From Trial-and-Error to Improvement: A Systematic Analysis of LLM Exploration Mechanisms in RLVR

Jia Deng*, Jie Chen*, Zhipeng Chen, Daixuan Cheng, Fei Bai, Beichen Zhang, Yinqian Min, Yanzipeng Gao, Wayne Xin Zhao†, Ji-Rong Wen

Gaoling School of Artificial Intelligence, Renmin University of China dengjia0510@outlook.com, ptyzchenjie@ruc.edu.cn, batmanfly@gmail.com

Abstract

Reinforcement learning with verifiable rewards (RLVR) has emerged as a powerful paradigm for enhancing the reasoning capabilities of large language models (LLMs). Unlike traditional RL approaches, RLVR leverages rule-based feedback to guide LLMs in generating and refining complex reasoning chains—a process critically dependent on effective exploration strategies. While prior work has demonstrated RLVR's empirical success, the fundamental mechanisms governing LLMs' exploration behaviors remain underexplored.

This technical report presents a systematic investigation of exploration capacities in RLVR, covering four main aspects: (1) exploration space shaping, where we develop quantitative metrics to characterize LLMs' capability boundaries; (2) entropy-performance exchange, analyzed across training stages, individual instances, and token-level patterns; and (3) RL performance optimization, examining methods to effectively translate exploration gains into measurable improvements. By unifying previously identified insights with new empirical evidence, this work aims to provide a foundational framework for advancing RLVR systems. We release our resources at the STILL project website: https://github.com/RUCAIBox/Slow_Thinking_with_LLMs.³.

1 Introduction

Recently, reinforcement learning (RL) has significantly enhanced the complex reasoning abilities of large language models (LLMs) [2]. A key distinction from early RL approaches (e.g., RLHF [3] with trained reward models) for training LLMs is the incorporation of *verifiable rewards*. The underlying idea is simple, yet the effect can be surprisingly powerful: when provided with verifiable signals, LLMs learn to produce lengthy reasoning chains that ultimately yield correct answers [4]. A well-known precedent for this approach is AlphaGo [5], which used rule-based rewards to train Go models. This methodology is now widely recognized as *reinforcement learning with verifiable rewards* (RLVR) [6].

In RLVR, LLMs first generate rollout responses to training problem prompts and then leverage these self-generated responses to improve model performance. This learning process iterates until performance gains become negligible. A crucial aspect of RLVR is enabling effective exploration within the vast state space of natural language. Research [7] has shown that the exploration capabilities

^{*}Equal contribution.

[†]Correspondence to Wayne Xin Zhao.

³This work is an extension of our previous work, "Decomposing the Entropy-Performance Exchange: The Missing Keys to Unlocking Effective Reinforcement Learning [1]".

of LLMs not only influence immediate learning progress but also determine the ultimate performance of the models. Thus, developing a systematic understanding of LLMs' exploration abilities—and how they drive performance improvements—is essential for RLVR.

To investigate the exploration mechanism in RLVR, we first revisit the fundamental exploration-exploitation trade-off in the classic RL literature [8]. An RL agent must strategically balance *exploration* (testing novel actions to discover improved strategies) and *exploitation* (leveraging known optimal actions to earn immediate rewards). This balance is crucial: excessive exploration delays convergence, while insufficient exploration may lead to locally optimal but globally subpar policies. In RLVR, verifiable rewards enable LLMs to guide their exploration in a task-aligned manner. The framework uses exploratory actions to identify potentially correct solutions to reasoning tasks, then reinforces successful solutions while pruning unsuccessful attempts—creating a self-improving cycle of reasoning refinement.

Given the pivotal role of exploration mechanisms in RLVR, this domain has drawn considerable research interest, spanning investigations of entropy mechanisms [9, 7] (where entropy reduction enhances performance) to various enhancement techniques [10, 11] (e.g., Clip-higher). However, despite these advances, current studies have predominantly examined either isolated or coarse-grained aspects of exploration mechanisms. A comprehensive understanding of several fundamental issues remains lacking, particularly regarding how to properly structure the exploration space, how exploration precisely translates to performance gains, and how to effectively augment exploration capabilities.

In this technical report, we conduct a systematic investigation of the fundamental exploration mechanisms employed by LLMs in RLVR. Our methodology integrates a comprehensive literature review with rigorous empirical analysis. The discussion is organized around three key dimensions:

- Exploration space structure (Section 2): We investigate methods to structure the exploration space for LLMs, with particular focus on developing quantitative metrics to characterize their capability boundaries. This involves determining both the solvable and unsolvable problems within practical LLM rollout constraints. Furthermore, we also compare how two primary post-training approaches—SFT and RL—influence LLM exploration capabilities and overall performance.
- Entropy-performance interplay (Section 3): We investigate the relationship between entropy (a key indicator of exploration capability) and model performance. Our analysis extends beyond reviewing recent advances in this area to include a multi-granularity empirical examination across three levels: stage-level dynamics, instance-level efficiency, and token-level significance.
- Performance improvement (Section 4): We discuss approaches to enhancing reasoning abilities, with a particular focus on two main aspects: (1) expanding exploration capacities and (2) enhancing the performance conversion efficacy. Concretely, we first review recent advancements in strengthening the exploration abilities of LLMs. Moreover, we conduct experiments to investigate how to preserve Pass@k performance during training and propose two simple methods to boost the RL efficiency.

Overall, this report establishes a foundational framework for understanding LLMs' exploration mechanisms in RLVR and their role in enhancing reinforcement learning performance. Through an integrated approach combining literature synthesis with novel empirical analysis, we offer a comprehensive investigation of these mechanisms and their practical implications.

2 Quantifying Exploration Abilities in RLVR

In the context of RL, the exploration abilities of LLMs refer to their capacity to discover effective solutions through iterative trial-and-error and environmental interaction [12], particularly in complex reasoning or planning tasks. A model with stronger exploration ability can solve problems more efficiently, requiring fewer attempts. These successful attempts then serve as training data for improving the model's capabilities through RL algorithms like GRPO [13]. The exploration process is fundamentally constrained by the model's *ability boundary*, *i.e.*, the upper limit of its problem-solving capacity [14]. For RLVR training to be effective, the LLM must generate successful attempts within this explorable space; problems beyond this boundary cannot be adequately solved through RL optimization. Therefore, we propose to quantify LLMs' exploration capabilities by measuring their ability boundaries. Specifically, this section introduces two key metrics for this assessment.

2.1 Pass@k Metric

The Pass@k metric, widely adopted in prior studies [15, 7, 11], evaluates whether LLMs can solve problems within k attempts. It has been adopted to estimate models' ability boundaries [16, 14, 17].

In this part, we first formalize the Pass@k metric and its unbiased estimator. We then present the extensions for adapting Pass@k to assess LLMs' exploration abilities.

Formulation for Pass@k Metric. Given a question q, the LLM generates k responses denoted by $\{o_i\}_{i=1}^k$, each evaluated by a verifier that assigns a binary reward: 1 for correct responses and 0 for incorrect ones. The Pass@k score for the problem equals 1 if at least one response is correct, and 0 otherwise. Formally, the Pass@k metric is defined as:

$$\operatorname{Pass}@k = \mathbb{E}_{q \sim \mathcal{D}, \{o_i\}_{i=1}^k \sim \pi_{\theta}(\cdot|q)} \left[\max \left(R(o_1), \dots, R(o_k) \right) \right], \tag{1}$$

where π_{θ} refers to the policy model with parameters θ , and $R(o_i)$ denotes the reward of i-th response generated by policy model. However, the choice of k significantly impacts the stability of Pass@k calculation, with small k values leading to high variance. To mitigate this, prior work [16] introduced an unbiased estimator for Pass@k. Concretely, the model first generates the n responses $\{o_i\}_{i=1}^n$ based on the given question q ($n \geq k$), and then Pass@k metric is computed by the expection of whether k responses contain the positive one among $\{o_i\}_{i=1}^n$. This leads to the following unbiased estimation:

$$\operatorname{Pass@k} = \mathbb{E}_{q \sim \mathcal{D}, \{o_i\}_{i=1}^n \sim \pi_{\theta}(\cdot | q)} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right], \tag{2}$$

where c is the number of the positive responses among the sampled responses $\{o_i\}_{i=1}^n$. Since the unbiased estimation of Pass@k provides stable evaluation of model capabilities, we adopt this method for computing Pass@k in our subsequent analysis.

k-rollout Unsolvable Problems. The Pass@k metric allows us to identify the limits of a model's exploration abilities by sampling numerous responses and determining which problems consistently lack a correct solution. Our empirical observations reveal that once the sample size becomes sufficiently large, additional sampling rarely leads to new correct solutions, effectively stabilizing the set of problems the model cannot solve. This motivates our k-rollout Unsolvable Problems metric, which identifies the set of problems that remain unsolvable after k attempts as a practical measure of a model's capability limitations. With this definition, when comparing models, we can use Venn diagrams to visualize the overlaps among their sets of k-rollout unsolvable problems. By analyzing the overlap and relative sizes of the sets of unsolvable problems, we can identify which models possess a higher capability boundary (i.e., a smaller set of unsolvable problems).

2.2 Entropy of Policy Distribution

In addition to the Pass@k metric, the entropy of policy distribution serves as another important measure for assessing LLM exploration abilities [9]. This metric captures the diversity and uncertainty in LLM behaviors during problem-solving. In this section, we first formalize the token-level entropy computation, then introduce an extended metric—the rollout branching factor—to quantify LLM exploration capabilities.

Formulation of Token-level Entropy. In the context of LLMs, token-level entropy quantifies the uncertainty in the model's token generation process. Following standard information theory definitions [18], the entropy H_i of the *i*-th token is computed as:

$$H_i = -\sum_{t_i \in \mathcal{V}} \pi_{\theta}(t_i | t_{< i}) \log \pi_{\theta}(t_i | t_{< i}), \tag{3}$$

where V and $t_{< j}$ denote the model vocabulary and the previously generated tokens, respectively. This formulation implies that for a given problem, higher values of H_i indicate greater uncertainty in the LLM's generation of the i-th token, reflecting stronger exploratory behavior at that token position.

Rollout Branching Factor. The uncertainty of a model's token generation can be measured by the number of candidate tokens with relatively high probabilities, which we term the *Rollout Branching Factor*. Following common decoding hyperparameters (top-p=0.95), we consider all tokens within the top 95% probability mass as potential candidates for generation. We define these as the candidate tokens. Crucially, a larger number of candidate tokens indicates greater generation diversity for the LLM, corresponding to stronger exploration ability and a higher ability boundary.

2.3 Extended Discussion

The Pass@k metric and policy distribution entropy represent two widely-used methods for evaluating LLM ability boundaries. Previous work [11, 9] has additionally demonstrated that the entropy of LLM-generated responses reflects the model's problem-solving uncertainty, where greater uncertainty may indicate higher potential solution diversity and thus greater problem-solving capacity [7]. Furthermore, generalization performance on unseen tasks serves as another effective indicator models with higher ability boundaries typically exhibit stronger zero-shot task transfer capabilities.

However, a major issues with Pass@k is that it may yield correct final answers in RLVR without producing valid reasoning chains. This phenomenon has been empirically verified through manual inspection in previous work [19]. Several studies have attempted to address this limitation by developing modified methods that verify the correctness of reasoning chains in addition to final answers. A potential approach is to use a language model to evaluate the reasoning chains and assess whether any errors are contained [20]. Specifically, two levels of evaluation granularity can be conducted for LLM assessment, *i.e.*, process-level evaluation [21] and instance-level evaluation [22]. To better evaluate the reasoning chains, existing studies [23, 24] employ reasoning models as evaluators, making the evaluation results more precise and reliable.

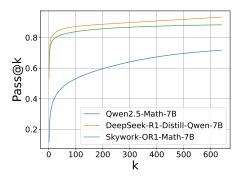
Currently, we do not employ any specific methods to mitigate this type of error. We conducted manual inspections of a random sample of outputs to verify the accuracy of the reasoning chains, finding that most correct answers are generally supported by valid reasoning rather than chance—this pattern is particularly evident for larger values of k. This observation aligns with findings reported in previous research [14]. That said, we recognize that investigating the integrity and faithfulness of reasoning chains generated by language models constitutes an important research direction, which we reserve for future work.

2.4 Comparison of SFT and RL Effects

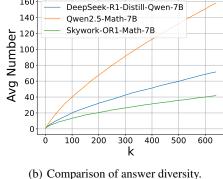
Experimental Setup. Our analysis investigates the effects of SFT and RL by comparing three models selected to represent a sequential training pipeline: Qwen2.5-Math-7B serves as the base model; DeepSeek-R1-Distill-Qwen-7B is an SFT-enhanced version of the base; and Skywork-OR1-Math-7B is a version of the SFT model that has been further trained with RL. All evaluations are performed on 60 questions from AIME24 and AIME25. For the analysis of the rollout branching factor, we also introduce two additional SFT models derived from the base model using the OpenThoughts dataset: SFT-Math, trained only on the math domain, and SFT-All, trained on all domains (*i.e.*, code, math, science, and puzzle). The SFT process for these models used a maximum context length of 20,000 tokens, a batch size of 96, and a learning rate of 1×10^{-5} . All experiments are conducted on a challenging set of 60 problems from AIME24 and AIME25, using k = 640 attempts. We evaluate the models using Pass@k, answer diversity, k-rollout unsolvable problems, and the rollout branching factor. We set the temperature to 0.6, the top-p to 0.95, and allow a maximum generation of 8192 tokens.

Results and Analysis. Our investigation compares SFT and RL across three key dimensions: their impact on overall problem-solving success, their effect on the capability boundary, and their influence on token-level generation diversity.

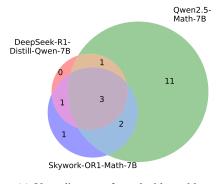
• SFT expands the Pass@k boundary while RL sharpens Pass@1 performance. We first compare how SFT and RL affect problem-solving ability and answer diversity. The experiment contrasts the base, SFT, and RL models on the AIME dataset. As shown in Figure 1(a), SFT achieves a significant improvement in model's Pass@k score compared to the base model (Qwen2.5-Math-7B v.s. DeepSeek-R1-Distill-Qwen-7B). In contrast, RL shows no improvement in Pass@k, indicating a decline in the model's exploration capability (DeepSeek-R1-Distill-Qwen-7B v.s. Skywork-OR1-



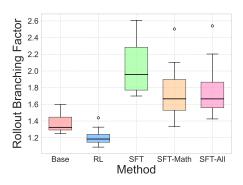
(a) Unbiased estimation of pass@k.



160



(c) Venn diagram of unsolvable problems.



(d) Comparison of rollout branching factor.

Figure 1: Comparison of different models' exploration capabilities across various metrics.

Math-7B). However, RL is known to improve Pass@1, sharpening the model's ability to solve problems on the first attempt. This focus on exploitation comes at a cost: Figure 1(b) shows that both training methods reduce answer diversity, but the RL-trained model exhibits a much more pronounced decrease. This suggests that SFT effectively expands the model's capability boundary by learning from diverse external solutions, whereas RL optimizes and reinforces existing solution paths, leading to a more deterministic policy that improves exploitation (Pass@1) but suppresses exploration (Pass@k and answer diversity).

- Both SFT and RL shift the capability boundary. To gain a more granular understanding beyond aggregate metrics, we analyze how the set of solvable problems changes after training. We compare the sets of k-rollout unsolvable problems for the three 7B models using the same AIME24 and AIME25 test set. The Venn diagram in Figure 1(c) reveals no complete set inclusion. This demonstrates that both SFT and RL induce bidirectional shifts in capability: problems solvable by the base model can become unsolvable after training, and vice-versa. This confirms that training does not simply expand the ability boundary but induces a shift in it. Notably, the RL model's set of unsolvable problems is larger than that of the SFT model. This suggests that RL is more prone to narrowing the model's exploration space.
- SFT fosters token-level diversity, whereas RL leads to a more constrained policy. Finally, we investigate how training impacts exploration at the token level by measuring the rollout branching factor. We compare the base model against the RL model (Skywork-OR1-Math-7B) and three SFT variants (DeepSeek-R1-Distill-Qwen-7B, SFT-Math, and SFT-All). The results in Figure 1(d) show that all SFT models, whether trained on a single domain (SFT-Math) or multiple domains (SFT-All, which is trained on code, math, science, and puzzle), significantly increase the rollout branching factor over the base model. This shows they generate a more diverse set of candidate tokens at each step. In contrast, the RL model fails to increase the branching factor. This indicates that access to high-quality external data via SFT is a critical mechanism for enhancing a model's intrinsic exploration capacity. RL, constrained by its self-generated data, reinforces existing high-probability pathways and does not foster the token-level diversity needed for broader exploration.

2.5 Effects of External Tools

RL directs model improvement through reward signals but relies primarily on self-generated data, restricting capacity expansion. In contrast, tool-integrated reasoning typically involves invoking external tools—such as code interpreters—to execute model-generated content, a process fundamentally distinct from the reasoning patterns of standard text-based language models. We therefore hypothesize that integrating external tools may significantly extend a model's reasoning capabilities. To test this hypothesis, we conducted an empirical study⁴.

Environmental Setup. To investigate how tool-integrated reasoning(TIR) influences a model's capability boundaries, we evaluated four baseline models—all derived from Qwen2.5-Math-7B—on the AIME2024 and AIME2025 benchmarks using the Pass@k metric. For all experiments, we set the temperature to 0.6, the top-p value to 0.95, and allowed a maximum generation of 16,384 tokens. The four baselines are as follows:

- Base_{text}: The base model, utilizing standard text-based reasoning.
- Base_{code}: The base model, utilizing code-integrated reasoning via the direct prompt.
- *RL*_{text}: The model trained with vanila RL, utilizing text-based reasoning. It is trained in a standard RLVR setting with GRPO.
- *RL_{code}*: The model trained with tool-augmented RL, utilizing code-integrated reasoning. It is trained using an approach that integrates a code interpreter; refer to the paper [25] for details.

Results and Analysis. As shown in Figure 2, TIR enhances Pass@k performance relative to standard text-based reasoning. This demonstrates that TIR can expand a model's capability boundary by facilitating external computation and structured tool utilization. However, consistent with prior findings [14], we observe that as k increases, models trained with RL are eventually matched or even outperformed by their non-RL counterparts—a pattern that underscores the inherent limitations of RL in extending a model's capability boundary.

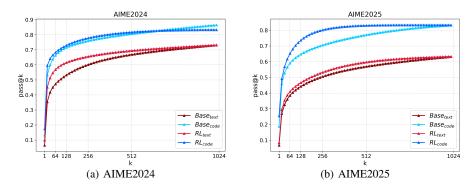


Figure 2: Pass@k Accuracy on AIME2024 and AIME2025.

TAKEAWAY FOR SHAPING LLM ABILITY BOUNDARIES (SECTION 2):

- SFT expands the model's exploration boundary by leveraging external data to increase token-level diversity and Pass@k performance, whereas RL sharpens exploitation for better Pass@1 at the cost of narrowing the exploration space and reducing answer diversity.
- Integrating external tools provides a powerful pathway to expand the ability boundary by facilitating external computation and structured tool utilization to significantly boost Pass@k, although this approach still suffers from the inherent limitations of RL in extending a model's capability boundary.

⁴This part is adapted from our earlier technical report, "*Towards Effective Code-Integrated Reasoning*[25]". It is included here to ensure the completeness of the discussion regarding ability boundaries.

3 Understanding the Entropy-Performance Interplay

From a broader perspective, exploration at the core of RLVR embodies a shift from *uncertainty* to *performance*. Initially, this uncertainty manifests as an expansive exploration space, enabling a diverse range of behavioral attempts. As exploration progresses, the model reinforces successful behavioral trajectories, gradually building confidence in these solutions—thereby driving performance improvements—while mitigating behavioral uncertainty. Ultimately, the model settles into a subtle trade-off between uncertainty and performance.

Specifically, building on the discussion in Section 2, we adopt the entropy of the policy distribution as a metric for uncertainty and investigate the entropy-performance trade-off. We begin by reviewing existing literature on entropy-performance exchange mechanism, then present a fine-grained analysis supported by empirical experiments.

3.1 Reviewing Existing Studies

Recent advancements in RLVR for LLMs have underscored the critical role of entropy in facilitating effective exploratory reasoning. Overall, these studies yield two key insights into the entropy-performance relationship.

Dynamics of Entropy-Reward Exchange. Despite the complexity of the learning process, the underlying mechanism governing entropy-reward dynamics exhibits remarkable simplicity. Empirical results [9] reveal an exponential relationship between performance and entropy, where the reward R follows a predictable exponential decay with respect to policy entropy H. Formally, this relationship can be expressed as: $R = -a \cdot \exp(H) + b$, where a, b > 0 are constants that can be fit from empirical data. are constants that can be fit to empirical data. Using the fitted values of a and b, we can establish a macroscopic performance trend as a function of the varying entropy values. Specifically, a governs the efficiency of coverting entropy to performance, while b-a defines the theoritical maximum reward achievable through RL training.

Exploration Signals from High-Entropy Tokens. As observed in [7], entropy dynamics are strongly linked to *exploratory reasoning behaviors* in LLMs, with high-entropy tokens/sentences frequently serving as pivotal elements that guide or connect reasoning steps (*e.g.*, first, because, however), enable reflective actions like self-verification and correction, and facilitate rare reasoning strategies and under-explored behaviors by the base model. These findings suggest that entropy acts not only as a regularization signal of the RL algorithm but also as an intrinsic indicator of exploration capabilities. Besides, analysis in [26] has revealed that only a minority of tokens exhibit high entropy, referred to as *forking tokens*. These tokens serve as critical decision points where the model's reasoning trajectory can diverge into multiple plausible paths.

3.2 Fine-grained Analysis

Despite these progresess, current investigations of the entropy-performance trade-off operate at a coarse granularity, treating RLVR training as a monolithic process or categorizing tokens into groups. These studies primarily examine aggregate performance changes before and after training states, failing to provide a fine-grained analysis of how entropy dynamics interact with model performance throughout the training trajectory. In essence, RLVR training constitutes a complex learning process shaped by multiple involving elements [9]. These factors dynamically influence model behavior [27], with entropy effects varying across training stages, token positions, and sampled instances—each contributing distinctively to overall performance. Building on the above discussion, in this section, we conduct a systematic study of the entropy-performance interplay in RLVR, focusing on three key aspects: stage-level dynamics, instance-level efficiency, and token-level significance.

3.2.1 Token-Level Metrics for RL Algorithmic Analysis

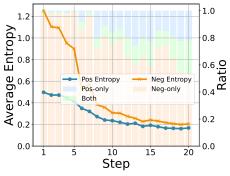
To enable a deeper analysis of RL algorithms in the RLVR setting, we introduce three fine-grained metrics that quantify token-level algorithmic behavior.

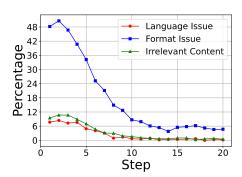
Entropy	0.88- 0.66- 0.44- 0.22-	Entropy Average Acc	0.77 Average Accuracy
	0.00	200 400 600 800 100 Step	0.56 00

Entropy ↓	Frequency ↑				
Û, erot, whim,	\ (, 2, 1, =, {, -,				
Dead,ther,	+, frac, }, }{, 3,5,),_,				
flirt,ĉquery,erot,	$\ \$, x, 7, \[,)., sqrt,				
whim, Dead,	times, cdot,**, 9, (
ther, flirt, ĉquery,	$8, :, 5,), 6, [, 0, 4, \land),$				
pornography, @Sup-], left, right, k, y, n,				
pressWarnings,	pi, theta,)\ }\ {x,				
Ethics, Aad, UDA,	c, k, _n, mod, cos,):,				
è©, åĩĢå̼,);)), f, sin, number,				
	is,				

The red line marks the transition from the rising rising stage. stage to the plateau stage.

(a) Entropy and accuracy trends on AIME24, (b) Example of generated tokens with significant AIME25 and MATH500 during GRPO training. entropy decrease or frequency increase during the





(c) Entropy dynamics. The bar chart shows the (d) Proportion of model responses containing sample distribution of the top 20% tokens exhibit- quality issues across different training steps. ing the fastest entropy drop.

Figure 3: Overview of entropy trends, accuracy evolution, token dynamics, and response issues during GRPO training's rising stage.

- Entropy. Following Section 2.2, we use token-level entropy H_t (defined in Equation 3) to quantify uncertainty in the policy π_{θ} 's predictions at generation step t. Higher H_t values indicate greater uncertainty in token selection, reflecting exploration potential during generation.
- Gradient. To analyze how tokens drive policy updates, we estimate each token's contribution to policy updates by computing the gradient of the GRPO objective $J_{\mathrm{GRPO}}(o_t^i)$ with respect to the language model head layer and taking its Frobenius norm as the update magnitude proxy [27]. Formally, the Frobenius norm of the resulting gradient for the t-th token is computed as:

$$G_t = \left\| \alpha_t \left(\mathbf{e}(o_t) - \boldsymbol{\pi}_{\theta} \right) \cdot \boldsymbol{h}^{\top} \right\|_F, \tag{4}$$

where $\alpha_t = \hat{r}_t \cdot \min(\hat{A}_t, \operatorname{clip}(\hat{A}_t, 1 - \epsilon, 1 + \epsilon))$, $e(o_t)(o_t^i) \in \mathbb{R}^V$ is the one-hot vector for token o_t , and $\pi_\theta \in \mathbb{R}^V$ is the policy distribution. $h \in \mathbb{R}^d$ is the output of the last transformer layer. The full derivation is in Appendix A.1.

• Performance Impact. To quantitatively assess the impact of tokens on reasoning accuracy, we design a token replacement intervention stragety. For any token o_i^t within a generated sequence, we substitute it with the highest-probability alternative token under the current policy:

$$\tilde{o}_t^i = \arg \max_{v_k \in V \setminus \{o_t^i\}} \pi_\theta(v_k \mid q, o_{< t}). \tag{5}$$

Subsequent k continuations are generated independently from both the original token o_i^t and the substituted token \tilde{o}_i^t . The divergence in average solution accuracy between these paired continuation paths serves as a metric for the token's influence on downstream reasoning correctness:

$$I_{t} = \frac{1}{k} \sum_{j=1}^{k} \left(Acc_{j}(q, o_{< t}, o_{t}) \right) - \frac{1}{k} \sum_{j=1}^{k} \left(Acc_{j}(q, o_{< t}, \tilde{o}_{t}) \right).$$
 (6)

Here, $Acc(\cdot)$ is a binary function that returns 1 if the completed sequence leads to a correct solution, and 0 otherwise.

3.2.2 Stage-level Dynamics

Prior work [9] identifies two distinct stages in RLVR training dynamics: (1) a rapid *rising stage* with quick performance improvements and decreasing policy entropy, followed by (2) a stable *plateau stage* with marginal gains (Figure 3(a)). This bimodal behavior naturally raises the question: what underlying mechanisms drive performance improvements in each stage?

Rising Stage. To understand the rapid performance gains in this stage, we analyze the source of entropy reduction and its effects on model behavior. We divide the model responses at each training step into positive and negative sets, and track their entropy dynamics, revealing two main phenomena:

- Entropy reduction mainly stems from negative samples. As shown in Figure 3(c), negative samples consistently exhibit higher average policy entropy than positive samples. More importantly, their entropy declines at a substantially more rapid rate during the rising stage. Also, tokens that appear exclusively in negative samples experience the fastest decline in entropy. This suggests that penalizing incorrect reasoning paths plays an important role in the model's initial learning signal, reducing the vast space of potential errors.
- Entropy reduction solidifies effective reasoning patterns. Our analysis of token distributions (Table 3(b)) reveals that the most significant entropy reductions occur in tokens unrelated to the task objective, while reasoning-critical tokens show increased frequency. Furthermore, we categorize low-quality responses into three types: format violations (unboxed or multiply-boxed answers), irrelevant content (garbled or repetitive text), and language mixing (multilingual responses). For format violations, we count the occurrences of "\boxed{" in the response string. To identify irrelevant content, we utilize Qwen2.5-32B-Instruct to determine if the response contains such content. For language mixing, we employ a Regular Expression to check if any token's Unicode encoding falls within the range of Chinese characters. As shown in Figure 3(d), this entropy shaping leads to a marked decrease in all three key types of defective outputs.

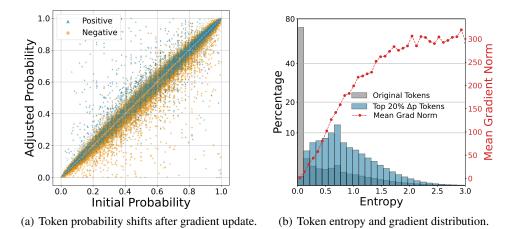


Figure 4: Token-level update patterns.

Plateau Stage. In this stage, as performance gains become incremental and entropy change flattens, we conduct a fine-grained investigation into the underlying mechanisms driving continued refinement.

Specifically, we examine the distribution of token-level probability updates, analyzing both the magnitude of learning signals received by different tokens and their relationship to entropy dynamics and semantic roles.

- Learning concentrates on a small subset of high-entropy, high-gradient tokens. Unlike the rising stage, our analysis of token probability updates reveals that most token probabilities remain stable during the plateau stage, with over 99% of tokens experiencing a probability change of less than 0.06 after parameter updates. As illustrated in Figure 4(a), learning is instead concentrated on a small fraction of tokens where probabilities in positive samples are reinforced while those in negative samples are suppressed. In Figure 4(b), these impactful updates primarily target high-entropy tokens. These tokens tend to produce larger gradients during backpropagation (Eq. 4). This indicates that progress in this stage is mainly driven by resolving uncertainty at critical "forks" in reasoning paths [26].
- Updates are most sensitive for tokens associated with formal reasoning. To further characterize these critical tokens, we categorize them by their semantic roles and analyze which types experience the largest probability changes: formal reasoning tokens enable symbolic manipulation for computation and modeling; logical structuring tokens manage the flow of reasoning; metacognitive tokens guide the process through self-monitoring; and semantic support tokens provide linguistic elements for fluency, coherence, and informativeness. We provide examples of each token category in Table 1. Our results show that among the top 20% of tokens with the greatest probability updates, those associated with formal reasoning (e.g., numerals, mathematical symbols) have the highest proportion (0.039), followed by metacognitive reasoning tokens (0.034), general semantic tokens (0.033), and logical structuring tokens (0.031). This targeted refinement of critical, uncertain tokens indicates a shift towards mastering the nuanced logic and precise calculations required for advanced reasoning, rather than merely reproducing structural patterns.

Table 1: Examples of Token Categories in RLVR.

Category	Examples
Formal Reasoning	Numbers (<i>e.g.</i> , '1', '3.14'), operators (<i>e.g.</i> , '+', '*', '='), variables (<i>e.g.</i> , 'x', 'y'), and symbols (<i>e.g.</i> , ' π ', ' $\sqrt{2}$ ', ' \sum ').
Logical Structuring	Causal (<i>e.g.</i> , 'therefore', 'because'), contrastive (<i>e.g.</i> , 'however', 'but'), progressive (<i>e.g.</i> , 'first', 'next', 'finally'), and parallel (<i>e.g.</i> , 'and', 'also').
Metacognitive	Verifying (e.g., 'Let's check'), revising (e.g., 'Correction', 'Wait'), summarizing (e.g., 'In summary'), and planning (e.g., 'First, I will').
Semantic Support	Grammatical elements (<i>e.g.</i> , 'the', 'is', 'of'), domain entities (<i>e.g.</i> , 'problem', 'solution'), and adjectives (<i>e.g.</i> , 'correct', 'final').

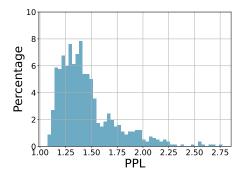
TAKEAWAY FOR STAGE-LEVEL ANALYSIS (SECTION 3.2.2):

- During the rising phase, entropy reduction is primarily driven by negative examples, facilitating the emergence of reasoning patterns.
- During the plateau phase, probability shifts are concentrated in a small set of high-entropy, high-gradient tokens associated with formal reasoning.

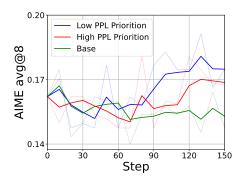
3.2.3 Instance-level Efficacy

As not all samples contribute equally to learning [28], to understand how instance quality affects optimization, we analyze the role of instance-level PPL, which can be regarded as a measure of the model's uncertainty over a whole sequence. Since low-PPL responses are generally more fluent and semantically coherent [29], we hypothesize that these low-PPL instances are more critical for effective RLVR, which is confirmed by the following three findings from our analysis:

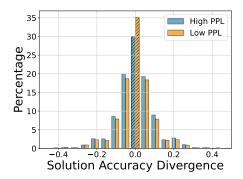
• Learning signals are concentrated in low-PPL samples. To explore where learning occurs most actively, we analyze the magnitude of token probability changes during RLVR updates. As shown in



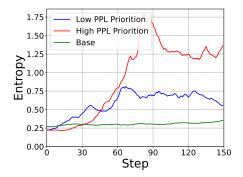
(a) Distribution of PPL values for tokens with the top 20% largest probability shifts in the training set.



(c) Model accuracy under advantage schemes favoring high or low PPL examples..



(b) Accuracy change after replacing high-entropy tokens in samples with high vs. low PPL.



(d) Average token entropy under advantage schemes favoring high or low PPL examples.

Figure 5: Analysis of token behavior and model performance under different manipulations related to PPL.

Figure 5(a), we observe a clear concentration of high-magnitude probability updates in the low-PPL region, indicating that the model's learning is more active within these generations.

- Low-PPL instances represent more robust reasoning paths. To understand the differences between samples, we apply token-level intervention analysis (Eq. 6) to instances sampled from both low-PPL (bottom 20%) and high-PPL (top 20%) groups. The results in Figure 5(b) show that replacing tokens in low-PPL responses leads to smaller changes in the final solution's accuracy compared to the same intervention in high-PPL responses, indicating that the model exhibits more robust and stable reasoning in low-PPL instances.
- Prioritizing low-PPL instances enhances RLVR effectiveness. To verify the importance of low-PPL instances, we conduct the experiment by dynamically re-weighting token advantages based on PPL. First, we compute a standardized log-PPL weight for each response o^i :

$$w_{\rm ppl}(o^i) = \frac{\ln \mathrm{PPL}(o^i) - \mu}{\sigma}.\tag{7}$$

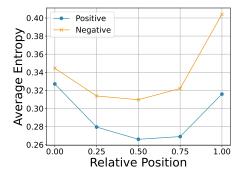
Here μ and σ are the mean and standard deviation of the log-PPL values across the sampled responses for the same query q, and α is a hyperparameter. We then compare two opposing strategies: one that adjusting the advantage with a factor of $(1-\alpha \cdot w_{\rm ppl}(o^i))$ of sampled instances, and another that using the factor of $(1+\alpha \cdot w_{\rm ppl}(o^i))$. As shown in Fig. 5(c), the former one results in superior performance gains. In contrast, focusing on high-PPL samples leads to much higher policy entropy, as shown in Figure 5(d). Further analysis of the model's generated responses on the test set reveals that this approach degrades response quality, with the frequency of responses containing quality issues rising to approximately 7%, compared to about 3% for the low-PPL strategy. This confirms that focusing RL updates on low-PPL samples is a more effective optimization strategy.

TAKEAWAY FOR INSTANCE-LEVEL ANALYSIS (SECTION 3.2.3):

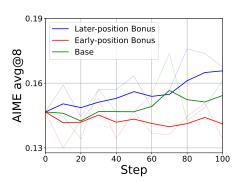
Low PPL samples are crucial for model self-improving, as they exhibit larger token probability shifts, demonstrate more robust and stable behavior, and offer higher optimization efficiency.

3.2.4 Token-level Significance

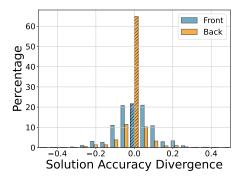
To understand how a token's effect on learning varies throughout a sequence, we analyze the interplay between token position, entropy, and optimization impact. We investigate the distribution of token entropy and importance across different positions, finding that although entropy is high at both the beginning and end of sequences, the tokens toward the end are more critical for effective RL.



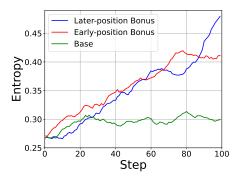
(a) Average token entropy by position across the training set, computed over 1k steps.



(c) Model accuracy under positional advantage schemes with fixed $\gamma=1.0.$.



(b) Accuracy change from replacing high-entropy tokens at top and bottom 20% positions.



(d) Average token entropy under positional advantage schemes with fixed $\gamma=1.0$.

Figure 6: Token position analysis and model performance under position-based reward schemes.

- Token entropy follows a U-shaped distribution, with higher values at the start and end of sequences. As illustrated in Fig 6(a), we observe that higher entropy concentrates at the beginning and end of a response. High entropy at the beginning reflects a broad exploration space where the model considers multiple initial approaches. In contrast, high entropy near the end of a sequence indicates uncertainty in the final decision-making process, which is directly linked to the task objective. As noted in prior work [30], there is a high correlation between model confidence in the last few tokens and overall accuracy.
- Initial high-entropy tokens govern outcomes; terminal high-entropy tokens reflect reasoning uncertainty. We use token-level intervention analysis in Eq. 6 and reveal the distinct functional roles of these two high-entropy regions. As Figure 6(b) illustrates, replacing early-position tokens significantly alters the final solution's accuracy. This highlights the inherent uncertainty in the initial language space, which broadens the exploration scope and results in higher entropy. Conversely, while late-position tokens also exhibit high entropy, their minimal impact on accuracy suggests

a more constrained semantic space. Interestingly, the entropy of late-position tokens in negative examples is higher than in positive ones. This subtly indicates that the model might, in the later stages of inference for incorrect solutions, implicitly detect its errors, leading to greater confusion and, consequently, elevate entropy.

• Optimizing tokens in later positions provides a more efficient learning signal. To verify this, we conduct a comparative experiment by applying a positional bonus to the token advantages, defined as follows:

$$b_t^i = \gamma \cdot \sigma(d \cdot r_t^i). \tag{8}$$

where γ is a hyperparameter, σ is the sigmoid function, r_t^i represents the token's relative position, and the direction parameter d determines the focus of the bonus. Setting d=1 rewards tokens appearing later in the sequence, while setting d=-1 rewards tokens appearing earlier. For positive samples, this bonus is added to the original advantage to increase the reward, while for negative samples, it is subtracted to amplify the penalty. Our experiment results in Fig. 6(c) shows that reinforcing tokens later in the sequence yields superior performance compared to both baselines with no positional bonus and the strategy that gives bonuses to early tokens. While applying the positional bonus in either direction increases policy entropy (Figure 6(d)), further analysis of the generated responses reveals that rewarding early positions leads to shorter average response lengths (904 tokens) compared to rewarding later positions (1146 tokens). This suggests that optimizing the latter parts of reasoning can extend the model's reasoning time [31], thereby improving accuracy.

TAKEAWAY FOR TOKEN-LEVEL ANALYSIS (SECTION 3.2.4):

Positions towards the end are more crucial for model learning, as they have higher entropy, more stable semantics, and better optimization efficiency.

4 Exploration-Enhanced RL Approaches

In this section, we discuss how to enhance model performance using exploration-enhanced reinforcement learning (RL) approaches. We start by introducing the RLVR baseline—the GRPO method—followed by a literature survey of related methods. Next, we present experimental results and discuss strategies to enhance exploration capabilities based on the Pass@k metric. Finally, we introduce several simple methods to improve RL performance, drawing on the findings from Section 3.

4.1 The RLVR Baseline - GRPO

GRPO [32] is a representative RL algorithm for LLMs. It optimizes a policy π_{θ} to maximize the expected reward over sampled reasoning trajectories. Given an old policy $\pi_{\theta_{\text{old}}}$ and the current policy π_{θ} , GRPO maximizes the following objective:

$$\mathcal{J}(\theta) = \mathbb{E}_{q \sim \mathcal{D}, \ o \sim \pi_{\theta_{\text{old}}}} \left[\sum_{t=1}^{|o|} \min \left(r_t \hat{A}_t, \ \text{clip}(r_t, \ 1 - \epsilon, \ 1 + \epsilon) \hat{A}_t \right) - \beta \cdot \text{KL} \left[\pi_{\theta}(\cdot \mid q, o_{< t}) \parallel \pi_{\text{ref}}(\cdot \mid q, o_{< t}) \right] \right], \tag{9}$$

where q and o denote the input prompt and the sampled response respectively, drawn from the prompt dataset \mathcal{D} and the old policy $\pi_{\theta_{\text{old}}}$. The importance sampling ratio r_t is $r_t = \frac{\pi_{\theta}(o_t|q,o_{< t})}{\pi_{\theta_{\text{old}}}(o_t|q,o_{< t})}$, and \hat{A}_t denotes the token-level advantage estimate. The hyperparameter ϵ controls the clipping range, and β weights the KL regularization term against a fixed reference policy π_{ref} .

To compute \hat{A}_t , GRPO applies a group-relative normalization scheme. In outcome-supervised settings, for each prompt q, it samples G responses $\{o^1, o^2, \dots, o^G\}$ using the old policy π_{old} , and assigns each response a binary reward R^i : 1 if correct and 0 otherwise. Since the reward is uniformly broadcast over all tokens, the token-level advantage for the t-th token in the i-th response o^i is

$$\hat{A}_t^i = \frac{R^i - \text{mean}(\{R^j\}_{j=1}^G)}{\text{std}(\{R^j\}_{j=1}^G)}.$$
(10)

4.2 The Exploration-Exploitation Trade-off in RLVR

The core of RLVR lies in balancing exploration and exploitation—a classic challenge in reinforcement learning algorithms. Initially, the model has limited knowledge of the target task and can only make meaningful attempts based on prior knowledge (*i.e.*, knowledge acquired from training data). The environment or verifier then provides feedback to guide the model's learning [33, 34]. Unlike SFT, however, all training data involved in RLVR—whether correct or incorrect solutions—are generated by the model itself. A key distinction between RL and SFT is thus: RL reinforces correct behaviors while discouraging incorrect ones, whereas SFT merely imitates correct demonstrations [35]. This makes RL focus on learning from self-generated behaviors; over time, this can lead to overconfidence in intermediate actions, which in turn leads to entropy collapse and performance saturation [9].

Once the model has acquired certain skills through self-generated data, developing new skills requires exploratory behaviors to attempt novel approaches. This dynamic embodies the fundamental exploration-exploitation trade-off. For strong performance in RLVR, two critical factors emerge: first, the model must be capable of diverse, meaningful exploration, with a sufficiently large exploration space to sample correct solutions; second, the model should be trained efficiently to identify correct solutions with minimal effort. These considerations give rise to two key research questions: how to enhance LLMs' exploration capabilities and how to translate these exploration abilities into performance gains?

In what follows, we first review the previous work that attempts to enhance the RLVR method in Section 4.3. Then we conduct the empirical experiments to enhance the model capabilities in Section 4.4.

4.3 Reviewing Existing Studies on Enhancing Exploration

To improve the training performance of RLVR, recent studies have proposed methods to preserve or enhance the model's exploration capacity during training. These approaches aim to mitigate distributional narrowing—evidenced by declining policy entropy or Pass@k metrics—ensuring the model retains sufficient reasoning diversity to discover and refine correct solutions. Concretely, these methods intervene in various components of the RL pipeline, including advantage shaping, token-level gradient modulation, KL-based regularization, and external tool integration. Below, we categorize and summarize these exploration-enhancing techniques across four primary areas, as detailed in Table 2.

Table 2: Exploration enhancement methods in different categories. We also report the metrics for measuring the exploration capacities.

Category	Method	Metric
Advantage Refinement	Entropy-based Advantage Shaping [7] Negative Sample Reinforcement [11] Pass@k Training [17]	Pass@k Pass@k Pass@k
Token/Gradient Selection	Clip-Higher [10] Covariance-aware Gradient Detach [9] Forking Tokens [26]	Entropy Entropy Entropy
KL Regularization	KL Reference Reset [15] Covariance-aware KL Penalty [9]	Pass@k Entropy
Tool Augmentation	Progressive Tool Integration [25] Entropy-based Adaptive Rollout [36]	Pass@k Entropy

Advantage Refinement. Modifying the advantage term during policy optimization is a straightforward yet effective approach to enhance LLMs' exploration ability. Entropy-based Advantage

Shaping [7] introduces a clipped, gradient-detached entropy bonus into the advantage function, which boosts learning signals for high-entropy (*i.e.*, exploratory) tokens while maintaining optimization stability. Negative Sample Reinforcement [11] decomposes the RL objective into positive and negative sample reinforcement, explicitly penalizing incorrect completions to suppress wrong reasoning paths and preserve exploration space. Pass@k Training [17] extends the advantage estimation to reward all correct completions within top-k outputs, ensuring that exploration of multiple correct solutions is retained before fine-tuning the model toward precise Pass@1 performance. These methods share a common goal: enhancing exploration by ensuring that the advantage term continues to reward diverse reasoning paths throughout the RL training process.

Token/Gradient Selection. Exploration can be improved by selectively controlling token-level gradient updates. The Clip-Higher method [10] relaxes the upper clipping bound on the importance sampling ratio in PPO/GRPO, allowing low-probability tokens to increase their likelihood more freely and thus preventing early entropy collapse. Covariance-Aware Gradient Detach [9] identifies tokens with unusually high covariance between their log-probabilities and advantages, clipping a small fraction of these tokens out from the gradient computation to avoid overfitting and maintain entropy. Forking Tokens Selection [26] focuses optimization on a top fraction of high-entropy tokens within each batch, discarding gradients from low-entropy tokens to concentrate learning on key decision points that promote diverse reasoning paths. All these methods aim to preserve tokens with high exploratory value during training to sustain policy diversity and effective exploration.

KL Regularization. KL-based methods stabilize exploration by penalizing divergence between the current policy and a reference policy, preventing excessive policy drift and improving training stability. ProRL [15] applies a KL penalty throughout training but periodically resets the reference policy to the current policy and reinitializes optimizer states. This reset prevents the KL term from dominating the loss and freezing updates, enabling longer and more stable training while maintaining exploration. Covariance-Aware KL Penalty [9] targets tokens with the highest covariance between their log-probabilities and advantages, applying intensified KL penalties only to a small subset of these tokens. By selectively regularizing these influential tokens, KL-Cov prevents premature entropy collapse without overly constraining the entire policy. Together, these techniques balance global and token-level control over policy shifts to sustain exploration and avoid overfitting during RL training.

Tool Augmentation. Integrating external tools into LLM reasoning processes effectively enhances exploration capabilities. ETIR [25] demonstrates that code-integrated reasoning—by executing generated code with external interpreters—significantly expands the model's capability boundaries and improves exploration, as it enables the model to access new reasoning pathways beyond text generation alone. The close coupling of code generation and execution encourages the model to explore diverse solutions that would be unreachable without tool interaction. ARPO [36] leverages an entropy-based adaptive rollout strategy to dynamically allocate exploration resources during tool interactions. By monitoring token-level entropy changes after each tool call, ARPO selectively branches new reasoning paths where uncertainty increases, thereby focusing exploration on promising but uncertain regions. This entropy-guided sampling enhances the model's ability to discover effective multi-step tool-use strategies and improves overall exploration efficiency.

The methods reviewed above focus on mitigating the exploration space collapse that inherently occurs during RLVR training. By modifying advantage computation, selectively regulating token updates, applying targeted regularization, these approaches help maintain a sufficiently wide exploration space, which is essential for enabling diverse reasoning strategies. Preserving this exploration capacity sets the foundation for RL to effectively search, discover, and eventually refine correct solutions, ensuring that exploration potential can be effectively translated into performance gains on Pass@1 performance.

4.4 Empirical Experiments on Enhancing Model Capabilities

In the preceding part, we provided a brief overview of existing progress in enhancing the effectiveness of reinforcement learning training. Here, we present empirical experiments we conducted to examine the efficacy of various strategies for improving model performance. Specifically, we consider two lines of experiments: the first focuses on maintaining the exploratory capabilities of LLMs during the

training process, while the second explores more effective training approaches based on the findings in Section 3.

4.4.1 Maintaining Exploration Capabilities: Retaining Pass@k

In Section 2, we have compared the effects of SFT and RL noting a key distinction: SFT utilizes external demonstration data, whereas RL relies on self-generated data. When trained on self-generated data, the uncertainty of LLMs is gradually suppressed, leading to a significant reduction in their exploratory capabilities. Accordingly, this section focuses on strategies to preserve such exploratory capacities, with Pass@k serving as the core evaluation metric.

Methods. Considering the high computational cost of RL, we conduct our experiments using Rejection-sampling Fine-Tuning (RFT). RFT iteratively refines a model by sampling multiple responses from the model itself, evaluating those responses, and then fine-tuning the model on a selected subset of these samples. To maintain and enhance the model's exploration capabilities during this process, we propose three data selection strategies for each RFT iteration, applied on top of standard rule-based filtering:

- *Incorporating Noisy Data*. To prevent the model from becoming overly deterministic and to encourage a broader exploration space, we integrate a small proportion of "noisy" or incorrect samples into the training data. Specifically, after initial rule-based filtering, we include 5% of negative samples from the generated rollouts into the training batch.
- Selecting High-Entropy Data. High entropy in a generated response indicates greater uncertainty and diversity in the model's token distribution, reflecting a more exploratory behavior. For each candidate response obtained through rejection sampling, we calculate the average token-level entropy across all its tokens. We then prioritize and select a batch of samples for fine-tuning that exhibit the highest average entropy.
- Selecting High-Rollout Branching Factor (RBF) Data. Building on the concept of token-level entropy, the rollout branching factor (as defined in Section 2.2) quantifies the diversity of plausible next tokens at each generation step. We compute the average rollout branching factor for each response and select samples with the highest average values for fine-tuning.

Experimental Setup. We conduct experiments using the Qwen2.5-32B model and the STILL-3 dataset. Our RFT process involves multiple iterations, with each iteration utilizing 1.1k selected data points for fine-tuning. The fine-tuning parameters are set as follows: a maximum context length of 20000 tokens, a batch size of 32, and a learning rate of 1×10^{-5} . Before applying the specific data selection strategies, we employ a rigorous rule-based filtering process to ensure the quality and relevance of the self-generated responses. This filtering is primarily applied to all samples from the rollouts. The filtering criteria includes:

- Answer Box Presence: Only responses containing a properly formatted answer within "\boxed" are considered. Responses without this specific format are discarded.
- *Numerical Content:* For mathematical reasoning tasks, responses are required to contain at least one digit to ensure relevance to the problem type.
- *Post-Answer Content Truncation:* Considering that the base model might produce redundant output after generating the "\boxed" answer, we process all responses by removing any content that followed the first "\boxed" marker.
- Redundancy Check: A 3-gram redundancy metric is calculated for each response. Responses with a 3-gram redundancy exceeding 0.31 (indicating high repetition or garbled text) are excluded to maintain content quality.

These rule-based filters ensure that only well-formed, relevant, and non-repetitive incorrect responses are considered for further processing and selection by our proposed strategies.

For evaluation, we assess the models on three benchmarks: AIME24, AMC23, and MINERVA. During evaluation, we set top-p= 0.95 and temperature 0.6, sampling 128 times for each problem on AIME24 and 64 times for the other two benchmarks to estimate Pass@1 and Pass@k.

Results and Analysis. Our experimental results in Table 3 lead to the following key conclusions:

Table 3: Comparison of Pass@1 and Pass@k performance of Qwen2.5-32B with different RFT data selection strategies on AIME24, AMC23, and MINERVA benchmarks. "Intermediate Methods" are experiments run for fewer iterations, while the "Final Method" is our best combined approach trained for the full duration. "RBF" refers to the rollout branching factor.

Method	AIN	IE24	AMC23		MINERVA	
	Pass@1	Pass@k	Pass@1	Pass@k	Pass@1	Pass@k
Baselines						
Qwen2.5-32B (Base)	10.50	60.00	46.60	95.00	27.50	63.52
Qwen-2.5-32B-SimpleRL-Zoo	27.20	65.75	67.50	95.00	36.38	69.85
Intermediate Methods (fewer iterations	5)					
RFT (Rule-filtered only)	20.29	58.76	56.17	92.50	/	/
RFT + Noisy Data (5%)	18.62	63.02	57.24	92.50	/	/
RFT + High-Entropy Data	16.43	68.30	56.60	97.50	/	/
RFT + High-RBF Data	18.83	68.81	58.16	97.50	/	/
Final Method (most iterations)						
RFT + Noisy Data (5%) + High-RBF	23.23	66.62	67.46	100.00	32.23	72.43

- Controlled noise injection enhances exploration. Incorporating a small fraction (5%) of noisy data during RFT improves Pass@k performance with minimal impact on Pass@1 accuracy. This suggests that preventing the model from becoming overly deterministic by exposing it to a controlled amount of negative samples helps it diversify its exploration.
- Directly optimizing for exploration metrics is effective. Data selection strategies based on high entropy and, particularly, high rollout branching factor yield the most significant improvements in Pass@k. This confirms that explicitly selecting for samples that exhibit high token-level diversity is a powerful method for enhancing a model's underlying exploration capabilities.
- RFT with exploration-aware data selection surpasses strong RL baselines in Pass@k. After several iterations, our RFT model trained with the noise injection and high rollout branching factor strategy outperforms the Qwen-2.5-32B-SimpleRL-Zoo baseline in Pass@k across all benchmarks, while its Pass@1 performance remains competitive. This demonstrates that our method effectively maintains and enhances the model's ability to find diverse correct solutions.
- Enhanced exploration provides a foundation for long-term improvement. While prioritizing exploration can introduce a slight, temporary trade-off in Pass@1 precision, the iterative nature of RFT allows the model to translate these exploratory gains into robust performance improvements over time. The substantial increase in Pass@k is particularly crucial, as it potentially provides a larger pool of successful trajectories, thereby offering significant room for further performance gains in complex reasoning tasks.

TAKEAWAY FOR MAINTAINING EXPLORATION CAPABILITIES (SECTION 4.4.1):

In RFT, using data selection strategies that favor diversity—such as incorporating controlled noise or prioritizing samples with high rollout branching factor—can effectively maintain exploration capabilities and boost Pass@k performance.

4.4.2 Enhancing Optimization Efficacy: PPL- and Position-Based Advantage Shaping

In Section 3, we have found that perplexity (PPL) and position exert potential influences on reinforcement learning (RL) training. Here, we investigate how to leverage these insights to enhance the training of RLVR.

Methods. In this part, we consider examining two simple adayantage shaping methods, described as follows:

• PPL-based Advantage Shaping. As the first strategy, we adjust token advantages to favor low-PPL samples, where learning is concentrated. For each response o^i in a batch, we compute its standardized log-PPL weight $w_{\rm ppl}(o^i)$ using Eq. 7. The advantage A_t for each token t in that response is then

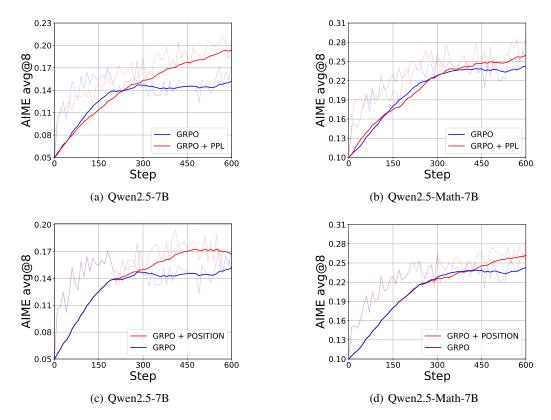


Figure 7: Comparison of average accuracy change curves.

Table 4: Results on math benchmarks across The pass@k results is shown in A.2.

Method	AIME24		AIME25		AMC23		MATH500		Avg.	
1,10,110,11	avg@8	maj@8	avg@8	maj@8	avg@8	maj@8	avg@8	maj@8	avg@8	maj@8
Qwen2.5-7B										
BASE	7.50	9.88	1.67	1.56	38.44	52.50	58.63	75.40	26.56	34.84
GRPO	21.30	24.17	15.40	15.54	59.38	65.00	80.83	84.80	44.23	47.38
GRPO+PPL	22.08	25.03	18.75	20.24	60.31	67.50	82.90	86.00	46.01	49.69
GRPO+POSITION	20.00	21.40	<u>17.08</u>	<u>17.68</u>	63.44	75.00	<u>81.33</u>	<u>85.20</u>	<u>45.46</u>	49.82
Qwen2.5-Math-7B										
BASE	15.42	20.78	7.50	13.38	42.77	52.47	57.60	67.41	30.82	38.51
GRPO	27.08	31.25	25.00	25.85	67.81	72.50	86.65	89.00	51.64	54.65
GRPO+PPL	31.25	37.42	25.42	26.24	73.44	82.50	86.73	88.80	54.21	58.74
GRPO+POSITION	33.75	39.51	22.92	24.02	<u>71.56</u>	<u>75.00</u>	86.52	88.20	<u>53.69</u>	<u>56.68</u>

modulated as follows:

$$\tilde{A}_t^i = A_t^i \cdot \left(1 - \alpha \cdot w_{\text{ppl}}(o^i)\right). \tag{11}$$

This method down-weights the updates from high-PPL samples, focusing the model's learning on more in-distribution reasoning paths.

• Position-based Advantage Shaping To focus optimization on the latter parts of reasoning sequences, we apply a position bonus to the token advantages. As motivated by our empirical analysis, we use the positional bonus b_t^i defined in Eq. 8. This bonus increases toward the end of the sequence and is applied based on the sign of the original advantage:

$$\tilde{A}_t^{i'} = A_t^i + \operatorname{sign}(A_t^i) \cdot b_t^i. \tag{12}$$

This approach encourages the model to allocate more learning effort toward the latter parts of its reasoning process.

Training Details. For the PPL-based reward shaping method, we apply the advantage adjustment throughout the entire RLVR training process, as PPL's measure of the model's uncertainty over a sequence is consistently applicable across the entire training period. We set the scaling hyperparameter $\alpha=0.01$. For the positional reward shaping method, as shown in Fig. 6(d), our empirical analysis reveals that applying a positional bonus can cause a rapid rise in entropy. Therefore, we apply this method selectively. The bonus is only applied during the plateau stage, beginning at step 200 and continuing for 100 steps. Also, we set a small scaling factor $\gamma=0.1$ to moderate the entropy increase. We set the bonus direction d=1.0. The token's relative position score r_t^i is calculated as $r_t^i=m\cdot(l_t^i-n)$, where $l_t^i\in[0,1]$ is the token's relative position in the sequence, with scaling and shifting parameters m=15 and n=-0.5.

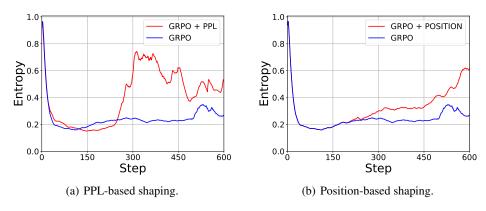


Figure 8: Entropy dynamics for Qwen2.5-7B.

Overall Performance. We evaluate our proposed methods on mathematical reasoning benchmarks and analyze their impact on model behavior. As shown in Table 4, our approaches achieve subtantial improvements across the evaluation benchmarks. Compared to the GRPO baseline, they outperform it by an average of 1.51% for the Qwen2.5-7B model and by 2.31% for the Qwen2.5-Math-7B model, demonstrating the effectiveness of our targeted reward shaping. Moreover, our evaluations on GPQA and HumanEval reveal that both approaches exhibit enhanced generalization capabilities over the GRPO baseline.

Entropy Dynamics. As illustrated in Fig. 8, our approaches sustain a higher level of entropy during the later stages of the plateau stage. It exhibits a higher entroy trend compared to the GRPO baseline. This indicates that our method enables the model to retain substantial exploratory capability even in the later stages of training.

Table 5: Comparison of average response length and token type counts in test set responses for Qwen2.5-7B.

Method	Mean Length	Formal Reasoning Tokens	Logical Structuring Tokens	Metacognitive Reasoning Tokens
GRPO	969.06	501.24	26.31	0.02
GRPO+PPL	1841.44	1007.07	44.80	0.18
GRPO+POSITION	1121.28	607.625	38.10	0.04

Response Pattern Analysis. We further analyze changes in response patterns by quantifying the distribution of token categories across all test sets. As shown in Table 5, both methods result in longer responses compared to the baseline, with a notable increase in tokens related to formal reasoning and logic. Formal reasoning tokens show the most significant increase, while the other categories,

particularly metacognitive reasoning tokens, see smaller gains. This suggests that improving advanced cognitive abilities is inherently more difficult and may require more training steps. Case studies in Appendix A.3 further reveal that both methods yield more detailed step-by-step breakdowns and a deeper display of the computational process compared to the baseline. Notably, the positional method encourages the model to attempt and backtrack from erroneous approaches, indicating a deeper reasoning process.

TAKEAWAY FOR ENHANCING OPTIMIZATION EFFICACY (SECTION 4.4.2):

The optimization efficiency can be consistently improved by leveraging the advantages of low-PPL instances and tokens appearing in later positions.

5 Conclusion

In this report, we present a systematic investigation of exploration mechanisms in LLMs under RL with verifiable rewards (RLVR). Our methodology combines a comprehensive literature review with original empirical analysis, focusing on four key dimensions: (1) exploration space shaping, (2) entropy-performance interplay, and (3) RL performance optimization. The major findings of our study include:

- For exploration space shaping, we introduce two new metrics: k-rollout unsolvable problems and rollout branching factor—to augment conventional evaluation measures (pass@k and policy distribution entropy). Additionally, we propose to use the Venn diagram for comparative analysis of LLMs' reasoning capacities across different exploration regimes. We observe that the sets of unsolvable problems for different methods, when compared to the base model, do not exhibit complete inclusion. This confirms that training does not simply expand the ability boundary but reconfigures it. Furthermore, tool augmentation can effectively expand the capacity boundary of large language models.
- Through a detailed analysis of entropy-performance interplay in RLVR, we identify three core phenomena: stage-level dynamics, instance-level efficiency, and token-level significance. Our findings show that during the rising stage, models establish formal reasoning patterns primarily through entropy reduction in negative samples. Furthermore, plateau-stage analysis demonstrates that significant entropy changes occur predominantly in low-PPL responses and are concentrated in later-position tokens that contribute most to final decisions.
- We consider improving RL performance improvement through two main aspects: (1) expanding exploration capacities and (2) enhancing the performance conversion efficacy. For exploration capacity enhancement, we systematically summarize existing methods and empirically validate several data selection strategies for RFT, confirming that prioritizing sample diversity is an effective method for enhancing exploration. Regarding performance conversion, we mainly review entropy-based optimization techniques and introduce two simple yet effective reward shaping methods that leverage instance perplexity and token position information.

RLVR offers a promising approach for enhancing LLMs' reasoning capabilities, while it also inherits limitations from conventional RL methods—such as training instability and low sample efficiency. We argue that a deeper understanding of RLVR algorithms is crucial for developing more capable reasoning models. Our work establishes a foundational framework that integrates existing insights with new empirical analysis, and we aim to further explore LLM exploration mechanisms in future research.

References

- [1] Jia Deng, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, and Ji-Rong Wen. Decomposing the entropy-performance exchange: The missing keys to unlocking effective reinforcement learning. *arXiv preprint arXiv:2508.02260*, 2025.
- [2] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

- [3] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [4] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. CoRR, abs/2501.12948, 2025.
- [5] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.
- [6] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.
- [7] Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*, 2025.
- [8] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.
- [9] Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.
- [10] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- [11] Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*, 2025.
- [12] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 9426–9439. Association for Computational Linguistics, 2024.
- [13] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.
- [14] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *CoRR*, abs/2504.13837, 2025.

- [15] Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025.
- [16] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.
- [17] Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. Pass@k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*, 2025.
- [18] Claude E. Shannon. A mathematical theory of communication. Bell Syst. Tech. J., 27(3):379–423, 1948.
- [19] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.
- [20] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- [21] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics*, *ACL 2025*, *Vienna, Austria, July 27 August 1, 2025*, pages 10495–10516. Association for Computational Linguistics, 2025.
- [22] Jinhao Jiang, Zhipeng Chen, Yingqian Min, Jie Chen, Xiaoxue Cheng, Jiapeng Wang, Yiru Tang, Haoxiang Sun, Jia Deng, Wayne Xin Zhao, Zheng Liu, Dong Yan, Jian Xie, Zhongyuan Wang, and Ji-Rong Wen. Technical report: Enhancing LLM reasoning with reward-guided tree search. *CoRR*, abs/2411.11694, 2024.
- [23] Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *CoRR*, abs/2504.02495, 2025.
- [24] Wenlei Shi and Xing Jin. Heimdall: test-time scaling on the generative verification. *CoRR*, abs/2504.10337, 2025.
- [25] Fei Bai, Yingqian Min, Beichen Zhang, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. Towards effective code-integrated reasoning. *arXiv* preprint arXiv:2505.24480, 2025.
- [26] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for Ilm reasoning. arXiv preprint arXiv:2506.01939, 2025.
- [27] Jiakang Wang, Runze Liu, Fuzheng Zhang, Xiu Li, and Guorui Zhou. Stabilizing knowledge, promoting reasoning: Dual-token constraints for rlvr. *arXiv preprint arXiv:2507.15778*, 2025.
- [28] Zhipeng Chen, Kun Zhou, Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. Not everything is all you need: Toward low-redundant optimization for large language model alignment. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 15337–15351. Association for Computational Linguistics, 2024.

- [29] Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*, 2020.
- [30] Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025.
- [31] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Oiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. CoRR, abs/2501.12948, 2025.
- [32] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.
- [33] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024.
- [34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [35] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.
- [36] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

A Appendix

A.1 Gradient Derivation

We derive the gradient of the GRPO objective J_{GRPO} with respect to the logits $\mathbf{z} \in \mathbb{R}^V$. Recall the policy probability for token o_t^i :

$$\pi_{\theta}(o_t^i) = \operatorname{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{V} e^{z_j}},$$

where V is the vocabulary size. The gradient of $\pi_{\theta}(o_t^i)$ w.r.t. z_k is:

$$\frac{\partial \boldsymbol{\pi}_{\theta}(o_{t}^{i})}{\partial z_{k}} = \boldsymbol{\pi}_{\theta}(o_{t}^{i}) \left(\mathbb{I}(o_{t}^{i} = v_{k}) - \boldsymbol{\pi}_{\theta}(v_{k}) \right),$$

with $\mathbb{I}(\cdot)$ the indicator function and v_k the k-th vocabulary token. Applying the chain rule to J_{GRPO} :

$$\begin{split} \frac{\partial J_{\text{GRPO}}}{\partial z_k} &= \left[\hat{r}_t \cdot \min \left(\hat{A}_t, \text{clip}(\hat{A}_t, 1 - \epsilon, 1 + \epsilon) \right) \right] \\ & \cdot \frac{1}{\pi_{\theta}(o_t^i)} \cdot \frac{\partial \pi_{\theta}(o_t^i)}{\partial z_k} \\ &= \alpha_t \left(\mathbb{I}(o_t^i = v_k) - \pi_{\theta}(v_k) \right). \end{split}$$

Table 6: Results for pass@k. All values are pass@8, except for HumanEval, which is pass@4.

Method	GPQA	HumanEval	AIME24	AIME25	AMC23	MATH500	Avg.
Qwen2.5-7B							
BASE	63.13	14.18	20.30	7.32	77.50	87.00	44.91
GRPO	48.98	22.10	35.91	27.50	70.00	90.00	49.08
GRPO+PPL	50.50	21.49	35.48	24.35	85.00	92.40	51.54
GRPO+LOCATION	<u>61.11</u>	23.02	29.75	<u>25.36</u>	80.00	90.00	51.54
Qwen2.5-Math-7B							
BASE	66.67	26.68	33.76	21.61	70.96	84.46	50.69
GRPO	55.55	26.37	40.69	37.44	85.00	93.60	56.44
GRPO+PPL	54.55	33.38	56.89	33.18	87.50	94.40	59.98
GRPO+LOCATION	<u>55.55</u>	26.07	<u>55.33</u>	30.85	<u>85.00</u>	93.20	57.67

Vectorizing over the vocabulary V, the gradient is:

$$\frac{\partial J_{\text{GRPO}}}{\partial \mathbf{z}} = \alpha_t \left(\mathbf{e}(o_t) - \boldsymbol{\pi}_{\theta} \right), \tag{13}$$

where $e(o_t) \in \mathbb{R}^V$ is the one-hot vector for token o_t , $\pi_\theta \in \mathbb{R}^V$ is the policy distribution, and $\alpha_t = \hat{r}_t \cdot \min(\hat{A}_t, \operatorname{clip}(\hat{A}_t, 1 - \epsilon, 1 + \epsilon))$.

Crucially, the policy update operates on the language model head weights $\mathbf{W} \in \mathbb{R}^{V \times d}$, where $\mathbf{z} = \mathbf{W} h$ and $h \in \mathbb{R}^d$ is the last transformer layer's output. By the chain rule:

$$\frac{\partial J_{\text{GRPO}}}{\partial \mathbf{W}} = \frac{\partial J_{\text{GRPO}}}{\partial \mathbf{z}} \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{W}} = \underbrace{\alpha_t \left(\boldsymbol{e}(o_t) - \boldsymbol{\pi}_{\theta} \right)}_{\boldsymbol{\epsilon} \in \mathbb{R}^V} \cdot \boldsymbol{h}^{\top},$$

yielding a gradient matrix $\frac{\partial J_{\text{GRPO}}}{\partial \mathbf{W}} \in \mathbb{R}^{V \times d}$. The magnitude of this update is quantified by its Frobenius norm:

$$G_t = \left\| \alpha_t \left(\mathbf{e}(o_t) - \boldsymbol{\pi}_{\theta} \right) \mathbf{h}^{\top} \right\|_{F}, \tag{14}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This serves as the token-wise update magnitude proxy.

A.2 Pass@k Results

Results of pass@k on six benchmarks are shown in Tab. 6. It can be seen that the average scores of our method on both the out-of-domain and in-domain benchmarks are higher than those of the GRPO baseline. However, all three methods struggle to surpass the performance of the base model on out-of-domain benchmarks, suggesting that applying reinforcement learning in the mathematics domain alone may weaken capabilities in other fields.

A.3 Case Study

We compared the answers to the same question from models trained using three different methods: GRPO, GRPO+PPL, and GRPO+POSITION. The results are presented in Tab. 7,Tab. 8, and Tab. 9 respectively. We found that the responses from the GRPO+PPL and GRPO+POSITION models were noticeably more granular, with more detailed formula derivations, making them significantly easier to understand than those from the GRPO model.

Question

Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of s kilometers per hour, the walk takes her 4 hours, including t minutes spent in the coffee shop. When she walks s+2 kilometers per hour, the walk takes her 2 hours and 24 minutes, including t minutes spent in the coffee shop. Suppose Aya walks at $s+\frac{1}{2}$ kilometers per hour. Find the number of minutes the walk takes her, including the t minutes spent in the coffee shop.

Answer from GRPO

To determine the number of minutes the walk takes Aya when she walks at a speed of $s+\frac{1}{2}$ kilometers per hour, we need to first set up the problem using the information given in the problem. Let's start by defining the variables and the relationships involved. The total distance of the walk is 9 kilometers, and Aya spends t minutes in the coffee shop. The total time for the walk at a speed of s kilometers per hour takes 4 hours. This total time includes both the time taken to walk and the time spent in the coffee shop. Therefore, the time taken to walk 9 kilometers at a speed of s kilometers per hour is given by:

Time taken to walk =
$$\frac{9}{s}$$
 hours

The total time taken for the walk including the time spent in the coffee shop is 4 hours. So, we have:

Time taken to walk + Time spent in coffee shop = 4 hours

Therefore,

$$\frac{9}{s} + \frac{t}{60} = 4$$
 (1)

For the second scenario, where Aya walks at a speed of s+2 kilometers per hour, the total time taken for the walk including the time spent in the coffee shop is 2 hours and 24 minutes, which is equivalent to 2.4 hours. So we have:

$$\frac{9}{s+2} + \frac{t}{60} = 2.4 \quad (2)$$

Now we have two equations (1) and (2): ...

Subtracting equation (2) from equation (1), we get:...

Simplifying the left side of the equation, we have:...

Combining the fractions on the left side:...

Multiplying both sides by s(s + 2):...

Dividing both sides by 1.6:...

So, we have the quadratic equation:...

We solve this quadratic equation using the quadratic formula $s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, where a = 1, b = 2, and c = -11.25:...

This gives two solutions:...

Since the speed s must be positive, we discard the negative solution. Therefore,

$$s=2.5~\mathrm{km/h}$$

...

Therefore, the number of minutes the walk takes Aya, including the time spent in the coffee shop, when she walks at a speed of $s + \frac{1}{2}$ km/h is 204.

Question

Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of s kilometers per hour, the walk takes her 4 hours, including t minutes spent in the coffee shop. When she walks s+2 kilometers per hour, the walk takes her 2 hours and 24 minutes, including t minutes spent in the coffee shop. Suppose Aya walks at $s+\frac{1}{2}$ kilometers per hour. Find the number of minutes the walk takes her, including the t minutes spent in the coffee shop.

Answer from GRPO+PPL

*Step 1: Analyze the first scenario where Aya walks at a speed of s kilometers per hour.**-**Walking Time: ** When Aya walks at a constant speed of s kilometers per hour, the time she spends walking is the distance divided by her speed. The distance she walks is 9 kilometers. -The walking time can be calculated as: ...

- **Total Time: ** The total time for the walk, including the time spent in the coffee shop, is 4 hours. - The time spent in the coffee shop is t minutes, which we convert to hours by dividing by 60: t minutes = $\frac{t}{60}$ hours. - Thus, the total time is the sum of the walking time and the time spent in the coffee shop, which equals 4 hours. Thus,...

Substitute the expression for the walking time into this equation:...

Multiply through by 60 to clear the fractions:...

*Step 2: Analyze the second scenario where Aya walks at a speed of s+2 kilometers per hour.**-**Walking Time:** When Aya walks at a speed of s+2 kilometers per hour, the walking time is the distance divided by this new speed, which is $\frac{9}{s+2}$ hours.- **Total Time:** The total time for the walk, including the time spent in the coffee shop, is 2 hours and 24 minutes. Convert 2 hours and 24 minutes to hours. - 2 hours and 24 minutes is equal to $2 + \frac{24}{60} = 2 + 0.4 = 2.4$ hours. - Thus, the total time is the sum of the walking time and the time spent in the coffee shop, which equals 2.4 hours. Thus,...

Substitute the expression for the walking time into this equation:...

Multiply through by 60 to clear the fractions:...

Now we have two equations: 1. $\frac{540}{s} + t = 2402$. $\frac{540}{s+2} + t = 144$ Let's subtract equation (2) from equation (1):...

Simplify the left side:...

Combine the fractions:...

Thus,...

Now we solve the equation:...

Multiply both sides by s(s+2) to clear the fraction:...

Divide both sides by 96:...

Thus, we have:...

This leads to the quadratic equation:...

Multiply through by 4 to clear the fraction:...

Rearrange the terms:...

Now we solve the quadratic equation $4s^2 + 8s - 45 = 0$ for s using the quadratic formula $s=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$ where a=4,b=8, and c=-45. The quadratic formula is:...

Substitute the coefficients into the formula:...

Calculate the discriminant:...

Thus,...

Since $\sqrt{784} = 28$, we have two possible solutions for s:...

Since speed cannot be negative, we discard $s=-\frac{9}{2}$. Thus,... Now that we have the value of s, we substitute $s=\frac{5}{2}$ back into one of the original equations to find t. We will use Eq. (1):...

Substitute $s = \frac{5}{2}$:... Calculate $\frac{540}{\frac{5}{2}}$:...

Thus,...

Solve for t:...

So, t=24 minutes.**Step 3: Determine the time it takes Aya to complete the walk if she walks at a speed of $s + \frac{1}{2}$ kilometers per hour.**First, we need to determine the speed $s + \frac{1}{2}.s = \frac{5}{2}...$ So, the total time in minutes is: 204

Question

Every morning Aya goes for a 9-kilometer-long walk and stops at a coffee shop afterwards. When she walks at a constant speed of s kilometers per hour, the walk takes her 4 hours, including t minutes spent in the coffee shop. When she walks s+2 kilometers per hour, the walk takes her 2 hours and 24 minutes, including t minutes spent in the coffee shop. Suppose Aya walks at $s+\frac{1}{2}$ kilometers per hour. Find the number of minutes the walk takes her, including the t minutes spent in the coffee shop.

Answer from GRPO+Location

To find the value of s, t and then the time Aya takes when walking at $s+\frac{1}{2}$ kilometers per hour, we start by setting up equations based on the given information about Aya's walking speed and the total time for her walk including the time at the coffee shop. First, consider the case where Aya walks at a constant speed of s kilometers per hour. The total distance she walks is 9 kilometers and the total time for her walk including the t minutes at the coffee shop is 4 hours. The walking time at speed s can be written as $\frac{9}{s}$ hours. Since the total time for the walk and the coffee shop visit is 4 hours, the walking time is $4-\frac{t}{60}$ hours. So we have the equation for the walking time at speed s:

$$\frac{9}{s} = 4 - \frac{t}{60}$$

Next, consider the case where Aya walks at a constant speed of s+2 kilometers per hour. The total time for her walk including the t minutes at the coffee shop is 2 hours and 24 minutes, which is $2+\frac{24}{60}=2.4$ hours. The walking time at speed s+2 can be written as $\frac{9}{s+2}$ hours. Since the total time for the walk and the coffee shop visit is 2.4 hours, the walking time is $2.4-\frac{t}{60}$ hours. So we have the equation for the walking time at speed s+2:...

Now we have a system of two equations:...

We can now set these two expressions for $4 - \frac{t}{60}$ and $2.4 - \frac{t}{60}$ equal to each other:...

Rearrange the equation to get:...

This equation seems problematic, so let's solve the system of equations for s and t first and then find the value of the walking time when Aya walks at $s+\frac{1}{2}$ kilometers per hour. From the first equation, we have:...

From the second equation, we have:...

Now we have two equations in terms of s and t:...

We can now set the right sides of these two equations equal to each other:...

This simplifies to 4 = 2.4, which is clearly false. So we need to solve these two equations for s and t separately. From the first equation:...

Multiply both sides by 60 to clear the fractions:...

Rearrange to get:...

From the second equation:...

Multiply both sides by 60 to clear the fractions:...

Now we have the system of equations:...

We now have two equations in terms of s and t. We can solve these two equations for s and t. First, we can equate the two expressions for t from these two equations. From equation (1):... From equation (2):...

Now we have an equation in terms of s:...

Rearrange terms to combine the fractions:...

Simplify the left side of the equation:...

Simplify the expression in the numerator of the right side:...

Now we can cross multiply to obtain:...

Simplify the equation:...

So now we have the equation:...

Rearrange terms to form a quadratic equation:...

We now have a quadratic equation to solve for s. We can use the quadratic formula $s = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$, where a = 4, b = 8, and c = -45. First compute the discriminant:...

So $\sqrt{b^2 - 4ac} = \sqrt{784} = 28$. Now we can apply the formula for the roots of the equation:...

This gives two potential solutions for s:...

••••

Thus the number of minutes the walk takes her including t minutes spent in the coffee shop when she walks at $s + \frac{1}{2}$ kilometers per hour is 204.