Where to Spend Rollouts: Hit-Utility Optimal Rollout Allocation for Group-Based RLVR

Tao Wang[1] Shuo Li[1*] Yan Sun[2] Dongsheng Ding[3] Edgar Dobriban[1]

[1]University of Pennsylvania
[2]New Jersey Institute of Technology [3]University of Tennessee, Knoxville

May 11, 2026

Abstract

Reinforcement learning with verifiable rewards (RLVR) has emerged as a central paradigm for improving the reasoning capabilities of large language models. Group-based policy optimization methods, such as GRPO, typically allocate a fixed number of rollouts to every prompt. This uniform allocation can be inefficient: it over-allocates compute to prompts whose sampled groups are already saturated while under-exploring prompts for which additional samples may reveal useful correct trajectories. To address this limitation, we introduce *hit utility*, the posterior probability that at least one rollout in a proposed additional allocation for a prompt will be correct. Building on this notion, we propose Hit-Utility Optimal Rollout Allocation (HORA), a learning-free rollout allocation policy that maximizes total posterior hit utility within each allocation batch. HORA adaptively reallocates rollout budgets while leaving the downstream reward evaluation and group-based advantage estimator unchanged. Across four mathematical reasoning benchmarks and three model scales, HORA preserves comparable Pass@1 and improves Pass@K over compute-matched GRPO in ten of twelve model–benchmark configurations, with one tie and one saturated exception. It is also drop-in compatible with other group-based estimators such as RLOO. Ablation studies indicate that the uniform prior used by HORA is competitive with five prompt-conditioned learned-prior alternatives.

Contents

*Now at Google DeepMind

[1]tawan@wharton.upenn.edu, lishuo1@seas.upenn.edu, dobriban@wharton.upenn.edu

[2]yan.sun@njit.edu

[3]dongshed@utk.edu

1 Introduction

Reinforcement learning (RL) has become a central technique for improving the reasoning capabilities of large language models (LLMs) [Guo et al., 2025]. Recent works have demonstrated that RL-based post-training can substantially enhance problem-solving performance, especially in domains where rewards can be verified automatically, such as mathematical reasoning and coding [Wen et al., 2025]. A widely used framework for reinforcement learning with verifiable rewards (RLVR) is group-based policy optimization, such as Group Relative Policy Optimization (GRPO) [Shao et al., 2024], which samples a group of rollouts per prompt and constructs a relative advantage estimate. This relative-advantage formulation avoids the need to train a high-quality value function, which can be computationally expensive [Shao et al., 2024, Ahmadian et al., 2024].

Although RL fine-tuning often improves overall accuracy, it has also been observed to degrade performance under Pass@K evaluation [Yue et al., 2025], where success is defined as at least one of K independently sampled solutions is correct [Chen et al., 2021]. Since Pass@K for large K is dominated by the model's ability to solve challenging problems, such degradation suggests a potential loss of capability on hard prompts. In group-based policy optimization methods, the same number of rollouts is typically generated for every prompt. This uniform rollout-allocation strategy can limit exploration of difficult prompts, where additional samples may be necessary to obtain even one correct solution, while wasting computation on prompts whose sampled groups are already nearly saturated [Nguyen et al., 2026].

Existing work on mitigating Pass@K degradation mainly operates at the *reward- and advantage-shaping* layer. These methods modify how the training signal is computed or weighted, for example by adding an entropy term to the advantage estimate [Cheng et al., 2026] or by computing gradients only from high-entropy tokens [Wang et al., 2025]. However, such approaches involve a trade-off between single-sample correctness and multi-sample coverage [Gai et al., 2025, Wu et al., 2025], and methods based on entropy regularization are sensitive to the regularization strength [Zhang et al., 2025].

These limitations motivate us to look beyond reward and advantage shaping. In particular, we consider a new direction: addressing Pass@K degradation at the *rollout-allocation* layer. Prior work on rollout allocation has mainly focused on training stability or computational efficiency Qu et al. [2026a,b], Nguyen et al. [2026]. In contrast, we show that appropriate rollout allocation can directly improve

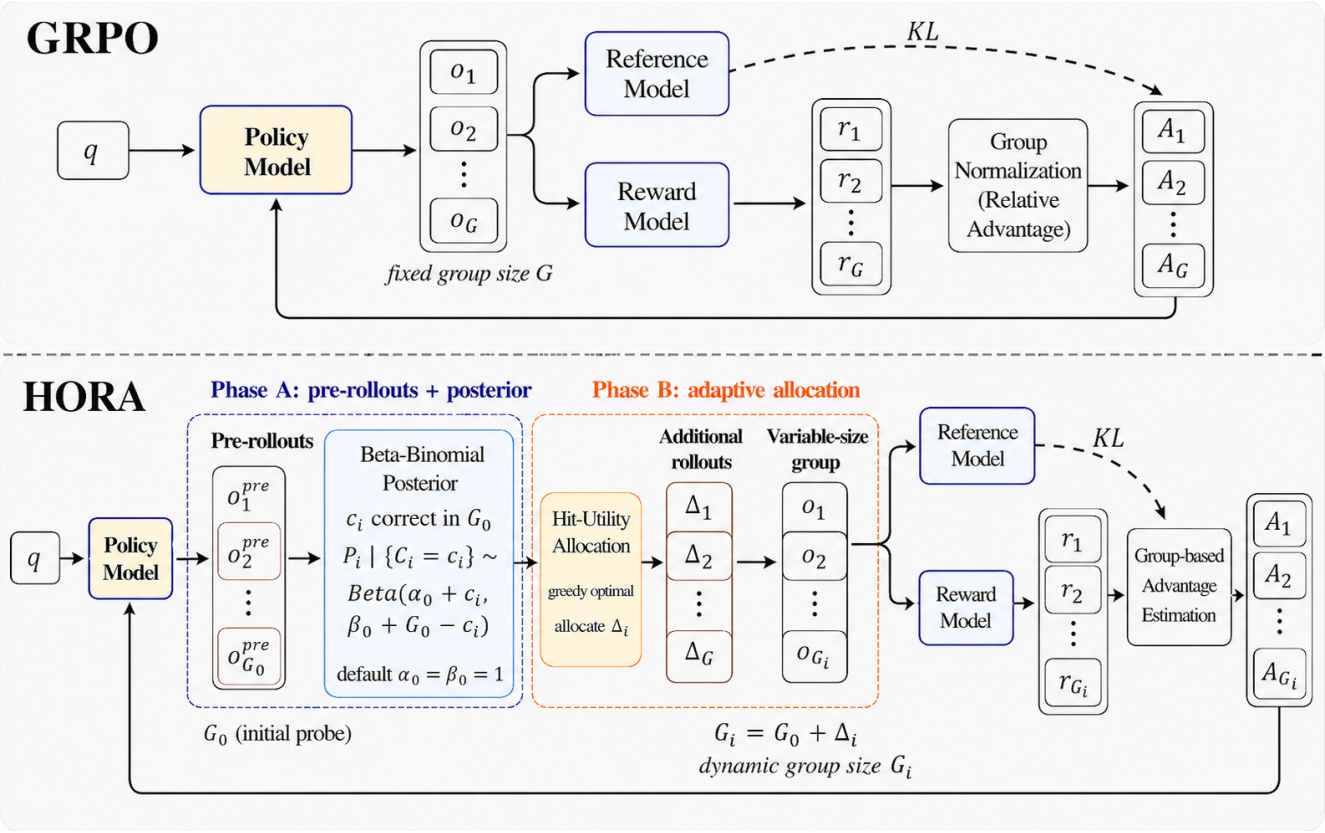

Figure 1: Comparison of fixed-group GRPO (top) and HORA (bottom). GRPO allocates a uniform group of size G to every prompt. HORA instead splits the rollout layer into two stages: Phase A draws G_0 pre-rollouts per prompt and forms a same-step posterior from the observed correct count c_i; Phase B reallocates the remaining rollout budget by maximizing the hit utility, yielding Δ_i additional rollouts and a variable final group size $G_i = G_0 + \Delta_i$. The downstream loss update is then computed on the resulting variable-size group.

Pass@K performance. Our proposed rollout allocation policy is architecturally orthogonal to reward- and advantage-shaping methods and can be combined with other group-based policy optimization framework.

We propose a learning-free rollout allocation policy, termed Hit-Utility Optimal Rollout Allocation (HORA), which prioritizes prompts whose posterior marginal hit utility remains large after accounting for diminishing returns, thereby shifting rollout budget toward the exploration frontier under a compute budget. To formalize this idea, we introduce the concept of *hit utility*: the posterior probability that at least one additional rollout for a given prompt will be correct. At each training step, HORA performs allocation in two phases. First, it conducts a small number of pre-rollouts per prompt to estimate prompt-level success probabilities. Second, it distributes the remaining rollout budget across prompts so as to maximize the aggregate hit utility. Please refer to Figure 1 for a schematic overview of HORA and its comparison with GRPO. Our contributions are three-fold:

- We propose a novel rollout-allocation module that is compatible with group-based policy optimization methods. Our module uses a utility function to prioritize prompts on the exploration frontier, and we show that the optimal allocation that maximizes the utility can be computed efficiently.
- We conduct experiments across LLMs of different scales and evaluate performance on a variety of datasets. HORA improves Pass@K across a wide range of K while maintaining comparable Pass@1 to group-based policy optimization methods with default uniform allocation. These results demonstrate that HORA improves the trade-off between correctness and coverage over group-based

policy optimization.

- We conduct ablation studies to examine the key design choices in the allocation policy. We show that the proposed computationally efficient allocation obtains comparable results to more sophisticated methods, demonstrating the simplicity and robustness of HORA.

2 Related Work

Reinforcement Learning with Verifiable Rewards (RLVR). In domains such as mathematical reasoning and code generation, rewards are often obtained automatically from answer checkers, unit tests, or other verifiers, making RLVR a scalable post-training paradigm [Guo et al., 2025, Zeng et al., 2025, Yang et al., 2025]. Group-based policy optimization is widely used in this setting [Guo et al., 2025, Yang et al., 2025]. Representative examples include Group Relative Policy Optimization (GRPO) [Shao et al., 2024], Dr. GRPO [Liu et al., 2025], Dynamic Sampling Policy Optimization (DAPO) [Yu et al., 2025], and RLOO [Ahmadian et al., 2024]. Compared with classical Proximal Policy Optimization (PPO) [Schulman et al., 2017], these methods avoid learning a value function, which is expensive and unstable in large-scale LLM training. They have been adopted in the training pipelines of powerful open-source LLMs [Guo et al., 2025, Yang et al., 2025]. Our work improves these methods by designing a more effective rollout-allocation policy.

Mitigating Pass@K Degradation in RLVR. Although RL fine-tuning improves overall accuracy, it has also been observed to degrade performance under Pass@K evaluation [Yue et al., 2025]. Existing work largely attributes this phenomenon to entropy collapse, where the policy becomes overly deterministic after RL optimization, reducing output diversity and limiting the benefit of multiple sampling [He et al., 2025, Yue et al., 2025]. To address this issue, one line of work explicitly controls entropy. For example, Cheng et al. [2026] augment the advantage with an entropy term to encourage longer exploratory reasoning chains. A second line of work restricts gradients to selected tokens or trajectories. For instance, Wang et al. [2025] apply RL updates only through the roughly 20% of tokens with the highest entropy; Gai et al. [2025] apply reward smoothing only to correct trajectories. A third line of work reweights samples: Zhu et al. [2025] upweight negative-sample reinforcement to suppress incorrect generations and redistribute probability mass toward alternative candidates. While these methods can help mitigate entropy collapse, they often involve a trade-off between correctness and coverage [Gai et al., 2025]. Our method modifies the rollout-allocation policy, which is orthogonal to existing approaches that alter reward or advantage computation.

Efficient Training via Dynamic Batching. Existing dynamic-allocation methods for group-based policy optimization mainly focus on training efficiency and stability. Examples include filtering all-correct or all-incorrect groups [Yu et al., 2025], predicting prompt difficulty with auxiliary models [Qu et al., 2026a,b], and framing rollout allocation as a variance-reduction problem for stable training [Nguyen et al., 2026]. Our work complements this direction by using rollout allocation as a mechanism for improving multi-sample reasoning performance, not only throughput, filtering, or variance reduction.

3 HORA: Hit-Utility Optimal Rollout Allocation

3.1 Background on RLVR

We consider RLVR methods based on group-based advantage estimation, where advantages are computed relative to multiple rollouts sampled for the same prompt. Let $\mathcal{Q}$ denote the prompt dataset and let $\pi_{\theta_{\text{old}}}$

be the rollout-generating policy. For a prompt $q \sim \mathcal{Q}$, $\pi_{\theta_{\text{old}}}$ samples a group of G responses $\{o_i\}_{i=1}^{G}$. The general objective function for this class of methods is

$$\mathcal{J}(\theta) = \mathbb{E}_{q\sim\mathcal{Q},\, o_i\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min\Big(\rho_{i,t}(\theta)\hat{A}_{i,t},\, \text{clip}(\rho_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon)\, \hat{A}_{i,t} \Big) \right], \tag{1}$$

where $\rho_{i,t}(\theta) := \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$ is the per-token importance ratio, and $\hat{A}_{i,t}$ is the corresponding per-token advantage. In standard GRPO, after evaluating the reward vector $\mathbf{R} = (R(q,o_1), \ldots, R(q,o_G))$, the group-relative advantage is calculated as $\hat{A}_{i,t}^{\text{GRPO}} = \frac{R(q,o_i) - \text{mean}(\mathbf{R})}{\text{std}(\mathbf{R})}$. Here the mean and standard deviation are computed over the G rewards of the responses; when the empirical standard deviation is zero, the group-relative advantage is set to zero, yielding no policy-gradient signal from that group.

While methods in this class differ in their specific advantage estimators and loss formulations, they share a default rollout interface: each prompt receives a fixed group size G. This fixed-group design is limiting when prompt difficulty varies widely. Mathematically, if the probability of generating a correct rollout for a prompt is p, then the event that a group of size G contains at least one correct rollout has probability $1-(1-p)^G$. This probability saturates quickly for easy prompts, while it remains small for prompts at the edge of the policy's exploration frontier.[1]. A uniform group size therefore fails to account for heterogeneous sampling utility across prompts.

3.2 Posterior-guided rollout allocation

To address this bottleneck, we propose Hit-Utility Optimal Rollout Allocation (HORA), a learning-free rollout-allocation policy that modifies the rollout-allocation layer while keeping the downstream reward evaluation and group-based advantage estimation unchanged. For a training batch of Γ prompts $\{q_i\}_{i=1}^{\Gamma}$, standard training uses a fixed group size G, resulting in a total of ΓG rollouts. HORA preserves the same total rollout budget but distributes it non-uniformly across prompts. Specifically, HORA divides the sampling process into two phases. Phase A draws a uniform pre-rollout group of size $G_0 < G$ for every prompt. Phase B re-allocates the remaining $\Gamma(G-G_0)$ rollouts across prompts according to the outcome of pre-rollouts. The resulting per-prompt group size satisfies $G_i = G_0 + \Delta_i$ and $\sum_{i=1}^{\Gamma} \Delta_i = \Gamma\,(G-G_0)$, where $\Delta_i \geqslant 0$ is the number of additional rollouts to be allocated to prompt q_i in Phase B. HORA uses the Phase A correctness counts to estimate each prompt's current success probability, and then determines the Phase B allocation by maximizing total hit utility. Concretely, Phase A and Phase B proceed as follows.

Phase A: Beta-Binomial model of prompt success rates. Let $R_{\text{acc}}(q,o) \in \{0,1\}$ denote whether rollout o yields a correct final answer for prompt q. For each prompt q_i, let C_i denote the random Phase-A correctness count and let c_i denote its observed realization. After drawing G_0 pre-rollouts $\{o_{i,j}^{\text{pre}}\}_{j=1}^{G_0}$ from $\pi_{\theta_{\text{old}}}$, we observe $c_i = \sum_{j=1}^{G_0} R_{\text{acc}}\left(q_i, o_{i,j}^{\text{pre}}\right) \in \{0,1,\ldots,G_0\}$. We model the prompt-level success probability p_i with a Bayesian binomial model. This is a working model chosen for conjugacy, tractability, and its ability to avoid zero-probability empirical estimates. With a Beta prior $P_i \sim \text{Beta}(\alpha_0, \beta_0)$ and binomial likelihood $C_i \mid P_i = p_i \sim \text{Binomial}(G_0, p_i)$, the posterior follows a Beta distribution by conjugacy:

$$P_i \mid C_i = c_i \;\sim\; \text{Beta}(\alpha_i, \beta_i), \quad \text{where } \alpha_i = \alpha_0 + c_i \text{ and } \beta_i = \beta_0 + G_0 - c_i. \tag{2}$$

HORA uses the uniform prior $\alpha_0 = \beta_0 = 1$ unless otherwise specified; this introduces no learned prompt-difficulty model.

[1] More formally, for a test set with n prompts, let p_i denote the probability that the current policy generates a correct solution for problem i. Then the average Pass@K metric is $\text{Pass@K} = \sum_{i=1}^{n} \left(1-(1-p_i)^K\right)/n$, therefore $1-\text{Pass@K} = \sum_{i=1}^{n}(1-p_i)^K/n$. Note that $\sqrt[K]{\sum_{i=1}^{n}(1-p_i)^K/n} \to \max_i 1-p_i$ as $K \to +\infty$. Therefore, for large K, Pass@K is dominated by problems with the smallest p_i, i.e., the hardest questions in the set.

Algorithm 1 HORA rollout construction for one training step.

Require: batch of prompts $\{q_i\}_{i=1}^{\Gamma}$; group size G; pre-rollout count $G_0 < G$; prior (α_0, β_0); behavior policy $\pi_{\theta_{\text{old}}}$

Ensure: variable-size rollout groups $\{(o_{i,1}, \ldots, o_{i,G_i})\}_{i=1}^{\Gamma}$

Phase A: pre-rollouts and posterior

for $i = 1, \ldots, \Gamma$ **do**
  draw G_0 pre-rollouts $\{o_{i,j}^{\text{pre}}\}_{j=1}^{G_0} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q_i)$
  $c_i \leftarrow \sum_{j=1}^{G_0} R_{\text{acc}}(q_i, o_{i,j}^{\text{pre}})$
  $(\alpha_i, \beta_i) \leftarrow (\alpha_0 + c_i,\ \beta_0 + G_0 - c_i)$
  $\Delta_i \leftarrow 0$
end for

Phase B: greedy hit-utility allocation

$\mathcal{B} \leftarrow \Gamma(G - G_0)$
for $b = 1, \ldots, \mathcal{B}$ **do**
  $i^\star \leftarrow \arg\max_i\ B(\alpha_i + 1,\ \beta_i + \Delta_i)/B(\alpha_i, \beta_i)$
  $\Delta_{i^\star} \leftarrow \Delta_{i^\star} + 1$
end for

Additional rollouts and group assembly

for $i = 1, \ldots, \Gamma$ **do**
  draw Δ_i additional rollouts $\{o_{i,j}^{\text{add}}\}_{j=1}^{\Delta_i} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q_i)$
  pool $\{o_{i,j}^{\text{pre}}\}_{j=1}^{G_0}$ and $\{o_{i,j}^{\text{add}}\}_{j=1}^{\Delta_i}$ into $\{o_{i,j}\}_{j=1}^{G_i}$, with $G_i = G_0 + \Delta_i$
end for

return $\{(o_{i,1}, \ldots, o_{i,G_i})\}_{i=1}^{\Gamma}$

Phase B: hit-utility allocation. Given the posterior in (2), we define the per-prompt *hit utility* as

$$U_i^{\text{hit}}(\Delta_i) \ := \ \mathbb{E}_{P_i \mid C_i = c_i}\left[\, 1 \,-\, (1 \,-\, P_i)^{\Delta_i} \,\right], \tag{3}$$

which is the posterior probability that at least one of the Δ_i additional rollouts for prompt q_i is correct. HORA then allocates the remaining rollout budget by maximizing the total hit utility across prompts subject to the budget constraint:

$$\max_{\Delta_1, \ldots, \Delta_\Gamma \in \mathbb{Z}_{\geqslant 0}} \sum_{i=1}^{\Gamma} U_i^{\text{hit}}(\Delta_i) \qquad \text{s.t.} \qquad \sum_{i=1}^{\Gamma} \Delta_i \ = \ \Gamma\,(\,G \,-\, G_0\,). \tag{4}$$

Greedy allocator. The optimization problem (4) can be solved exactly by a greedy allocation rule. Let $M_i(\ell)$ denote the *marginal hit utility* of assigning the $(\ell+1)$-th additional rollout to prompt q_i:

$$M_i(\ell) \ = \ U_i^{\text{hit}}(\ell + 1) \,-\, U_i^{\text{hit}}(\ell) \ = \ \frac{B(\alpha_i + 1, \beta_i + \ell)}{B(\alpha_i, \beta_i)}, \tag{5}$$

where $B(\cdot, \cdot)$ denotes the beta function. In implementation, the same marginal can be updated stably using $M_i(0) = \alpha_i/(\alpha_i + \beta_i)$ and $M_i(\ell + 1) = M_i(\ell)(\beta_i + \ell)/(\alpha_i + \beta_i + \ell + 1)$, avoiding repeated beta-function evaluations. The greedy rule starts from $\Delta_i = 0$ for all prompts and repeatedly assigns the next rollout to the prompt with the largest current marginal gain $M_i(\Delta_i)$ until the Phase B budget $\Gamma(G - G_0)$ is exhausted. Algorithm 1 summarizes the complete rollout construction procedure, and the following proposition establishes optimality of its Phase B allocation.

Proposition 3.1. *The Phase B allocation returned by Algorithm 1 is a globally optimal solution to the integer allocation problem* (4)*.*

The proof is provided in Appendix B.1. The key observation is that each marginal sequence $M_i(0), M_i(1), \ldots$ is decreasing, so the optimal allocation is obtained by selecting the largest $\Gamma(G - G_0)$ marginal gains across prompts.

Remark 3.2. *Using the posterior distribution rather than the plug-in estimate C_i/G_0 is important for allocation. Under the plug-in estimate, any prompt with $c_i = 0$ has zero estimated success probability and therefore zero marginal hit utility, causing it to receive no additional rollouts in Phase B except in degenerate cases where all remaining marginal hit utilities are also zero. The uniform Beta prior avoids this zero-probability degeneracy while remaining learning-free. A formal statement is given in Appendix B.2. We also consider the prompt-conditioned learned priors in Section 4.2, but these variants do not provide consistent gains in our experiments.*

Training objective with variable-size groups. After allocation, the additional rollouts are drawn from $\pi_{\theta_{\text{old}}}$ and pooled with the pre-rollouts to form a final group of size $G_i = G_0 + \Delta_i$ for prompt q_i. Reward evaluation and group-based advantage estimation are computed on this group. In the policy loss of (1), we replace $1/G$ with $1/G_i$ so that each prompt has equal weight regardless of how many additional rollouts it receives. Thus, HORA changes only the rollout-allocation layer and remains compatible with RLVR methods based on group-based advantage estimation.

4 Experiments

We evaluate the benefits of HORA on mathematical reasoning across model scales and benchmark difficulty. Using standard practice Gai et al. [2025], Zhu et al. [2025], we experiment three backbone models up to 7B parameters: Qwen2.5-1.5B-Instruct, Qwen2.5-3B, and Qwen2.5-7B [Yang et al., 2025]. Each model is fine-tuned on MATH12k [Hendrycks et al., 2021] and evaluated on four benchmarks: MATH500 Hendrycks et al. [2021], AMC23 Cao et al. [2025], AIME 2024 Patel et al. [2024], and AIME 2025 Ospanov et al. [2025]. During training, each rollout uses a verifiable reward $(R_{\text{fmt}}(q, o) + R_{\text{acc}}(q, o))/2$, where $R_{\text{fmt}}, R_{\text{acc}} \in \{0, 1\}$. The format reward R_{fmt} checks whether the completion follows the required `<think>...</think><answer>...</answer>` structure, with reasoning enclosed in `<think>` tags and the final answer enclosed in `<answer>` tags. The accuracy reward R_{acc} checks whether the extracted answer is correct. All evaluation metrics are computed using R_{acc}.

For each prompt q in the evaluation dataset $\mathcal{D}$, we draw a pool of N independent samples, $o_1, \ldots, o_N$, from the evaluated policy, where $r_j(q) = R_{\text{acc}}(q, o_j)$ denotes the correctness of the j-th sample. In line with Yue et al. [2025], we assess reasoning performance using the Pass@1 and Pass@K metrics. Specifically, for any $K \leqslant N$, we employ the low-variance unbiased estimator introduced by [Chen et al., 2021]:

$$\widehat{\text{Pass}}@K \;:=\; \mathbb{E}_{q \sim \mathcal{D}}\left[1 - \binom{N - c(q)}{K} \Big/ \binom{N}{K}\right], \tag{6}$$

where $c(q) := \sum_{j=1}^{N} r_j(q)$ represents the total number of correct samples within the evaluation pool for prompt q. To accommodate varying benchmark difficulties, we follow the standard practice Yue et al. [2025] and tailor our sample sizes and evaluation range. For MATH500 and AMC23, we set $N = 256$ and evaluate Pass@K for $K \in \{8, 16, 32, 64, 128, 256\}$. For the AIME 2024 and AIME 2025 datasets, we scale the pool size to $N = 1024$ and evaluate across $K \in \{8, 16, 32, 64, 128, 256, 512, 1024\}$.

We use GRPO as a baseline and compare with HORA on top of GRPO (denoted as HORA). Our implementation uses the Dr. GRPO length normalization; we refer to the compute-matched fixed-allocation baseline as GRPO for readability. We set the group size to $G = 32$ and the HORA pre-rollout count to $G_0 = 8$. We set the KL penalty coefficient to zero throughout. For a training batch of Γ prompts $\{q_i\}_{i=1}^{\Gamma}$, HORA and GRPO use the same total rollout budget per training step, ΓG, so the comparison is compute-matched and differs only in how rollouts are allocated across prompts. We compare HORA

against GRPO with the same group size and against the corresponding pre-RL checkpoint, denoted as "Base".[2] Per-model training configuration details are listed in Appendix C.

4.1 Main Results

Table 1 reports estimated Pass@1 and Pass@K across all twelve model-benchmark configurations. Figure 2 presents representative Pass@K curves for Qwen2.5-7B on MATH500 and AIME 2025; the remaining Pass@K curves are provided in Appendix D.

Table 1: Pass@1 and Pass@K (%) across Qwen2.5 model scales and four benchmarks. We use $K = 256$ for MATH500 and AMC23, and $K = 1024$ for AIME 2024 and AIME 2025. The best value per column within each model scale is highlighted in green.

Scale	Method	MATH500		AMC23		AIME 2024		AIME 2025	
		Pass@1	Pass@K	Pass@1	Pass@K	Pass@1	Pass@K	Pass@1	Pass@K
1.5B-Inst.	Base	46.5	**91.4**	22.0	92.5	2.2	**66.7**	0.9	**63.3**
	GRPO	**50.8**	89.4	27.4	87.5	**3.1**	53.3	**1.0**	56.7
	GRPO + HORA	50.6	90.4	**27.8**	**95.0**	3.0	63.3	**1.0**	60.0
3B	Base	27.6	**93.6**	11.1	**100.0**	1.2	53.3	0.7	56.7
	GRPO	**59.9**	91.8	35.9	97.5	**6.2**	56.7	**2.4**	63.3
	GRPO + HORA	59.1	92.0	**36.3**	97.5	5.7	**60.0**	2.0	**70.0**
7B	Base	41.9	94.2	24.5	97.5	4.8	70.0	2.6	70.0
	GRPO	70.8	93.4	49.8	**100.0**	11.7	70.0	8.3	66.7
	GRPO + HORA	**73.9**	**94.4**	**53.9**	97.5	**12.0**	**73.3**	**9.2**	**73.3**

HORA achieves a superior trade-off between Pass@1 and Pass@K. In terms of Pass@1, *HORA outperforms GRPO on all four benchmarks at the 7B* scale with gains of +3.1 points on MATH500 and +4.1 points on AMC23, while performing comparably at the 1.5B and 3B scales. These findings suggest that HORA preserves the average correctness achieved by standard GRPO and may even enhance it for sufficiently large models.

Furthermore, the Pass@K metrics demonstrate a more consistent advantage over GRPO. *Across the twelve model-benchmark configurations, HORA improves upon GRPO on ten settings* and matches its performance on one; the single observed decrease occurs on Qwen2.5-7B AMC23, where GRPO has already saturated at Pass@256 = 100.0. The most substantial gains emerge on benchmarks where additional exploration is highly beneficial. For instance, HORA improves Pass@K by +10.0 points on Qwen2.5-1.5B AIME 2024, +7.5 points on Qwen2.5-1.5B AMC23, +6.7 points on Qwen2.5-3B AIME 2025, and +6.6 points on Qwen2.5-7B AIME 2025. On the four AIME configurations at the 3B and 7B scales, HORA also surpasses the corresponding Base model.

As illustrated in Figure 2, this behavior persists across the full Pass@K curve. While GRPO often drops below the Base model at large values of K, HORA consistently remains above it in representative 7B settings. At the 1.5B scale, the Base model remains strongest in Pass@K across three of the four benchmarks, indicating limited exploration headroom for the smallest instruction-tuned models. Nevertheless, HORA bridges this gap, performing closer to the Base model than GRPO across every 1.5B evaluation.

HORA effectively reallocates rollout budgets. We inspect how HORA allocates the Phase B budget. Figure 3 shows that the pre-rollout outcomes are highly non-uniform: many prompts are already saturated after Phase A, while a smaller fraction receive no correct pre-rollouts. HORA reallocates budget

[2]For the 1.5B experiments, we use `Qwen2.5-1.5B-Instruct` as the base checkpoint because `Qwen2.5-1.5B` rarely follows the required output format.

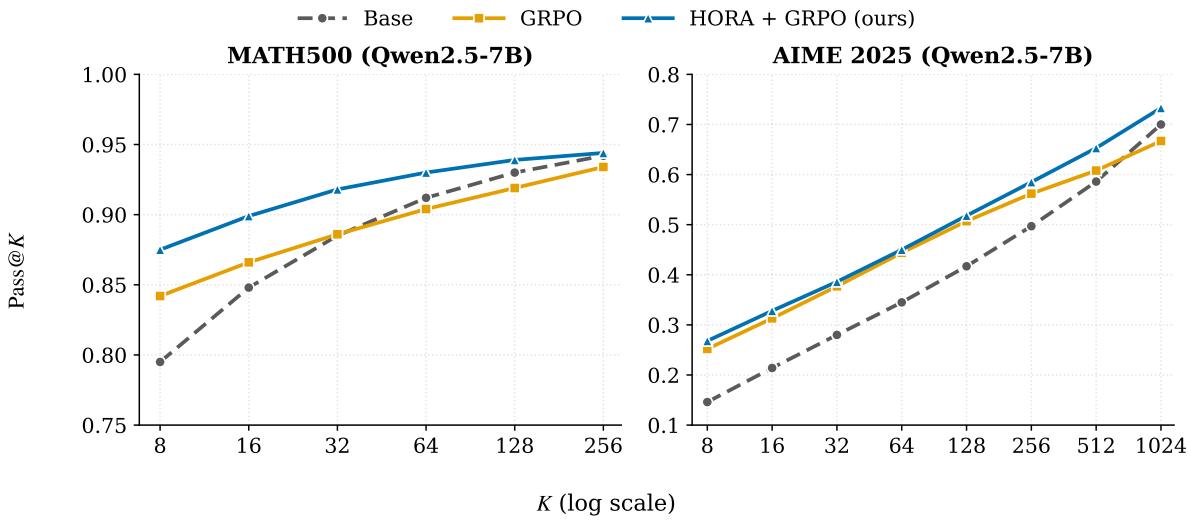

Figure 2: Pass@K curves on Qwen2.5-7B for in-distribution MATH500 (left) and out-of-distribution AIME 2025 (right). HORA (blue) lies strictly above the Base checkpoint (gray) at every K we evaluate. GRPO (orange) falls below the base model on MATH500 from $K = 64$ onward and on AIME 2025 at $K = 1024$ (66.7 vs. 70.0), whereas HORA (blue) remains above the Base model across the evaluated range in both cases.

toward hard prompts. Averaged over training, prompts with $c_i = 0$ account for 15% of the Phase A input distribution but receive 58% of the Phase B budget. Conversely, prompts with $c_i = G_0 = 8$ account for 51% of the input distribution but receive only 12% of the Phase B budget. This nonzero allocation to $c_i = 8$ is expected under the marginal utility rule: an all-correct pre-rollout group has a high first marginal hit probability, but its subsequent marginal utilities drop quickly. These additional rollouts also reveal residual incorrect responses in otherwise all-correct pre-rollout groups and thereby restore nonzero group-relative signal. Over training, the allocation to $c_i = 8$ grows gradually as the policy improves and more prompts become saturated after Phase A. HORA also produces longer responses on average and maintains a heavier entropy tail than GRPO, suggesting that the allocation policy preserves more exploratory sampling behavior; details are reported in Appendix E.

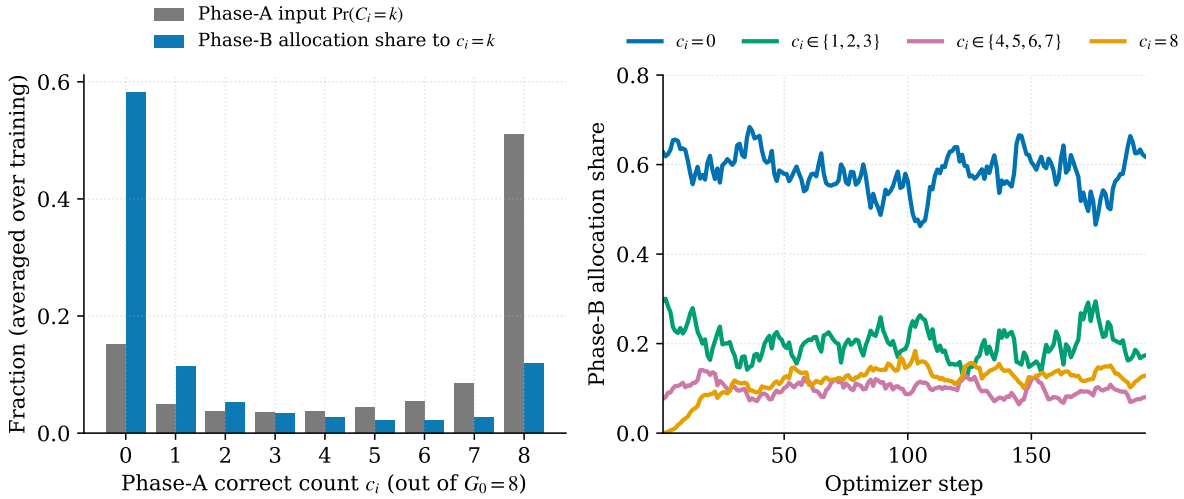

Figure 3: Allocation dynamics on the Qwen2.5-7B run. Left: average phase-A input fraction Pr($C_i = k$) and average phase-B allocation share for each observed pre-rollout correct-count bucket $c_i = k$, averaged over 196 optimizer steps. Right: phase-B allocation share over training, grouped by $c_i = 0$, $c_i \in \{1, 2, 3\}$, $c_i \in \{4, 5, 6, 7\}$, and $c_i = 8$; curves are smoothed with a 6-step moving average.

HORA is compatible with GRPO-family objectives. As discussed, HORA should be drop-in compatible with other prompt-normalized GRPO-family advantage estimators. To investigate this, we stack HORA on top of REINFORCE Leave-One-Out (RLOO) [Ahmadian et al., 2024] on Qwen2.5-7B and compare it with a compute-matched RLOO baseline using fixed group size $G = 32$. Figure 4 shows that HORA preserves the same qualitative effect under RLOO. On MATH500, RLOO improves Pass@K over Base at small K but falls below Base at large K, while HORA + RLOO remains above both curves across the evaluated range. On AMC23, Base catches up with the RLOO baseline at high K, whereas HORA + RLOO remains higher and reaches Pass@256 = 100.0. These results support that HORA can transfer to other advantage estimators.

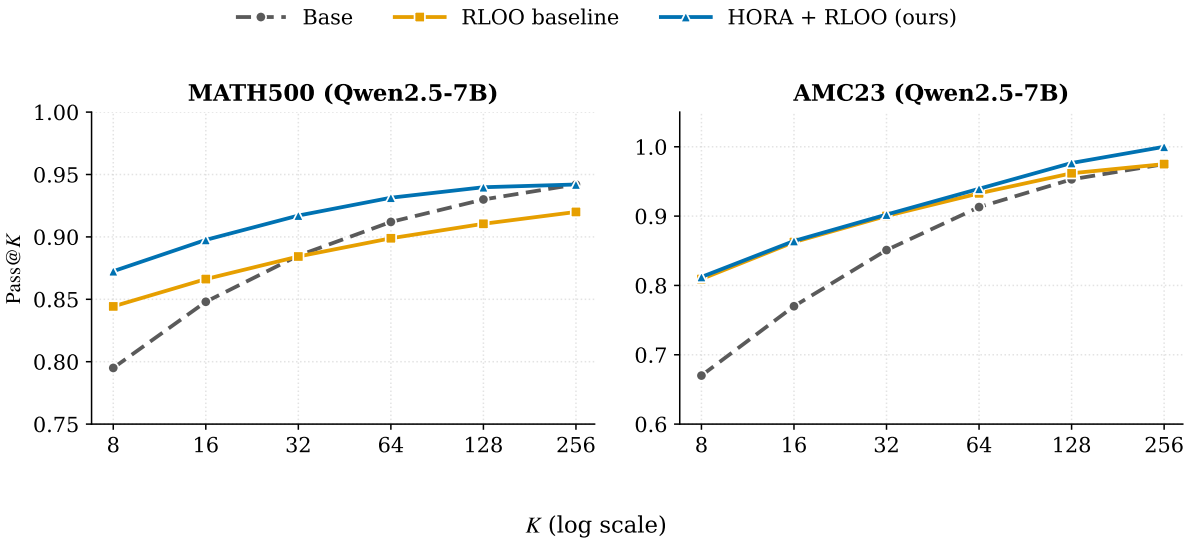

Figure 4: Pass@K curves for Base, RLOO, and HORA + RLOO on Qwen2.5-7B. HORA changes only the rollout-allocation stage, while the downstream advantage estimator is RLOO for both RL methods.

4.2 Ablations

We next ablate the main design choices in HORA. These experiments address three questions: (1) is hit-utility allocation needed beyond simpler hard-first heuristics, (2) how much budget should be spent on pre-rollouts, and (3) do auxiliary prompt-difficulty predictors improve over the fixed-prior posterior used by HORA. Unless otherwise stated, all ablations are conducted on Qwen2.5-3B with the same total rollout budget as the main experiments.

Table 2: Allocation-rule and pre-rollout-budget ablations on Qwen2.5-3B. The hard-first heuristic allocates the phase-B budget only to prompts with $c_i = 0$ after the G_0 pre-rollouts. The $G_0 = 16$ variant doubles the pre-rollout budget while keeping the total group size fixed at $G = 32$. Pass@1 and Pass@K (%) follow the evaluation setup in Section 4.

Variant	MATH500		AMC23		AIME 2024		AIME 2025	
	Pass@1	Pass@K	Pass@1	Pass@K	Pass@1	Pass@K	Pass@1	Pass@K
Hard-first ($c_i = 0$)	56.1	91.2	32.7	**97.5**	5.6	**60.0**	1.7	56.7
HORA ($G_0 = 8$)	59.1	**92.0**	**36.3**	**97.5**	**5.7**	**60.0**	**2.0**	70.0
HORA ($G_0 = 16$)	**59.5**	91.2	36.0	92.5	5.5	**60.0**	1.8	**73.3**

Comparison with the hard-first heuristic. We ablate two design choices in HORA on Qwen2.5-3B (Table 2). The first ablation tests a natural hard-first heuristic motivated by the limitation of uniform group size: after the G_0 pre-rollouts, allocate the remaining budget only to prompts with $c_i = 0$, i.e., prompts for which all pre-rollouts are incorrect. This heuristic performs worse than the hit-utility allocator,

reducing Pass@1 on all four benchmarks and lowering AIME 2025 Pass@K from 70.0 to 56.7. This supports that marginal hit utility brings benefits beyond simply routing the budget to the hardest prompts.

Pre-rollout budget G_0. The second ablation doubles the pre-rollout budget from $G_0 = 8$ to $G_0 = 16$ while keeping the total group size fixed at $G = 32$. Increasing G_0 provides more within-step evidence about prompt difficulty, but it also reduces the adaptive phase-B budget from 24 to 16 rollouts per prompt on average. Empirically, $G_0 = 16$ gives similar Pass@1 but does not uniformly improve Pass@K: it improves AIME 2025 (70.0 $\rightarrow$ 73.3) while reducing AMC23 (97.5 $\rightarrow$ 92.5). We therefore use $G_0 = 8$ as the default trade-off between estimating prompt difficulty and retaining budget for adaptive allocation.

Prior design. We next test whether the fixed Beta$(1, 1)$ prior in HORA can be improved by incorporating prompt-conditioned prior estimates. We consider both parametric and nonparametric alternatives: (a) linear probes over frozen base-model hidden states and (b) linear probes over frozen MPNet embeddings Song et al. [2020], as well as (c) the Gaussian-process predictor from VIP [Nguyen et al., 2026]; see Appendix F for details. These alternatives are more expressive, but they also introduce additional complexity. Parametric probes must be fitted or updated online from noisy rollout outcomes, making their quality sensitive to the choice of update schedule, learning rate, and distribution shift during RL training. Nonparametric predictors such as the Gaussian process used in VIP [Nguyen et al., 2026] require maintaining a kernel matrix over training-prompt embeddings and performing recursive covariance updates, introducing memory and computation costs that scale with the prompt set size.

Figure 5 shows that these more complex priors do not yield a clear advantage over the fixed prior. On AMC23, several prompt-conditioned priors are slightly ahead at intermediate values of K, but the fixed prior catches up by $K = 256$. On AIME 2025, the fixed prior is consistently better across the evaluated range and reaches Pass@1024 = 70.0, compared with at most 66.7 among the predictor-based variants. These results suggest that, in our current setting, the same-step pre-rollout outcomes already provide a strong allocation signal. Designing prompt-conditioned priors that reliably improve over this simple baseline remains an open question.

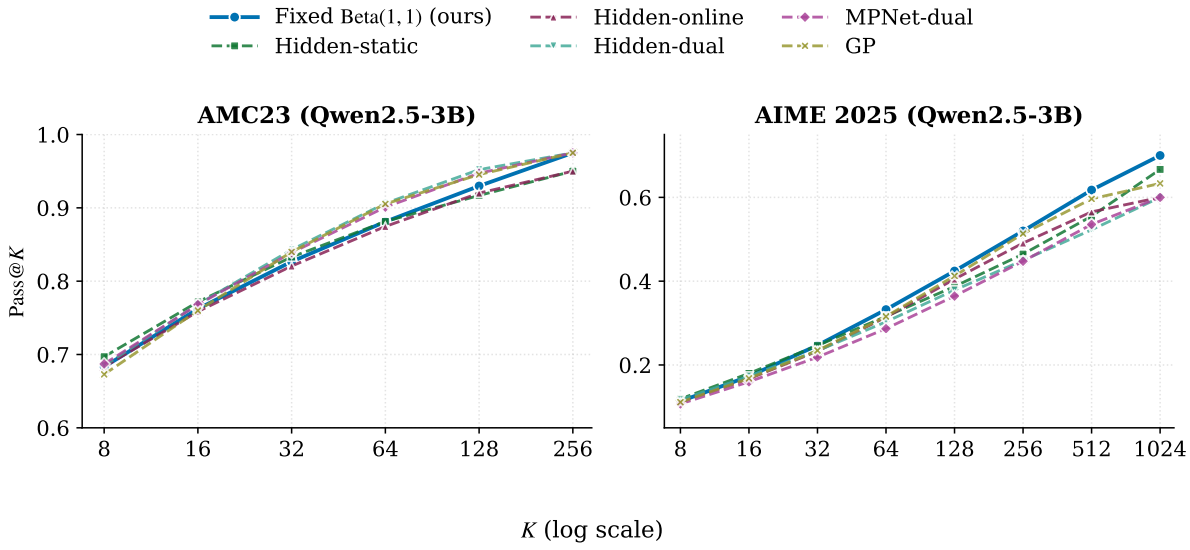

Figure 5: Pass@K curves for prior-design ablations on Qwen2.5-3B. The fixed Beta$(1, 1)$ prior is compared with prompt-conditioned priors based on policy hidden states, MPNet embeddings, and a Gaussian-process predictor.

5 Conclusion and Discussion

We introduce HORA, a novel rollout-allocation policy that prioritizes prompts near the exploration frontier in group-based RLVR training. Across multiple model scales and mathematical reasoning benchmarks, HORA improves the trade-off between correctness, measured by Pass@1, and multi-sample coverage, measured by Pass@K, compared with GRPO. These results suggest that, even without modifying the reward model, advantage estimator, or policy architecture, adaptive rollout allocation can serve as a layer for mitigating Pass@K degradation under fixed compute budgets. The rollout-allocation view also provides a flexible module for future extensions, including larger-scale training, refined prior estimation, and alternative utility specifications.

Acknowledgments

This work was partially supported by the NSF, ARO, AFOSR, ONR, and the Sloan Foundation. Portions of this work was performed on High Performance Computing resources provided by the Advanced Research Computing Services team at NJIT.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267, 2024.

Lang Cao, Yingtian Zou, Chao Peng, Renhong Chen, Wu Ning, and Yitong Li. Step guided reasoning: Improving mathematical reasoning using guidance generation and step reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21112–21129, 2025.

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

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 30377–30385, 2026.

Jingchu Gai, Guanning Zeng, Huaqing Zhang, and Aditi Raghunathan. Differential smoothing mitigates sharpening and improves llm reasoning. *arXiv preprint arXiv:2511.19942*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Andre Wang He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting grpo beyond distribution sharpening. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25559–25571, 2025.

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Hieu Trung Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2602.01601*, 2026.

Azim Ospanov, Zijin Feng, Jiacheng Sun, Haoli Bai, Xin Shen, and Farzan Farnia. Hermes: Towards efficient and verifiable mathematical reasoning in llms. *arXiv preprint arXiv:2511.18760*, 2025.

Bhrij Patel, Souradip Chakraborty, Wesley A Suttle, Mengdi Wang, Amrit Singh Bedi, and Dinesh Manocha. Aime: Ai system optimization via multiple llm evaluators. *arXiv preprint arXiv:2410.03131*, 2024.

Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. Can prompt difficulty be online predicted for accelerating rl finetuning of reasoning models? In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 1240–1250, 2026a.

Yun Qu, Qi Wang, Yixiu Mao, Heming Zou, Yuhang Jiang, Weijie Liu, Clive Bai, Kai Yang, Yangkun Chen, Saiyong Yang, et al. Small generalizable prompt predictive models can steer efficient rl post-training of large reasoning models. *arXiv preprint arXiv:2602.01970*, 2026b.

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

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

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.

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 llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, et al. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025.

Fang Wu, Weihao Xuan, Ximing Lu, Mingjie Liu, Yi Dong, Zaid Harchaoui, and Yejin Choi. The invisible leash: Why rlvr may or may not escape its origin. *arXiv preprint arXiv:2507.14843*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

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

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.

Xiaoyun Zhang, Xiaojian Yuan, Di Huang, Wang You, Chen Hu, Jingqing Ruan, Kejiang Chen, and Xing Hu. Revisiting entropy regularization: Adaptive coefficient unlocks its potential for llm reinforcement learning. *arXiv preprint arXiv:2510.10959*, 2025.

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.

A Limitations and Broader Impact

Limitations. Due to computational resource constraints, we conduct experiments up to the 7B scale. Larger-scale training, refined prior estimation, and alternative utility specifications remain important directions for future work.

Broader Impact. Our method provides a drop-in rollout-allocation module for group-based RLVR methods. On mathematical reasoning tasks, it improves the Pass@K performance of RL-fine-tuned models, suggesting stronger capability on challenging problems. This improvement may have potential positive societal impacts.

B Proofs

B.1 Proof of Proposition 3.1

We first show that the marginal gains $M_i(0), M_i(1), \ldots$ for each prompt form a decreasing sequence. For $P_i \mid C_i = c_i \sim \mathrm{Beta}(\alpha_i, \beta_i)$, the hit utility in (3) admits the closed form

$$\begin{aligned} U_i^{\mathrm{hit}}(\Delta_i) &= 1 - \mathbb{E}_{P_i|C_i=c_i}\left[(1-P_i)^{\Delta_i}\right] \\ &= 1 - \frac{1}{B(\alpha_i,\beta_i)} \int_0^1 p^{\alpha_i-1}(1-p)^{\beta_i+\Delta_i-1}\, dp \\ &= 1 - \frac{B(\alpha_i, \beta_i+\Delta_i)}{B(\alpha_i,\beta_i)}, \end{aligned} \tag{7}$$

where $B(a,b) = \int_0^1 p^{a-1}(1-p)^{b-1}\, dp$ denotes the beta function.

Recall that we define the marginal gain of the $(\ell+1)$-th additional rollout for prompt i as

$$M_i(\ell) := U_i^{\mathrm{hit}}(\ell+1) - U_i^{\mathrm{hit}}(\ell), \qquad \ell \in \mathbb{Z}_{\geqslant 0}. \tag{8}$$

Using (7), we have

$$\begin{aligned} M_i(\ell) &= \frac{B(\alpha_i, \beta_i+\ell)}{B(\alpha_i,\beta_i)} - \frac{B(\alpha_i,\beta_i+\ell+1)}{B(\alpha_i,\beta_i)} \\ &= \frac{B(\alpha_i+1, \beta_i+\ell)}{B(\alpha_i,\beta_i)}, \end{aligned} \tag{9}$$

where the last equality uses the fact that $B(a,b) = B(a+1,b) + B(a,b+1)$.

Note that the allocation objective can be written as a sum of marginal gains:

$$\sum_{i=1}^{\Gamma} U_i^{\mathrm{hit}}(\Delta_i) = \sum_{i=1}^{\Gamma} \sum_{\ell=0}^{\Delta_i-1} M_i(\ell). \tag{10}$$

Moreover, each marginal sequence is strictly decreasing:

$$\frac{M_i(\ell+1)}{M_i(\ell)} = \frac{B(\alpha_i+1, \beta_i+\ell+1)}{B(\alpha_i+1,\beta_i+\ell)} = \frac{\beta_i+\ell}{\alpha_i+\beta_i+\ell+1} < 1, \tag{11}$$

since $\alpha_i > 0$. Hence

$$M_i(0) > M_i(1) > M_i(2) > \cdots > 0. \tag{12}$$

We next prove the optimality of the greedy allocation rule by comparing its objective value with that of any feasible allocation. Let $\Delta^{\mathrm{g}} = (\Delta_1^{\mathrm{g}}, \ldots, \Delta_\Gamma^{\mathrm{g}})$ be the allocation returned by the greedy procedure, which repeatedly assigns the next rollout to the prompt with the largest current marginal gain. By the greedy property, we have the following condition:

$$M_i(\Delta_i^{\mathrm{g}} - 1) \geqslant M_j(\Delta_j^{\mathrm{g}}) \qquad \text{for every } i \text{ with } \Delta_i^{\mathrm{g}} > 0 \text{ and every } j. \tag{13}$$

Indeed, when greedy assigned the last rollout to prompt i, the available marginal for prompt i was $M_i(\Delta_i^{\mathrm{g}} - 1)$. If prompt j had already received r_j rollouts at that time, then $r_j \leqslant \Delta_j^{\mathrm{g}}$. By monotonicity, $M_j(r_j) \geqslant M_j(\Delta_j^{\mathrm{g}})$, and since greedy selected prompt i at that step,

$$M_i(\Delta_i^{\mathrm{g}} - 1) \geqslant M_j(r_j) \geqslant M_j(\Delta_j^{\mathrm{g}}).$$

Now consider any feasible allocation $\Delta = (\Delta_1, \ldots, \Delta_\Gamma)$ satisfying

$$\sum_{i=1}^{\Gamma} \Delta_i = \Gamma(G - G_0).$$

Define

$$A = \{i : \Delta_i > \Delta_i^{\mathrm{g}}\}, \qquad D = \{j : \Delta_j < \Delta_j^{\mathrm{g}}\}.$$

Because Δ and Δ^{g} use the same total budget, we have

$$\sum_{i \in A} (\Delta_i - \Delta_i^{\mathrm{g}}) = \sum_{j \in D} (\Delta_j^{\mathrm{g}} - \Delta_j).$$

Relative to the greedy allocation, prompts in A receive extra marginal gains

$$M_i(\Delta_i^{\mathrm{g}}),\ M_i(\Delta_i^{\mathrm{g}} + 1),\ \ldots,\ M_i(\Delta_i - 1),$$

each of which is at most $M_i(\Delta_i^{\mathrm{g}})$. Prompts in D lose marginal gains

$$M_j(\Delta_j),\ M_j(\Delta_j + 1),\ \ldots,\ M_j(\Delta_j^{\mathrm{g}} - 1),$$

each of which is at least $M_j(\Delta_j^{\mathrm{g}} - 1)$.

By the condition in (13), every extra marginal gain added by Δ is no larger than some marginal gain removed from Δ^{g}. Therefore,

$$\sum_{i=1}^{\Gamma} \sum_{\ell=0}^{\Delta_i - 1} M_i(\ell) \leqslant \sum_{i=1}^{\Gamma} \sum_{\ell=0}^{\Delta_i^{\mathrm{g}} - 1} M_i(\ell).$$

Using (10), this implies

$$\sum_{i=1}^{\Gamma} U_i^{\mathrm{hit}}(\Delta_i) \leqslant \sum_{i=1}^{\Gamma} U_i^{\mathrm{hit}}(\Delta_i^{\mathrm{g}}).$$

Thus no feasible allocation has larger objective value than the greedy allocation, so the greedy allocator exactly solves (4).

B.2 Degeneracy of Plug-in Hit Utility

Here, we explain the reason that we use the posterior distribution to model the success probability p_i rather than the plug-in estimate $\hat{p}_i = C_i/G_0$. Using the point estimate, the corresponding hit utility will be defined as

$$\widetilde{U}_i^{\mathrm{hit}}(\Delta_i) = 1 - (1 - \hat{p}_i)^{\Delta_i}.$$

The key difference is that our Bayesian formulation (3) computes the expectation over the posterior $P_i|C_i = c_i$. If $c_i = 0$, then $\hat{p}_i = 0$ and hence $\widetilde{U}_i^{\mathrm{hit}}(\Delta_i) = 0$ for all $\Delta_i \geqslant 0$. Its marginal gain is therefore

$$\widetilde{M}_i(\ell) = \widetilde{U}_i^{\mathrm{hit}}(\ell + 1) - \widetilde{U}_i^{\mathrm{hit}}(\ell) = 0, \qquad \forall \ell \geqslant 0.$$

For any prompt j with $0 < \hat{p}_j < 1$, the marginal gain is

$$\widetilde{M}_j(\ell) = \hat{p}_j(1 - \hat{p}_j)^{\ell} > 0, \qquad \forall \ell \geqslant 0.$$

Thus, under the greedy allocation rule, a prompt with $c_i = 0$ is never selected while any prompt with positive marginal gain remains. Intuitively, the plug-in estimator treats a zero-count prompt as if its success probability were exactly zero. The greedy allocation rule therefore withholds additional rollouts from such prompts, which conflicts with the goal of continuing to explore challenging prompts whose true success probability may be small but nonzero.

C Training configurations

Tables 3–5 list the per-model HORA training configurations used for the main-results runs in Table 1. GRPO baselines use identical hyperparameters except for the HORA-specific rows (pre-rollout count G_0 and Beta prior). All runs use the TRL implementation of GRPO with vLLM rollout in colocate mode. To accelerate format learning, we run vanilla GRPO as a warm-up at the start of each HORA run and switch to HORA once the global format-reward mean stays at or above 0.85 for three consecutive generation cycles; in practice all three models cross this threshold within about 10 optimizer steps ($\lesssim 5\%$ of total training). Phase-B allocation is run on four mini-batches of $\Gamma/4$ prompts rather than the full batch in one shot. Thus the optimality result in Proposition 3.1 applies within each allocation shard, while the total rollout budget over the full training batch remains unchanged. Rows that match TRL defaults (linear LR scheduler, max grad norm 1.0, weight decay 0, entropy coefficient 0) are omitted.

Table 3: Configuration for Qwen2.5-1.5B-Instruct.

Parameter	**Value**	**Parameter**	**Value**
Pretrained model	Qwen2.5-1.5B-Instruct	Training set	MATH12k
Prompts per batch Γ	60	Generations per prompt G	32
Pre-rollout count G_0	8	Beta prior (α_0, β_0)	$(1, 1)$
Training steps	200	Learning rate	1.0×10^{-6}
Clip ratio ε	0.2	KL coefficient	0.0
Loss type	Dr.GRPO	Importance sampling	token-truncate
Max prompt length	2048	Max response length	2048
Reward weights (fmt / acc)	0.5 / 0.5	Gradient updates per RL step	1
Rollout temperature	0.7	Precision	bfloat16
Rollout engine	vLLM (colocate)	Device	3 × Nvidia-A100

D Additional Pass@K curves

Figures 6–8 report Pass@K curves for all twelve (model, benchmark) configurations summarized in Table 1, including the two panels already shown in Figure 2.

E Length and entropy statistics

To corroborate the claim in Section 4.1 that HORA preserves more exploratory sampling behavior, Figure 9 plots mean response length and mean token entropy over training for the Qwen2.5-7B HORA run against

Table 4: Configuration for Qwen2.5-3B.

Parameter	**Value**	**Parameter**	**Value**
Pretrained model	Qwen2.5-3B	Training set	MATH12k
Prompts per batch Γ	60	Generations per prompt G	32
Pre-rollout count G_0	8	Beta prior (α_0, β_0)	$(1, 1)$
Training steps	200	Learning rate	1.0×10^{-6}
Clip ratio ε	0.2	KL coefficient	0.0
Loss type	Dr.GRPO	Importance sampling	token-truncate
Max prompt length	2048	Max response length	2048
Reward weights (fmt / acc)	0.5 / 0.5	Gradient updates per RL step	1
Rollout temperature	0.7	Precision	bfloat16
Rollout engine	vLLM (colocate)	Device	3 × Nvidia-A100

Table 5: Configuration for Qwen2.5-7B.

Parameter	**Value**	**Parameter**	**Value**
Pretrained model	Qwen2.5-7B	Training set	MATH12k
Prompts per batch Γ	60	Generations per prompt G	32
Pre-rollout count G_0	8	Beta prior (α_0, β_0)	$(1, 1)$
Training steps	200	Learning rate	1.0×10^{-6}
Clip ratio ε	0.2	KL coefficient	0.0
Loss type	Dr.GRPO	Importance sampling	token-truncate
Max prompt length	2048	Max response length	2048
Reward weights (fmt / acc)	0.5 / 0.5	Gradient updates per RL step	1
Rollout temperature	0.7	Precision	bfloat16
Rollout engine	vLLM (colocate)	Device	6 × Nvidia-A100

the compute-matched GRPO baseline. HORA produces noticeably longer completions throughout training ($\approx$ 300-token gap) and maintains a heavier entropy tail after the initial format-collapse transient, indicating that the policy continues to allocate non-trivial probability mass to alternative continuations.

F Prior-design predictor details

This section specifies the five learned-prior variants used in the prior ablation of Section 4.2. All variants replace the fixed Beta$(1, 1)$ prior used by HORA with a prompt-conditioned predictor that produces a per-prompt success-rate estimate $\hat{p}_i$ (and, for the dual-head variants, a per-prompt confidence s_i). The Beta posterior in (2) is then formed as

$$\alpha_i = s_i\,\hat{p}_i + c_i, \qquad \beta_i = s_i\,(1 - \hat{p}_i) + G_0 - c_i, \tag{14}$$

which collapses to the HORA fixed prior when $s_i = 2$ and $\hat{p}_i = 0.5$. Single-head variants and the Gaussian Process (GP) [Nguyen et al., 2026] use a fixed $s_i = 2$, while the two dual-head variants learn s_i jointly with $\hat{p}_i$. All variants are trained on the same Qwen2.5-3B / MATH12k pipeline as the main HORA run; only the prior is changed. Figure 10 reports Pass@K across all four benchmarks.

Hidden-state linear probe (static / online / dual). The hidden-state variants extract the last-token hidden state at the $\lfloor 0.6L \rfloor$-th transformer block (mid-depth, L = number of hidden layers) of the frozen base model backbone, pass it through a LayerNorm, and predict $\hat{p}_i$ with a linear head. The probe is pretrained offline on held-out rollouts using the per-prompt binomial NLL

$$\mathcal{L}_{\text{bin}} = -c_i \log \hat{p}_i - (G_0 - c_i) \log(1 - \hat{p}_i).$$

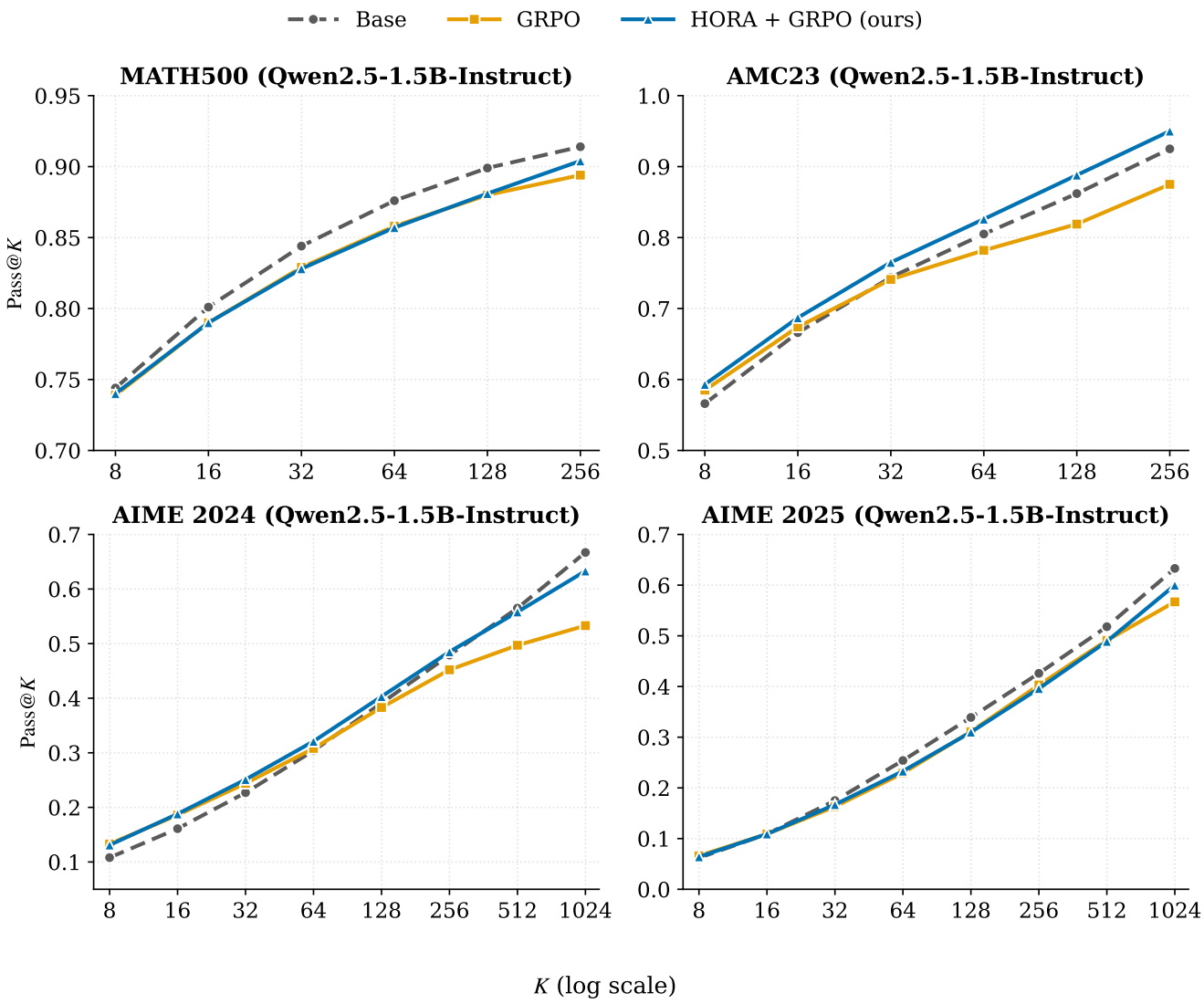

Figure 6: Pass@K curves for Qwen2.5-1.5B-Instruct across all four benchmarks.

Three update modes are evaluated: **Hidden-static** keeps the pretrained probe frozen during RL training (no online updates). **Hidden-online** updates the probe online: each optimizer step replays the most recent 10 generation cycles, takes 5 Adam steps with learning rate 10^{-3}, weight decay 10^{-2}, geometric importance weight $\gamma^t = 0.97^t$, and maintains an EMA shadow with decay 0.99 that is used at inference. **Hidden-dual** replaces the single-head probe with a dual-head probe that outputs $(\hat{p}_i, s_i)$ via two parallel linear heads; s_i is parametrized through a softplus and clamped to $[0.25, 16]$. The probe is trained with the per-prompt Beta-Binomial NLL

$$\mathcal{L}_{\mathrm{BB}} = -\log \frac{B(c_i + s_i\hat{p}_i,\, G_0 - c_i + s_i(1-\hat{p}_i))}{B(s_i\hat{p}_i,\, s_i(1-\hat{p}_i))},$$

using the same online schedule as Hidden-online.

MPNet-dual. Replaces the base policy hidden state with frozen sentence embeddings from `all-mpnet-base-v2` [Song et al., 2020]. The probe architecture (LayerNorm + dual linear heads), Beta-Binomial loss, and online update schedule are identical to Hidden-dual; only the input feature is different.

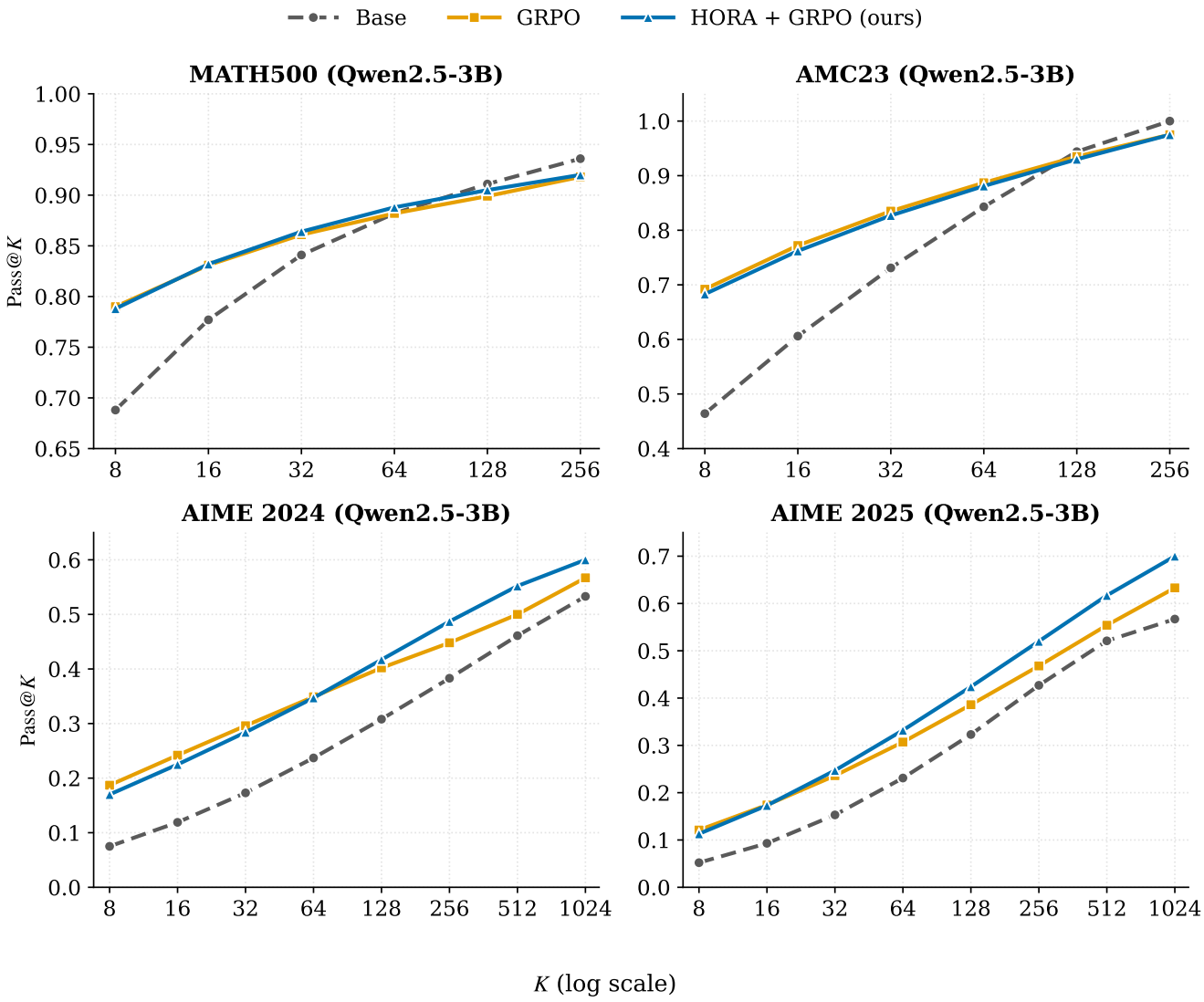

Figure 7: Pass@K curves for Qwen2.5-3B across all four benchmarks.

GP. We adopt the recursive Gaussian-process predictor proposed by VIP [Nguyen et al., 2026]: an RBF kernel over MPNet features with median-distance bandwidth, prior mean $\mu_0 = -1$ in logit-space (so the prior $\hat{p}_i \approx 0.27$), and recursive Bayesian updates on the observed empirical rates c_i/G_0. We use the implementation provided by the original authors and refer the reader to that paper [Nguyen et al., 2026] for the full formulation.

G Compatibility with the RLOO advantage estimator

Table 6 reports the per-benchmark Pass@1 and Pass@K results of RLOO [Ahmadian et al., 2024] and RLOO + HORA discussed in Section 4.1. Both methods use the same total rollout budget with $G = 32$ on Qwen2.5-7B, and differ only in the within-group advantage estimator: the RLOO baseline applies the leave-one-out advantage [Ahmadian et al., 2024] to a fixed-group rollout of size $G = 32$, while HORA + RLOO replaces the rollout stage with HORA's two-phase posterior-guided allocator and uses the same RLOO advantage calculation. The pattern observed in Section 4.1 recurs here: HORA improves Pass@K on three of four benchmarks while matching it on the fourth, and matches the RLOO baseline on average

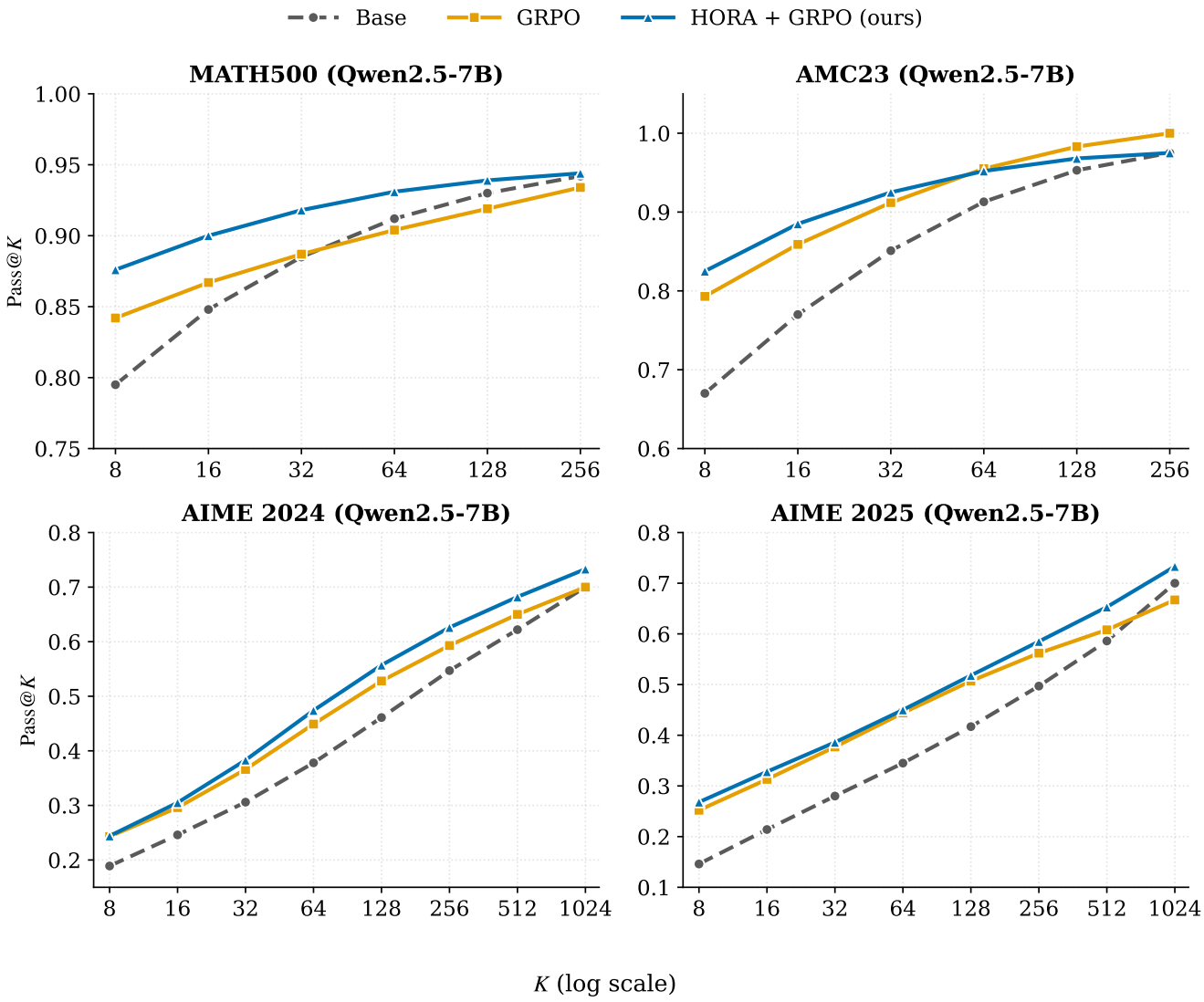

Figure 8: Pass@K curves for Qwen2.5-7B across all four benchmarks. The MATH500 and AIME 2025 panels reproduce Figure 2.

Pass@1.

Table 6: Pass@1 and Pass@K (%) for RLOO and HORA + RLOO on Qwen2.5-7B. Both methods use the same total rollout budget with $G = 32$. We use $N = 256$ for MATH500 and AMC23, and $N = 1024$ for AIME 2024 and AIME 2025. Bold marks the better value in each column.

Method	MATH500		AMC23		AIME 2024		AIME 2025	
	Pass@1	Pass@256	Pass@1	Pass@256	Pass@1	Pass@1024	Pass@1	Pass@1024
RLOO baseline	71.0	92.0	49.6	97.5	**12.3**	70.0	**10.2**	**73.3**
HORA + RLOO	**71.9**	**94.2**	**51.1**	**100.0**	11.6	**73.3**	8.8	**73.3**

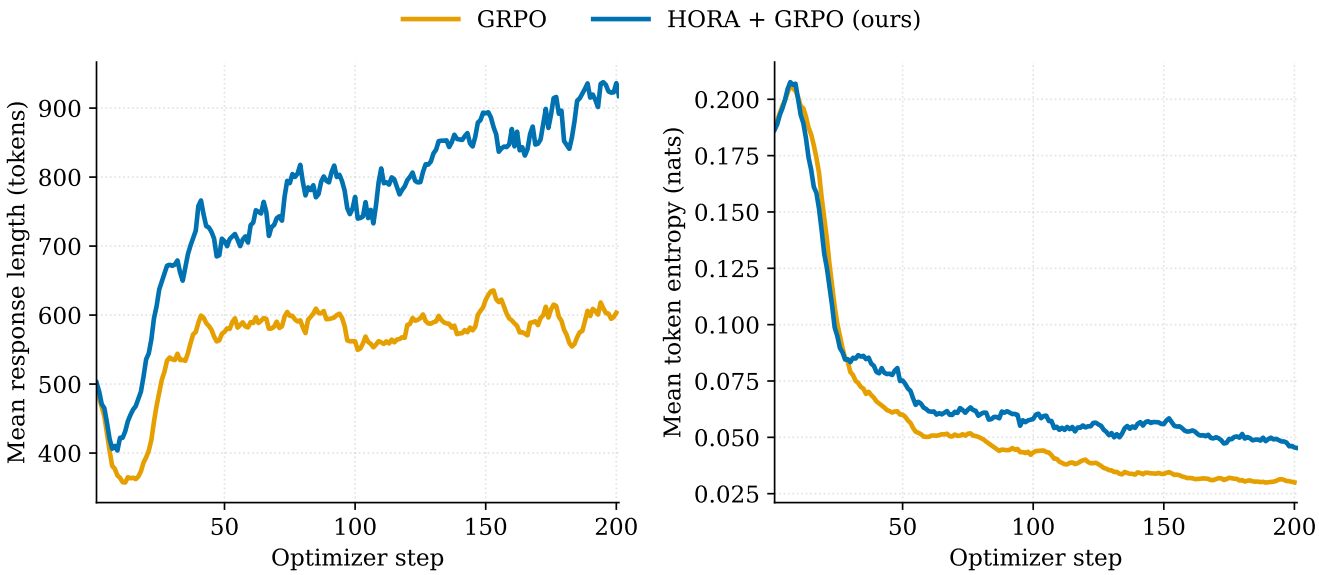

Figure 9: Mean response length (left) and mean token entropy (right) over training for HORA + GRPO and the GRPO baseline on Qwen2.5-7B. Both runs use the same total rollout budget. Curves are smoothed with a centered moving average of window $\max(5, n/30)$ where n is the number of optimizer steps logged; the smoother shrinks its window at the boundaries to avoid edge artifacts.

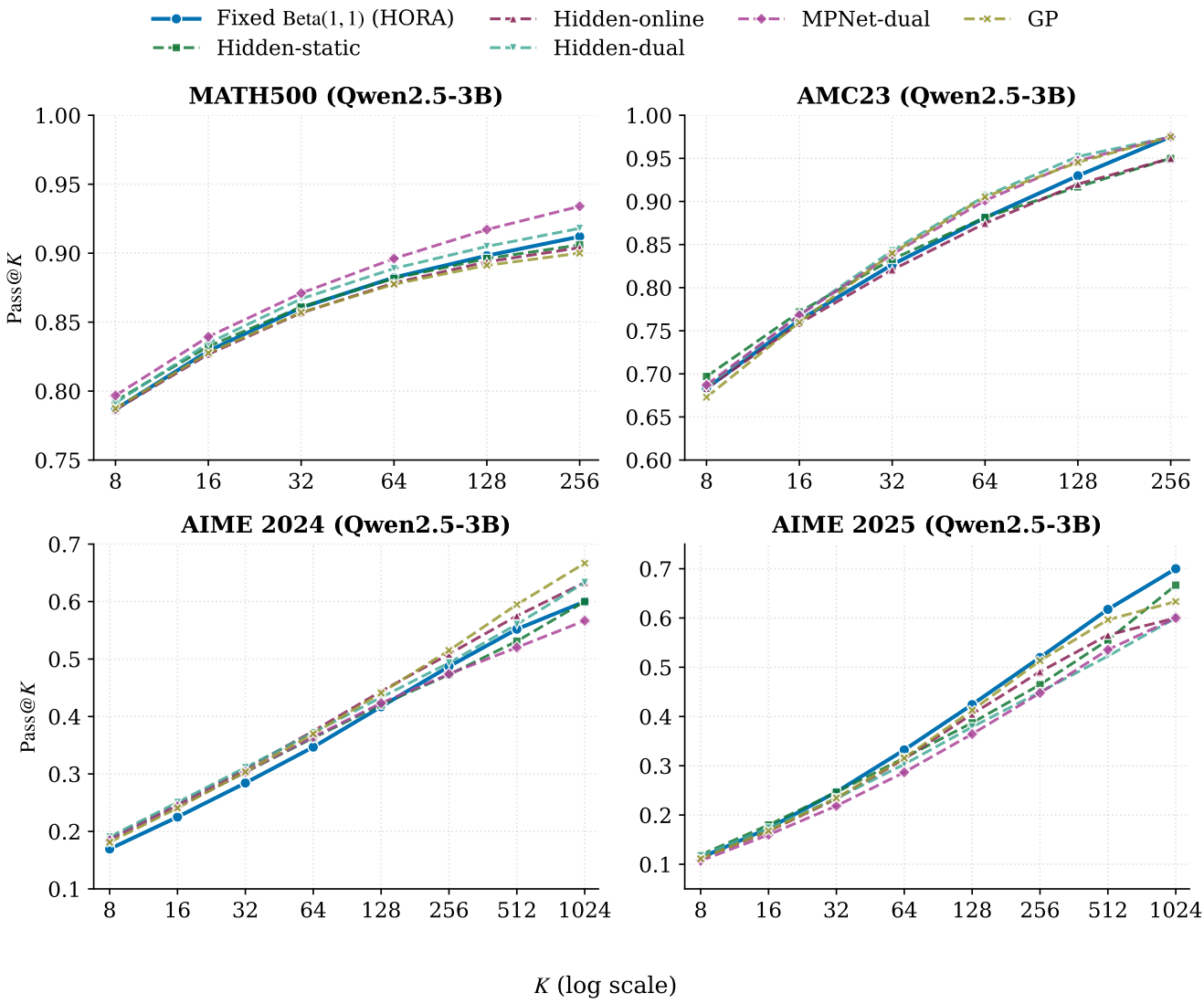

Figure 10: Prior ablation across all four benchmarks on Qwen2.5-3B. The fixed Beta(1, 1) prior used by HORA (solid blue) is compared against the five learned-prior variants described above. No learned variant uniformly dominates the fixed prior across all four benchmarks: on AIME 2025 the fixed prior is the strongest at $K = 1024$; on MATH500 the MPNet-dual variant pulls slightly ahead at large K; individual variants take the lead on different intermediate K values, but no consistent ranking emerges across benchmarks. This supports the negative finding in Section 4.2: the same-step pre-rollout signal already saturates the information available for allocation in this regime.