$ while delivering superior performance gains over vanilla slow-thinking RL. On Qwen3-1.7B, Fast-RL improves AIME24 by $+10.0$ and $+13.7$ in slow- and fast-thinking evaluation, respectively. Vanilla slow-thinking RL, in contrast, drops slow-thinking accuracy by -8.8 and yields only $+2.8$ in fast-thinking evaluation. The code will be released at https://github.com/JIA-Lab-research/Fast-RL.

1 Introduction

Large Reasoning Models (LRMs) [18, 30] have marked a major breakthrough on complex multimodal and linguistic reasoning tasks, exemplified by frontier models such as OpenAI o3 [23], Gemini 2.5 Pro [5], and DeepSeek-R1 [11]. A pivotal driver behind this progress is a fundamental shift of the model's thinking paradigm from fast-thinking to slow-thinking. Fast-thinking generates a concise, deterministic reasoning trajectory; slow-thinking instead engages in deliberate exploration and reflection before producing the final solution (more details in Appendix A).

However, slow-thinking poses a severe challenge to efficient Reinforcement Learning (RL) [25, 26], the most effective method for enhancing reasoning ability. For example, RL training of a 1.5B-parameter LRM on a modest 40K-sample dataset already consumes tens of thousands of A100 GPU

*Corresponding author

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

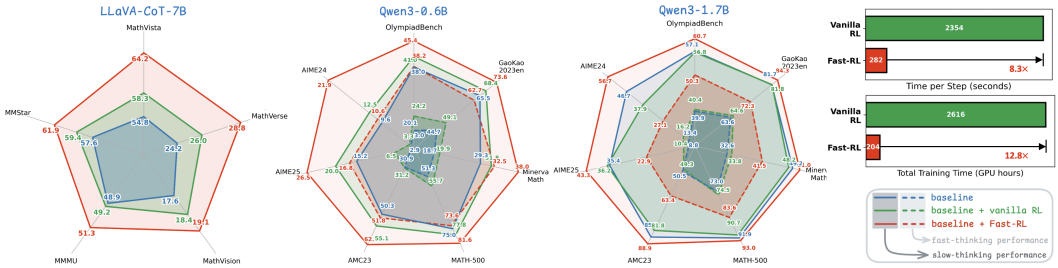


Figure 1: Performance and speed comparison between Fast-RL and vanilla RL. Vision-R1-7B is trained on VerMulti-65K [24] and Vision-R1-10K [15], while Qwen3 models are trained on DeepScaleR-40K [22]. Speed tests are conducted on DeepSeek-R1-Distill-Qwen-7B with DeepScaleR-40K. For Qwen3 models we report both slow- and fast-thinking performance, as they natively support both modes. See Sec. 4 for more results.

hours [22]. This high cost stems from inefficient online sampling: RL inherently relies on massive trajectory collection to estimate the policy gradient, and slow-thinking sampling is far slower than its fast-thinking counterpart due to the long-form nature of slow-thinking trajectories. Such prohibitive training costs not only impede the development and scaling of LRMs, but also concentrate progress within a small subset of well-resourced institutions, exacerbating accessibility barriers in the research community. This dilemma compels us to revisit a tacit convention: *must the thinking mode used during RL online sampling match the mode used at inference?*

In this work, we challenge this convention (see Sec. 2) and uncover a foundational insight: *slow-thinking online sampling is not a prerequisite for enhancing slow-thinking reasoning ability.* Through systematic investigation, we find that RL with fast-thinking online sampling can robustly improve performance under slow-thinking evaluation, indicating that the core reasoning skills (such as logical decomposition, hypothesis verification, and strategic planning) reside in a substrate shared across thinking modes and can therefore be optimized through either fast or slow trajectories. Most importantly, low-cost fast-thinking RL achieves performance gains comparable to, or even exceeding, those of high-cost slow-thinking RL. Fast-thinking sampling thus serves as a highly efficient and effective proxy for its expensive counterpart, providing a solid foundation for accelerating RL.

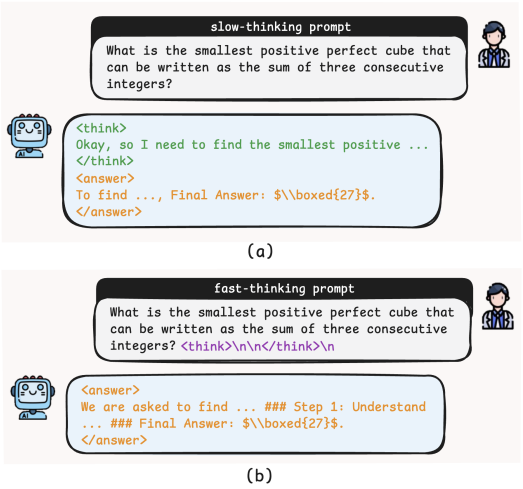


Figure 2: Illustration of (a) slow-thinking behavior under a standard prompt (question only) and (b) fast-thinking behavior under the augmented prompt (question ending with an empty thinking tag).

Inspired by this insight, we propose **Fast-RL**, a novel RL framework that enables fast training by replacing the default slow-thinking online sampling with fast-thinking sampling. However, making an LRM accustomed to slow-thinking stably perform fast-thinking is non-trivial. We therefore design a two-stage framework: (1) **Stage 1** establishes a stable prompt-conditioned thinking-mode selection mechanism, which lays the foundation for fast-thinking sampling in Stage 2. As illustrated in Figure 2, the model is trained to perform native slow-thinking when given the question alone and to think fast (*i.e.,* bypass the thinking phase) when the input prompt ends with an empty thinking block ("`<think>\n\n</think>\n`"). We achieve this with RL on mixed fast/slow prompts using a binary reward that measures thinking-mode correctness. Empirically, this capability is acquired in tens of training steps with only $\sim$1K examples, incurring minimal overhead. We further adopt LoRA rather than full fine-tuning to preserve the model's original capability and to isolate mode-control parameters from those governing reasoning. (2) **Stage 2** enhances reasoning ability through RL while accelerating training via our key design: *exclusive* fast-thinking online sampling. Crucially, the

LoRA mode controller is *frozen* and only the main branch is updated, which strictly preserves the prompt-based mode control acquired in Stage 1 and prevents mode contamination.

Extensive experiments on diverse multimodal and text-only LRMs demonstrate that Fast-RL dramatically accelerates RL training while consistently bringing larger performance gains than vanilla slow-thinking RL. We summarize four key advantages: (1) About $10\times$ faster training, by simultaneously reducing per-step cost and the number of steps needed for convergence (Sec. 4.2.1). (2) Superior performance gains: Fast-RL consistently outperforms vanilla RL, with improvements in some cases more than doubling or even tripling those of vanilla RL (Sec. 4.2.2). (3) Breaking the slow-thinking convergence bottleneck: Fast-RL further improves models that have already converged under vanilla RL on the same training data (Sec. 4.2.3). (4) Broad compatibility with diverse policy-optimization algorithms, such as PPO [25], GRPO [26], and DAPO [33] (Sec. 4.3). Together, these results establish Fast-RL as a scalable, robust, and high-performance framework for efficient LRM training.

2 Foundational Insight and Motivation

In this section, we establish the central insight that motivates our work and provides a viable foundation for accelerating RL training of LRMs.

Core Insight

Slow-thinking online sampling is not a prerequisite for enhancing the slow-thinking reasoning ability of LRMs through RL. Remarkably, fast-thinking sampling serves as a highly effective proxy for slow-thinking sampling: despite its brevity, it yields comparable or even superior improvements while drastically reducing sampling cost.

This insight is far from self-evident. It emerges from a step-by-step empirical investigation that mirrors our own exploratory journey. We begin in Sec. 2.1 by introducing a prompting trick that elicits fast-thinking behavior from slow-thinking LRMs. This trick lets us probe a previously uninvestigated question in Sec. 2.2: *does slow-thinking RL improve the model's fast-thinking performance?* Our experiments confirm this surprising result, which in turn motivates the reverse question: *can fast-thinking RL also enhance slow-thinking performance?* Sec. 2.3 answers this affirmatively. Together, these findings show that the benefits of RL are not confined to the thinking mode used during online sampling, but consistently extend to the alternative mode. Sec. 2.4 analyzes the underlying rationale of this cross-mode generalization and discusses its practical implications for accelerating RL training. Finally, Sec. 2.5 highlights two critical limitations of the simple implementation in Sec. 2.3, which motivate our two-stage RL framework presented in Sec. 3.

2.1 A Prompting Trick to Unlock Fast-Thinking in Slow-Thinking LRMs

It has long been a convention that LRMs operate exclusively in slow-thinking mode. This is not merely an observed behavior but an explicit design objective: substantial effort has been devoted to robustly instilling slow-thinking, ensuring the model always engages in a deep reasoning process before producing the final solution. As a result, slow-thinking has naturally become the default mode for RL online sampling. In our preliminary exploration, we found that a simple prompting trick can induce slow-thinking LRMs to skip the thinking phase and produce the solution directly. As shown in Figure 2 (b), this trick appends an empty thinking tag "`<think>\n\n</think>\n`" to the end of the question, leading the model to treat the thinking phase as already complete and to output the solution directly, resembling a fast-thinking response. The success rate ρ of this trick varies across models, typically in the range of $40\% \sim 80\%$ (Figure 3).

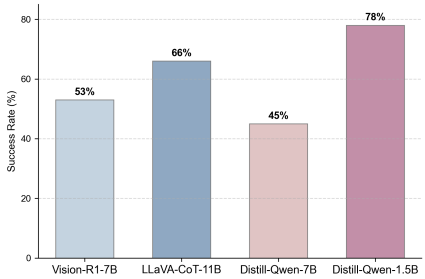


Figure 3: The percentage of fast-thinking trajectories elicited by the prompting trick (*i.e.,* its success rate) on different LRMs.

2.2 Initial Discovery: Slow-Thinking RL Improves Fast-Thinking Performance

The prompting trick in Sec. 2.1 raises an overlooked but intriguing question:

> *How does the LRM's fast-thinking performance change after it undergoes standard RL training with slow-thinking sampling?*

To investigate this, we conduct an empirical study and detail our setup and findings below.

Setup. We adopt the multimodal LRM Vision-R1-7B [15] as the baseline and fine-tune it with GRPO [26] under slow-thinking online sampling. More details are provided in Appendix B.1.

Results. Figure 4 (a) reports both fast- and slow-thinking performance and reveals two key findings:

- **As expected, slow-thinking performance improves**, e.g., $+0.5$ on MathVista and $+0.8$ on MMMU, confirming the effectiveness of RL.
- **Surprisingly, fast-thinking performance also improves**: the RL-finetuned model gains $+1.2$ on MathVision and $+0.5$ on MMMU under fast-thinking evaluation, despite the fact that *no* fast-thinking trajectory is used for policy optimization. This provides the first strong empirical evidence that the reasoning gains induced by slow-thinking RL are not confined to the slow-thinking mode, but generalize to fast-thinking, suggesting that RL strengthens a more fundamental reasoning substrate that remains accessible when the model is constrained to think fast.

2.3 Reverse Exploration: Can Fast-Thinking RL Boost Slow-Thinking Performance?

Given the surprising finding in Sec. 2.2 that gains from slow-thinking RL generalize to fast-thinking, a natural follow-up question arises:

> *Can RL training with fast-thinking sampling, conversely, improve the slow-thinking performance?*

To investigate this, we conduct an empirical study analogous to Sec. 2.2, but with the training-time thinking mode reversed from slow to fast. More details are provided in Appendix B.2.

2.3.1 Fast-thinking Evaluation Results

We observe two phenomena. **(1) Fast-thinking performance clearly improves.** As shown in Figure 4 (b), training with fast-thinking sampling brings $+2.5$ on MathVision and $+1.1$ on MMMU, confirming the feasibility of RL under fast-thinking sampling. **(2) These gains exceed those from slow-thinking RL.** Comparing Figure 4 (a) and (b), fast-thinking RL nearly doubles the fast-thinking gains achieved by slow-thinking RL ($+2.5$ vs. $+1.2$ on MathVision; $+1.1$ vs. $+0.5$ on MMMU). This indicates that, for enhancing fast-thinking reasoning, RL under the in-distribution setting (fast-thinking online sampling) is more direct and effective.

2.3.2 Slow-thinking Evaluation Results

The slow-thinking evaluation reveals more surprising findings. **(1) Fast-thinking RL also boosts slow-thinking performance.** As shown in Figure 4 (b), the model trained exclusively on fast-thinking trajectories yields $+1.0$ on MathVista and $+0.6$ on MMMU under slow-thinking evaluation. This affirmatively answers the question raised at the beginning of this subsection: fast-thinking RL not only improves fast-thinking ability but also enhances slow-thinking reasoning, again indicating that RL benefits are not confined to the training-time thinking mode. **(2) The slow-thinking gains are comparable to or even larger than those from slow-thinking RL.** The slow-thinking RL in Sec. 2.2 delivers $+0.5$ on MathVista and $+0.8$ on MMMU, whereas fast-thinking RL achieves a similar $+0.6$ on MMMU and an even larger $+1.0$ on MathVista. Fast-thinking sampling thus achieves comparable or better improvements in reasoning ability, highlighting its potential as the default sampling mode for RL.

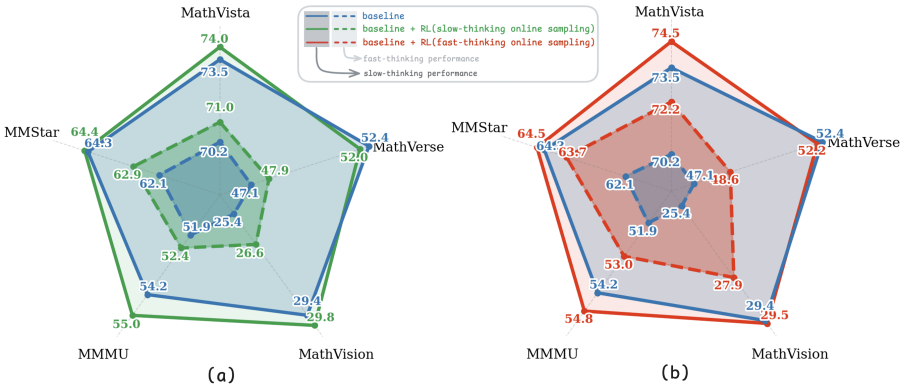


Figure 4: Comparison of the Vision-R1-7B model before and after RL with (a) slow-thinking and (b) fast-thinking online sampling.

2.4 Analysis and Implications

The experiments in Sec. 2.2 and Sec. 2.3 jointly reveal a consistent and striking pattern: *reasoning capabilities in LRMs transfer bidirectionally across thinking modes.* Whether trained with slow- or fast-thinking sampling, the resulting gains generalize to the opposite mode with only minor loss.

Underlying rationale. This cross-mode bidirectional generalization indicates that the core reasoning skills enhanced by RL are not bound to the surface format of the reasoning trajectory, but instead contain a substantial mode-agnostic component shared by fast and slow thinking. Although the two modes are phenotypically different, they draw upon a shared substrate of fundamental reasoning skills, such as problem decomposition, logical verification, knowledge retrieval, and strategic planning. RL with either fast- or slow-thinking sampling strengthens this shared substrate, which is why the gains acquired in one mode remain accessible when the model is prompted to operate in the other. This view aligns with dual-process theories of human cognition [16], which posit that fast and slow thinking are dual expressions of a unified cognitive architecture rather than disjoint systems.

Implication: a viable path to acceleration. From the perspective of training efficiency, this finding has a transformative implication: fast-thinking sampling can serve as an efficient proxy for slow-thinking sampling in RL. Replacing costly slow-thinking trajectories with concise fast-thinking trajectories dramatically accelerates RL training while delivering similar or larger performance gains. This alleviates the core efficiency bottleneck of RL for LRMs, shortens development cycles, and lowers the resource barrier for broader participation from the research community.

2.5 Remaining Challenges

While fast-thinking online sampling offers great potential to accelerate RL, the simple implementation in Sec. 2.3, which relies solely on the prompting trick, is inherently unreliable and suffers from two limitations that hinder its practical applicability.

- **Limitation 1 (unreliable fast-thinking sampling).** As shown in Sec. 2.1, the prompting trick has a low and unstable success rate. Consequently, many slow-thinking trajectories are still sampled during training, preventing the acceleration scheme from reaching its full potential.
- **Limitation 2 (test-time mode contamination).** Training exclusively on fast-thinking trajectories can compromise the model's native slow-thinking behavior. Since RL increases the sampling probability of high-reward trajectories and suppresses low-reward ones, optimizing solely on fast-thinking trajectories inadvertently reinforces fast-thinking. The RL-finetuned model may then skip the deliberation phase even under a standard prompt (question only), violating our core principle that acceleration must not compromise the model's native slow-thinking reasoning.

To fully leverage proxy fast-thinking sampling for RL acceleration, we need a principled framework that simultaneously (1) enables stable and efficient elicitation of fast-thinking behavior during training, addressing Limitation 1, and (2) preserves the model's native slow-thinking behavior, addressing Limitation 2. This motivates our two-stage framework, Fast-RL, which satisfies both conditions through a carefully designed separation between thinking-mode control and reasoning-capability optimization (Sec. 3).

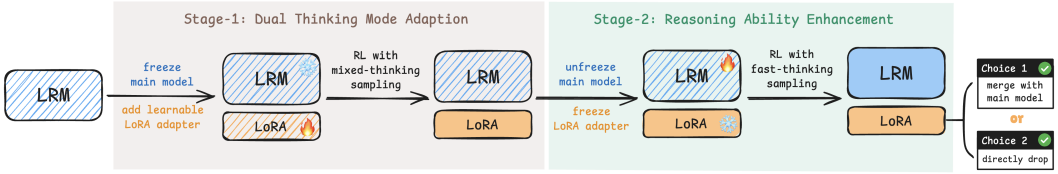


Figure 5: Overview of Fast-RL, a two-stage framework for accelerating RL training of Large Reasoning Models (LRMs).

3 Fast-RL: A Two-Stage Robust and Efficient RL Framework

To overcome the limitations identified in Sec. 2.5, we propose *Fast-RL*, a two-stage framework that enables efficient RL for LRMs as illustrated in Figure 5. Before presenting its technical details in Sec. 3.2 and Sec. 3.3, we first articulate the core design principles in Sec. 3.1, providing an intuitive explanation of how each limitation is addressed.

3.1 Design Principles

Our design starts by considering independent solutions for each of the two limitations identified in Sec. 2.5.

Resolving unreliable sampling (for Limitation 1). The unreliability of the prompting trick does not stem from the trick itself, but from its training-free use. To enhance reliability and achieve deterministic control, the natural solution is to fine-tune the model so as to explicitly strengthen its internal mapping from prompt structure to thinking-mode selection.

Preventing mode contamination (for Limitation 2). The root cause of mode contamination is the implicit coupling between reasoning-ability optimization and thinking-mode preference enhancement. Under our acceleration principle, all trajectories used for policy optimization are fast-thinking, so the optimization objective unavoidably reinforces fast-thinking behavior. Explicitly disentangling the mode-preference effect from the optimization objective or policy gradient is non-trivial. We instead decouple the two functions *at the parameter level*: we spatially isolate the parameters responsible for thinking-mode selection from the rest of the model. With this separation, we can *freeze* the mode-selection parameters during fast-thinking RL, filtering out the unintended reinforcement of fast-thinking behavior and preserving the learned prompt-to-mode mapping.

Overview. The two strategies above, making thinking-mode selection learnable and then freezing mode-selection parameters during reasoning optimization, naturally converge into a unified two-stage framework: (1) ***Stage 1*** optimizes the model's ability to select the appropriate thinking mode based on the prompt structure, addressing Limitation 1. Empirically, this skill is acquired in only tens of training steps with minimal overhead. (2) ***Stage 2*** enhances reasoning ability and improves training efficiency by performing *exclusive* fast-thinking online sampling, leveraging the insight from Sec. 2. Freezing the mode-selection parameters in this stage prevents mode contamination, resolving Limitation 2.

3.2 Stage 1: Dual Thinking Mode Adaptation

This stage equips the model with a reliable and prompt-conditioned thinking-mode selection ability. The model should consistently generate slow-thinking trajectories under a standard prompt (question only), and fast-thinking responses under the trick-augmented prompt (question followed by "`<think>\n\n</think>\n`"). This capability is essential for deterministic fast-thinking sampling in Stage 2 while preserving the native slow-thinking behavior.

To this end, we perform **LoRA fine-tuning** while freezing the main model. Compared with full fine-tuning, LoRA is critical for two reasons: **(1). Preservation of original capability.** Mode selection is a simple task that can be learned from a small amount of data. Full fine-tuning on such limited data risks a catastrophic distributional shift, potentially degrading the model's pre-trained knowledge. Freezing the main model and updating only a lightweight LoRA adapter maximally preserves the model's original reasoning ability. **(2). Isolation of mode-control parameters.** As

discussed in Sec. 3.1, the mode-selection parameters must be frozen in Stage 2. With full fine-tuning, the mode-related parameters would be dispersed across the entire model, requiring complex attribution analysis to identify them, a task that is theoretically complex, computationally expensive, and practically unreliable. With LoRA, the adapter naturally serves as a dedicated module for mode selection, eliminating this isolation challenge. The LoRA adapter is *not* merged into the main model after Stage 1, since it requires further treatment in Stage 2.

We adopt reinforcement learning (RL) rather than supervised fine-tuning (SFT), as it offers higher sample efficiency and stronger generalization. During online sampling, we use a mixed prompting strategy: with probability β we use a slow-thinking prompt P_{slow} (question only), and with probability $1-\beta$ a fast-thinking prompt P_{fast} (question with "`<think>\n\n</think>\n`"). In practice β is small (*e.g.*, ≤ 0.25), since the model already exhibits robust slow-thinking under P_{slow} and only a minimal rehearsal signal is needed to preserve it. The remaining budget is allocated to fast-thinking prompts, whose triggering success rate is the primary target of Stage 1.

The reward signal in this stage optimizes only for mode correctness, ignoring final-answer accuracy and reasoning quality. We define a binary mode reward R_{mode}: let $f(\tau) \in \{0, 1\}$ indicate whether trajectory τ is in fast-thinking format ($f(\tau)=1$) or slow-thinking format ($f(\tau)=0$). The reward is then

$$R_{\text{mode}}(\tau, P) = \begin{cases} 1 - f(\tau) & \text{if } P = P_{\text{slow}} \\ f(\tau) & \text{if } P = P_{\text{fast}} \end{cases} \tag{1}$$

The LoRA adapter is updated via GRPO [26] to maximize $\mathbb{E}[R_{\text{mode}}]$. With this mode-only objective and no supervision on answer correctness or reasoning quality, the model rapidly acquires reliable prompt-conditioned mode selection within tens of training steps. Remarkably, the model's reasoning ability remains well-preserved after this stage even without a correctness reward. This stability stems from two factors: (i) LoRA introduces minimal perturbations to the frozen base model, and (ii) RL exhibits strong generalization that helps avoid overfitting and capability degradation.

3.3 Stage 2: Reasoning Ability Enhancement

Stage 2 is designed primarily for **training acceleration**. We replace computationally expensive slow-thinking online sampling with efficient fast-thinking sampling, drastically reducing training overhead. Leveraging the prompt-conditioned mode control established in Stage 1, we append the "`<think>\n\n</think>\n`" token to each query, prompting the model to bypass exploratory reasoning and directly generate fast-thinking trajectories for policy optimization. These trajectories are optimized using standard reward signals (e.g., binary outcome rewards or fine-grained process rewards). Crucially, this acceleration mechanism is algorithm-agnostic and seamlessly compatible with mainstream policy-gradient methods such as PPO [25], GRPO [26], and DAPO [33].

A critical architectural choice in Stage 2 is to **freeze the LoRA adapter** learned in Stage 1 and update only the base model parameters. This serves two essential purposes: (i) it shields the prompt-conditioned mode-selection mechanism from RL gradients, preventing the mode contamination identified in Sec. 2.5; and (ii) it ensures that all reasoning improvements are encoded directly into the base model rather than being absorbed by the adapter.

Upon completion of Stage 2, the LoRA adapter functions as a disposable **training scaffold**. It can be merged into the backbone to retain dynamic mode-switching, or safely discarded without degrading the enhanced slow-thinking reasoning capabilities. This confirms a clean functional separation: the base model absorbs all reasoning gains, while the LoRA solely manages prompt-driven mode control. Comprehensive ablations validating each design choice are provided in Appendix C.4.

4 Experiments

4.1 Settings

Our experiments cover both multimodal and text-only LRMs. **Models:** For multimodal LRMs we use LLaVA-CoT-11B [31] and Vision-R1-7B [15], two popular and strong models. For text-only LRMs we use the slow-thinking DeepSeek-R1-Distill-Qwen models [11] (1.5B and 8B) and the natively hybrid-mode Qwen3 models [32] of various sizes (0.6B, 1.7B, 4B, 8B). **Baseline:** We compare *Fast-RL* against *vanilla RL*, the standard RL scheme for LRMs that employs the default

slow-thinking online sampling. **Training:** Unless otherwise specified, multimodal LRMs are trained on a 75K dataset combining VerMulti-65K [24] and Vision-R1-10K [15]; text-only LRMs are trained on DeepScaleR-40K [22]. For Qwen3 models, which natively support stable fast-thinking, we skip Stage 1 and directly perform Stage 2 without LoRA. More configurations are detailed in Appendix B.3 and Appendix B.4. **Evaluation:** We evaluate multimodal models on three multimodal math benchmarks and two general multimodal benchmarks. Text-only models are evaluated on seven widely used challenging math benchmarks. Details are in Appendix B.5.

4.2 Main Results

4.2.1 Training Acceleration

As shown in Figure 1, Fast-RL accelerates RL training in two complementary ways. **(1). Reduced time per step.** This is the primary source of acceleration. Fast-thinking trajectories are short-form (*e.g.*,, $<$2K tokens), in stark contrast to slow-thinking ones ($\sim$8K–32K tokens). Collecting a batch of trajectories for a policy update is therefore $5 \sim 8\times$ faster. **(2). Fewer steps to convergence.** Fast-RL often requires fewer training steps to converge, especially under outcome-based rewards. We attribute this to the higher signal-to-noise ratio provided by concise and deterministic fast-thinking trajectories. Appendix C.1.2 provides more analysis. We further provide speed analysis in Appendix C.1.1, including comparisons across different models and the detailed training cost of each stage in Fast-RL.

4.2.2 Superior Performance Gains

Beyond acceleration, Fast-RL consistently delivers larger performance gains than vanilla slow-thinking RL across all models and benchmarks. This indicates that Fast-RL is not only an accelerator but also a more effective and robust training framework that yields stronger reasoning ability at lower cost.

For multimodal LRMs, Table 1 shows that Fast-RL significantly outperforms both the base model and vanilla RL. On LLaVA-CoT-11B, Fast-RL reaches an average accuracy of 45.1 ($+4.5$ over the base model and $+2.8$ over vanilla RL), with particularly strong gains on MathVista ($+9.4$ over baseline, $+5.9$ over vanilla RL) and MMStar ($+4.3$ over baseline, $+2.5$ over vanilla RL). On Vision-R1-7B, vanilla RL barely brings any meaningful improvement ($+0.2$ on average), whereas Fast-RL delivers a robust $+2.1$ gain across the five benchmarks.

For text-only LRMs, the gains are even more pronounced. Table 2 shows that Fast-RL achieves a $+9.4$ average gain on DeepSeek-R1-Distill-1.5B, more than doubling the $+3.8$ gain of vanilla RL. The results on Qwen3 models are particularly striking. As shown in Figure 1, on Qwen3-0.6B Fast-RL obtains an average slow-thinking gain of $+9.4$ points, nearly triple the $+3.4$ gain of vanilla RL, with exceptional improvements on challenging benchmarks such as AIME24 ($+12.3$ over baseline, $+9.4$ over vanilla RL) and AMC23 ($+12.5$ over baseline, $+6.2$ over vanilla RL). On Qwen3-1.7B, vanilla RL even degrades slow-thinking performance by -2.0 on average. In contrast, Fast-RL improves both slow- and fast-thinking performance by $+4.4$ and $+11.4$ points, respectively. Appendix C.5 contains more results for larger baselines.

4.2.3 Unlocking Vanilla RL Convergence Bottleneck

Interestingly, for a model already converged with vanilla RL on a given dataset, applying Fast-RL on the same dataset can further boost its performance. This indicates that Fast-RL can overcome

Table 1: Comparison of **multimodal LRMs** trained under different RL pipelines with GRPO as core policy update algorithm.

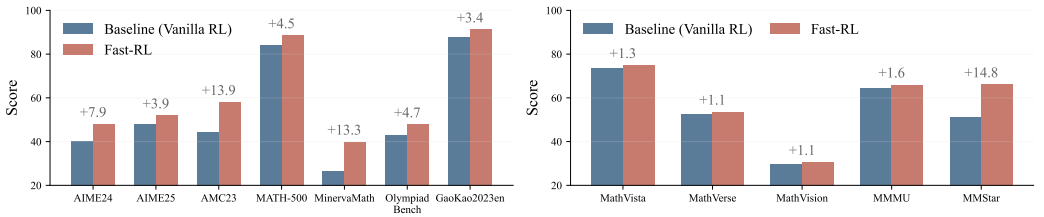
Model	MathVista	MathVerse	MathVision	MMMU	MMStar	Avg
LLaVA-CoT-11B	54.8	24.2	17.6	48.9	57.6	40.6
+ vanilla RL	58.3 (+3.5)	26.0 (+1.8)	18.4 (+0.8)	49.2 (+0.3)	59.4 (+1.8)	42.3 (+1.7)
+ Fast-RL	64.2 (+9.4)	28.8 (+4.6)	19.1 (+1.5)	51.3 (+2.4)	61.9 (+4.3)	45.1 (+4.5)
Vision-R1-7B	73.5	52.4	29.4	54.2	64.3	54.8
+ vanilla RL	74.0 (+0.5)	52.0 (-0.6)	29.8 (+0.4)	55.0 (+0.8)	64.4 (-0.4)	55.0 (+0.2)
+ Fast-RL	75.5 (+2.0)	53.6 (+1.2)	31.3 (+1.9)	57.0 (+2.8)	67.0 (+2.7)	56.9 (+2.1)

Table 2: Comparison of **text-only LRMs** trained under different RL pipelines with GRPO as core policy update algorithm. We use DS-1.5B to denote the model DeepSeek-R1-Distill-Qwen-1.5B.

Model	AIME24	AIME25	AMC23	MATH-500	MinervaMath	OlympiadBench	GaoKao2023en	Avg
DS-1.5B	28.9	22.8	62.9	83.9	26.5	43.3	73.1	48.8
+ vanilla RL	35.8 (+6.9)	24.5 (+1.7)	70.3 (+7.4)	85.2 (+1.3)	30.1 (+3.6)	47.1 (+3.8)	75.3 (+2.2)	52.6 (+3.8)
+ Fast-RL	39.3 (+10.4)	30.6 (+7.8)	77.5 (+14.6)	88.9 (+5.0)	41.4 (+14.9)	50.6 (+7.3)	79.4 (+6.3)	58.2 (+9.4)

Table 3: Compatibility of Fast-RL with PPO [25] and DAPO [33]. All experiments use DeepSeek-R1-Distill-Qwen-1.5B as the baseline and DeepScaleR-40K as the training dataset.

Model	AIME24	AIME25	AMC23	MATH-500	MinervaMath	OlympiadBench	GaoKao2023en
DeepSeek-R1-Distill-Qwen-1.5B	28.9	22.8	62.9	83.9	26.5	43.3	73.1
+ Vanilla RL (PPO)	31.3	20.5	72.5	83.1	33.3	40.5	72.4
+ Fast-RL (PPO)	35.8	24.6	75.4	86.8	39.2	45.4	78.5
+ Vanilla RL (DAPO)	36.1	25.0	70.8	85.1	30.9	47.5	76.0
+ Fast-RL (DAPO)	41.5	32.3	78.9	89.2	41.9	51.1	80.0



(a) DeepScaleR-1.5B-Preview **(b) Vision-R1-7B**

Figure 6: Fast-RL further boosts the performance of models already converged under vanilla RL, even using the same training data.

the bottleneck at which vanilla RL converges. As shown in Figure 6, DeepScaleR-1.5B-Preview, a model trained to convergence with vanilla slow-thinking RL on DeepScaleR-40K, gains an additional $+6.7$ on average when continued with Fast-RL on the same dataset, with notable improvements on MinervaMath ($+13.1$) and AIME24 ($+7.9$). Similarly, Vision-R1-7B, converged under vanilla RL on Vision-R1-10K, gains an extra $+1.7$ on average from continued Fast-RL training on the same data. Fast-RL is therefore not just a faster alternative but also a complementary optimization tool. More discussions and experiments are provided in Appendix C.2.

4.3 Further Study

Compatibility Test. Fast-RL is a general framework compatible with a wide range of policy-optimization algorithms. As shown in Table 3, Fast-RL consistently outperforms the vanilla slow-thinking RL baseline across both algorithms. With DAPO, Fast-RL improves AIME24 from 36.1 to 41.5 ($+5.4$) and AIME25 from 25.0 to 32.3 ($+7.3$); similar gains are observed with PPO. This confirms that Fast-RL serves as a plug-and-play acceleration module compatible with diverse policy-optimization strategies.

Ablation Study. We thoroughly ablate each key design in Fast-RL, including the contribution of each stage, the parameter-isolation mechanism, the alternating parameter freeze, and the disposition of the LoRA adapter after Stage 2. Full results and analyses are provided in Appendix C.4.

5 Conclusion

Fast-RL addresses the prohibitive computational bottleneck of RL for slow-thinking LRMs by establishing a foundational insight: fast-thinking trajectories serve as highly efficient and effective proxies for reasoning optimization. By decoupling thinking-mode control from reasoning updates via a frozen, lightweight LoRA scaffold, our framework accelerates training while rigorously preserving native slow-thinking capabilities. Extensive experiments demonstrate that Fast-RL achieves a $\sim 10\times$ speedup and delivers consistent, superior performance gains across diverse text-only and multimodal architectures, in stark contrast to the degradation often observed in vanilla slow-thinking RL.

Acknowledgements

This work was supported in part by the Research Grants Council under the Areas of Excellence scheme grant AoE/E-601/22-R. The work has been supported by Hong Kong Research Grant Council-General Research Fund Scheme (Grant No. 17202422, 17212923, 17215025), Theme-based Research (Grant No. T45-701/22-R), and Strategic Topics Grant (Grant No. STG3/E-605/25-N). Part of the described research work is conducted in the JC STEM Lab of Robotics for Soft Materials, funded by The Hong Kong Jockey Club Charities Trust.

References

[1] Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., Hooker, S.: Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. arXiv preprint arXiv:2402.14740 (2024)

[2] AIME: American invitational mathematics examination (2024), https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions

[3] AMC: American mathematics competitions (2023), https://artofproblemsolving.com/wiki/index.php/2023_AMC_12A

[4] Chen, L., Li, J., Dong, X., Zhang, P., Zang, Y., Chen, Z., Duan, H., Wang, J., Qiao, Y., Lin, D., et al.: Are we on the right way for evaluating large vision-language models? Advances in Neural Information Processing Systems **37**, 27056–27087 (2024)

[5] Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al.: Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261 (2025)

[6] Cui, G., Yuan, L., Wang, Z., Wang, H., Zhang, Y., Chen, J., Li, W., He, B., Fan, Y., Yu, T., et al.: Process reinforcement through implicit rewards. arXiv preprint arXiv:2502.01456 (2025)

[7] Fei, W., Kong, H., Liang, S., Lin, Y., Yang, Y., Tang, J., Chen, L., Hua, X.: Self-guided process reward optimization with redefined step-wise advantage for process reinforcement learning. arXiv e-prints pp. arXiv–2507 (2025)

[8] Fu, W., Gao, J., Shen, X., Zhu, C., Mei, Z., He, C., Xu, S., Wei, G., Mei, J., Wang, J., et al.: Areal: A large-scale asynchronous reinforcement learning system for language reasoning. arXiv preprint arXiv:2505.24298 (2025)

[9] Gao, Z., Kim, J., Sun, W., Joachims, T., Wang, S., Pang, R.Y., Tan, L.: Prompt curriculum learning for efficient llm post-training. arXiv preprint arXiv:2510.01135 (2025)

[10] GaoKao, C.: Gaokao2023en (2024), https://huggingface.co/datasets/MARIO-Math-Reasoning/Gaokao2023-Math-En

[11] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948 (2025)

[12] He, C., Luo, R., Bai, Y., Hu, S., Thai, Z.L., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., et al.: Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. arXiv preprint arXiv:2402.14008 (2024)

[13] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., Steinhardt, J.: Measuring mathematical problem solving with the math dataset. arXiv preprint arXiv:2103.03874 (2021)

[14] Hu, J.: Reinforce++: A simple and efficient approach for aligning large language models. arXiv preprint arXiv:2501.03262 (2025)

[15] Huang, W., Jia, B., Zhai, Z., Cao, S., Ye, Z., Zhao, F., Xu, Z., Hu, Y., Lin, S.: Vision-r1: Incentivizing reasoning capability in multimodal large language models. arXiv preprint arXiv:2503.06749 (2025)

[16] Kahneman, D.: Thinking, fast and slow. Penguin, London (2012)

[17] Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al.: Solving quantitative reasoning problems with language models. Advances in neural information processing systems **35**, 3843–3857 (2022)

[18] Li, Y., Liu, Z., Li, Z., Zhang, X., Xu, Z., Chen, X., Shi, H., Jiang, S., Wang, X., Wang, J., Huang, S., Zhao, X., Jiang, B., Hong, L., Wang, L., Tian, Z., Huai, B., Luo, W., Luo, W., Zhang, Z., Hu, B., Zhang, M.: Perception, reason, think, and plan: A survey on large multimodal reasoning models. arXiv preprint arXiv:2505.04921 (2025)

[19] Li, Z., Xu, T., Zhang, Y., Yu, Y., Sun, R., Luo, Z.Q.: Remax: A simple, effective, and efficient method for aligning large language models. arXiv preprint arXiv:2310.10505 (2023)

[20] Liu, L., Yao, F., Zhang, D., Dong, C., Shang, J., Gao, J.: Flashrl: 8bit rollouts, full power rl (Aug 2025), https://fengyao.notion.site/flash-rl

[21] Lu, P., Bansal, H., Xia, T., Liu, J., Li, C., Hajishirzi, H., Cheng, H., Chang, K.W., Galley, M., Gao, J.: Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. arXiv preprint arXiv:2310.02255 (2023)

[22] Luo, M., Tan, S., Wong, J., Shi, X., Tang, W.Y., Roongta, M., Cai, C., Luo, J., Li, L.E., Popa, R.A., Stoica, I.: Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2 (2025), notion Blog

[23] OpenAI: Introducing openai o3 and o4-mini (2025), https://openai.com/index/introducing-o3-and-o4-mini/

[24] Peng, Y., Zhang, G., Zhang, M., You, Z., Liu, J., Zhu, Q., Yang, K., Xu, X., Geng, X., Yang, X.: Lmm-r1: Empowering 3b lmms with strong reasoning abilities through two-stage rule-based rl. arXiv preprint arXiv:2503.07536 (2025)

[25] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[26] Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al.: Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300 (2024)

[27] Tang, X., Zhang, Z., Liu, Y., Zhao, W.X., Wen, Z., Zhang, Z., Zhou, J.: Towards high data efficiency in reinforcement learning with verifiable reward. arXiv preprint arXiv:2509.01321 (2025)

[28] Wang, K., Pan, J., Shi, W., Lu, Z., Ren, H., Zhou, A., Zhan, M., Li, H.: Measuring multimodal mathematical reasoning with math-vision dataset. Advances in Neural Information Processing Systems **37**, 95095–95169 (2024)

[29] Xie, G., Shi, Y., Tian, H., Yao, T., Zhang, X.: Capo: Towards enhancing llm reasoning through verifiable generative credit assignment. arXiv e-prints pp. arXiv–2508 (2025)

[30] Xu, F., Hao, Q., Shao, C., Zong, Z., Li, Y., Wang, J., Zhang, Y., Wang, J., Lan, X., Gong, J., et al.: Toward large reasoning models: A survey of reinforced reasoning with large language models. Patterns **6**(10) (2025)

[31] Xu, G., Jin, P., Wu, Z., Li, H., Song, Y., Sun, L., Yuan, L.: Llava-cot: Let vision language models reason step-by-step. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2087–2098 (2025)

[32] Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report. arXiv preprint arXiv:2505.09388 (2025)

[33] Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al.: Dapo: An open-source llm reinforcement learning system at scale. arXiv preprint arXiv:2503.14476 (2025)

[34] Yue, X., Ni, Y., Zhang, K., Zheng, T., Liu, R., Zhang, G., Stevens, S., Jiang, D., Ren, W., Sun, Y., et al.: Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9556–9567 (2024)

[35] Zhang, R., Jiang, D., Zhang, Y., Lin, H., Guo, Z., Qiu, P., Zhou, A., Lu, P., Chang, K.W., Qiao, Y., et al.: Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In: European Conference on Computer Vision. pp. 169–186. Springer (2024)

[36] Zhang, R., Arora, D., Mei, S., Zanette, A.: Speed-rl: Faster training of reasoning models via online curriculum learning. arXiv preprint arXiv:2506.09016 (2025)

[37] Zhang, Y., Lv, N., Wang, T., Dang, J.: Fastgrpo: Accelerating policy optimization via concurrency-aware speculative decoding and online draft learning. arXiv preprint arXiv:2509.21792 (2025)

[38] Zheng, H., Zhou, Y., Bartoldson, B.R., Kailkhura, B., Lai, F., Zhao, J., Chen, B.: Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. arXiv preprint arXiv:2506.02177 (2025)

Broader impacts

This paper introduces Fast-RL, a reinforcement learning framework that drastically accelerates the post-training of large reasoning models by leveraging efficient fast-thinking trajectories as high-quality proxies for slow-thinking optimization. By reducing training compute and iteration cycles by approximately an order of magnitude, our approach significantly lowers energy consumption and democratizes access to advanced LRM development, enabling faster deployment of reasoning-capable systems for scientific discovery, education, and complex decision support. Nevertheless, like any capability-advancing AI methodology, this efficiency gain carries dual-use implications: lowering the computational barrier to training sophisticated reasoning models could theoretically facilitate the development of systems optimized for malicious applications, such as automated cyber-offense, large-scale disinformation generation, or evasion of safety alignment filters. In light of these risks, we strongly advocate for parallel advancements in scalable oversight and robust alignment protocols, and urge the research community and policymakers to establish collaborative governance frameworks that ensure accelerated reasoning capabilities are developed and deployed responsibly, maximizing societal utility while proactively mitigating potential harms.

Limitations

A current limitation lies in the scope of domain verification. We have demonstrated robust gains across diverse multimodal and text-only benchmarks in math and general reasoning, but Fast-RL has not yet been extensively validated on open-ended domains such as creative writing or complex role-playing. In such subjective domains, "correct reasoning" is less rigorously defined and the boundary between fast and slow thinking is more ambiguous. Extending the fast-thinking-as-proxy insight to these broader scenarios is an important direction for future work.

Ethics Statement

This research adheres strictly to established guidelines for responsible AI development. Our work focuses on improving the computational efficiency and convergence stability of reinforcement learning for large reasoning models, without introducing new data collection pipelines. All experiments utilize exclusively publicly available benchmarks and datasets; we do not collect, process, or store any personally identifiable information (PII) or sensitive user data. We acknowledge the dual-use nature of accelerated reasoning capabilities and emphasize the importance of deploying such optimizations alongside robust safety and alignment frameworks.

Reproducibility Statement

We provide complete implementation details to ensure independent verification, including model configurations, training pipelines, optimization hyperparameters, and computational environments, all documented in the main text and Appendix. To facilitate replication and future research, we will publicly release the full source code, training scripts, and configuration files upon paper acceptance. All reported results can be reproduced using the provided artifacts and standard academic computing resources.

LLM Usage Statement

During the preparation of this manuscript, large language models were used exclusively for editorial purposes, including typo correction, grammar checking, and stylistic polishing. All research conceptualization, data analysis, results interpretation, and scientific writing were performed independently by the authors. The authors assume full responsibility for the accuracy, integrity, and originality of this work.

A Thinking Modes of Large Reasoning Models

Analogous to human System 1 cognition [16], the *fast-thinking* mode operates as an intuitive process in which the model directly produces a concise and deterministic chain-of-thought (CoT) trajectory toward the final answer. While efficient, it often falls short on tasks that require deep planning, multi-step logical reasoning, or extensive information integration.

In contrast, the *slow-thinking* mode aligns with human System 2 cognition [16] and emphasizes deliberate, reflective reasoning. The model produces a long-form CoT trajectory composed of two distinct phases: (i) an exploratory phase, typically enclosed within "`<think>`" and "`</think>`" tags, in which the model explores alternative reasoning paths, verifies hypotheses, and performs self-correction; and (ii) a solution phase, enclosed within "`<answer>`" and "`</answer>`" tags, in which the final answer is constructed based on the preceding exploration. This slow-thinking paradigm has demonstrated substantially superior performance over its fast-thinking counterpart and has become the de facto standard for state-of-the-art large reasoning models (LRMs).

B Experimental Settings

B.1 Setup for Sec. 2.2 in the Main Paper

We adopt two representative LRMs as baselines: the multimodal Vision-R1-7B [15] and the text-only DeepSeek-R1-Distill-Qwen-1.5B [11]. Each baseline is fine-tuned with GRPO [26] under slow-thinking online sampling, yielding the corresponding RL-finetuned variants. We then evaluate both fast- and slow-thinking performance for all models.

The multimodal LRM is trained on the Vision-R1 dataset [15], while the text-only LRM is trained on the DeepScaleR-40K dataset [22]. To mitigate sampling randomness, we generate K trajectories per question for each thinking mode and report the average pass@1 accuracy ($K = 8$ by default). Slow-thinking evaluation uses a standard prompt containing only the question, while fast-thinking evaluation employs the prompting trick described in Sec. 2.1 of the main paper. Since this trick does not always elicit fast-thinking behavior, we repeatedly sample for each question, discarding any reverted slow-thinking trajectories, until K valid fast-thinking trajectories are collected. Other RL hyper-parameters follow the vanilla RL setting in Appendix B.3 and Appendix B.4.

B.2 Setup for Sec. 2.3 in the Main Paper

We use the same baselines, Vision-R1-7B [15] and DeepSeek-R1-Distill-Qwen-1.5B [11], but fine-tune them with GRPO [26] under *fast-thinking* online sampling. The prompting trick from Sec. 2.1 is used to elicit fast-thinking trajectories, and we repeatedly resample until M valid fast-thinking trajectories are obtained per question ($M = 8$ by default; slow-thinking trajectories are discarded). The training data and evaluation protocol are identical to those of Sec. 2.2; other RL hyper-parameters follow Stage 2 of Fast-RL in Appendix B.3 and Appendix B.4.

B.3 Training Setup for Multimodal LRMs

All multimodal models are trained for one epoch with a global batch size of 128 and a constant learning rate of 1×10^{-6}. We sample $G = 8$ responses per prompt with temperature 1.0 and top-p 1.0. Weight decay is set to 0.01 and the KL coefficient is set to 0.001.

B.4 Training Setup for Text-only LRMs

Shared settings. For both vanilla RL and Fast-RL, we sample $G = 8$ responses per prompt with temperature 1.0 and top-p 1.0. The batch size is 256 across all models, with a constant learning rate of 1×10^{-6}, weight decay 0.01, and KL coefficient 0.001.

Method-specific settings. For vanilla RL with slow-thinking online sampling, the maximum output length is 16,384 tokens, and we train until convergence: 800 steps for DeepSeek-R1-Distill-Qwen-1.5B, 500 steps for DeepSeek-R1-Distill-Qwen-7B, and 250 steps for Qwen3 models. For Stage 1 of Fast-RL, all models are trained for 50 steps with a maximum output length of 16,384 tokens. For Stage 2 of Fast-RL, the maximum output length is reduced to 4,096 tokens, and we train until

convergence: 350 steps for DeepSeek-R1-Distill-Qwen-1.5B, 250 steps for DeepSeek-R1-Distill-Qwen-7B, and 100 steps for Qwen3 models.

B.5 Evaluation Settings

For multimodal models, we evaluate on three multimodal math benchmarks (MathVista [21], MathVerse [35], and MathVision [28]) and two general multimodal benchmarks (MMMU [34] and MMStar [4]). For text-only models, we evaluate on seven challenging math benchmarks: AIME24, AIME25 [2], AMC23 [3], MATH-500 [13], MinervaMath [17], OlympiadBench [12], and GaoKao2023en [10]. For each test instance, we sample 8 responses with temperature 0.6 and top-p 0.95, and report the average `pass@1` accuracy.

Table 4: Detailed training-time breakdown on DeepSeek-R1-Distill-Qwen-1.5B and 7B. "Stage 1 Time" refers to the mode-adaptation phase (Fast-RL only). "Time per Step" is the average wall-clock time per RL step. "Total Time" is the wall-clock time of the entire training process.

Model	Method	Stage 1 Time (GPU h)	Stage 2 Time per Step (s)	Stage 2 Steps to Converge	Stage 2 Time (GPU h)	Total Time (GPU h)	Speedup
1.5B	Vanilla RL	-	403	800	716.4	716.4	-
	Fast-RL	21.1	66	350	51.3	72.4	**9.9**×
7B	Vanilla RL	-	2354	500	2615.6	2615.6	-
	Fast-RL	47.2	282	250	156.7	203.9	**12.8**×

C More Experiments

C.1 Speed Analysis

C.1.1 Speed Comparison across Different Model Scales

Table 4 provides a fine-grained breakdown of the training-time costs for DeepSeek-R1-Distill-Qwen-1.5B and 7B. Three observations are worth highlighting.

- **Negligible Stage 1 overhead.** The cost of Stage 1 is small relative to the total training budget (e.g., $\sim$2% on the 7B model). Our two-stage design therefore introduces virtually no "alignment tax".
- **Drastic per-step time reduction.** The dominant source of acceleration is the reduction in sampling time per step: fast-thinking trajectories are substantially shorter than slow-thinking ones, and the gap is amplified by RL algorithms that require multiple samples per prompt.
- **Acceleration scales with model size.** The speedup grows from 9.9× on the 1.5B model to 12.8× on the 7B model, since larger models tend to produce longer slow-thinking chains and therefore benefit more from bypassing them. This indicates that Fast-RL is especially well-suited for training large-scale reasoning models.

C.1.2 Analysis of Faster Convergence

Beyond per-step savings, Fast-RL also requires fewer total training steps to reach the same or better performance. The reason is that fast-thinking trajectories, being concise and deterministic, yield a higher signal-to-noise ratio for policy optimization: each sample delivers a clearer reward signal with less redundant exploration. In vanilla RL, slow-thinking trajectories are long-form and highly exploratory, so two trajectories that reach the same correct answer may follow vastly different reasoning paths. The resulting trajectory variance translates into high-variance gradients that slow convergence. Consequently, Fast-RL attains comparable or superior performance in fewer steps than vanilla slow-thinking RL.

C.2 Relation between Fast- and Slow-Thinking Online Sampling

In the main paper, we position fast-thinking sampling as an efficient proxy for slow-thinking sampling. A natural question follows: are the two modes mutually exclusive, or are they complementary?

To investigate this, we adopt a sequential training strategy: starting from a model already converged with Fast-RL (Stage 2), we continue training it with slow-thinking online sampling.

As shown in Table 6, this brief slow-thinking phase yields a small further gain on top of the Fast-RL baseline. For example, on DeepSeek-R1-Distill-Qwen-1.5B, the AIME24 accuracy improves from

Table 5: Compatibility of Fast-RL with PPO [25] and DAPO [33]. All experiments use DeepSeek-R1-Distill-Qwen-1.5B as the baseline and DeepScaleR-40K as the training dataset. Evaluation is conducted on seven math reasoning benchmarks.

Model	AIME24	AIME25	AMC23	MATH-500	MinervaMath	OlympiadBench	GaoKao2023en
DeepSeek-R1-Distill-Qwen-1.5B	28.9	22.8	62.9	83.9	26.5	43.3	73.1
+ Vanilla RL (PPO)	31.3	20.5	72.5	83.1	33.3	40.5	72.4
+ Fast-RL (PPO)	35.8	24.6	75.4	86.8	39.2	45.4	78.5
+ Vanilla RL (DAPO)	36.1	25.0	70.8	85.1	30.9	47.5	76.0
+ Fast-RL (DAPO)	41.5	32.3	78.9	89.2	41.9	51.1	80.0

Table 6: Complementarity between fast-thinking and slow-thinking online sampling during RL. All experiments use DeepSeek-R1-Distill-Qwen-1.5B as the baseline and DeepScaleR-40K as the training dataset. Evaluation is conducted on seven math reasoning benchmarks.

Model	AIME24	AIME25	AMC23	MATH-500	MinervaMath	OlympiadBench	GaoKao2023en
DeepSeek-R1-Distill-Qwen-1.5B	28.9	22.8	62.9	83.9	26.5	43.3	73.1
+ Vanilla RL (slow-thinking sampling)	35.8	24.5	70.3	85.2	30.1	47.1	75.3
+ Fast-RL (fast-thinking sampling)	39.3	30.6	77.5	88.9	41.4	50.6	79.4
+ Fast-RL → Slow-thinking RL	39.8	30.7	77.5	89.1	41.5	50.8	79.9

39.3 (Fast-RL) to 39.8 (Fast-RL followed by slow-thinking RL). These results suggest that the two sampling modes are complementary:

- Fast-thinking online sampling efficiently establishes the core cognitive primitives (*e.g.,*, calculation, logical rules) and general reasoning capability via high-throughput, low-variance gradients.
- Once the core capability is in place, a brief period of slow-thinking online sampling helps the model adapt these primitives to the structural requirements of long-form exploration (*e.g.,*, complex self-correction loops or long-context management).

In summary, while Fast-RL alone is sufficient to match or surpass standard baselines, a short tail of slow-thinking sampling can further push reasoning performance.

C.3 Compatibility with Different RL Algorithms

Fast-RL is a general framework: its core acceleration mechanism, replacing slow-thinking sampling with an efficient fast-thinking proxy, is orthogonal to the underlying policy-optimization algorithm. To verify compatibility beyond GRPO, we integrate Fast-RL with two other mainstream RL algorithms, PPO [25] and DAPO [33].

As shown in Table 5, Fast-RL consistently outperforms the vanilla slow-thinking RL baseline across both algorithms. With DAPO, Fast-RL improves AIME24 from 36.1 to 41.5 ($+5.4$) and AIME25 from 25.0 to 32.3 ($+7.3$); similar gains are observed with PPO. This confirms that Fast-RL serves as a plug-and-play acceleration module compatible with diverse policy-optimization strategies.

C.4 Ablation Studies

C.4.1 Necessity of Stage 1

To validate the necessity of the "Dual Thinking-Mode Adaptation" phase (Stage 1), we compare our two-stage Fast-RL against a baseline variant termed "Fast-RL w/o Stage 1". In this variant we skip Stage 1 and directly perform fast-thinking RL (Stage 2) using the prompting trick from Sec. 2.1 to elicit fast-thinking behavior. Removing Stage 1 also necessitates removing the LoRA design, so this baseline reduces to full fine-tuning of the original model parameters.

As shown in Table 7, removing Stage 1 leads to two critical failures:

- **Compromised acceleration due to unstable sampling.** Without Stage 1 fine-tuning, the model relies solely on the raw prompting trick, whose success rate is low and unstable (around $40\% \sim 80\%$, see Figure 3 in the main paper). Consequently, $20\% \sim 60\%$ of sampled trajectories revert to the slow-thinking format, and the resulting long-form sequences consume substantial sampling time, dragging down the overall training speed. In contrast, Fast-RL (with Stage 1) achieves nearly 100% fast-thinking sampling, yielding consistent and stable fast-thinking RL and maximizing the speedup.

Table 7: Ablation on the necessity of Stage 1 in Fast-RL: "two-stage Fast-RL" versus "Fast-RL w/o Stage 1". Both experiments use DeepSeek-R1-Distill-Qwen-1.5B as the baseline and DeepScaleR-40K as the training dataset. Evaluation is conducted on seven math reasoning benchmarks. For each benchmark we report Acc@1 (problem-solving correctness) and $\mathcal{P}_{\text{slow}}$ (%) (percentage of responses following the slow-thinking format, measuring mode stability). Higher is better for both. The last column reports the proportion of slow-thinking trajectories generated during Stage 2 online sampling, where lower values indicate higher training efficiency.

Stage 1	Stage 2	Evaluation Benchmarks																Training Data
		AIME24		AIME25		AMC23		MATH-500		MinervaMath		OlympiadBench		GaoKao2023en		Avg		DeepScaleR-40K
		Acc@1	$\mathcal{P}_{\text{slow}}$ (↑)	Acc@1	$\mathcal{P}_{\text{slow}}$ (↑)	Acc@1	$\mathcal{P}_{\text{slow}}$ (↑)	Acc@1	$\mathcal{P}_{\text{slow}}$ (↑)	Acc@1	$\mathcal{P}_{\text{slow}}$ (↑)	Acc@1	$\mathcal{P}_{\text{slow}}$ (↑)	Acc@1	$\mathcal{P}_{\text{slow}}$ (↑)	Acc@1	$\mathcal{P}_{\text{slow}}$ (↑)	$\mathcal{P}_{\text{slow}}$ (↓)
✘	✔	36.6	82.1	25.8	81.3	75.0	82.5	85.8	70.4	37.3	81.9	47.3	79.6	76.4	80.0	54.9	79.7	22.0
✔	✔	39.3	100.0	30.6	100.0	77.5	100.0	88.9	100.0	41.4	100.0	50.6	100.0	79.4	100.0	58.2	100.0	0.08

Table 8: Comparison of *full fine-tuning* and *LoRA fine-tuning* for Stage 1 in Fast-RL. Both experiments use DeepSeek-R1-Distill-Qwen-1.5B as the baseline and DeepScaleR-40K as the training dataset. Evaluation is conducted on seven math reasoning benchmarks. For each benchmark we report Acc@1 (problem-solving correctness) and Mode@1 (the percentage of responses whose generated thinking mode matches the prompt instruction). Higher is better for both.

Stage 1 Fine-Tuning	AIME24		AIME25		AMC23		MATH-500		MinervaMath		OlympiadBench		GaoKao2023en		Avg	
	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1
-	28.9	73.5	22.8	75.4	62.9	83.0	83.9	84.5	26.5	75.2	43.3	72.0	73.1	80.6	48.8	77.7
Full	28.3	100.0	22.0	100.0	62.5	100.0	83.0	100.0	25.9	100.0	42.8	100.0	72.9	100.0	48.2	100.0
LoRA	28.6	100.0	23.0	100.0	62.8	100.0	83.9	100.0	26.6	100.0	43.5	100.0	73.0	100.0	48.8	100.0

Table 9: Comparison of *supervised fine-tuning (SFT)* and *reinforcement learning (RL)* for Stage 1 in Fast-RL. Both experiments use DeepSeek-R1-Distill-Qwen-1.5B as the baseline and DeepScaleR-40K as the training dataset. Evaluation is conducted on seven math reasoning benchmarks. For each benchmark we report Acc@1 (problem-solving correctness) and Mode@1 (the percentage of responses whose generated thinking mode matches the prompt instruction). Higher is better for both.

Stage 1 Fine-Tuning	AIME24		AIME25		AMC23		MATH-500		MinervaMath		OlympiadBench		GaoKao2023en		Avg	
	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1
-	28.9	73.5	22.8	75.4	62.9	83.0	83.9	84.5	26.5	75.2	43.3	72.0	73.1	80.6	48.8	77.7
SFT	25.5	92.4	20.8	95.8	60.4	93.2	82.1	95.3	23.0	90.9	39.6	92.1	68.9	89.9	45.8	92.8
RL	28.6	100.0	23.0	100.0	62.8	100.0	83.9	100.0	26.6	100.0	43.5	100.0	73.0	100.0	48.8	100.0

Table 10: Ablation on the reward design for Stage 1 in Fast-RL. Both experiments use DeepSeek-R1-Distill-Qwen-1.5B as the baseline and DeepScaleR-40K as the training dataset. Evaluation is conducted on seven math reasoning benchmarks. For each benchmark we report Acc@1 (problem-solving correctness) and Mode@1 (the percentage of responses whose generated thinking mode matches the prompt instruction). Higher is better for both.

Mode Reward R_{mode}	Correctness Reward $R_{\text{correctness}}$	AIME24		AIME25		AMC23		MATH-500		MinervaMath		OlympiadBench		GaoKao2023en		Avg	
		Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1	Acc@1	Mode@1
-	-	28.9	73.5	22.8	75.4	62.9	83.0	83.9	84.5	26.5	75.2	43.3	72.0	73.1	80.6	48.8	77.7
✔	✘	28.6	100.0	23.0	100.0	62.8	100.0	83.9	100.0	26.6	100.0	43.5	100.0	73.0	100.0	48.8	100.0
✔	✔	30.3	98.5	23.4	96.9	63.5	95.6	84.0	95.5	26.3	94.1	44.0	97.6	73.8	94.9	49.3	96.2

Table 11: Ablation on the post-Stage-2 handling strategies for the LoRA adapter in Fast-RL. Both experiments use DeepSeek-R1-Distill-Qwen-1.5B as the baseline and DeepScaleR-40K as the training dataset. Evaluation is conducted on seven math reasoning benchmarks; for each benchmark we report the pass@1 accuracy (Acc@1).

Merge	Discard	AIME24	AIME25	AMC23	MATH-500	MinervaMath	OlympiadBench	GaoKao2023en	Avg
✔	✘	39.1	30.6	77.4	89.0	41.5	50.8	79.3	58.2
✘	✔	39.3	30.6	77.5	88.9	41.4	50.6	79.4	58.2

- **Severe mode contamination.** The model trained without Stage 1 suffers from severe mode contamination: under standard prompts, it frequently fails to engage slow-thinking, resulting in inferior performance on complex reasoning tasks compared to our two-stage Fast-RL.

Stage 1 is therefore not merely a warm-up but a prerequisite for Fast-RL's efficiency and robustness: it converts a probabilistic prompting trick into a deterministic thinking-mode controller, enabling the subsequent parameter isolation that protects the model's native capabilities.

C.4.2 Impact of Stage 1 on Native Reasoning Capability

A natural concern in multi-stage training is whether the Dual Thinking-Mode Adaptation phase (Stage 1) degrades the model's native reasoning capability. We therefore evaluate the model immediately after Stage 1 (trained solely for mode control) and compare it against the original baseline.

As shown in Table 8 (rows 1 and 3), Stage 1 has a negligible impact on reasoning performance: the post-Stage-1 model is on par with the baseline across all benchmarks, with differences within the noise margin. This non-destructive adaptation is attributable to two design choices in Stage 1:

- **Minimal parameter perturbation via LoRA.** Freezing the backbone and updating only a lightweight adapter keeps the bulk of the model's pre-trained knowledge undisturbed (see Appendix C.4.3).
- **RL-based adaptation.** Compared with supervised fine-tuning (SFT), our RL-based approach better preserves the original reasoning performance (see Appendix C.4.4).

This confirms that Stage 1 establishes a reliable mode controller while serving as a lossless foundation for the subsequent reasoning enhancement in Stage 2.

C.4.3 Full vs. LoRA Fine-Tuning in Stage 1

We ablate the fine-tuning strategy (full-parameter vs. LoRA) used in Stage 1. The ideal outcome is to establish a reliable mode controller without degrading the model's pre-trained reasoning capability.

As shown in Table 8, both strategies are highly effective at the primary objective: each achieves nearly 100% Mode@1, successfully learning to switch thinking modes based on the prompt.

A clear difference, however, emerges in reasoning performance:

- **Full fine-tuning** causes a slight degradation of reasoning accuracy (Avg Acc@1 drops from 48.8 to 48.2). This indicates that updating all parameters introduces a minor distributional shift or "forgetting" of general reasoning capability.
- **LoRA fine-tuning** maintains the reasoning performance at the exact baseline level (Avg Acc@1 stays at 48.8).

LoRA is therefore the preferred choice for Stage 1: it introduces only minimal perturbations to the base model, enabling the mode-control logic while strictly preserving the original reasoning substrate. It also provides the architectural foundation for the parameter isolation ("safety lock") required in Stage 2.

C.4.4 SFT vs. RL Fine-Tuning in Stage 1

We further justify the choice of reinforcement learning (RL) over supervised fine-tuning (SFT) for the mode-adaptation phase (Stage 1). Table 9 reveals three distinct advantages of RL for this task:

- **Preservation of reasoning capability.** SFT incurs a noticeable degradation, dropping Avg Acc@1 from 48.8 (baseline) to 45.8. RL preserves the original reasoning capability (48.8), confirming that optimizing a sparse reward (mode correctness) is less disruptive than fitting specific SFT target tokens.
- **Superior generalization in mode selection.** RL achieves perfect deterministic mode control (100% Mode@1 on all benchmarks). SFT fails to generalize as cleanly (averaging 92.8%), incorrectly switching modes on a non-trivial fraction of queries.
- **Higher training efficiency.** RL converges to optimal behavior in only $\sim$50 steps, whereas SFT requires roughly 400 steps to reach comparable adherence, making RL a more lightweight choice for Stage 1.

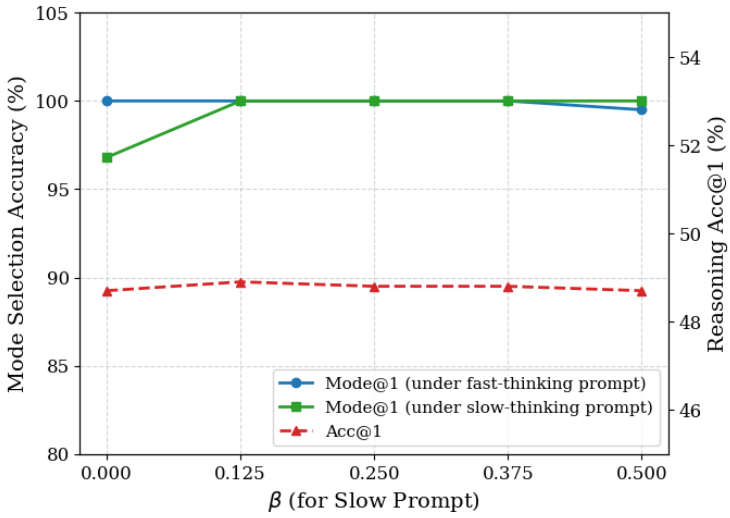


Figure 7: Ablation study on the impact of the mixing probability β (ratio of slow-thinking prompts) in Stage 1. The left axis reports the Mode Selection Accuracy (Mode@1) for eliciting fast-thinking (blue circle) and maintaining slow-thinking (green square). The right axis (red dashed line) tracks the average pass@1 reasoning accuracy across seven benchmarks.

C.4.5 Effect of the Mixing Probability β in Stage 1

As described in Sec. 3.2 of the main paper, Stage 1 employs a mixed-prompt sampling strategy: the standard slow-thinking prompt P_{slow} is sampled with probability β, and the fast-thinking prompt P_{fast} with probability $1-\beta$. Since the model's native slow-thinking capability is already robust, the primary objective in this stage is to learn the new fast-thinking trigger ($P_{\text{fast}} \rightarrow$ Fast). A smaller β is therefore preferable, as it allocates more training budget to the new task and accelerates convergence.

To verify this and select an optimal ratio, we vary β over $\{0, 0.125, 0.25, 0.375, 0.5\}$ and measure three quantities after Stage 1: (i) Mode accuracy under P_{fast} (success rate of eliciting fast-thinking), (ii) Mode accuracy under P_{slow} (success rate of maintaining slow-thinking), and (iii) average pass@1 reasoning accuracy over seven benchmarks.

As illustrated in Figure 7, the results show that:

- When $\beta = 0$ (training exclusively on fast prompts), the slow-thinking maintenance rate drops slightly to $\sim$96.8%. Adding even a small fraction of slow-thinking prompts ($\beta = 0.125$) restores it to 100%, confirming that only a minimal "rehearsal" signal is needed to prevent catastrophic forgetting of the native mode.
- The average reasoning accuracy stays stable ($\sim$48.8) across all β values, confirming that the lightweight LoRA adaptation is safe and does not compromise core reasoning capability regardless of the mixing ratio.

A small β (e.g., 0.125 or 0.25) is therefore sufficient to preserve native slow-thinking behavior without slowing the learning of the new fast-thinking mode. We use $\beta = 0.25$ by default.

C.4.6 Effect of Reward Design in Stage 1

In the main paper we state that Stage 1 optimizes exclusively for mode alignment (R_{mode}), deliberately omitting any reward for reasoning correctness ($R_{\text{correctness}}$). While adding $R_{\text{correctness}}$ is standard in RL, we justify its exclusion based on the principle of model-parameter isolation.

To analyze this, we trained a Stage 1 variant using a combined reward $R = R_{\text{mode}} + R_{\text{correctness}}$ and compared it against our default "Mode Reward Only" setting. Table 10 reports the performance of the model immediately after Stage 1.

- **Reasoning performance.** Adding $R_{\text{correctness}}$ yields an immediate reasoning boost (Avg Acc@1 rises from 48.8 to 49.3). Our default setting maintains the baseline (48.8).

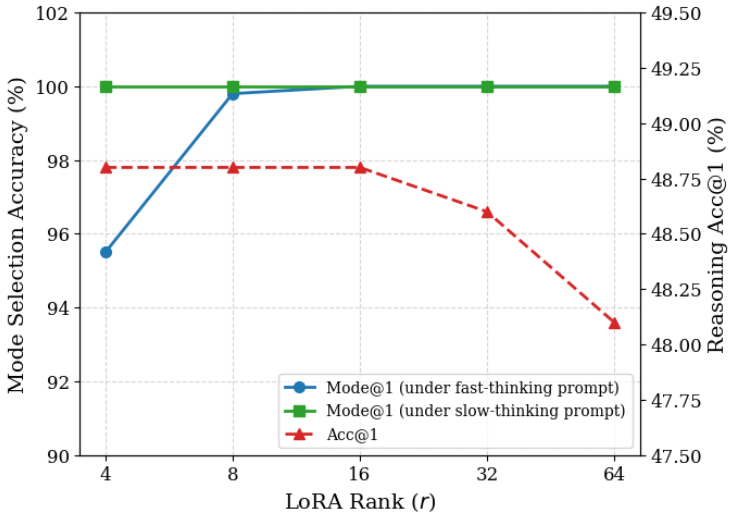


Figure 8: Ablation study on the impact of LoRA rank. The left axis reports the Mode Selection Accuracy (Mode@1) for eliciting fast-thinking (blue circle) and maintaining slow-thinking (green square). The right axis (red dashed line) tracks the average pass@1 reasoning accuracy across seven benchmarks.

- **Mode selection accuracy.** Incorporating $R_{\text{correctness}}$ complicates the optimization landscape: the model must simultaneously balance complex reasoning logic and behavioural mode switching. Within the same training budget, the combined reward only reaches 96.2% Mode@1 on average, falling short of the perfect 100.0% achieved by the mode-only reward.

More importantly, introducing $R_{\text{correctness}}$ violates the parameter-isolation principle: the LoRA adapter would no longer act solely as a mode controller but would also encode reasoning knowledge. Discarding the LoRA after Stage 2 would then forfeit these reasoning gains, forcing a dependency on adapter merging and limiting deployment flexibility.

We therefore exclude $R_{\text{correctness}}$ in Stage 1. The mode-only reward suffices to preserve the native reasoning capability while attaining superior mode-selection accuracy, and it maintains a clean functional separation: the LoRA remains a pure, disposable training scaffold for mode control, ensuring that all subsequent reasoning enhancements are channeled exclusively into the base model.

C.4.7 Effect of LoRA Rank

We investigate the sensitivity of our method to the LoRA rank r. Figure 8 compares $r \in \{4, 8, 16, 32, 64\}$. The results show:

- **Mode selection accuracy.** All ranks $r \geq 8$ converge to 100% accuracy for both fast and slow modes. An extremely small rank ($r = 4$) is slightly under-parameterized and fails to reach perfect convergence ($\sim$96%) within the fixed training budget.
- **Reasoning performance.** Larger ranks can in fact harm reasoning performance. While $r = 16$ maintains the baseline reasoning accuracy (48.8), increasing the rank to 32 or 64 leads to a noticeable decline. We attribute this to "over-parameterization" for the simple mode-control task: a larger LoRA introduces greater perturbations to the frozen base model, potentially drifting away from the optimal reasoning manifold.

Based on these results, we set $r = 16$ as the default: it ensures perfect mode control while minimizing both memory overhead and the risk of reasoning degradation.

C.4.8 Necessity of Freezing LoRA in Stage 2

We investigate the necessity of spatially isolating the mode controller (LoRA) and freezing it during Stage 2. To validate this design, we compare our default setting against two alternatives, both starting from the same Stage-1-adapted model:

- **Variant 1 (Merge & Full Fine-tune).** After Stage 1, we merge the LoRA adapter into the base model and fully fine-tune all parameters under the fast-thinking RL objective. This eliminates the parameter isolation between mode control and reasoning parameters.
- **Variant 2 (Joint Update, no freeze).** The LoRA adapter is kept separate (same architecture as Fast-RL) but remains trainable in Stage 2. Both the base model (θ) and the adapter (ϕ) are updated. This preserves spatial isolation but removes the freezing constraint.

The results in Table 12 reveal that both variants fail to prevent mode contamination:

- **Failure of Variant 1.** Once parameters are merged, the mode-control logic becomes entangled with the reasoning logic. During fast-thinking RL, the gradients uniformly reinforce fast-thinking behavior across the entire parameter space. Slow-thinking is catastrophically forgotten (mode-selection accuracy drops to 52.4%), reverting to the contamination issues of naive fine-tuning.
- **Failure of Variant 2.** Maintaining separate parameters offers a partial buffer against contamination, leading to better mode retention than Variant 1 (78.1% vs. 52.4%), but it still fails. Since Stage 2 rewards exclusively fast-thinking trajectories, the optimizer continues to update the trainable LoRA toward maximizing this reward, gradually overwriting the prompt-to-mode mapping learned in Stage 1 and converting the adapter into a biased fast-thinking trigger.

In contrast, Fast-RL with our default setting (LoRA frozen in Stage 2) maintains 100% mode-selection accuracy. The combination of keeping parameters separate *and* freezing the controller is the only way to strictly decouple reasoning enhancement from behavioural modification, yielding a robust and contamination-free outcome.

Table 12: Ablation on the parameter-freezing and isolation strategy in Stage 2. We compare our default setting (freeze LoRA) against two variants that allow the mode controller to be updated. "Trainable Params" indicates which parts of the model are updated during the fast-thinking RL phase. "Acc@1" denotes the average pass@1 over seven benchmarks. "Mode@1" represents the percentage of responses that correctly follow the slow-thinking format under a standard prompt.

Update Strategy (Stage 2)	**Trainable Params**	**Acc@1**	**Mode@1**
Base Model (after Stage 1)	-	48.8	100.0
Variant 1: Merge & Full Fine-tune	Merged Base + LoRA	58.3	52.4
Variant 2: Joint Update (no freeze)	Base Model + LoRA	58.2	78.1
Fast-RL (freeze LoRA)	Base Model only	58.2	100.0

C.4.9 Merging vs. Discarding LoRA after Stage 2

We compare two strategies for handling the LoRA adapter after Stage 2 training: (i) merging it into the base model, and (ii) discarding it and using only the base model (main branch) for inference.

As shown in Table 11, discarding the LoRA induces a negligible difference in slow-thinking reasoning performance compared with the merged variant. This empirically confirms the success of our spatial decoupling design: reasoning capabilities are infused exclusively into the base-model parameters, while the LoRA functions strictly as a detachable mode controller.

In our main experiments we discard the LoRA by default. In practice, however, we recommend saving the trained adapter rather than permanently deleting it: it can be reused as a plug-and-play mode controller (bypassing Stage 1) or loaded on demand to enable fast-thinking behavior.

C.5 More Experiments on Larger Models

To assess scalability, we extend our experiments to larger models, namely Qwen3-4B and Qwen3-8B. As shown in Table 13, vanilla RL struggles to provide significant gains on these strong baselines, occasionally even leading to minor regressions (e.g., -0.6 on AIME24 for Qwen3-4B).

In contrast, Fast-RL delivers consistent improvements across both thinking modes. On Qwen3-4B, it lifts slow-thinking AIME24 to 76.3 ($+2.8$ over the baseline), substantially outperforming vanilla RL. Fast-thinking performance benefits even more dramatically (*e.g.,*, nearly doubling from 24.5 to 44.2 on AIME24), demonstrating that Fast-RL scales effectively to larger models and unlocks potential that standard RL fails to capture.

Table 13: Results on Qwen3-4B and Qwen3-8B trained under different RL pipelines with GRPO as the policy-update algorithm. Results are reported as "slow-thinking / fast-thinking".

Model	AIME24	AIME25	AMC23	MATH-500	MinervaMath	OlympiadBench	GaoKao2023en
Qwen3-4B	73.5 / 24.5	62.5 / 21.6	95.4 / 65.1	95.2 / 85.3	56.4 / 46.4	67.5 / 51.0	76.4 / 72.1
+ Vanilla RL	72.9 / 22.9	61.2 / 21.6	97.8 / 68.1	95.4 / 85.1	56.0 / 47.6	66.9 / 51.3	76.6 / 72.3
+ Fast-RL	76.3 / 44.2	71.7 / 37.5	98.1 / 89.7	96.8 / 90.4	58.1 / 60.3	69.3 / 51.0	87.9 / 80.3
Qwen3-8B	76.6 / 25.3	66.6 / 22.4	95.0 / 71.0	96.2 / 84.3	57.6 / 46.4	68.9 / 54.0	83.0 / 70.2
+ Vanilla RL	77.5 / 27.5	65.0 / 25.4	95.6 / 74.1	96.1 / 85.7	57.9 / 46.8	66.7 / 53.5	82.6 / 69.7
+ Fast-RL	79.2 / 51.3	74.6 / 39.6	97.5 / 90.3	96.9 / 89.9	59.3 / 52.5	70.0 / 59.3	88.7 / 76.5

D Related Work

D.1 Reinforcement Learning for LRMs

Reinforcement learning (RL) has become the dominant procedure for enhancing the reasoning ability of LRMs. Early approaches such as Proximal Policy Optimization (PPO) [25] adopt an actor-critic architecture, training a critic alongside the policy to learn a value function for advantage estimation. However, the auxiliary critic introduces additional memory and compute, and often causes training instability. To mitigate these drawbacks, recent work has shifted to critic-free policy optimization, where the advantage function is formulated as a normalized reward using a tailored comparison baseline rather than a value network. ReMax [19] introduces a greedy-decoding baseline: for each input, a single greedy response is generated and its reward serves as the reference for advantages computed on stochastically sampled responses. Although simple, this requires an additional inference pass per sample and roughly doubles training latency. GRPO [26] improves on ReMax by leveraging multiple samples per input: for each prompt, multiple responses are generated, and their average reward serves as a shared baseline. Compared with ReMax, this reduces variance via intra-prompt consistency, though it can introduce slight bias when responses are highly correlated. RLOO [1] refines this idea with a leave-one-out (LOO) normalization: for each sampled response, the baseline is the mean reward of all other responses in the group. This estimator is theoretically unbiased under exchangeability and reduces the correlation between the advantage and the policy gradient, leading to more stable updates. REINFORCE++ [14] proposes a global normalization strategy: the baseline is the mean reward across all responses in the training batch, regardless of input prompt. The authors prove that this global averaging reduces the bias of the advantage estimator under mild assumptions, while keeping low variance through batch-level smoothing. REINFORCE++ is particularly effective in low-data regimes and when response diversity is high.

D.2 RL Acceleration Strategies

Efficiency is a central challenge in RL, leading to acceleration strategies that fall into three broad categories.

Algorithmic efficiency via rewards and data. One line of work accelerates convergence by refining the learning signal. PRIME [6], SPRO [7], and CAPO [29] introduce fine-grained process rewards or step-level supervision that provide denser feedback than sparse outcome rewards. Another line is data-centric: methods such as DAPO [33], SPEED-RL [36], PCL [9], DEPO [27], and GRESO [38] accelerate training by dynamically selecting high-value samples or organizing the data into an effective curriculum (e.g., easy-to-hard).

System and inference optimization. A complementary direction targets the computational bottleneck of generation itself. FastGRPO [37] leverages speculative decoding to speed up token generation during rollout. FlashRL [20] employs quantization to reduce memory bandwidth and compute. Asynchronous frameworks such as AREAL [8] optimize resource scheduling to maximize GPU utilization across sampling and training.

Relation to Fast-RL. Most existing acceleration methods are designed for general-purpose LLM RL. In contrast, Fast-RL targets the specific bottleneck of slow-thinking LRMs: the prohibitive cost of generating long-form reasoning chains. Crucially, our approach is orthogonal to the strategies above and can be combined with them (e.g., Fast-RL plus a data curriculum) to achieve cumulative acceleration benefits.

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: See the corresponding part.

   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: See the corresponding part.

   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: [Yes]

Justification: See the corresponding part.

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: See the corresponding part.

   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: [N/A]

Justification: Code will be released after the peer-review stage.

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: See the corresponding part.

   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: [Yes]

   Justification: Yes, see experiments.

   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: See experiment sections.

   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:

   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 have procided it.

    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: This work does not involve the release of models and data. This paper merely presents a new algorithm.

    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:

    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:

    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:

    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:

    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: We have provided it.

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.