HIGAP: BRIDGING THE OVERHEAD GAP IN HISTORY-BASED SPECULATIVE ROLLOUT FOR MOE RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Rollout generation is a major bottleneck in reinforcement learning (RL) for large language models. History-based speculative rollout can reduce this cost by reusing trajectories from previous training steps as drafts, but its rollout-stage savings do not necessarily translate into end-to-end speedup, especially for mixture-of-experts (MoE) models. Expensive actor verification, redundant old-log-probability computation, and actor drift can substantially offset the benefits of history reuse. We present **HiGAP**, a history-based speculative rollout framework that bridges this overhead gap through full-pipeline optimization. HiGAP adaptively bounds historical draft lengths to regulate history reuse and verification overhead, leverages verification probabilities to partially bypass old-log-probability recomputation, and mitigates history staleness under actor drift through post-update probability caching and advantage-guided history updates. Across three MoE and one dense models, HiGAP consistently reduces end-to-end step time and improves training throughput while maintaining competitive training effectiveness. On Qwen3.5-35B-A3B, HiGAP reduces step time by 11.6% and improves throughput by 12.7% over vanilla GRPO, while improving the average score by 12.3 points across nine mathematical reasoning benchmarks.

1 INTRODUCTION

Reinforcement learning (RL) has become an important post-training paradigm for improving the reasoning capabilities of large language models (LLMs). Methods such as proximal policy optimization (PPO) (Schulman et al., 2017), reinforcement learning from human feedback (Ouyang et al., 2022), and group relative policy optimization (GRPO) (DeepSeek-AI, 2025) optimize language models using responses sampled from the evolving policy. Unlike supervised fine-tuning, however, RL training repeatedly interleaves online response generation, reward evaluation, and policy optimization. In reasoning-oriented RL, increasingly long responses and multiple rollout trajectories per prompt make rollout generation a major contributor to the wall-clock cost of each training iteration. This bottleneck is further amplified by long-tail generations in synchronous training, where a small number of slow responses can delay the entire training step(Hu et al., 2026; Qin et al., 2026; Xu et al., 2026). Improving rollout efficiency has therefore become an important systems problem for scaling LLM reinforcement learning.

Speculative decoding provides a natural opportunity to reduce this cost. Instead of generating every token autoregressively, it proposes multiple draft tokens and verifies them in parallel with the target model (Leviathan et al., 2023; Chen et al., 2023). Recent work has adapted this idea to RL rollout using either auxiliary drafters (Kim et al., 2026; Zhang et al., 2026; Hu et al., 2026) or previously generated trajectories (Liu et al., 2025a; He et al., 2025; Chang et al., 2026; Qin et al., 2026; Xu et al., 2026). The latter is particularly attractive for RL because historical trajectories are produced naturally during training and require no additional draft model to track the continuously evolving actor. When similar responses recur across training iterations, a historical response can serve as a speculative prefix, potentially replacing a substantial amount of expensive autoregressive generation with parallel verification.

However, reducing autoregressive generation does not necessarily translate into end-to-end RL speedup. Historical drafts still require actor verification, while PPO/GRPO training introduces additional actor-side computation such as old-log-probability evaluation. Meanwhile, policy updates can

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

make historical trajectories increasingly stale, reducing their reuse efficiency. Consequently, rollout-stage savings may be offset by verification and training-side overheads. We refer to this discrepancy between *rollout-stage savings* and the realized *end-to-end RL speedup* as the **overhead gap**.

The overhead gap becomes particularly pronounced for mixture-of-experts (MoE) models. MoE models activate only a subset of experts for each token, providing a favorable compute-to-capacity ratio for large-scale language models (Jiang et al., 2024; Dai et al., 2024). However, multi-token speculative verification can activate a larger set of experts, increasing expert-weight movement and memory-bandwidth pressure during verification (Huang et al., 2025; Saxena et al., 2025; McDanel et al., 2026). Meanwhile, as the RL policy evolves, historical trajectories can become increasingly stale, shortening their reusable prefixes (Ma et al., 2026). Consequently, history-based speculation may incur substantial verification cost while reusing only a small fraction of historical trajectories. Indeed, our experiments show that naive history reuse can reduce rollout generation yet make the overall RL step *slower* than vanilla GRPO on all three evaluated MoE actors.

To address this issue, we present **HiGAP**, a history-based speculative rollout framework designed to bridge the overhead gap in MoE RL. HiGAP optimizes history reuse across the full RL pipeline. It adaptively bounds historical draft lengths to regulate verification cost, and reuses verification probabilities to partially bypass redundant old-log-probability computation. To mitigate history staleness under actor drift, HiGAP further employs post-update probability caching and advantage-guided history updates. Together, these mechanisms regulate the cost and utility of history reuse, allowing rollout-stage savings to translate into end-to-end training speedup.

We make the following contributions:

- We identify and characterize the *overhead gap* in history-based speculative rollout for MoE RL, where rollout-stage savings can be offset by expensive verification, short reusable prefixes, redundant old-log-probability computation, and actor drift.
- We propose HiGAP, a full-pipeline history-based speculative rollout framework that regulates verification and history reuse, reuses verification probabilities for policy optimization, and mitigates history staleness as the actor evolves.
- We evaluate HiGAP on three MoE actors and one dense actor. On the three MoE actors, naive history reuse increases end-to-end step time by 0.9–32.5%, whereas HiGAP consistently reduces it by 11.6–14.9% over vanilla GRPO, while maintaining comparable reward trajectories and competitive downstream performance.

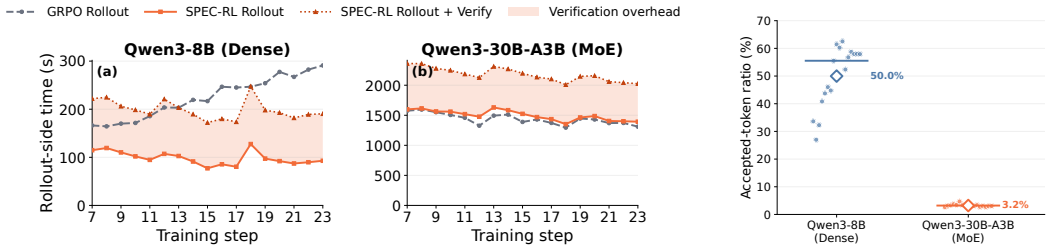
2 MOTIVATION

History-based speculative rollout replaces part of autoregressive generation with historical draft reuse, but reducing rollout generation alone does not necessarily translate into end-to-end efficiency. Figure 1 illustrates two factors behind this overhead gap.

Verification can offset rollout-stage savings. Historical drafts must be verified by the current actor before they can be reused, introducing additional actor-side computation. To expose this trade-off directly, Figure 1(a) compares the per-step rollout-generation time of vanilla GRPO and SPEC-RL, while the shaded region shows the additional verification cost incurred by SPEC-RL. For the dense Qwen3-8B model, history reuse substantially reduces rollout generation time, although part of this saving is offset by verification. In contrast, for Qwen3-30B-A3B, SPEC-RL provides little reduction in rollout generation time while still incurring substantial verification overhead. These results show that reducing autoregressive generation alone is insufficient: the cost of verifying historical drafts can substantially reduce, or even eliminate, the rollout-stage benefit of history reuse.

Limited draft acceptance further restricts history reuse. The effectiveness of history reuse also depends on how much of a historical draft can actually be accepted by the current actor. Figure 1(b) shows the accepted-token ratios across post-warm-up training steps. Qwen3-8B achieves a mean accepted-token ratio of 50.0%, whereas Qwen3-30B-A3B averages only 3.2%. Thus, in the evaluated MoE actor, only a small fraction of each historical draft can be reused, while verification is still required for the proposed draft. This combination of limited effective reuse and substantial verification overhead makes it difficult for history reuse to yield meaningful rollout-stage savings.

These observations suggest that effective history reuse should be optimized at the level of the full RL pipeline rather than rollout generation alone. They motivate two complementary design goals in HiGAP: **(1) reduce and reuse verification cost**, and **(2) mitigate history staleness under actor drift**.



(a) Verification overhead.

(b) History reuse efficiency.

Figure 1: The overhead gap in history-based speculative rollout. (a) Per-step rollout-generation time for GRPO and SPEC-RL. The shaded region denotes the additional actor-verification cost incurred by SPEC-RL on top of rollout generation. (b) Distribution of accepted-token ratios across post-warm-up training steps; diamonds denote the mean.

3 METHOD

3.1 OVERVIEW

Figure 2 illustrates the overall pipeline of **HiGAP**, which integrates history-based speculative rollout into PPO/GRPO training without an additional draft model. HiGAP combines adaptive history pruning with *partial log-probability bypass* to control verification overhead and reuse rollout-side computation (Figure 2(b,d)). It further maintains a policy-refreshed history cache and applies *advantage-guided history updates* to improve the freshness and utility of cached trajectories (Figure 2(c,e)). These components form two complementary design paths: *reducing and reusing verification cost* (Section 3.2) and *mitigating history staleness under actor drift* (Section 3.3), enabling rollout-stage savings to translate into end-to-end RL speedup.

3.2 REDUCING AND REUSING VERIFICATION COST

HiGAP reduces actor-side verification overhead from two complementary directions: it bounds the verification scope of historical drafts and reuses verification probabilities to avoid redundant old-log-probability computation.

Adaptive History Pruning History-based speculative rollout requires the current actor to verify historical draft tokens before they can be reused. Let L_{draft} denote the number of historical tokens submitted for verification. The verification cost can be generally expressed as

$$T_{\text{verify}} = f(L_{\text{draft}}, B, \mathcal{M}), \tag{1}$$

where B denotes the verification batch size and $\mathcal{M}$ captures the execution characteristics of the actor model. For a fixed model and batch configuration, T_{verify} increases with the amount of historical context being verified. For MoE actors, verification additionally involves expert routing and inter-device communication (Jiang et al., 2024; Dai et al., 2024), making unnecessary verification particularly costly.

Motivated by the verification cost analysis above, HiGAP controls verification cost by bounding the length of historical drafts before actor verification. A retrieved historical response can be substantially longer than the prefix that is likely to remain reusable under the current actor. Verifying the entire response therefore increases verification cost, especially for MoE actors where additional tokens may incur expensive expert-side computation. Meanwhile, aggressively reusing long historical prefixes reduces the amount of fresh generation performed by the current policy. HiGAP therefore uses adaptive history pruning to balance verification cost and history reuse, rather than simply maximizing speculative acceptance.

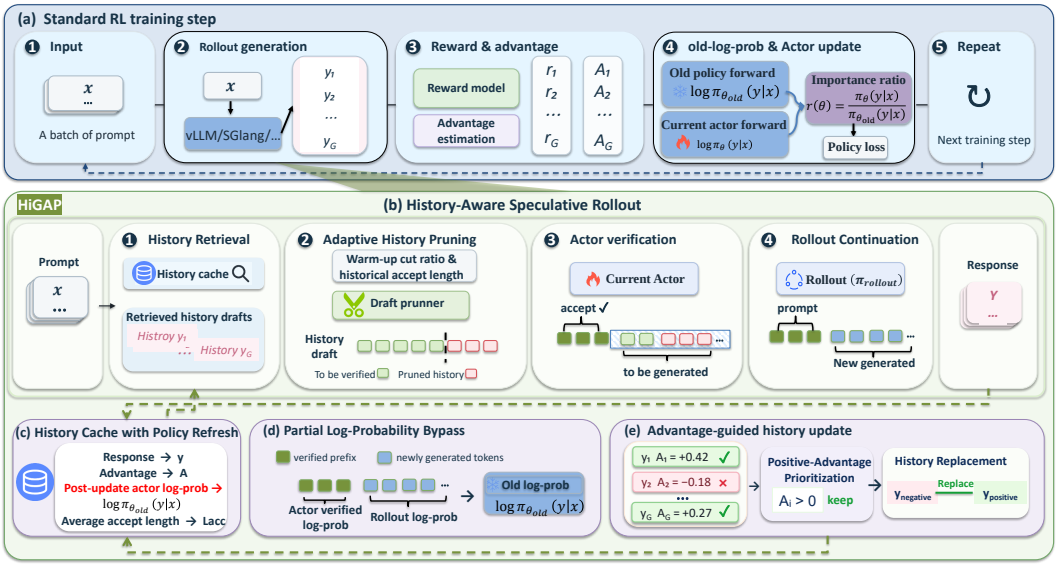


Figure 2: Overview of HiGAP. HiGAP bridges the overhead gap by reducing and reusing verification cost while mitigating history staleness under actor drift.

For each training step t, let $C_{t,i}$ denote the verification cut position of the i-th historical response, i.e., the position up to which the historical draft remains reusable after speculative verification. HiGAP records the batch-level average

$$\bar{C}_t = \frac{1}{B}\sum_{i=1}^{B} C_{t,i}, \tag{2}$$

where B is the number of historical responses in the batch. This statistic provides an empirical estimate of the reusable-prefix extent observed at the current stage of training.

To obtain a stable verification-length estimate without introducing additional online profiling, HiGAP reuses the cut statistic from the corresponding step in the previous epoch. Let S denote the number of training steps per epoch. At step t, the verification budget is set as

$$R_{\text{cut}}^{(t)} = \min\left(R_{\max}, \max\left(1, \left\lfloor (1+\rho)\bar{C}_{t-S} \right\rfloor\right)\right), \tag{3}$$

where $R_{\max}$ is the length of the retrieved historical response and ρ is a small slack ratio. We use $\rho = 0.1$ by default, retaining a small margin beyond the previously observed cut position to accommodate changes in acceptance as the actor evolves.

During warm-up, when the corresponding previous-epoch statistic is not yet available, HiGAP initializes the verification budget to half of the average valid length of the retrieved responses. The response tokens, attention masks, cached probabilities, and MoE routing metadata are then consistently truncated to $R_{\text{cut}}^{(t)}$ before verification. In this way, HiGAP adapts the verification scope to the reusable prefixes observed at a similar stage of the previous epoch, while retaining sufficient headroom for changes in acceptance as the actor evolves. Section 4.4 empirically examines the resulting efficiency–quality trade-off.

Partial Log-Probability Bypass History-based speculative rollout creates an opportunity to reuse computation across rollout and policy optimization. In PPO/GRPO-style training, policy updates rely on token-level importance ratios to account for the difference between the policy being optimized and the policy that generated the training responses. For a response token y_t, the standard ratio is

$$r_t(\theta) = \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t})}, \tag{4}$$

where $\pi_{\theta_{\text{old}}}$ denotes the actor policy that generated the current response and π_θ denotes the policy being optimized. The ratio measures the token-level policy change and is used by the clipped

PPO/GRPO objective to constrain policy updates. Obtaining the denominator through a separate training-side forward pass introduces additional actor-side computation.

Prior work has shown that independently recomputing token probabilities in the rollout and training stages can also introduce training–inference mismatch due to differences in their underlying implementations (Zhong et al., 2026). Motivated by this observation and the bypass principle of reusing rollout-time probabilities, HiGAP extends probability reuse to history-based speculative rollout. Specifically, it reuses the probabilities obtained during verification for the accepted historical prefix and those obtained during generation for the newly generated suffix:

$$\pi_{\theta_{\text{old}}}(y_t \mid x, y_{<t}) = \begin{cases} \pi_{\text{verify}}(y_t \mid x, y_{<t}), & t \le L_{\text{acc}}, \\ \pi_{\text{generate}}(y_t \mid x, y_{<t}), & t > L_{\text{acc}}, \end{cases} \tag{5}$$

where L_{acc} is the accepted historical-prefix length, and π_{verify} and π_{generate} denote probabilities obtained during verification and generation, respectively, both under the same old policy $\pi_{\theta_{\text{old}}}$. This cross-stage probability reuse preserves the original GRPO objective while partially bypassing the old-log-probability forward pass and reusing rollout-time probabilities for policy optimization.

3.3 Mitigating History Staleness under Actor Drift

HiGAP mitigates policy–history mismatch by refreshing cached policy information after actor updates and prioritizing positive-advantage trajectories during history updates.

Post-update Probability Caching A trajectory entering actor optimization is initially associated with old-policy probabilities from the rollout stage. However, actor optimization is performed over a sequence of micro-batches, during which the policy is progressively updated. Consequently, the old-policy probabilities can become stale with respect to the actor state later in the training step.

HiGAP exploits the forward computation already performed during actor optimization to refresh the cached policy information. Let $\pi_{\theta_k^{(m)}}$ denote the actor used to process micro-batch m at training step k. For trajectories in this micro-batch, HiGAP stores the log-probabilities computed during the actor-update forward pass:

$$\log \mu_{\text{cache}}(y_t \mid x, y_{<t}) \leftarrow \log \pi_{\theta_k^{(m)}}(y_t \mid x, y_{<t}). \tag{6}$$

Compared with the old-policy log-probabilities obtained before actor optimization, these values are computed from a more recently updated actor within the current training step and therefore provide fresher policy information for subsequent history reuse.

Importantly, HiGAP does not perform an additional forward pass with the final updated actor. The cached log-probabilities are directly reused from the actor-update computation of each micro-batch. This provides a low-overhead approximation to refreshing the history cache toward the updated policy, mitigating history staleness as the actor evolves.

Advantage-Guided History Update Policy freshness alone does not determine which trajectories are most useful for future reuse. During GRPO training, multiple responses are sampled for the same prompt and assigned advantages according to their relative rewards. HiGAP uses this signal to preferentially retain trajectories that are favored by the current optimization objective.

For responses $\{y_i\}_{i=1}^{G}$ sampled for the same prompt x, let A_i denote their group-relative advantages. HiGAP partitions them into

$$\mathcal{Y}^+ = \{y_i \mid A_i > \tau_+\}, \qquad \mathcal{Y}^- = \{y_i \mid A_i < \tau_-\}, \tag{7}$$

where τ_+ and τ_- are set to zero by default. Among the eligible positive responses, HiGAP selects the highest-advantage trajectory as the donor:

$$y^* = \arg\max_{y_i \in \mathcal{Y}^+} A_i. \tag{8}$$

Each eligible negative response uses y^* as its replacement cache source, subject to a maximum number of replacements per prompt. If either set is empty, no replacement is performed. This best-positive replacement rule biases the history cache toward trajectories reinforced by the current

GRPO objective without introducing an additional trajectory-quality model. Together with post-update probability caching, it maintains the history cache from two complementary perspectives: the former refreshes cached *policy information*, while the latter improves *trajectory selection*. Combined with the verification-cost optimizations in Section 3.2, these mechanisms jointly control the cost and utility of history reuse during RL training.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate HiGAP on Qwen3.5-35B-A3B (Team, 2026), Qwen3-30B-A3B (Yang et al., 2025), Moonlight-16B-A3B (Liu et al., 2025b)) and Qwen3-8B (Yang et al., 2025)) using GRPO (DeepSeek-AI, 2025) with the veRL framework (veRL Contributors, 2026). All models are trained on DeepMath dataset (DeepMath Contributors, 2026). We compare against vanilla GRPO and SPEC-RL (Liu et al., 2025a). We evaluate on nine mathematical reasoning benchmarks, detailed experimental settings and evaluation protocols are provided in Appendix A.

4.2 END-TO-END EFFICIENCY

Overall Efficiency. Table 1 summarizes the rollout-stage and end-to-end efficiency across all four actors. For the three MoE models, HiGAP consistently reduces step time by 11.6–14.9% and improves throughput by 11.8–16.0% across all three MoE actors. The Qwen3.5-35B-A3B results further highlight the difference between rollout-stage and end-to-end efficiency: SPEC-RL reduces rollout-generation time by 31.9%, compared with 15.3% for HiGAP, but increases the overall step time by 0.9%; HiGAP instead reduces step time by 11.6% and improves throughput by 12.7%. This comparison shows that larger rollout-generation savings do not necessarily translate into better end-to-end efficiency. On the dense Qwen3-8B actor, HiGAP reduces rollout-generation time by 35.3% and step time by 31.3%, while improving throughput by 10.9%. Figure 3 further shows that these gains are sustained throughout training. After the initial warm-up phase, HiGAP consistently maintains lower step time and higher throughput over the subsequent epochs despite normal fluctuations.

Table 1: Main rollout and end-to-end efficiency results averaged after excluding the initial warm-up steps. Percentage changes are relative to vanilla GRPO.

Method	Qwen3.5-35B-A3B	Qwen3-30B-A3B	Moonlight-16B-A3B	Qwen3-8B
Rollout generation time (s) ↓				
vanilla GRPO	1043.7	1437.3	311.6	224.2
SPEC-RL	**710.5** (↓31.9%)	1498.4 (↑4.3%)	345.0 (↑10.7%)	**98.3** (↓56.2%)
HiGAP	884.3 (↓15.3%)	**1359.2** (↓5.4%)	**308.9** (↓0.9%)	145.0 (↓35.3%)
Step time (s) ↓				
vanilla GRPO	2813.5	4833.1	1462.3	764.4
SPEC-RL	2838.5 (↑0.9%)	5671.7 (↑17.4%)	1937.8 (↑32.5%)	660.6 (↓13.6%)
HiGAP	**2487.5** (↓11.6%)	**4178.6** (↓13.5%)	**1245.0** (↓14.9%)	**525.0** (↓31.3%)
Throughput (tokens/s/GPU) ↑				
vanilla GRPO	488.1	685.4	794.4	1072.6
SPEC-RL	485.3 (↓0.6%)	601.5 (↓12.2%)	674.8 (↓15.1%)	971.9 (↓9.4%)
HiGAP	**549.9** (↑12.7%)	**766.0** (↑11.8%)	**921.3** (↑16.0%)	**1189.9** (↑10.9%)

Efficiency Breakdown. Figure 4 decomposes the end-to-end training step into its major components and reveals why naive history reuse is insufficient. SPEC-RL reduces rollout generation from 1044 (s) to 710 (s), but introduces 312 (s) of actor verification while leaving old-log-probability computation almost unchanged. The rollout saving is therefore largely offset by additional actor-side computation, making the overall step slightly slower than vanilla GRPO. HiGAP reduces verification to 140 (s) through adaptive history pruning and nearly eliminates the separate old-log-probability

pass through probability reuse. History reuse thus yields a net end-to-end benefit rather than merely shifting computation from rollout generation to actor-side overhead.



Figure 3: Per-step end-to-end efficiency across four evaluated models. The final checkpoint-saving step is omitted.

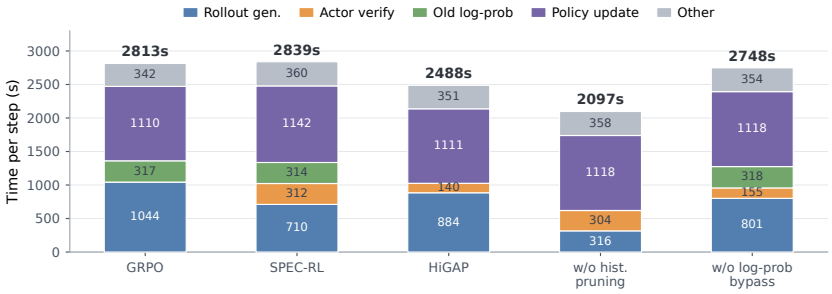


Figure 4: Wall-clock time breakdown on Qwen3.5-35B-A3B.

4.3 Accuracy Analysis

Training dynamics. Figure 5 compares the training reward trajectories of vanilla GRPO, SPEC-RL, and HiGAP across all four evaluated models. HiGAP generally follows reward trends comparable to the baselines throughout training, suggesting that its end-to-end efficiency gains do not substantially alter the observed optimization dynamics.

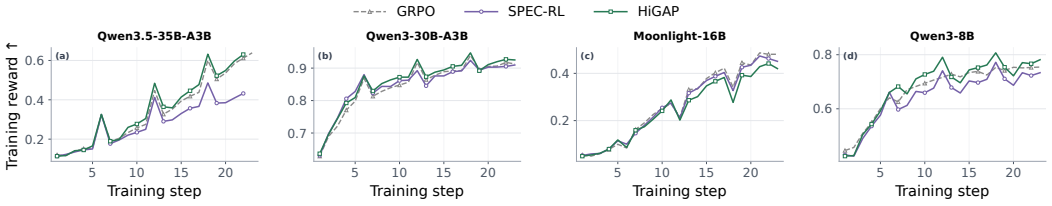


Figure 5: Training reward trajectories across the four evaluated models.

Downstream performance. Table 2 evaluates the resulting checkpoints on nine mathematical reasoning benchmarks. HiGAP achieves strong downstream performance on the two Qwen MoE actors, improving the average score from 65.2 to 77.5 on Qwen3.5-35B-A3B and from 64.3 to 65.9 on Qwen3-30B-A3B relative to vanilla GRPO. On Moonlight-16B-A3B, HiGAP achieves an average score of 34.3, slightly above SPEC-RL (33.8), although below vanilla GRPO (37.1). On Qwen3-8B, HiGAP achieves 57.4, while removing advantage-guided history updates improves the score to 58.7,

outperforming SPEC-RL (58.1) and approaching vanilla GRPO (59.1). Overall, these results suggest that HiGAP's system-level speedup generally preserves competitive downstream performance rather than relying on degraded training quality.

Table 2: Accuracy (%) across nine mathematical reasoning benchmarks. Base denotes the checkpoint before GRPO. Minerva/Olympiad/College denote Minerva Math/OlympiadBench/College Math. Bold and underlined values indicate the best and second-best results.

Model	Method	MATH-500	GSM8K	Minerva	Olympiad	MMLU-STEM	AMC23	MATH	College	AIME24	Avg.
Qwen3.5-35B-A3B	Base	89.6	88.9	26.1	55.1	66.5	85.0	90.2	**52.1**	46.7	66.7
	vanilla GRPO	83.4	93.0	34.9	61.8	41.0	92.5	84.1	36.0	60.0	65.2
	SPEC-RL	83.4	90.4	22.1	56.4	30.1	87.5	81.7	34.4	56.7	60.3
	HiGAP	**94.6**	**95.2**	**57.0**	**62.5**	**80.8**	**95.0**	**94.3**	47.8	**70.0**	**77.5**
Qwen3-30B-A3B	Base	68.0	92.6	30.9	28.3	**57.4**	42.5	67.5	34.7	10.0	48.0
	vanilla GRPO	**88.6**	94.7	50.0	**52.0**	51.0	80.0	89.3	**40.2**	33.3	64.3
	SPEC-RL	88.0	95.3	47.1	48.7	49.8	72.5	89.1	39.6	33.3	62.6
	HiGAP	88.4	**95.8**	**52.9**	51.4	52.3	**82.5**	**90.2**	39.6	**40.0**	**65.9**
Moonlight-16B-A3B	Base	18.4	42.5	5.9	5.8	19.6	7.5	18.3	11.4	**3.3**	14.7
	vanilla GRPO	**54.0**	**74.7**	**30.1**	**18.5**	**42.7**	25.0	**52.6**	**33.3**	**3.3**	**37.1**
	SPEC-RL	50.4	71.3	18.0	15.3	40.9	**31.0**	47.2	30.0	0.0	33.8
	HiGAP	49.6	71.0	24.3	17.0	40.3	27.5	49.1	30.2	0.0	34.3
Qwen3-8B	Base	65.0	84.2	32.4	30.5	58.9	40.0	65.4	33.7	6.7	46.3
	vanilla GRPO	**81.4**	**93.6**	48.2	44.0	**75.0**	50.0	81.3	41.9	**16.7**	**59.1**
	SPEC-RL	80.2	**93.6**	**49.3**	**44.1**	72.4	50.0	**81.4**	**42.3**	10.0	58.1
	HiGAP	78.4	92.5	44.1	41.6	70.1	60.0	79.7	40.2	10.0	57.4
	HiGAP*	78.0	92.7	44.1	42.4	73.0	**65.0**	78.9	41.1	13.3	58.7

HiGAP* without advantage-guided history update.

4.4 Ablation Study

We ablate the major components of HiGAP on Qwen3.5-35B-A3B. Table 3 summarizes their effects on system efficiency, history reuse, and downstream performance, with full benchmark results reported in Appendix A.5. Partial log-probability bypass provides a clear system benefit: removing it restores 318.5 s of old-log-probability computation and increases step time from 2487.5 s to 2747.5 s. Post-update probability caching improves history reusability; without it, the average accepted length decreases from 639.6 to 420.1 tokens and step time increases to 2613.8 s. Advantage-guided history update has a smaller but consistent effect on both efficiency and downstream performance.

History pruning exhibits an efficiency–quality trade-off. Without pruning, the accepted length increases from 639.6 to 1593.0 tokens and rollout time decreases from 884.3 s to 315.8 s, despite higher verification cost. Figure 6 shows that history reuse approaches 1.0 in later training, nearly eliminating autoregressive generation. However, this is accompanied by lower validation performance, with the average downstream score decreasing from 77.5 to 68.9. These results show that HiGAP benefits from regulating history reuse rather than simply maximizing speculative acceptance.

Table 3: Ablation study on Qwen3.5-35B-A3B.

Method	Overall Efficiency		Time Breakdown (s)			Reuse & Quality	
	Step ↓	Throughput ↑	Rollout	Verify	Old-log-prob	Acc. Len.	Avg. score ↑
HiGAP	2487.5	549.9	884.3	140.0	1.2	639.6	77.5
w/o adaptive history pruning	2097.1	664.8	315.8	303.7	1.0	1593.0	68.9
w/o partial log-prob bypass	2747.5	500.2	801.3	155.2	318.5	783.2	66.2
w/o advantage-guided history update	2544.2	539.1	912.4	155.7	1.3	630.0	75.1
w/o post-update probability caching	2613.8	525.7	974.7	156.8	1.4	420.1	75.7

5 Related Work

Efficient LLM RL systems. Recent systems improve LLM RL efficiency through resource orchestration, execution planning, and asynchronous training (Sheng et al., 2025; Mei et al., 2025; Fu et al., 2025). These approaches optimize pipeline execution and resource utilization, whereas HiGAP targets the overhead of history-based speculative rollout.

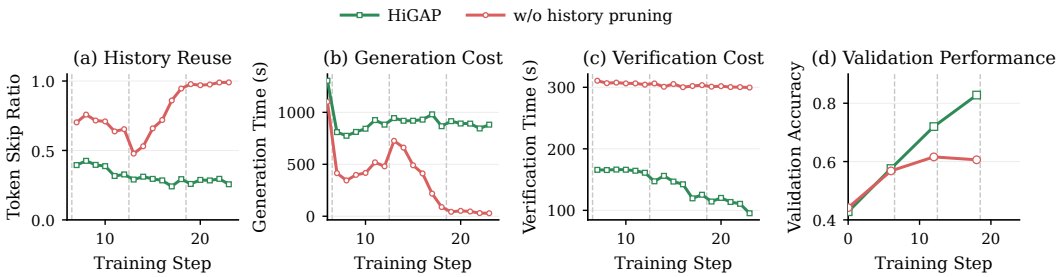


Figure 6: Effect of history pruning.

Speculative rollout. Speculative decoding accelerates autoregressive inference by verifying multiple draft tokens with the target model in parallel (Leviathan et al., 2023; Chen et al., 2023). Recent work extends this idea to RL rollout, where the target policy continuously evolves during training. One line of work maintains an explicit drafter: EfficientRollout derives a quantized drafter from the target actor (Kim et al., 2026), FastGRPO performs online draft learning under high-concurrency rollout (Zhang et al., 2026), and TLT adapts the drafter using idle resources during long-tail generation (Hu et al., 2026). While effective, these approaches require the drafter to remain synchronized with an evolving RL policy, introducing additional update or synchronization costs.

History-Based Speculative Rollout An alternative is to reuse trajectories generated during RL training. SPEC-RL verifies historical trajectory segments with the current actor (Liu et al., 2025a), while RhymeRL exploits cross-iteration trajectory similarity (He et al., 2025). SRT, SEER, and BubbleSpec further improve history caching, scheduling, and long-tail utilization (Chang et al., 2026; Qin et al., 2026; Xu et al., 2026). These methods enable speculative rollout without maintaining a separate draft model. Unlike prior work that primarily targets draft availability or rollout-stage efficiency, HiGAP focuses on translating history reuse into *end-to-end* RL speedup for MoE actors. MoE verification incurs additional expert routing and communication costs (Jiang et al., 2024; Dai et al., 2024), while policy updates can reduce the compatibility of cached trajectories with the evolving actor. HiGAP therefore jointly optimizes verification, probability reuse, and history maintenance at the full-pipeline level.

6 Limitations

Our evaluation focuses primarily on mathematical reasoning tasks and a limited set of MoE architectures, and further validation on broader RL tasks and model families would strengthen the generality of our findings. Some HiGAP components, such as adaptive history pruning, also rely on heuristic choices that may require adaptation across training settings. Moreover, while we observe a pronounced overhead gap on MoE actors, we do not isolate the effects of expert routing and communication. Future work will explore adaptive history-reuse policies and extend HiGAP to broader RL tasks and larger-scale MoE training.

7 Conclusion

We presented HiGAP, a history-based speculative rollout framework for efficient MoE RL training. HiGAP addresses the overhead gap between rollout-stage savings and end-to-end speedup through adaptive history pruning, verification-probability reuse, and drift-aware history updates. Experiments across MoE and dense actors show that HiGAP consistently reduces step time and improves training throughput while maintaining competitive training effectiveness. These results highlight that effective history reuse in RL requires jointly optimizing rollout generation, verification, and training-side overheads.

AI USE STATEMENT

Generative AI tools were used to assist with drafting, language editing, and organizing the manuscript based on author-provided notes and materials. All technical content, methodological descriptions, equations, analyses, and experimental results were reviewed and verified by the authors.

ETHICS STATEMENT

This work focuses on improving the training-system efficiency of reinforcement learning for large language models. It does not introduce new datasets, collect user data, or directly extend the capabilities of language models. By reducing the computational cost of RL training, the proposed techniques may lower the resources required for model post-training. As with other training-efficiency techniques, their broader impact ultimately depends on how the resulting models are developed and deployed.

REPRODUCIBILITY STATEMENT

We provide the model configurations, training and rollout settings, hardware and software environment, evaluation metrics, and benchmark protocols in Appendix A. The main text describes the key mechanisms of HiGAP, including adaptive history pruning, partial log-probability bypass, and history-update policies. We additionally report wall-clock breakdowns and component-level ablations to facilitate reproduction and analysis of the reported efficiency gains.

REFERENCES

Chi-Chih Chang, Siqi Zhu, Zhichen Zeng, Haibin Lin, Jiaxuan You, Mohamed S. Abdelfattah, Ziheng Jiang, and Xuehai Qian. Srt: Accelerating reinforcement learning via speculative rollout with tree-structured cache, 2026. Manuscript.

Charlie Chen et al. Accelerating large language model decoding with speculative sampling, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024.

DeepMath Contributors. Deepmath: Mathematical reasoning dataset. Dataset, 2026. URL `https://huggingface.co/datasets/DeepMath/DeepMath`. Accessed 2026-09-16.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. In *Advances in Neural Information Processing Systems*, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Jingkai He, Tianjian Li, Erhu Feng, Dong Du, Qian Liu, Tao Liu, Yubin Xia, and Haibo Chen. History rhymes: Accelerating llm reinforcement learning with rhymerl, 2025. Manuscript.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a.

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, 2021b.

Qinghao Hu, Shang Yang, Junxian Guo, Xiaozhe Yao, Yujun Lin, Yuxian Gu, Han Cai, Chuang Gan, Ana Klimovic, and Song Han. Taming the long-tail: Efficient reasoning rl training with adaptive drafter. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1933–1948, 2026.

Zongle Huang, Lei Zhu, ZongYuan Zhan, Ting Hu, Weikai Mao, Xianzhi Yu, Yongpan Liu, and Tianyu Zhang. MoESD: Unveil speculative decoding's potential for accelerating sparse MoE. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4176.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lelio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Theophile Gervet, Thibaut Lavril, Thomas Wang, Timothee Lacroix, and William El Sayed. Mixtral of experts, 2024.

Minseo Kim, Minjae Lee, Seunghyuk Oh, Kevin Galim, Donghoon Kim, Coleman Hooper, Harman Singh, Amir Gholami, Hyung Il Koo, and Wonjun Kang. Efficientrollout: System-aware self-speculative decoding for rl rollouts, 2026. Manuscript.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Chung-Ching Wu, et al. Solving quantitative reasoning problems with language models, 2022.

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 *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Bingshuai Liu, Ante Wang, Zijun Min, Liang Yao, Haibo Zhang, Yang Liu, Xu Han, Peng Li, Anxiang Zeng, and Jinsong Su. Spec-rl: Accelerating on-policy reinforcement learning with speculative rollouts, 2025a. Manuscript.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025b.

Changlian Ma, Zizheng Huang, Xiangyu Zeng, Yi Wang, Cheng Liang, Kun Tian, Xinhai Zhao, and Limin Wang. Balancing the experts: Unlocking lora-moe for grpo via mechanism-aware rewards. In *International Conference on Learning Representations*, 2026.

Mathematical Association of America. American mathematics competitions (amc). Competition, 2023. URL `https://maa.org/student-programs/amc/`. Accessed 2026-09-17.

Mathematical Association of America. American invitational mathematics examination (aime). Competition, 2024. URL `https://maa.org/math-competitions/american-invitational-mathematics-examination-aime`. Accessed 2026-09-17.

Bradley McDanel, Steven Li, Sruthikesh Surineni, and Harshit Khaitan. MoE-Spec: Expert budgeting for efficient speculative decoding. *arXiv preprint arXiv:2602.16052*, 2026.

Zhiyu Mei, Wei Fu, Kaiwei Li, Guangju Wang, Huanchen Zhang, and Yi Wu. Real: Efficient rlhf training of large language models with parameter reallocation. In *Proceedings of the Eighth Conference on Machine Learning and Systems (MLSys)*, 2025.

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, 2022.

Ruoyu Qin, Weiran He, Weixiao Huang, Yangkun Zhang, Yikai Zhao, Bo Pang, Xinran Xu, Yingdi Shan, Yongwei Wu, and Mingxing Zhang. Seer: Online context learning for fast synchronous LLM reinforcement learning. In *20th USENIX Symposium on Operating Systems Design and Implementation (OSDI 26)*, pp. 883–901, Seattle, WA, July 2026. USENIX Association. ISBN 978-1-939133-55-7. URL https://www.usenix.org/conference/osdi26/presentation/qin.

Anish Saxena, Po-An Tsai, Hritvik Taneja, Aamer Jaleel, and Moinuddin Qureshi. Utility-driven speculative decoding for mixture-of-experts. *arXiv preprint arXiv:2506.20675*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

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

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. Mathscale: Scaling instruction tuning for mathematical reasoning, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

veRL Contributors. verl: A reinforcement learning framework for large language models. GitHub repository, 2026. URL https://github.com/volcengine/verl. Accessed 2026-09-16.

Yuhang Xu, Kaibin Tian, Yang Tian, Zhice Yang, Yifeng Yu, Yan Li, Shengzhong Liu, Fan Wu, and Guihai Chen. Bubblespec: Turning long-tail bubbles into speculative rollout drafts for synchronous reinforcement learning, 2026. Manuscript.

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

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. In *Second Conference on Language Modeling*, 2025.

Yizhou Zhang, Ning Lv, Teng Wang, and Jisheng Dang. Fastgrpo: Accelerating policy optimization via concurrency-aware speculative decoding and online draft learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 59620–59636, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/61674667d642ae52f6bb281bea90ee29-Paper-Conference.pdf.

Tianle Zhong, Neiwen Ling, Yifan Pi, Zijun Wei, Tianshu Yu, Geoffrey Fox, Peng Wu, and Xiao Yu. Diagnosing training inference mismatch in llm reinforcement learning. *arXiv preprint arXiv:2605.14220*, 2026.

A Additional Implementation Details

A.1 Hardware and Software Environment

All experiments are conducted on NVIDIA A100-SXM4-80GB GPUs. Each compute node is equipped with 8 GPUs, with 80 GB of HBM memory per GPU, two Intel Xeon Platinum 8358P CPUs, and 1 TiB of host memory. For controlled wall-clock comparisons, GRPO, SPEC-RL, HiGAP, and the corresponding ablations use the same hardware allocation for each model. The main hardware and software configurations are summarized in Table 4.

Table 4: Hardware and software environment used in our experiments.

Component	Configuration
GPU	NVIDIA A100-SXM4-80GB
GPUs per node	8
GPU memory	80 GB per GPU
CPU	2× Intel Xeon Platinum 8358P
CPU cores	64 physical cores per node
Host memory	1 TiB per node
PyTorch	2.10.0+cu129
CUDA runtime	12.9
cuDNN	9.10.2
Transformers	5.12.1
veRL	0.8.0
vLLM	0.18.1
Ray	2.49.0

A.2 Training and Rollout Configurations

Table 5 summarizes the model-specific training, parallelism, and rollout configurations used in our experiments. For each model, GRPO, SPEC-RL, and HiGAP use the same underlying training and rollout configurations for controlled comparison. All experiments use a training batch size of 1024 and are trained for four epochs. The remaining configurations are adjusted according to model architecture and sequence length.

Table 5: Training, parallelism, and rollout configurations across evaluated models.

Model	Actor Parallelism		Training					Rollout		
	TP	EP	Batch	Mini-batch	Prompt	Response	Epochs	TP	n	Mem.
Qwen3.5-35B-A3B	2	8	1024	128	1k	2k	4	8	5	0.55
Qwen3-30B-A3B	4	2	1024	128	2k	4k	4	4	8	0.60
Moonlight-16B-A3B	4	4	1024	128	1k	4k	4	2	8	0.60
Qwen3-8B	2	–	1024	256	1k	4k	4	2	5	0.55

TP/EP: tensor/expert parallelism; Mem.: rollout GPU memory utilization. EP is not applicable to the dense Qwen3-8B actor.

A.3 Evaluation Metrics

System efficiency. We use end-to-end step time and training throughput as the primary efficiency metrics. The end-to-end step time measures the wall-clock duration of one complete RL training step, including rollout generation, actor verification when applicable, old-log-probability computation, policy optimization, and other framework overheads:

$$T_{\text{step}} = T_{\text{rollout}} + T_{\text{verify}} + T_{\text{old-logp}} + T_{\text{update}} + T_{\text{other}}. \tag{9}$$

Here, T_{rollout} is the autoregressive rollout-generation time, T_{verify} is the actor time spent verifying historical drafts, $T_{\text{old-logp}}$ is the time for computing old-policy log-probabilities, and T_{update} denotes policy-optimization time. T_{other} contains the remaining framework and data-processing overheads.

Training throughput measures the number of response tokens processed per second per GPU:

$$\text{Throughput} = \frac{N_{\text{response}}}{T_{\text{step}}\, N_{\text{GPU}}}, \tag{10}$$

where N_{response} is the total number of response tokens processed in a training step and N_{GPU} is the number of GPUs.

Training effectiveness. For downstream evaluation, we evaluate the resulting checkpoints on nine mathematical reasoning benchmarks. For each benchmark $\mathcal{D}$, accuracy is computed as

$$\mathrm{Acc}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathbb{I}[\hat{y}_i \equiv y_i], \tag{11}$$

where $\hat{y}_i$ is the extracted model answer, y_i is the reference answer, and $\equiv$ denotes equivalence under the answer checker. The same evaluation protocol is applied to all compared methods.

A.4 Reasoning Benchmarks

We evaluate the resulting checkpoints on the nine benchmarks summarized in Table 6. We use stochastic decoding with temperature 1.0, top-$p = 0.95$, and a maximum generation length of 4096 tokens. Each problem is evaluated with one sampled response ($n = 1$) using random seed 0. Generated answers are evaluated using the symbolic and numerical equivalence checker from the `simpleRL-reason` evaluation toolkit (Zeng et al., 2025). We report accuracy (%) for each benchmark and the unweighted mean accuracy across all nine benchmarks.

Table 6: Summary of the reasoning benchmarks used for evaluation.

Benchmark	**Focus**	**Description**
MATH-500 (Hendrycks et al., 2021b; Lightman et al., 2024)	Competition	500 MATH problems covering multi-step reasoning across diverse mathematical topics.
GSM8K (Cobbe et al., 2021)	Grade school	Mathematical word problems requiring multi-step arithmetic reasoning.
Minerva Math (Lewkowycz et al., 2022)	Math & Science	Quantitative problems emphasizing step-by-step mathematical and scientific reasoning.
OlympiadBench (He et al., 2024)	Olympiad	Challenging olympiad-level problems requiring advanced mathematical reasoning.
MMLU-STEM (Hendrycks et al., 2021a)	STEM	STEM subsets of MMLU covering science, technology, engineering, and mathematics.
AMC23 (Mathematical Association of America, 2023)	Competition	Problems from the 2023 American Mathematics Competitions.
MATH (Hendrycks et al., 2021b)	Competition	Competition mathematics spanning multiple subjects and difficulty levels.
College Math (Tang et al., 2024)	College	Reasoning problems involving college-level mathematical concepts.
AIME24 (Mathematical Association of America, 2024)	Competition	Challenging problems from the 2024 American Invitational Mathematics Examination.

A.5 Full Ablation Benchmark Results

Table 7 reports the complete downstream evaluation of the HiGAP ablations across nine mathematical reasoning benchmarks. The main paper reports only the average score to keep the ablation analysis focused on the trade-off between system efficiency and training effectiveness. As detailed, the full HiGAP model establishes a strong baseline, achieving the highest overall average score of 77.5. However, the breakdown in Table 7 reveals distinct trade-offs. Removing the *partial log-prob bypass* yields higher system throughput, but this efficiency comes at a severe cost to reasoning ability, with the average score plummeting to 66.2. This variant struggles particularly on MMLU-STEM (51.0) and College Math (36.9), indicating that the bypass mechanism is crucial for maintaining a stable training gradient. On the other hand, *w/o adaptive history pruning* also exhibits a performance drop to 68.9, though it maintains relatively strong results on MATH-500 (89.6). Interestingly, the

variants *w/o advantage-guided history update* and *w/o post-update probability caching* retain competitive average scores of 75.1 and 75.7, respectively. The latter even achieves the best individual scores on AMC23 (97.5) and College Math (48.0), suggesting that while these components are not strictly necessary for achieving high scores on specific benchmarks, they are essential for maximizing the aggregated performance. Overall, the comprehensive results confirm that the efficiency optimizations in HiGAP do not compromise its mathematical reasoning capabilities.

Table 7: Benchmark scores for ablation variants on Qwen3.5-35B-A3B. Minerva, Olympiad, and College denote Minerva Math, OlympiadBench, and College Math, respectively.

Method	MATH-500	GSM8K	Minerva	Olympiad	MMLU-STEM	AMC23	MATH	College	AIME24	Avg.
HiGAP	**94.6**	**95.2**	**57.0**	**62.5**	**80.8**	95.0	**94.3**	47.8	**70.0**	**77.5**
w/o adaptive history pruning	89.6	93.6	42.3	54.5	70.3	85.0	91.3	46.9	46.7	68.9
w/o partial log-prob bypass	90.0	92.2	32.4	59.9	51.0	95.0	88.4	36.9	50.0	66.2
w/o advantage-guided history update	94.0	95.0	44.9	62.2	71.3	**97.5**	94.8	**53.3**	63.3	75.1
w/o post-update probability caching	94.4	92.3	54.0	61.8	78.7	**97.5**	94.4	48.0	60.0	75.7