SPEC-RL: Accelerating On-Policy Reinforcement Learning with Speculative Rollouts

Anonymous ACL submission

Abstract

Large Language Models (LLMs) increasingly rely on reinforcement learning with verifiable rewards (RLVR) to elicit reliable chain-of-thought reasoning. However, the training process remains bottlenecked by the computationally expensive rollout stage. Existing acceleration methods—such as parallelization, objective- and data-driven modifications, and replay buffers—either incur diminishing returns, introduce bias, or overlook redundancy across iterations. We identify that rollouts from consecutive training epochs frequently share a large portion of overlapping segments, wasting computation. To address this, we propose **SPEC-RL**, a novel framework that integrates **SPEC**ulative decoding with the **RL** rollout process. SPEC-RL reuses prior trajectory segments as speculative prefixes and extends them via a draft-and-verify mechanism, avoiding redundant generation while ensuring policy consistency. Experiments on diverse math reasoning and generalization benchmarks, including AIME24, MATH-500, OlympiadBench, MMLU-STEM, and others, demonstrate that SPEC-RL reduces rollout time by 2–3× without compromising policy quality. As a purely rollout-stage enhancement, SPEC-RL integrates seamlessly with mainstream algorithms (e.g., PPO, GRPO, DAPO), offering a general and practical path to scale RLVR for large reasoning models.

1 Introduction

Large Language Models (LLMs) have recently achieved substantial progress on challenging reasoning-intensive tasks, such as mathematical problem-solving (Lewkowycz et al., 2022a), program synthesis (Chen et al., 2021; Li et al., 2022), and multi-step agentic planning (Yao et al., 2023b,a). A key enabler of these advances is reinforcement learning with verifiable rewards (RLVR) (Lambert et al., 2024; Guo et al., 2025; Yue et al., 2025), which has emerged as a widely adopted paradigm for incentivizing models to produce faithful and reliable chain-of-thought (CoT) reasoning (Wei et al., 2022). However, RLVR training pipelines remain constrained by the rollout stage, a fundamental efficiency bottleneck, despite its demonstrated efficacy (Zheng et al., 2025). During this stage, the model must generate large quantities of trajectories through interaction with the environment, a process that is computationally expensive and scales poorly with model size. As a result, the cost and latency of trajectory generation dominate overall training time, severely limiting the practicality of scaling RLVR to increasingly capable LLMs.

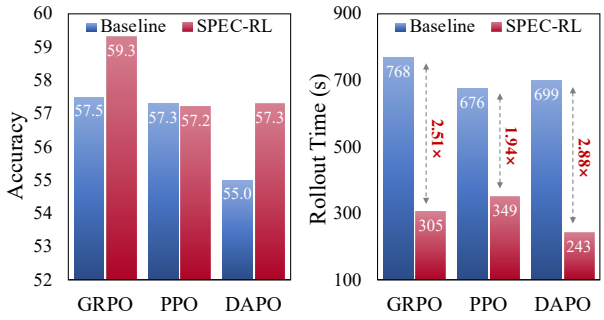

Figure 1: SPEC-RL achieves a 2–3× speedup in per-step rollout time without compromising average performance on Qwen3-8B-Base across various algorithms.

To mitigate rollout inefficiency, prior work has explored three directions. First, parallelized rollout generation increases throughput by producing many trajectories per iteration (Xu et al., 2025), but its benefits fade as computational and synchronization costs rise. Second, model-based accelerations reduce environment interaction through modified objectives (Brantley et al., 2025; Lin et al., 2025), data restructuring (Liu et al., 2025; Zhang et al., 2025b), or sample selection heuristics (Yu et al., 2025; Zheng et al., 2025), though these approaches often introduce bias and added complexity. Third, caching methods such as replay buffers reuse prior

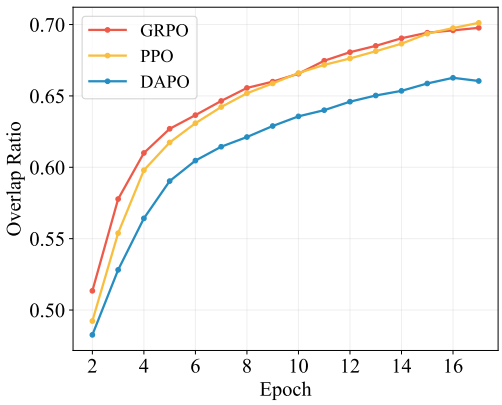

Figure 2: Token overlap ratios across different RL algorithms, computed with ROUGE-1 (Lin, 2004) by comparing rollout responses from consecutive training epochs.

trajectories (Zhang et al., 2025a), thereby improving data utilization, but still require fresh on-policy rollouts and struggle when policies shift significantly.

Existing research (Yang et al., 2025; Wang et al., 2025) has found that the quality of the RL training set is more critical than increasing its quantity. Consequently, these studies often adopt a rigorous approach to curating a small but high-quality dataset. This practice typically leads to training for tens or even hundreds of epochs. The extensive exploration over this same set of instances inevitably results in high redundancy between rollouts from consecutive training epochs, as illustrated in Figure 2. Such redundancy naturally arises due to incremental policy updates, with the current policy often behaving similarly to the previous one. Moreover, in environments with fixed initial states or tasks (e.g., repeated prompts in an LLM reasoning task), the early parts of trajectories tend to overlap across iterations. As a result, significant computation is wasted regenerating these overlapping segments. This motivates the central question of our work: ***can such redundancy be systematically exploited to accelerate rollouts?***

We answer this question by proposing **SPEC-RL**, a novel framework that integrates **SPEC**ulative decoding with the **RL** rollout process. Rather than regenerating full trajectories from scratch, SPEC-RL treats old rollouts from the previous epoch as implicit drafts: following the speculative decoding paradigm, old rollout tokens are verified under the current policy to form a verified prefix. When the first rejection position is reached, the current policy continues generation from that point onward, as illustrated in Figure 3. This approach is directly analogous to draft-and-verify methods in text generation, where a draft sequence is proposed and then validated in parallel by the target model (Leviathan et al., 2023). By incorporating the same mechanism into RL rollouts, SPEC-RL leverages cached rollouts to skip redundant computation while ensuring that the final outputs remain faithful to the current policy. The verified prefix is quickly extended by the latest policy, ensuring that the final trajectory remains consistent with the current policy's behavior.

Our experiments demonstrate that SPEC-RL substantially improves training efficiency across diverse tasks and model scales. Concretely, SPEC-RL consistently reduces rollout generation time by 2–3× on average, while maintaining or even improving final policy performance across a wide range of math reasoning benchmarks (AIME24 (Art of Problem Solving, 2024a), MATH-500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022b), OlympiadBench (He et al., 2024), AMC 2023 (Art of Problem Solving, 2024b)) and out-of-distribution benchmarks (MMLU-STEM (Hendrycks et al., 2020), IFEval (Zhou et al., 2023)). Importantly, SPEC-RL is designed as a modular enhancement to the data collection phase, making it readily applicable to a wide range of mainstream RLVR algorithms, such as GRPO, DAPO, and PPO.

In summary, we identify substantial rollout redundancy in RLVR and show that reusing overlapping trajectory segments can greatly reduce sampling cost. Building on this insight, we introduce SPEC-RL, the first framework to incorporate speculative decoding into RL rollouts by treating previous-epoch trajectories as implicit drafts and verifying reusable prefixes. This design integrates smoothly with mainstream RL algorithms and yields significant rollout acceleration while maintaining policy performance.

2 Preliminaries

2.1 On-Policy Reinforcement Learning

Reinforcement learning with verifiable rewards (RLVR) is a training paradigm that couples RL with objective, externally verifiable signals. It typically employs on-policy methods like Proximal Policy Optimization (PPO), where the agent learns exclusively from rollouts generated by its current,

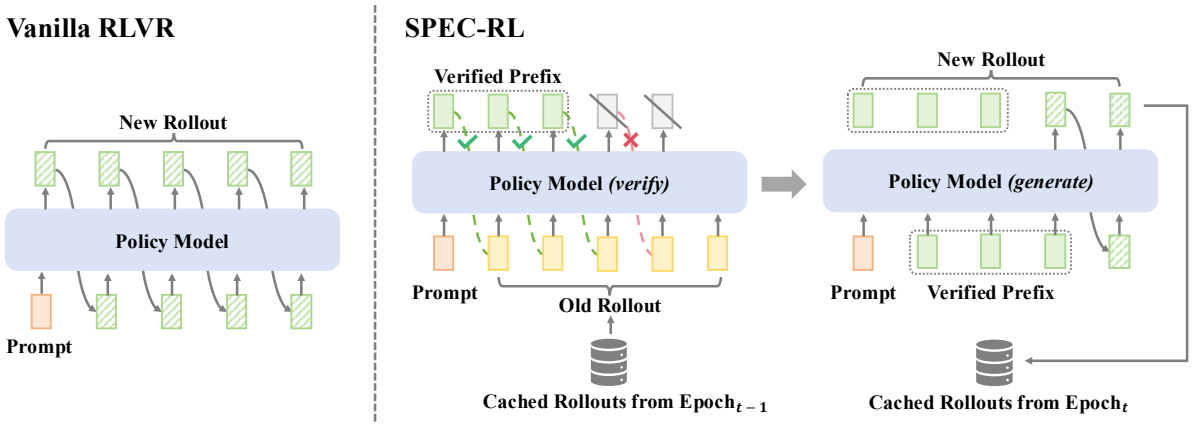

Figure 3: A comparison of the rollout processes in vanilla RLVR and our method. At each training step, vanilla RLVR regenerates full responses. In contrast, SPEC-RL, retrieves cached rollouts from the previous epoch. It then verifies these rollouts in parallel, keeps the verified prefixes, and resumes generation with the current policy to produce the final response.

cautious policy. Formally, for an instance $(\mathbf{x}, \mathbf{y}^*)$ from the training set $\mathcal{D}$—where $\mathbf{x}$ is a prompt and $\mathbf{y}^*$ is the ground-truth answer—a rollout is sampled from the policy as $\mathbf{y} \sim \pi(\cdot \mid \mathbf{x})$. A reward function $R(\mathbf{y}, \mathbf{y}^*)$ then evaluates whether $\mathbf{y}$ matches $\mathbf{y}^*$. This feedback is typically binary and tamper-resistant to ensure precision, as in the case of numerical comparisons or unit test results. The on-policy RL objective is to maximize the expected reward of these generated rollouts:

$$J(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y}^*)\sim\mathcal{D},\mathbf{y}\sim\pi(\cdot|\mathbf{x})}\big[R(\mathbf{y}, \mathbf{y}^*)\big]. \quad (1)$$

This approach ensures the training data remain consistent with the current policy distribution, carefully balancing exploration with stable, incremental updates to guarantee reliable performance. In this work, we retain the standard RL objective and policy update methods, focusing instead on improving the efficiency of the rollout stage.

2.2 Speculative Decoding

Speculative decoding follows a draft-and-verify paradigm: an efficient draft model p proposes multiple future tokens, which are then verified in parallel by the target model q. A drafted token $z_i \sim p(\cdot \mid \mathbf{x}, \mathbf{z}_{<i})$ is accepted with probability

$$\alpha_i = \min\Big(1, \frac{q(z_i \mid \mathbf{x}, \mathbf{z}_{<i})}{p(z_i \mid \mathbf{x}, \mathbf{z}_{<i})}\Big). \quad (2)$$

This ensures the process samples exactly from q, preserving fidelity while accelerating generation by reducing costly target computations. The speedup is governed by the acceptance rate and the cost gap between the draft and target models.

3 Method

The goal of SPEC-RL is to accelerate RL rollouts by avoiding redundant regeneration. Instead of sampling complete trajectories from scratch at every step, we leverage cached rollouts from the previous epoch and reuse as much of them as possible, only generating the minimal continuation that is inconsistent with the current policy (Figure 3). This reduces the number of decoded tokens and directly cuts rollout latency.

3.1 Speculative Decoding over Cached Rollouts with Lenience

The core of SPEC-RL is adapting speculative decoding to RL by treating previous rollouts as draft sequences. Formally, when sampling a rollout for a prompt $\mathbf{x}$ during epoch t, let $\mathbf{y}^{\text{prev}}$ denote a rollout generated for the same prompt in the previous epoch $t-1$ under the policy π_{prev}. Following the standard draft–and–verify formulation in Equation 2, we verify each token y_i^{prev} under the current policy π_{curr} and decide whether it can be reused. The acceptance probability for token i is $\min\Big(1, \pi_{\text{curr}}(y_i^{\text{prev}}|\mathbf{x},\mathbf{y}_{<i}^{\text{prev}}) / \pi_{\text{prev}}(y_i^{\text{prev}}|\mathbf{x},\mathbf{y}_{<i}^{\text{prev}})\Big)$. This ensures exact consistency with the current policy; however, it can be overly strict in practice, limiting the amount of reusable tokens.

To address this, we introduce a lenience parameter ℓ, following prior work (Chen et al., 2024). This relaxes the acceptance condition by shifting the decision boundary. Formally, the acceptance

Algorithm 1: Speculative Rollout

Input: Current policy π_θ; Prompt $\mathbf{x}$; Previous rollout $\mathbf{y}^{\text{prev}}$ with token probabilities $\mathbf{p}^{\text{prev}}$; Lenience parameter $\ell \geq 1$.

Compute current token probabilities:
$\mathbf{p}^{\text{curr}} \leftarrow \left[\pi_\theta(y_i^{\text{prev}} \mid \mathbf{x}, \mathbf{y}_{<i}^{\text{prev}})\right]_{i=1}^{|\mathbf{y}^{\text{prev}}|}$;

Compute per-token acceptance probability:
$\widetilde{\boldsymbol{\alpha}} \leftarrow \min\left(\mathbf{1},\ \ell \cdot \frac{\mathbf{p}^{\text{curr}}}{\mathbf{p}^{\text{prev}}}\right)$;

Initialize rejection position $n \leftarrow |\mathbf{y}^{\text{prev}}| + 1$;

for $i = 1$ **to** $|\mathbf{y}^{prev}|$ **do**
  Sample $u \sim \mathcal{U}(0, 1)$;
  if $u > \widetilde{\alpha}_i$ **then**
    $n \leftarrow i$;
    break;

Generate continuation from position n:
$\mathbf{y}_{\geq n}^{\text{curr}} \sim \pi_\theta(\cdot \mid \mathbf{x}, \mathbf{y}_{<n}^{\text{prev}})$;

Assemble final response:
$\mathbf{y}^{\text{curr}} \leftarrow \mathbf{y}_{<n}^{\text{prev}} \oplus \mathbf{y}_{\geq n}^{\text{curr}}$;

return $\mathbf{y}^{\text{curr}}$

probability becomes

$$\tilde{\alpha}_i = \min\Big(1,\ \ell \cdot \frac{\pi_{\text{curr}}(y_i^{prev} \mid \mathbf{x}, \mathbf{y}_{<i}^{prev})}{\pi_{\text{prev}}(y_i^{prev} \mid \mathbf{x}, \mathbf{y}_{<i}^{prev})}\Big). \quad (3)$$

A token is accepted if a random sample $u \sim \mathcal{U}(0, 1)$ satisfies $u \leq \tilde{\alpha}_i$, and rejected otherwise. The verified prefix $\mathbf{y}_{<n}^{\text{prev}}$ is determined by identifying the first rejected token at position n. The final rollout, $\mathbf{y}^{\text{curr}} = \mathbf{y}_{<n}^{\text{prev}} \oplus \mathbf{y}_{\geq n}^{\text{curr}}$, is formed by concatenating this verified prefix with a newly generated continuation $\mathbf{y}_{\geq n}^{\text{curr}} \sim \pi_{\text{curr}}(\cdot \mid \mathbf{x}, \mathbf{y}_{<n}^{\text{prev}})$. The overall process of verification, generation, and assembly is summarized in Algorithm 1.

Difference from speculative decoding SPEC-RL follows the draft–and–verify paradigm of speculative decoding, but in a simplified, single-round form. Vanilla speculative decoding typically requires a separate draft model, loading extra parameters, scheduling overhead, and multiple verification rounds. In contrast, SPEC-RL reuses the previous policy as the draft, with cached rollouts available "for free". This eliminates the need for auxiliary models while preserving the fidelity guarantees of speculative decoding.

The mechanism of lenience The parameter ℓ provides a flexible mechanism to balance rollout efficiency against on-policy fidelity: setting $\ell = 1$ recovers the original speculative decoding rule; $\ell > 1$ increases the acceptance rate, leading to longer reused prefixes; $\ell \to \infty$ corresponds to full prefix reuse; and $\ell \to 0$ reverts to standard RLVR without reuse. Although this introduces a parameter that requires manual tuning, we demonstrate empirically that a moderate value of ℓ yields significant speedup without degrading performance.

3.2 Implementing SPEC-RL in RL Training

To enable practical use in RL training pipelines, SPEC-RL introduces a lightweight cache module that stores rollouts from the previous epoch and continuously refreshes them during training. When the same prompt reappears, its cached response is retrieved to serve as a draft sequence for reuse. To ensure efficacy, all draft verification requests within a training batch are packed into a single call to the rollout engine. Verified prefixes and prompts are aligned via left padding, enabling parallel processing of further continuations without fragmentation. Once new rollouts are generated, the cache is updated for the corresponding prompts for later use. This ensures retrieved rollouts are always generated by the most recent policy, naturally increasing token reuse probability.

This design allows SPEC-RL to operate as a drop-in module: it modifies only the rollout stage, requires no changes to reward computation or policy updates, and is compatible with mainstream algorithms such as GRPO and PPO.

4 Experiments

4.1 Experimental Setup

Models and hyperparameters We evaluate SPEC-RL on Qwen3-1.7B-Base, Qwen3-8B-Base, and LLaMA-3.2-1B-Instruct across various RL algorithms, including GRPO, PPO, and DAPO. All models are trained using the veRL (Sheng et al., 2025) framework with vLLM (Kwon et al., 2023) as the rollout engine, on data sampled from DeepMath (6,144 examples, denoted as DeepMath-6K) (He et al., 2025b) and SimpleRL (8,192 examples, denoted as SimpleRL-8K) (Zeng et al., 2025). All experiments use a rollout batch size of 1,024 and a maximum response length of 4,096 tokens, conducted on a single node with 8× NVIDIA H100 GPUs. Rollout is performed at a temperature of 1.0. The actor learning rate is fixed at 5×10^{-7}, and for PPO we set the critic learning rate to 1×10^{-5}. Fur-

Algorithm	Rollout Efficiency		Math Reasoning					OOD		AVG
	Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Qwen3-1.7B-Base										
Base Model	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	**60.7**	24.4	34.9
↪ + SPEC-RL	**182.7**	**2.29×**	**38.3**	**9.1**	**68.0**	**29.4**	**29.3**	58.3	**28.8**	**37.3**
PPO	565.1	1.00×	35.3	8.3	63.0	**26.8**	25.3	**59.4**	25.5	34.8
↪ + SPEC-RL	**230.8**	**1.94×**	**36.9**	7.7	**64.8**	25.4	**25.9**	58.6	**25.9**	**35.0**
DAPO	543.1	1.00×	33.1	4.9	**60.8**	24.6	23.0	52.2	24.8	31.9
↪ + SPEC-RL	**171.6**	**2.17×**	33.1	4.9	60.0	**25.7**	**25.5**	**53.5**	**27.0**	**32.8**
Qwen3-8B-Base										
Base Model	-	-	40.2	11.5	67.4	27.2	34.1	60.4	29.9	38.7
GRPO	1033.1	1.00×	69.2	24.2	86.4	43.8	**53.0**	**84.6**	41.2	57.5
↪ + SPEC-RL	**336.6**	**2.51×**	**72.8**	**26.9**	**87.8**	**44.1**	51.0	84.5	**47.7**	**59.3**
PPO	984.0	1.00×	68.9	**26.4**	**85.8**	43.0	**51.6**	83.8	**41.6**	**57.3**
↪ + SPEC-RL	**400.1**	**1.94×**	**70.5**	25.1	85.2	**43.4**	50.8	**84.4**	41.0	57.2
DAPO	1052.2	1.00×	65.2	24.0	**84.8**	40.1	48.6	**82.4**	39.6	55.0
↪ + SPEC-RL	**326.2**	**2.88×**	**69.7**	**26.4**	84.4	**43.8**	**50.4**	82.2	**44.4**	**57.3**
LLaMA-3.2-1B-Instruct										
Base Model	-	-	7.5	0.6	14.2	4.0	2.8	32.6	37.0	14.1
GRPO	553.9	1.00×	7.5	1.4	19.2	**3.3**	4.9	33.1	37.0	15.2
↪ + SPEC-RL	**162.5**	**2.60×**	**8.8**	**1.7**	**19.4**	1.8	**5.0**	**34.5**	**37.2**	**15.5**
PPO	521.5	1.00×	7.8	0.8	**20.8**	4.0	**6.4**	34.3	**42.7**	16.7
↪ + SPEC-RL	**210.6**	**2.01×**	**9.2**	**1.4**	20.2	**5.5**	5.0	**35.3**	40.7	**16.8**
DAPO	482.6	1.00×	9.2	0.9	19.2	4.0	5.5	33.0	**38.6**	15.8
↪ + SPEC-RL	**123.1**	**2.48×**	**11.6**	**2.0**	**20.2**	4.0	5.5	**35.5**	38.4	**16.7**

Table 1: Main results across various models and RL algorithms on DeepMath-6K.

ther hyperparameters are detailed in Appendix A.1.

Benchmarks and metrics We evaluate rollout efficiency and accuracy on a broad suite of benchmarks. **Rollout efficiency** is reported as the number of generated tokens and the relative speedup (baseline time divided by method time). **Math reasoning benchmarks** include AMC 2023 (Art of Problem Solving, 2024b), AIME24 (Art of Problem Solving, 2024a), MATH-500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022b), and OlympiadBench (He et al., 2024). **Out-of-distribution (OOD) benchmarks** include MMLU-STEM (Hendrycks et al., 2020) and IFEval (Zhou et al., 2023), which evaluate the generalization capability of the model. Full evaluation details are provided in Appendix A.2.

4.2 Main Results

As shown in Table 1, across nine model–algorithm settings, SPEC-RL yields an average speedup of 2.31× by reducing generated tokens by 66%. The largest gain is with Qwen-3-8B-Base under DAPO (1,052.2M → 326.2M tokens; 2.88×), while even the smallest case (Qwen-3-8B-Base with PPO) achieves 1.94× without accuracy loss. These improvements closely track the reduction in generated tokens, confirming that token-level savings drive the acceleration.

On math benchmarks, accuracy remains broadly stable: larger models are highly robust, while smaller models show only small fluctuations. For OOD tasks, MMLU-STEM stays nearly unchanged, and IFEval improves in several cases—for example, +6.5 points on Qwen-3-8B-Base with GRPO. Overall, SPEC-RL accelerates rollout generation substantially without degrading reasoning quality, and sometimes even improves out-of-distribution generalization.

Additional analyses, including wall-clock breakdowns, performance on larger model scale, and the effect of training-set size, comparisons on DeepMath-6K and SimpleRL-8K, per-step training curves, are provided in Appendices B.1, B.2, B.3, B.6, C.1.

4.3 Ablation Studies and Analyses

The draft–and–verify strategy and the lenience parameter ℓ play critical roles in SPEC-RL. In this section, we conduct ablation experiments on different rollout reuse methods and various values of ℓ. We consistently use Qwen3-1.7B-Base with GRPO, trained on the DeepMath-6K dataset. Our analysis

Algorithm	Rollout Efficiency		Math Reasoning					OOD		AVG
	Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
SPEC-RL	**182.7**	2.29×	**38.3**	**9.1**	**68.0**	**29.4**	**29.3**	**58.3**	**28.8**	**37.3**
⇒ Random Reuse	304.5	**2.35**×	30.3	5.0	60.4	21.7	25.3	53.1	24.0	31.4
⇒ Delayed Reuse	308.8	1.44×	35.6	9.0	66.4	29.0	28.6	57.3	28.3	36.3

Table 2: Comparison between SPEC-RL and its variants when training Qwen3-1.7B-Base on DeepMath-6K with GRPO.

Algorithm	Rollout Efficiency		Math Reasoning					OOD		AVG
	Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
GRPO	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + SPEC-RL $\ell = 1$	419.1	1.22×	37.8	8.0	63.8	28.7	26.5	59.6	25.9	35.8
↪ + SPEC-RL $\ell = e^{0.2}$	246.7	1.86×	35.8	7.8	66.4	**29.8**	**29.6**	58.5	25.9	36.3
↪ + SPEC-RL $\ell = e^{0.5}$	182.7	2.29×	**38.3**	**9.1**	**68.0**	29.4	29.3	58.3	28.8	**37.3**
↪ + SPEC-RL $\ell = e^{0.8}$	144.8	2.64×	28.6	5.0	63.6	27.2	25.0	**61.7**	26.2	33.9
↪ + SPEC-RL $\ell = e^{2.0}$	114.4	3.05×	26.2	5.2	55.0	21.0	21.9	53.5	**29.0**	30.3
↪ + SPEC-RL $\ell = \infty$	**40.0**	**14.86**×	30.2	4.4	60.4	19.9	23.7	44.1	22.0	29.2

Table 3: Ablation on lenience parameter ℓ on DeepMath-6K. Here $\ell = 1$ corresponds to vanilla speculative decoding, while $\ell = \infty$ corresponds to full reuse.

proceeds as follows.

Comparison with different reuse variants To better understand our method, we design two variants:

- **Random Reuse:** Instead of using Equation 3 to compute the acceptance criterion, rejection positions are sampled uniformly at random. This results in roughly half of the tokens being reused on expectation.
- **Delayed Reuse:** Instead of retrieving rollouts from the most recent epoch as draft sequences, this variant uses rollouts from two epochs ago to validate the necessity of using the most recent rollouts.

As shown in Table 2, Random Reuse enjoys higher efficiency against SPEC-RL (2.35× vs. 2.29×) without verification cost, but it causes a substantial drop in accuracy (31.4 vs. 37.3). This highlights that naive reuse introduces harmful noise, whereas SPEC-RL leverages speculative verification to retain policy fidelity while accelerating training. Delayed Reuse shows competitive performance compared to the GRPO baseline (36.3 vs. 37.3); however, its efficiency is significantly reduced (1.44× vs. 2.29×). This occurs because the larger policy gap leads to fewer reused tokens, demonstrating the effectiveness of our immediate cache-updating strategy. The detailed results are provided in Appendix B.4.

Impact of lenience ℓ on efficacy As shown in Table 3 and Figure 4, increasing ℓ consistently improves rollout efficiency: starting from vanilla speculative decoding at $\ell = 1$ with a speedup of only 1.22×, the acceleration rises steadily and reaches 14.86× when $\ell \to \infty$. This improvement stems from the greater prefix length reused when ℓ increases. Interestingly, as training progresses, the reused prefix length exhibits a trend of first decreasing and then rising. This occurs because policy parameters are updated more aggressively in the early stages of training, widening the policy gap and reducing reuse. As training continues, however, the policy stabilizes, updates become less frequent, and reuse increases accordingly. Accuracy, however, does not follow the same trend—performance peaks at $\ell = e^{0.5}$ with 37.3, but declines when reuse becomes overly aggressive, dropping to 29.2 at $\ell \to \infty$. Overall, moderate lenience values strike the best balance, yielding 2–3× rollout speedups while preserving or slightly improving accuracy, whereas extreme reuse sacrifices performance despite dramatic acceleration. The detailed intermediate results throughout training are provided in Appendix B.5.

Impact of Lenience ℓ on diagnostic metrics Due to the complexity of modern RL algorithms, several metrics, such as policy entropy, KL divergence, and the clip fraction of rollouts, are commonly used to diagnose training health. We in-

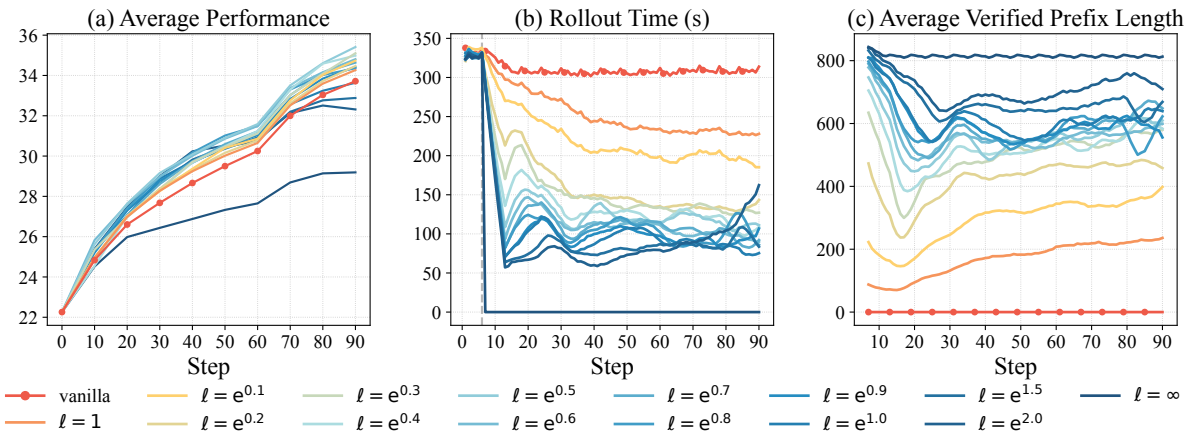

Figure 4: The effect of lenience ℓ on model performance and efficiency: (a) averaged test performance, (b) rollout time, and (c) averaged verified prefix length at different training steps.

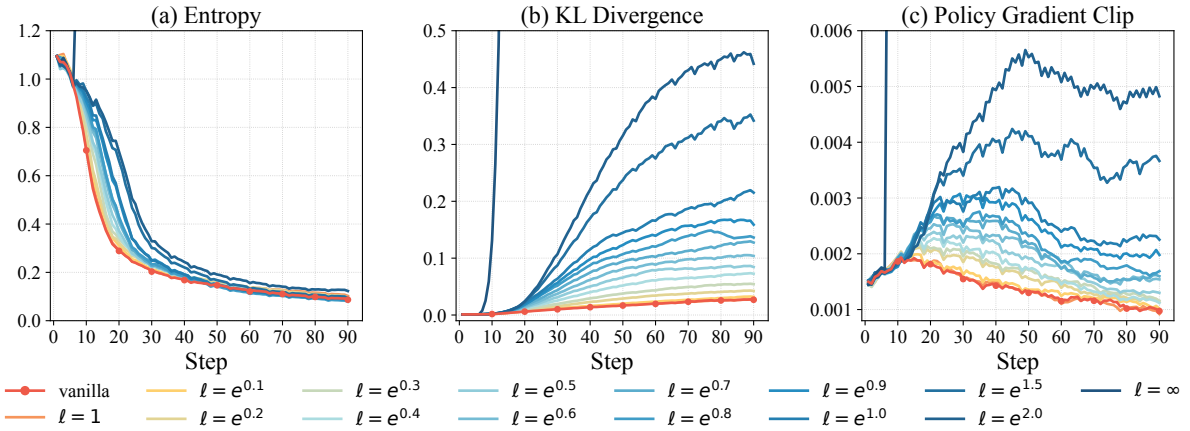

Figure 5: Training dynamics of SPEC-RL under different ℓ for three metrics: (a) entropy, (b) KL divergence, and (c) policy gradient clip ratio.

vestigate how the lenience parameter ℓ influences these metrics; the corresponding results are plotted in Figure 5. For $\ell = 1$, the metrics closely match those of the GRPO baseline. This is expected because rollouts are fully on-policy in this setting. As ℓ increases, all three metrics rise consistently, reflecting greater deviation of the rollout data from the current policy. In the extreme case where $\ell \to \infty$, rollouts are reused completely, causing values to increase sharply and exceed the plotted range, clearly indicating the training instability. Nevertheless, with a moderate choice of ℓ, the metrics remain within a stable region. For example, when $\ell < e^{0.5}$, the clip fraction stays below 0.0025 and KL divergence remains under 0.1. Additional case studies in Appendix C.2 demonstrate how SPEC-RL reuses validated prefixes while preserving the integrity of the reasoning chain. This illustrates why moderate reuse strikes an optimal balance, thereby allowing SPEC-RL to maintain strong performance without sacrificing training stability.

Analysis of rollout diversity We further examine whether speculative reuse affects rollout diversity by comparing SPEC-RL with the GRPO baseline at identical training steps. Diversity is measured using two standard metrics: **Distinct-1** (Li et al., 2016), which captures unigram variability, and **Self-BLEU** (Zhu et al., 2018), which measures similarity among samples within a batch. As shown in Figure 6(a)–(b), SPEC-RL consistently achieves *equal or higher* diversity than the GRPO baseline throughout training. Although SPEC-RL reuses prefixes generated by a previous policy, this reuse affects the similarity of the prefix region to the old policy's outputs. However, it does *not* constrain the diversity of the rollouts in a training batch at each step when compared to standard GRPO. This

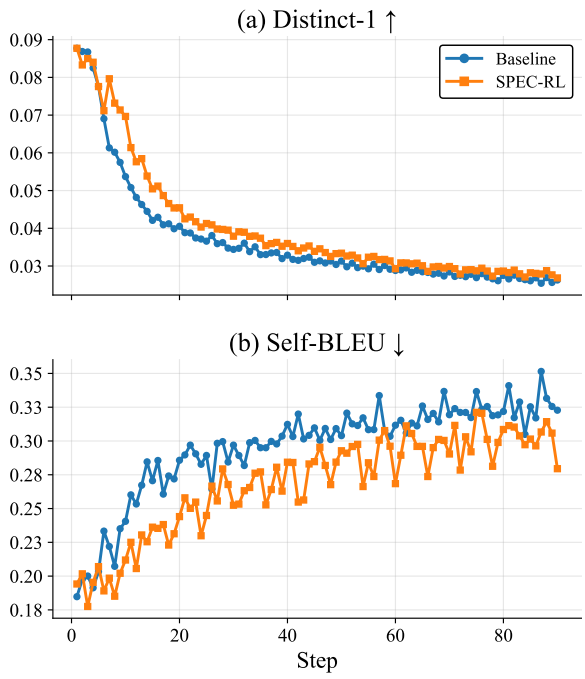

Figure 6: Effects of SPEC-RL on rollout diversity measured by Distinct-1 and Self-BLEU.

preservation of diversity can be attributed to the fact that, as RL training progresses, generated rollouts typically become less diverse. In contrast, the verified prefixes reused by SPEC-RL are drawn from earlier epochs and often retain greater diversity. These diverse prefixes naturally lead to varied subsequent rollouts. In practice, the trajectory variation at each training step, which is essential for exploration and effective learning, remains fully intact. The observed trends confirm that SPEC-RL maintains, and in some cases even enhances, final task performance.

5 Related Work

Efficiency in RLVR. Recent studies explore RLVR efficiency from multiple perspectives. Some parallelize rollout generation to increase throughput (Xu et al., 2025), though the gains are often limited by computation and synchronization overhead. Others improve efficiency by modifying optimization objectives or introducing additional constraints (Brantley et al., 2025; Lin et al., 2025), or by reorganizing training data to increase the proportion of informative samples (Liu et al., 2025; Zhang et al., 2025b). Heuristic and system-level designs further reduce wasted rollouts and stabilize training (Yu et al., 2025; Zheng et al., 2025). More recently, RhymeRL (He et al., 2025a) shows that suffix-tree based speculative decoding can reduce rollout latency by reusing draft tokens from historical responses. In contrast, our method adopts a lightweight and plug-and-play speculative reuse mechanism tailored for RLVR. Instead of maintaining complex suffix-tree structures, we directly reuse verified prefixes through speculative decoding with a lenience mechanism. This design significantly lowers system complexity and integration cost, while enabling more effective reuse and yielding stronger acceleration in practice, without modifying RLVR objectives or update rules.

Speculative decoding. Speculative decoding has become a widely studied technique for accelerating the generation process. (Leviathan et al., 2023) first proposed using a lightweight draft model to propose tokens, which are then verified by a stronger target model. Subsequent works extended this principle in several directions. Medusa (Cai et al., 2024) attaches multiple proposal heads to increase parallelism, while Cascade Speculative Drafting (Chen et al., 2024) chains multiple draft–target stages for additional speedups. Other variants study accepting multiple tokens at once (Qin et al., 2024) or verifying blocks of tokens instead of single steps (Sun et al., 2024). We adapt speculative decoding to RLVR training by reusing outputs from the prior policy as drafts and verifying them under the current policy, enabling prefix reuse during rollout without additional models and remaining compatible with high-throughput rollout engines.

6 Conclusion

We address the rollout bottleneck in reinforcement learning with verifiable rewards (RLVR) by introducing SPEC-RL, which integrates speculative decoding into rollout generation. Instead of regenerating trajectories from scratch, SPEC-RL treats previous-epoch rollouts as implicit drafts: tokens are verified under the current policy to form a verified prefix, then generation resumes from the first rejection position. A lenience parameter ℓ modulates the acceptance rule, trading off reuse and exploration. Experiments on Qwen and LLaMA models with GRPO, PPO, and DAPO show consistent 2–3$\times$ rollout speedups with largely preserved, and sometimes improved, reasoning and OOD performance. This demonstrates that rollout redundancy can be systematically exploited without modifying objectives, rewards, or update rules.

Limitations

The proposed approach relies on speculative reuse of cached responses generated by earlier policies, and its acceleration benefit therefore depends on the quality and diversity of the cache. When the cached pool becomes stale or overly homogeneous, speculative reuse may provide limited exploratory signal and reduce its effectiveness. While fixed lenience reuse settings are sufficient in our experiments, more principled adaptive lenience scheduling strategies remain to be explored. In addition, SPEC-RL does not provide acceleration in the very early stage of training, since speculative reuse requires a non-empty cache, leading to a cold-start phase in the first epoch. Finally, SPEC-RL focuses on single-turn reasoning tasks, and extending speculative reuse to more complex settings, such as multi-turn interactions, multimodal large language models, and agentic RL scenarios, remains an important direction for future work.

References

Art of Problem Solving. 2024a. Aime problems and solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions. Accessed: 2025-04-20.

Art of Problem Solving. 2024b. Amc problems and solutions. https://artofproblemsolving.com/wiki/index.php?title=AMC_Problems_and_Solutions. Accessed: 2025-04-20.

Kianté Brantley, Mingyu Chen, Zhaolin Gao, Jason D. Lee, Wen Sun, Wenhao Zhan, and Xuezhou Zhang. 2025. Accelerating rl for llm reasoning with optimal advantage regression. *Preprint*, arXiv:2505.20686.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *Preprint*, arXiv:2401.10774.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chang, and Jie Huang. 2024. Cascade speculative drafting for even faster llm inference. *Advances in Neural Information Processing Systems*, 37:86226–86242.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. 2023. Lighteval: A lightweight framework for llm evaluation.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Jingkai He, Tianjian Li, Erhu Feng, Dong Du, Qian Liu, Tao Liu, Yubin Xia, and Haibo Chen. 2025a. History rhymes: Accelerating llm reinforcement learning with rhymerl. *arXiv preprint arXiv:2508.18588*.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025b. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, and 1 others. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022a. Solving quantitative reasoning problems with language models.

Advances in neural information processing systems, 35:3843–3857.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022b. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 110–119.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, and 1 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Zhihang Lin, Mingbao Lin, Yuan Xie, and Rongrong Ji. 2025. Cppo: Accelerating the training of group relative policy optimization-based reasoning models. *Preprint*, arXiv:2503.22342.

Zikang Liu, Tongtian Yue, Yepeng Tang, Longteng Guo, Junxian Cai, Qingbin Liu, Xi Chen, and Jing Liu. 2025. Prefix grouper: Efficient grpo training through shared-prefix forward. *Preprint*, arXiv:2506.05433.

Zongyue Qin, Ziniu Hu, Zifan He, Neha Prakriya, Jason Cong, and Yizhou Sun. 2024. Optimized multi-token joint decoding with auxiliary model for llm inference. *arXiv preprint arXiv:2407.09722*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. 2024. Block verification accelerates speculative decoding. *arXiv preprint arXiv:2403.10444*.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. 2025. Reinforcement learning for reasoning in large language models with one training example. *Preprint*, arXiv:2504.20571.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yixuan Even Xu, Yash Savani, Fei Fang, and Zico Kolter. 2025. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

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

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

Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. 2025a. Rlep: Reinforcement learning with experience replay for llm reasoning. *arXiv preprint arXiv:2507.07451*.

Yiqi Zhang, Huiqiang Jiang, Xufang Luo, Zhihe Yang, Chengruidong Zhang, Yifei Shen, Dongsheng Li,

Yuqing Yang, Lili Qiu, and Yang You. 2025b. SortedRL: Accelerating RL training for LLMs through online length-aware scheduling. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. 2025. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. *Preprint*, arXiv:2506.02177.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 1097–1100.

A Experimental Details

This section provides additional details on experimental settings, shared training configurations, hyperparameters, and reward design. We describe the common rollout and optimization setups across models and algorithms, as well as the evaluation protocols used for all benchmarks.

A.1 Hyperparameters

We report the shared training settings (model families, rollout engine, batch size, sequence lengths, training steps, and optimizer details), as well as the algorithm-specific configurations. All experiments use Qwen-3-1.7B-Base, Qwen-3-8B-Base, Qwen-3-14B-Base, and LLaMA-3.2-1B as backbone models. Rollouts are generated using vLLM (rollout $N = 8$) with a global batch size of 1024. The maximum prompt length is 1,024 tokens, and the maximum response length is 4,096 tokens. For optimization, the actor is trained using AdamW (learning rate 5×10^{-7}, weight decay 0.01, and gradient clipping of 1.0). For PPO, the critic is additionally optimized with AdamW (learning rate 1×10^{-5}, weight decay 0.01, clipping 1.0). Algorithm-specific differences are as follows. GRPO enables KL regularization with a coefficient of 0.0001, whereas PPO and DAPO disable KL regularization. DAPO further adopts a wider clipping range (high = 0.28, $c = 10$) compared to GRPO and PPO (high = 0.2, $c = 3$). Additionally, DAPO utilizes dynamic sampling. To ensure fair comparison with GRPO and PPO, we control for the total amount of rollout data: each training step in DAPO corresponds to multiple generation steps, and the evaluation interval is reduced from every 10 steps to every 5 steps. SPEC-RL uses default lenience values of $e^{0.5}$ for GRPO, $e^{0.3}$ for PPO, and $e^{0.15}$ for DAPO, chosen via grid search to balance rollout efficiency and stability. All methods employ the `math-verify` reward, which assigns $+1$ if the final boxed or numeric answer matches the ground truth and 0 otherwise. This simple, deterministic design ensures that the reward is aligned with evaluation metrics across benchmarks.

We use a rule-based reward function that depends solely on the correctness of the final answer. Specifically, we utilize the `math-verify` library to verify each generated solution: if the predicted answer matches the reference, the model receives a reward of $+1$, and otherwise, 0. The `math-verify` library is responsible for parsing the model output, extracting the final boxed or numeric answer, and checking it against the ground truth. No format-based shaping or auxiliary heuristics are used. This choice maintains a simple, deterministic, and aligned reward signal across all benchmarks, aligning with the evaluation objective.

A.2 Evaluation Setups

Our evaluation setup largely follows prior work (Zeng et al., 2025; Yang et al., 2024), ensuring consistency and comparability with established baselines. For all math reasoning benchmarks, including AMC23, AIME24, MATH-500, Minerva Math, OlympiadBench, and MMLU-STEM, we use a maximum generation length of 16,000 tokens, with nucleus sampling ($p = 0.95$) and temperature set to 1.0. For the more challenging benchmarks, we increase the number of evaluation samples and report Pass@1 averaged over 16 samplings for AMC23 and over 32 samplings for AIME24. For IFEval, we employ the lighteval (Habib et al., 2023) framework for evaluation, maintaining the same decoding parameters as those used in the math reasoning benchmarks. This uniform setup ensures that all comparisons focus on the effects of SPEC-RL, rather than variations in decoding configurations. For experiments on DeepMath-6K, we report the performance at step 90 (corresponding to 15 epochs with 6,144 examples and a batch size of 1,024). For SimpleRL-8K, we report the performance at step 100.

B Additional Results and Analyses

This section reports additional experimental results and analyses that complement the main findings. We present extended ablations across datasets (DeepMath-6K vs. SimpleRL-8K) and training-set sizes (2K–6K), detailed efficiency analyses, and end-to-end time breakdowns. We further include full step-level training trajectories, additional baseline comparisons, and case studies to provide a more comprehensive view of the behavior of SPEC-RL throughout training.

B.1 End-to-End Time Breakdown

Table 4 reports the per-stage breakdown of training time. In the vanilla baseline, rollout generation dominates the runtime, often accounting for more than 60% of the total. With SPEC-RL, this cost is largely shifted into a lightweight verification stage, where cached rollouts are first verified in parallel under the current policy and then evaluated by the

Algorithm	End-to-end (h) Total	Average step time (s) Total	Δ vs. base	verification	rollout	assembly	reward	old-log-probs	ref	values	adv	update-critic	update-actor	others
Qwen-3-1.7B-Base														
GRPO	12.63	505.1	–	–	309.9	–	91.0	17.2	15.8	–	0.4	–	56.0	14.9
↪ + SPEC-RL	8.65	346.0	↓ 159.1	22.1	135.2 (2.29×)	1.5	81.0	17.1	16.3	–	0.5	–	56.2	16.2
PPO	14.10	563.9	–	–	308.1	–	100.5	17.2	–	14.0	4.7	46.0	56.5	16.9
↪ + SPEC-RL	10.78	431.2	↓ 132.7	22.7	158.6 (1.94×)	1.4	94.1	17.3	–	13.8	4.6	45.0	55.5	18.1
DAPO	11.10	443.8	–	–	301.3	–	93.1	8.6	–	–	0.3	–	25.9	14.6
↪ + SPEC-RL	7.90	316.0	↓ 127.9	21.0	139.0 (2.17×)	1.4	97.9	18.1	–	–	0.2	–	25.9	12.7
Qwen-3-8B-Base														
GRPO	31.66	1266.4	–	–	768.2	–	73.2	66.8	66.9	–	4.2	–	263.8	23.4
↪ + SPEC-RL	21.03	841.0	↓ 425.4	74.7	305.8 (2.51×)	1.3	61.4	63.8	62.4	–	4.9	–	248.8	18.0
PPO	34.85	1393.9	–	–	676.7	–	70.5	65.4	–	57.4	4.2	224.1	260.4	35.3
↪ + SPEC-RL	26.97	1078.8	↓ 315.1	71.5	349.3 (1.94×)	1.4	64.9	59.6	–	52.1	4.9	205.9	236.9	32.5
DAPO	24.29	971.8	–	–	699.2	–	64.4	66.3	–	–	0.1	–	121.1	20.7
↪ + SPEC-RL	12.90	515.9	↓ 455.9	51.0	243.0 (2.88×)	1.1	54.0	51.2	–	–	0.1	–	97.5	18.0
LLaMA-3.2-1B-Instruct														
GRPO	10.20	408.0	–	–	229.7	–	105.8	12.6	11.5	–	0.4	–	34.7	13.2
↪ + SPEC-RL	7.28	291.3	↓ 116.7	17.2	88.3 (2.60×)	1.4	110.4	13.0	11.9	–	0.5	–	34.4	14.4
PPO	10.94	437.6	–	–	218.9	–	117.6	12.5	–	10.0	4.8	10.0	32.6	31.3
↪ + SPEC-RL	8.60	344.0	↓ 93.6	17.5	108.9 (2.01×)	1.3	110.9	12.4	–	10.1	4.6	10.1	34.3	33.8
DAPO	9.77	328.4	–	–	198.4	–	100.8	11.2	–	–	0.1	–	9.6	8.5
↪ + SPEC-RL	6.97	238.4	↓ 90.0	13.4	80.0 (2.48×)	1.1	110.5	11.5	–	–	0.1	–	9.9	12.0

Table 4: End-to-end training time comparison across models and algorithms. We report both the **wall-clock training hours** ("End-to-end (h)") and the **average step time** ("Total (s)") with a detailed breakdown. *validation* refers to our newly introduced speculative decoding process that verifies old-policy rollouts in parallel; *assemble* denotes combining verified prefixes with newly generated continuations to form complete rollouts; the remaining parts (*reward*, *old-log-probs*, *ref*, *values*, *adv*, *update-critic*, *update-actor*, *others*) follow the standard pipeline of the verl ramework in execution order.

Algorithm	Rollout Efficiency		Math Reasoning					OOD		AVG
	Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Qwen3-14B-Base										
Base Model	-	-	45.2	9.1	68.6	28.3	33.5	68.3	43.6	42.4
GRPO	587.6	1.00×	66.2	21.1	**87.2**	43.4	49.5	84.7	49.7	57.4
↪ + SPEC-RL	**234.4**	**2.09×**	**67.8**	**26.2**	86.8	**46.3**	**50.7**	**84.9**	**54.9**	**59.7**
PPO	579.2	1.00×	67.5	22.1	**86.6**	41.9	50.2	**87.1**	**49.5**	57.8
↪ + SPEC-RL	**310.1**	**1.66×**	**75.5**	**29.1**	86.2	41.9	**53.6**	82.3	47.1	**59.4**
DAPO	641.1	1.00×	62.7	16.4	84.6	41.9	45.2	**72.9**	49.9	53.4
↪ + SPEC-RL	**242.8**	**2.46×**	**65.6**	**19.9**	**85.2**	**42.6**	**48.4**	72.4	**51.4**	**55.1**

Table 5: Results on *Qwen3-14B-Base* across different RLVR algorithms on DeepMath-6K.

speculative decoding rule to determine the rejection position, and a minimal assembly stage, where verified prefixes and regenerated suffixes are merged into complete responses. Both stages add only minor overhead (on Qwen-3-1.7B-Base, verification ∼20s and assembly ∼1–2s), while the total step time is reduced by about 129–161s, making the extra cost negligible compared to the savings from reduced rollout. For instance, on Qwen-3-8B-Base/GRPO, the rollout time decreases from 768.2s to 305.8s, while all other stages, such as reward computation and policy updates, remain nearly unchanged. Overall, although these new stages slightly increase non-rollout costs, the dominant effect is the 2–3 times reduction in rollout tokens, yielding substantially faster end-to-end training.

B.2 Scalability to Larger Models

To complement the main results in Table 1, we further evaluate SPEC-RL on a larger backbone, *Qwen3-14B-Base*, to examine whether the efficiency gains and performance trends observed on smaller models persist at increased scale. The main results are summarized in Table 5, and the complete training trajectories tables are reported below in Section C.1.

Across all three RLVR algorithms (GRPO, PPO, and DAPO), SPEC-RL consistently achieves substantial reductions in rollout tokens, yielding speedups of 1.66× to 2.46× compared to the corresponding vanilla baselines. These efficiency gains are comparable to, and in some cases larger than, those observed on smaller models, indicating that speculative reuse remains effective even when scaling to a 14B parameter backbone. In terms of rea-

Algorithm	Rollout Efficiency		Math Reasoning					OOD		AVG
	Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Deepmath-6K (Qwen-3-1.7B-Base)										
GRPO	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	**60.7**	24.4	34.9
↪ + SPEC-RL	**182.7**	**2.29×**	**38.3**	**9.1**	**68.0**	**29.4**	**29.3**	58.3	**28.8**	**37.3**
SimpleRL-8K (Qwen-3-1.7B-Base)										
GRPO	639.4	1.00×	40.5	8.6	68.2	27.2	30.5	49.4	24.0	35.5
↪ + SPEC-RL	**354.0**	**1.53×**	**41.2**	**12.9**	**72.2**	27.2	**32.1**	**57.4**	**27.7**	**38.7**

Table 6: Ablation study on different training datasets. Results show that our method maintains improvements in rollout efficiency and accuracy across both Deepmath-6K and SimpleRL-8K settings.

soning performance, SPEC-RL preserves or improves accuracy across both in-domain and OOD benchmarks. For GRPO and DAPO, SPEC-RL slightly improves average accuracy while reducing rollout cost by more than half. Under PPO, we observe notable gains on challenging math benchmarks such as AMC23 and AIME24, with overall performance remaining competitive despite the aggressive reduction in generation cost. These results suggest that the reuse mechanism does not degrade learning signals at larger scale, and may even stabilize optimization by reducing redundant sampling. Overall, the results on *Qwen3-14B-Base* demonstrate that SPEC-RL scales favorably to larger models, delivering consistent efficiency improvements without sacrificing reasoning performance. This provides further evidence that speculative reuse is a practical and robust acceleration strategy for RLVR training across model sizes.

B.3 Impact of Training Set Size on Acceleration

Since SPEC-RL accelerates training by reusing cached rollouts from the previous epoch, acceleration can only take effect starting from the second epoch. To study how dataset size influences this effect, we vary the training set size to 2K, 3K, 4K, 5K, and 6K samples, and train Qwen-3-1.7B-Base with GRPO. Figure 7 reports the rollout time across training steps.

We observe that smaller datasets lead to earlier reuse opportunities, since epochs finish more quickly and the second epoch arrives sooner. For example, with 2K samples, the rollout time drops sharply after step 3, whereas with 6K samples, the reduction is delayed until later steps. Across all settings, rollout time decreases steadily once reuse begins, with larger speedups achieved as training progresses. The markers in the figure denote the

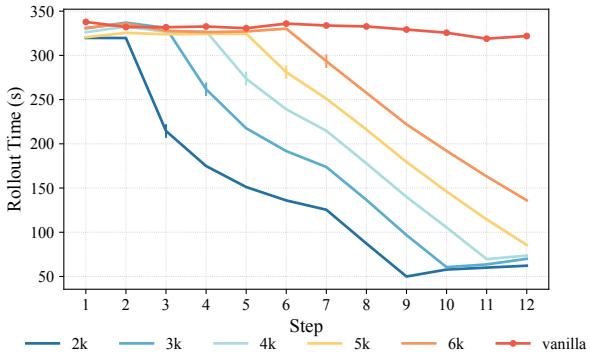

Figure 7: Rollout time under different training set sizes (2K–6K) with GRPO on Qwen-3-1.7B-Base. Markers highlight the first reuse points at the start of epoch 2, when SPEC-RL begins to accelerate rollouts.

first reuse points (the first step of epoch 2), where SPEC-RL begins to take effect. This analysis confirms that the efficiency gains of SPEC-RL depend not only on algorithm and model choice, but also on the dataset size, which determines how soon reuse can be activated during training.

B.4 Random Reuse and Delayed Reuse Results

For completeness, we also report the full training trajectories of the Random Reuse baseline on Qwen-3-1.7B-Base, trained with GRPO on DeepMath-6K.

Table 8 interleaves results of GRPO and Random Reuse every 10 training steps, providing a step-wise view of rollout efficiency and accuracy. While Table 2 summarizes the overall comparison, these detailed results illustrate how Random Reuse accelerates rollouts but produces unstable performance over the course of training.

Table 9 similarly reports the full training trajectories of the *Delayed Reuse* variant, interleaving GRPO and Delayed Reuse every 10 training steps.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	61.5	1.00×	31.7	4.6	57.6	19.1	22.7	44.4	21.1	28.7
↪ + SPEC-RL	10	50.5	1.16×	33.1	5.7	57.8	19.1	24.9	44.9	21.3	29.5
GRPO	20	123.1	1.00×	35.0	5.3	60.6	22.1	25.9	46.1	23.7	31.2
↪ + SPEC-RL	20	84.3	1.33×	34.5	6.9	62.2	20.6	28.0	46.4	20.9	31.4
GRPO	30	185.3	1.00×	33.9	6.5	63.8	24.3	27.0	46.1	23.7	32.2
↪ + SPEC-RL	30	112.2	1.47×	36.1	6.7	61.4	26.8	28.7	49.0	25.1	33.4
GRPO	40	247.7	1.00×	33.8	7.5	63.0	22.4	28.1	46.9	21.1	31.8
↪ + SPEC-RL	40	136.8	1.57×	39.4	8.6	65.8	24.3	29.5	49.8	26.1	34.8
GRPO	50	312.5	1.00×	35.9	7.4	65.2	27.2	26.8	48.7	23.1	33.5
↪ + SPEC-RL	50	171.4	1.58×	41.6	9.6	66.2	28.7	30.8	52.9	26.2	36.6
GRPO	60	377.7	1.00×	39.5	7.3	64.6	26.5	28.0	48.5	23.3	34.0
↪ + SPEC-RL	60	206.9	1.58×	39.5	9.5	67.8	26.8	32.0	53.5	25.5	36.4
GRPO	70	444.8	1.00×	39.5	8.3	66.0	26.5	26.5	48.6	20.9	33.8
↪ + SPEC-RL	70	246.5	1.56×	42.7	10.5	70.0	28.3	31.3	53.8	27.9	37.8
GRPO	80	512.9	1.00×	39.5	7.4	66.8	26.5	30.5	47.5	25.0	34.7
↪ + SPEC-RL	80	283.2	1.55×	44.8	10.7	68.4	26.5	31.7	55.5	25.0	37.5
GRPO	90	582.6	1.00×	38.6	7.5	66.6	25.7	29.5	48.4	24.6	34.4
↪ + SPEC-RL	90	324.8	1.53×	44.2	11.6	70.4	27.9	31.9	55.4	27.0	38.3
GRPO	100	639.4	1.00×	40.5	8.6	68.2	27.2	30.5	49.4	24.0	35.5
↪ + SPEC-RL	100	354.0	1.54×	41.2	12.9	72.2	27.2	32.1	57.4	27.7	38.7

Table 7: Intermediate training results of Qwen-3-1.7B-Base on **SimpleRL-8K** with GRPO and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with GRPO and its SPEC-RL variant interleaved.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + Random Reuse	10	58.0	1.11×	31.7	3.8	57.8	21.0	21.0	42.9	22.0	28.6
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + Random Reuse	20	98.9	1.43×	35.5	5.3	59.4	24.6	24.1	46.0	25.9	31.5
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + Random Reuse	30	134.1	1.61×	32.0	7.6	63.2	26.8	24.9	50.6	29.2	33.5
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + Random Reuse	40	165.4	1.75×	38.9	7.4	64.0	26.1	28.4	56.8	27.7	35.6
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + Random Reuse	50	194.1	1.89×	36.1	6.8	63.2	27.9	27.0	57.7	25.7	34.9
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + Random Reuse	60	221.8	2.03×	37.0	6.7	64.2	24.6	25.9	59.2	26.1	34.8
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + Random Reuse	70	249.6	2.14×	36.7	5.4	63.2	27.2	26.1	60.9	25.5	35.0
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + Random Reuse	80	277.5	2.25×	27.5	4.9	57.0	21.7	22.7	44.6	22.4	28.7
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + Random Reuse	90	304.5	2.35×	30.3	5.0	60.4	21.7	25.3	53.1	24.0	31.4

Table 8: Intermediate training results of Qwen-3-1.7B-Base on DeepMath-6K with GRPO and **Random Reuse**. We report rollout efficiency and accuracy every 10 training steps, with GRPO and its Random Reuse variant interleaved.

This presentation provides a step-wise view of rollout efficiency and accuracy under delayed reuse scheduling, allowing direct comparison with the Random Reuse baseline.

B.5 Full Lenience Ablation Results

For completeness, we provide the step-level results corresponding to the lenience ablation in Section 4.3. While Table 3 reports only the final-step outcomes for comparison across different lenience values, we include detailed intermediate results every 10 training steps in Tables 10, 11, 12, 13, 14, 15. These tables document how rollout efficiency and accuracy evolve throughout training under various lenience settings ($\ell = 1, e^{0.2}, e^{0.5}, e^{0.8}, e^{1.0}, e^{2.0}$, and $\ell \to \infty$), complementing the aggregated trends shown in Table 3.

Algorithm	Step	Rollout Efficiency		Math Reasoning						OOD	AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + Delayed Reuse	10	44.4	1.29×	32.2	4.9	57.4	21.7	22.1	44.0	19.6	28.8
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + Delayed Reuse	20	85.4	1.26×	34.4	5.5	58.4	21.7	24.9	46.5	21.3	30.4
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + Delayed Reuse	30	123.8	1.27×	33.3	6.5	61.0	26.5	26.1	51.2	24.2	32.7
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + Delayed Reuse	40	156.5	1.31×	35.5	6.8	62.6	27.9	24.3	53.2	25.1	33.6
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + Delayed Reuse	50	187.9	1.35×	35.8	7.4	65.0	27.6	28.4	55.4	25.9	35.1
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + Delayed Reuse	60	218.5	1.37×	35.3	7.8	66.4	27.6	27.0	55.9	26.2	35.2
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + Delayed Reuse	70	248.8	1.40×	31.4	8.9	64.6	28.3	26.8	57.5	24.6	34.6
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + Delayed Reuse	80	279.7	1.42×	32.8	8.8	65.4	29.0	27.3	57.2	26.4	35.3
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + Delayed Reuse	90	308.8	1.44×	35.6	9.0	66.4	29.0	28.6	57.3	28.3	36.3

Table 9: Interleaved comparison of GRPO and **Delayed Reuse**. Results are reported every 10 training steps.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + SPEC-RL $\ell = 1$	10	63.5	1.03×	28.6	5.0	55.2	20.2	21.2	44.5	18.9	27.7
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + SPEC-RL $\ell = 1$	20	118.3	1.05×	34.7	5.2	61.6	25.4	22.4	45.1	22.9	31.0
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + SPEC-RL $\ell = 1$	30	168.0	1.07×	34.1	6.1	61.2	25.0	25.0	49.9	23.7	32.1
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + SPEC-RL $\ell = 1$	40	213.5	1.11×	36.2	6.5	63.0	27.2	24.9	51.7	22.4	33.1
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + SPEC-RL $\ell = 1$	50	257.0	1.14×	35.0	6.9	63.2	29.0	24.7	54.0	24.4	33.9
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + SPEC-RL $\ell = 1$	60	298.5	1.17×	36.6	7.5	64.2	26.8	26.7	55.6	23.7	34.4
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + SPEC-RL $\ell = 1$	70	339.2	1.19×	36.4	6.6	62.6	28.7	28.0	58.0	26.8	35.3
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + SPEC-RL $\ell = 1$	80	379.7	1.20×	37.0	7.4	67.4	22.8	28.0	60.2	23.7	35.2
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + SPEC-RL $\ell = 1$	90	419.1	1.22×	37.8	8.0	63.8	28.7	26.5	59.6	25.9	35.8

Table 10: Intermediate training results of Qwen-3-1.7B-Base on DeepMath-6K with GRPO and SPEC-RL at lenience $\ell = 1$. We report rollout efficiency and accuracy every 10 training steps, illustrating the progression of model performance during training.

B.6 Generality Across Datasets

To examine whether the gains of SPEC-RL depend on a specific training corpus, we conduct experiments on two distinct datasets: DeepMath-6K and SimpleRL-8K. Results in Table 6 show that SPEC-RL consistently improves rollout efficiency across both settings. For example, on Qwen-3-1.7B-Base with GRPO, rollout tokens drop from 554.8M to 182.7M on DeepMath-6K and from 639.4M to 354.0M on SimpleRL-8K. Accuracy remains comparable or slightly improved, confirming that the efficiency benefits of SPEC-RL are robust to the choice of dataset. Intermediate performance on SimpleRL-8K is reported in Table 7, while the detailed results for DeepMath-6K can be found in Table 16. These results suggest that the efficiency improvements of SPEC-RL do not rely on a particular training distribution.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + SPEC-RL $\ell = e^{0.2}$	10	53.6	1.19×	30.5	4.0	55.6	19.1	22.1	42.4	20.0	27.7
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + SPEC-RL $\ell = e^{0.2}$	20	92.7	1.29×	33.3	5.2	60.6	26.5	25.5	46.2	20.0	31.0
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + SPEC-RL $\ell = e^{0.2}$	30	120.5	1.42×	34.8	6.2	60.8	26.8	27.0	49.5	22.9	32.6
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + SPEC-RL $\ell = e^{0.2}$	40	143.6	1.54×	34.2	6.7	61.8	28.3	26.4	51.8	22.2	33.1
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + SPEC-RL $\ell = e^{0.2}$	50	166.1	1.62×	33.4	8.5	65.2	27.6	25.9	54.4	25.5	34.4
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + SPEC-RL $\ell = e^{0.2}$	60	186.1	1.70×	35.9	7.5	63.6	29.4	25.5	55.0	24.4	34.5
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + SPEC-RL $\ell = e^{0.2}$	70	206.5	1.77×	34.1	7.3	64.6	27.2	29.5	58.4	25.1	35.2
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + SPEC-RL $\ell = e^{0.2}$	80	226.3	1.82×	37.3	5.9	63.4	29.4	29.9	57.9	24.0	35.4
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + SPEC-RL $\ell = e^{0.2}$	90	246.7	1.86×	35.8	7.8	66.4	29.8	29.6	58.5	25.9	36.3

Table 11: Intermediate training results of Qwen-3-1.7B-Base on DeepMath-6K with GRPO and SPEC-RL at lenience $\ell = e^{0.2}$. We report rollout efficiency and accuracy every 10 training steps, illustrating the progression of model performance during training.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + SPEC-RL $\ell = e^{0.5}$	10	43.6	1.41×	32.5	5.7	55.8	21.7	21.8	43.0	22.4	29.0
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + SPEC-RL $\ell = e^{0.5}$	20	67.2	1.66×	34.2	5.9	63.0	25.0	24.6	46.9	22.2	31.7
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + SPEC-RL $\ell = e^{0.5}$	30	85.1	1.85×	34.5	7.2	64.0	25.4	27.6	51.1	26.2	33.7
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + SPEC-RL $\ell = e^{0.5}$	40	102.9	1.96×	37.0	7.4	63.8	26.5	26.1	52.2	23.5	33.8
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + SPEC-RL $\ell = e^{0.5}$	50	119.4	2.06×	33.1	7.3	64.4	28.7	28.0	55.6	27.7	35.0
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + SPEC-RL $\ell = e^{0.5}$	60	135.1	2.14×	36.6	7.0	66.4	26.5	29.9	54.7	28.8	35.7
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + SPEC-RL $\ell = e^{0.5}$	70	153.2	2.18×	37.0	8.1	65.4	26.5	29.9	55.6	27.4	35.7
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + SPEC-RL $\ell = e^{0.5}$	80	168.1	2.24×	35.0	10.1	67.0	29.8	29.6	57.1	28.3	36.7
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + SPEC-RL $\ell = e^{0.5}$	90	182.7	2.29×	38.3	9.1	68.0	29.4	29.3	58.3	28.8	37.3

Table 12: Intermediate training results of Qwen-3-1.7B-Base on DeepMath-6K with GRPO and SPEC-RL at lenience $\ell = e^{0.5}$. We report rollout efficiency and accuracy every 10 training steps, illustrating the progression of model performance during training.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + SPEC-RL $\ell = e^{0.8}$	10	41.9	1.46×	31.7	3.8	57.0	19.9	21.6	44.8	20.1	28.4
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + SPEC-RL $\ell = e^{0.8}$	20	57.5	1.86×	35.5	5.3	60.0	26.1	24.4	46.2	23.3	31.5
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + SPEC-RL $\ell = e^{0.8}$	30	70.1	2.12×	33.4	7.6	62.4	28.7	24.0	51.5	27.4	33.6
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + SPEC-RL $\ell = e^{0.8}$	40	81.8	2.29×	37.7	7.4	63.8	26.8	27.7	53.8	28.5	35.1
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + SPEC-RL $\ell = e^{0.8}$	50	97.5	2.35×	35.8	6.8	63.4	28.3	26.5	57.0	28.7	35.2
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + SPEC-RL $\ell = e^{0.8}$	60	110.8	2.43×	36.2	6.7	61.8	26.8	25.8	57.7	25.5	34.4
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + SPEC-RL $\ell = e^{0.8}$	70	120.0	2.54×	37.2	5.4	62.2	26.8	25.9	58.9	27.9	34.9
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + SPEC-RL $\ell = e^{0.8}$	80	132.0	2.60×	25.0	4.9	63.6	27.6	26.5	58.4	28.3	33.5
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + SPEC-RL $\ell = e^{0.8}$	90	144.8	2.64×	28.6	5.0	63.6	27.2	25.0	61.7	26.2	33.9

Table 13: Intermediate training results of Qwen-3-1.7B-Base on DeepMath-6K with GRPO and SPEC-RL at lenience $\ell = e^{0.8}$. We report rollout efficiency and accuracy every 10 training steps, illustrating the progression of model performance during training.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + SPEC-RL $\ell = e^{2.0}$	10	40.4	1.50×	32.8	4.3	55.4	21.7	21.8	44.0	19.6	28.5
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + SPEC-RL $\ell = e^{2.0}$	20	43.9	2.24×	34.8	4.8	60.4	28.3	23.7	44.3	25.0	31.6
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + SPEC-RL $\ell = e^{2.0}$	30	51.2	2.59×	36.2	7.3	61.0	26.1	25.6	49.8	29.2	33.6
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + SPEC-RL $\ell = e^{2.0}$	40	54.5	2.93×	35.9	7.4	64.0	28.7	26.4	52.5	31.4	35.2
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + SPEC-RL $\ell = e^{2.0}$	50	60.1	3.14×	26.6	4.0	61.6	25.7	24.1	52.4	29.2	31.9
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + SPEC-RL $\ell = e^{2.0}$	60	67.2	3.24×	29.4	6.4	57.8	25.0	24.3	54.6	31.4	32.7
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + SPEC-RL $\ell = e^{2.0}$	70	76.8	3.28×	29.5	6.0	57.4	21.7	23.7	52.5	28.7	31.4
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + SPEC-RL $\ell = e^{2.0}$	80	90.5	3.24×	27.8	4.9	57.4	19.5	23.4	54.8	30.3	31.2
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + SPEC-RL $\ell = e^{2.0}$	90	114.4	3.05×	26.2	5.2	55.0	21.0	21.9	53.5	29.0	30.3

Table 14: Intermediate training results of Qwen-3-1.7B-Base on DeepMath-6K with GRPO and SPEC-RL at lenience $\ell = e^{2.0}$. We report rollout efficiency and accuracy every 10 training steps, illustrating the progression of model performance during training.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + SPEC-RL $\ell = \infty$	10	40.0	1.75×	29.2	3.5	53.8	16.9	21.6	42.5	20.0	26.8
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + SPEC-RL $\ell = \infty$	20	40.0	3.39×	31.6	4.8	56.4	21.0	22.7	44.4	21.4	28.9
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + SPEC-RL $\ell = \infty$	30	40.0	5.03×	30.2	3.8	53.2	21.7	21.3	43.0	21.4	27.8
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + SPEC-RL $\ell = \infty$	40	40.0	6.65×	33.0	4.9	57.0	17.3	22.4	43.3	22.6	28.6
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + SPEC-RL $\ell = \infty$	50	40.0	8.29×	34.1	4.3	57.0	23.2	22.5	43.3	22.7	29.6
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + SPEC-RL $\ell = \infty$	60	40.0	9.93×	31.9	3.8	58.8	23.5	22.8	42.6	23.7	29.6
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + SPEC-RL $\ell = \infty$	70	40.0	11.56×	33.1	4.5	58.6	23.2	22.4	44.0	20.9	29.5
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + SPEC-RL $\ell = \infty$	80	40.0	13.21×	31.1	4.0	60.2	23.9	24.3	42.8	23.3	29.9
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + SPEC-RL $\ell = \infty$	90	40.0	14.86×	30.2	4.4	60.4	19.9	23.7	44.1	22.0	29.2

Table 15: Intermediate training results of Qwen-3-1.7B-Base on DeepMath-6K with GRPO and SPEC-RL at lenience $\ell = \infty$. We report rollout efficiency and accuracy every 10 training steps, illustrating the progression of model performance during training.

C Full Tables and Case Studies

C.1 Full Performance over Training Steps

To provide a more complete view of model behavior and enhance the robustness of our method, we also report performance trajectories throughout training.

Training Dynamics and Efficiency Across Different RL Algorithms. We present the efficiency of our method across RL algorithms in Figures 8, 9, and compare rewards and rollout time against baselines in Figures 10, 11. Across all three algorithms, SPEC-RL substantially reduces rollout time while preserving learning quality: rewards match or exceed the vanilla baselines under PPO and GRPO, and are largely on par under DAPO (with a minor late-stage gap on Qwen3-8B). The efficiency gains align with stronger speculative reuse signals: the full reuse ratio quickly rises and stabilizes around 0.6–0.85 after early transients, and the average verified prefix length remains large (hundreds to 1.2k tokens) and generally increases over training—most prominently on Qwen3-8B for GRPO/DAPO. Together, these curves indicate that SPEC-RL learns to reuse long, verified prefixes, trading decoding for reuse, which yields lower per-step generation cost without compromising reward progress.

For each setting, results are shown every 10 steps, comparing the vanilla algorithm with its SPEC-RL variant, as shown in Tables 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27. This step-wise view complements the main results by illustrating how rollout efficiency and accuracy evolve consistently during training, rather than only at the final checkpoint.

C.2 Case Studies

To provide a more intuitive understanding of how SPEC-RL operates during training, we present several case studies comparing cached rollouts from previous epochs with newly generated rollouts under the current policy. These examples highlight how speculative prefixes are verified and reused, and how continuation is triggered once a rejection occurs. They also illustrate typical scenarios where SPEC-RL improves efficiency by avoiding redundant generation, while still correcting erroneous reasoning steps when necessary. Representative cases are shown in Figures 12, 13, 14, and 15.

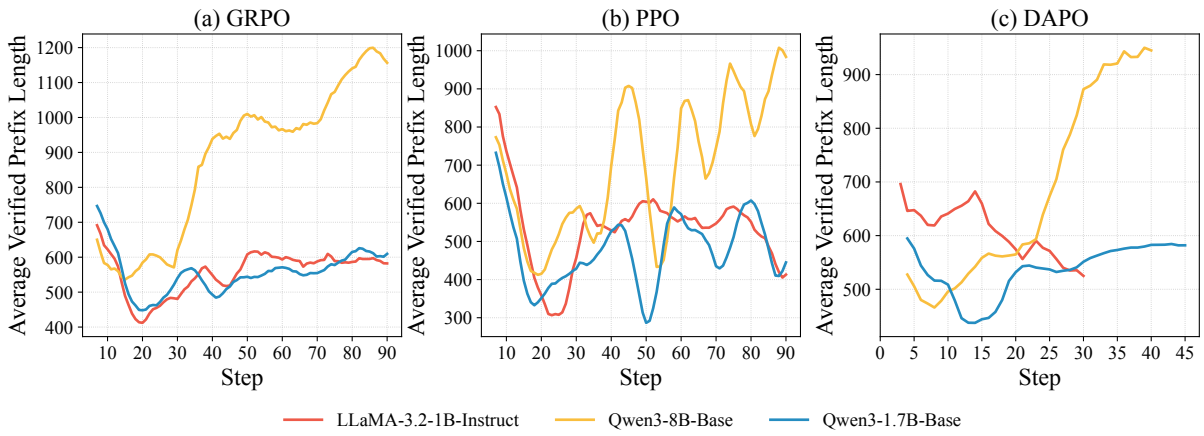

Figure 8: Average verified prefix length trajectories of SPEC-RL under three RL algorithms: (a) GRPO, (b) PPO, and (c) DAPO. The y-axis reports the average length of the verified speculative prefix per training step, and the x-axis is the training step. colors denote model backbones: red: LLaMA-3.2-1B-Instruct, yellow: Qwen3-8B-Base, blue: Qwen3-1.7B-Base.

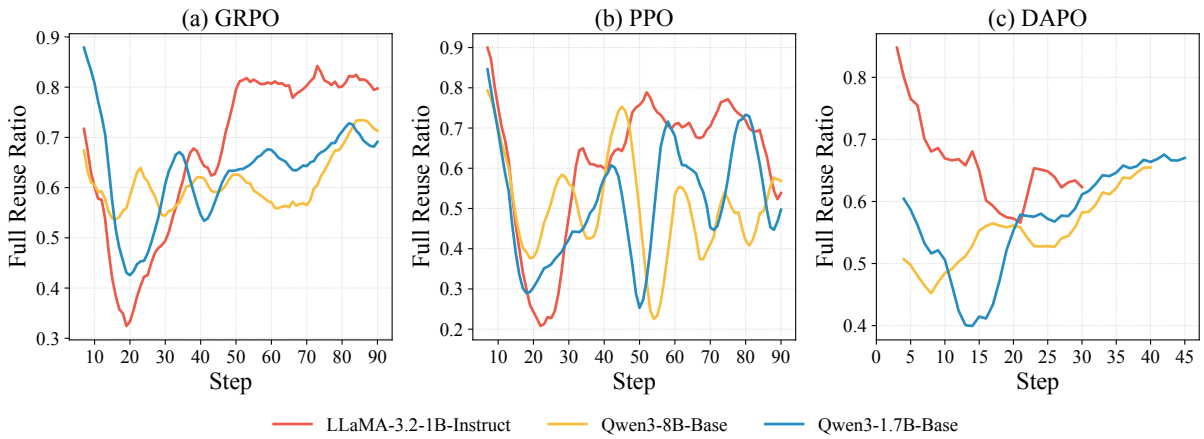

Figure 9: Full reuse ratio trajectories of SPEC-RL under three RL algorithms: (a) GRPO, (b) PPO, and (c) DAPO. The y-axis reports the fraction of rollouts per step that are fully reused, and the x-axis the training step. colors denote model backbones: red: LLaMA-3.2-1B-Instruct, yellow: Qwen3-8B-Base, blue: Qwen3-1.7B-Base. Across settings, SPEC-RL quickly stabilizes at a high full reuse ratio, indicating effective speculative reuse during training.

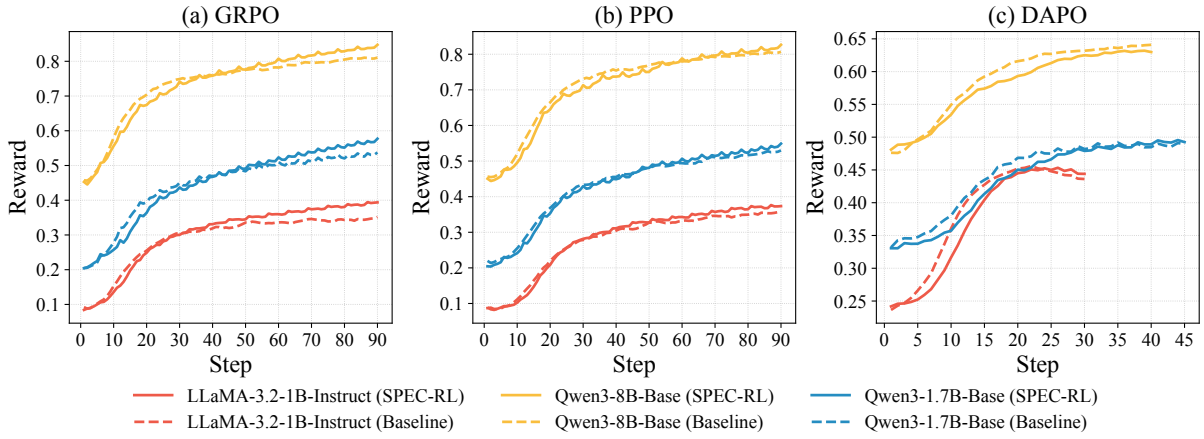

Figure 10: Training reward trajectories of SPEC-RL versus baseline under three RL algorithms: (a) GRPO, (b) PPO, and (c) DAPO. The y-axis reports reward, and the x-axis the training step. colors denote model backbones: red: LLaMA-3.2-1B-Instruct, yellow: Qwen3-8B-Base, blue: Qwen3-1.7B-Base, while solid lines indicate SPEC-RL and dashed lines the corresponding vanilla baselines. SPEC-RL matches or exceeds baseline rewards under different algorithms across all backbones.

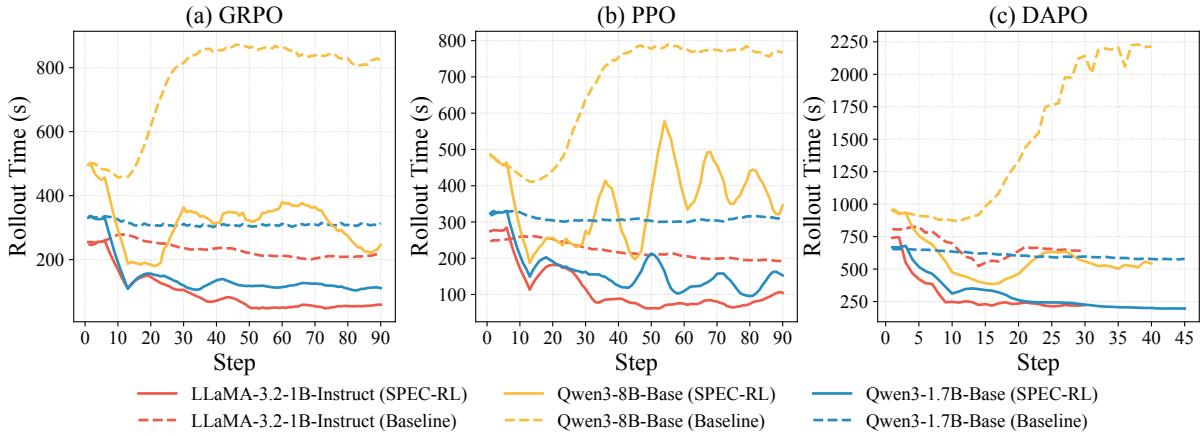

Figure 11: Training rollout time of SPEC-RL versus baseline under three RL algorithms: (a) GRPO, (b) PPO, and (c) DAPO. The y-axis reports rollout time (seconds) and the x-axis the training step. colors denote model backbones: red: LLaMA-3.2-1B-Instruct, yellow: Qwen3-8B-Base, blue: Qwen3-1.7B-Base, while solid lines indicate SPEC-RL and dashed lines the corresponding vanilla baselines. Across algorithms and models, SPEC-RL yields consistently lower rollout time than the baselines.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
GRPO	10	65.8	1.00×	27.8	4.3	54.0	18.8	22.8	43.9	20.3	27.4
↪ + SPEC-RL	10	43.6	1.41×	32.5	5.7	55.8	21.7	21.8	43.0	22.4	29.0
GRPO	20	127.4	1.00×	31.2	6.1	57.8	25.4	24.7	45.5	20.3	30.1
↪ + SPEC-RL	20	67.2	1.66×	34.2	5.9	63.0	25.0	24.6	46.9	22.2	31.7
GRPO	30	187.7	1.00×	33.8	5.9	60.4	22.4	25.0	47.9	20.9	30.9
↪ + SPEC-RL	30	85.1	1.85×	34.5	7.2	64.0	25.4	27.6	51.1	26.2	33.7
GRPO	40	248.3	1.00×	36.4	6.6	65.0	23.5	24.7	50.7	21.3	32.6
↪ + SPEC-RL	40	102.9	1.96×	37.0	7.4	63.8	26.5	26.1	52.2	23.5	33.8
GRPO	50	309.1	1.00×	36.6	5.9	64.2	25.7	24.6	53.7	25.1	33.7
↪ + SPEC-RL	50	119.4	2.06×	33.1	7.3	64.4	28.7	28.0	55.6	27.7	35.0
GRPO	60	370.4	1.00×	38.0	6.7	63.6	28.3	26.7	56.0	24.0	34.8
↪ + SPEC-RL	60	135.1	2.14×	36.6	7.0	66.4	26.5	29.9	54.7	28.8	35.7
GRPO	70	431.9	1.00×	35.3	8.0	61.8	26.5	28.1	55.5	26.2	34.5
↪ + SPEC-RL	70	153.2	2.18×	37.0	8.1	65.4	26.5	29.9	55.6	27.4	35.7
GRPO	80	493.5	1.00×	36.1	7.2	64.4	24.3	26.4	59.4	25.0	34.7
↪ + SPEC-RL	80	168.1	2.24×	35.0	10.1	67.0	29.8	29.6	57.1	28.3	36.7
GRPO	90	554.8	1.00×	35.0	7.8	64.4	26.5	25.5	60.7	24.4	34.9
↪ + SPEC-RL	90	182.7	2.29×	38.3	9.1	68.0	29.4	29.3	58.3	28.8	37.3

Table 16: Intermediate training results of **Qwen-3-1.7B-Base** on DeepMath-6K with **GRPO** and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with GRPO and its SPEC-RL variant interleaved.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	40.2	11.5	67.4	27.2	34.1	60.4	29.9	38.7
GRPO	10	71.9	1.00×	56.2	16.0	80.0	32.7	44.3	64.7	34.4	46.9
↪ + SPEC-RL	10	53.0	1.36×	56.2	17.1	80.0	37.1	43.9	64.2	37.2	48.0
GRPO	20	158.2	1.00×	64.7	20.8	82.8	40.4	49.3	77.3	39.0	53.5
↪ + SPEC-RL	20	76.4	1.96×	63.4	22.3	83.6	42.6	48.6	72.2	43.4	53.7
GRPO	30	278.1	1.00×	65.8	23.5	84.2	39.7	48.9	80.4	35.7	54.0
↪ + SPEC-RL	30	116.7	2.18×	66.6	24.4	85.0	43.0	49.5	80.4	47.9	56.7
GRPO	40	404.2	1.00×	66.2	24.6	85.2	40.8	50.2	82.0	37.9	55.3
↪ + SPEC-RL	40	156.4	2.31×	68.9	25.1	84.2	44.5	49.0	83.3	46.8	57.4
GRPO	50	532.0	1.00×	68.0	23.5	85.4	42.6	49.5	82.8	40.1	56.0
↪ + SPEC-RL	50	194.4	2.36×	72.2	25.3	84.8	44.1	52.3	83.2	45.7	58.2
GRPO	60	659.3	1.00×	70.6	24.4	84.8	44.1	51.4	83.0	38.8	56.7
↪ + SPEC-RL	60	235.7	2.36×	70.6	27.7	85.4	43.0	51.1	84.4	44.9	58.2
GRPO	70	785.6	1.00×	70.3	24.3	84.8	43.4	51.3	84.3	34.8	56.2
↪ + SPEC-RL	70	279.2	2.36×	70.9	27.1	87.0	43.8	51.7	84.7	47.5	59.0
GRPO	80	910.2	1.00×	70.6	25.3	85.8	43.4	50.2	84.7	40.1	57.2
↪ + SPEC-RL	80	311.1	2.42×	73.8	25.4	87.4	43.4	52.1	85.2	48.2	59.4
GRPO	90	1033.1	1.00×	69.2	24.2	86.4	43.8	53.0	84.6	41.2	57.5
↪ + SPEC-RL	90	336.6	2.51×	72.8	26.9	87.8	44.1	51.0	84.5	47.7	59.3

Table 17: Intermediate training results of **Qwen-3-8B-Base** on DeepMath-6K with **GRPO** and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with GRPO and its SPEC-RL variant interleaved.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	7.5	0.6	14.2	4.0	2.8	32.6	37.0	14.1
GRPO	10	72.1	1.00×	5.9	0.4	14.2	2.9	3.4	31.7	38.8	13.9
↪ + SPEC-RL	10	47.8	1.38×	6.9	0.1	12.8	2.6	3.4	32.8	38.6	13.9
GRPO	20	141.2	1.00×	6.4	1.0	15.6	2.6	3.9	33.5	39.7	14.7
↪ + SPEC-RL	20	78.5	1.57×	6.1	0.7	18.0	3.7	3.9	35.0	38.6	15.1
GRPO	30	204.5	1.00×	9.2	0.8	17.4	2.9	4.4	35.1	38.3	15.4
↪ + SPEC-RL	30	100.7	1.73×	6.9	0.9	18.4	2.6	4.9	33.4	40.5	15.4
GRPO	40	266.8	1.00×	7.3	0.5	15.4	3.3	4.4	33.2	41.2	15.0
↪ + SPEC-RL	40	115.0	1.94×	9.1	2.0	18.6	4.4	5.5	34.2	38.8	16.1
GRPO	50	326.2	1.00×	9.7	0.8	17.6	3.7	5.3	34.3	38.1	15.6
↪ + SPEC-RL	50	126.3	2.12×	8.0	1.0	20.2	4.4	4.7	36.0	39.6	16.3
GRPO	60	382.9	1.00×	9.1	0.6	17.8	3.3	5.2	34.0	40.5	15.8
↪ + SPEC-RL	60	134.7	2.29×	9.4	1.2	19.0	4.4	5.5	35.6	37.9	16.1
GRPO	70	438.3	1.00×	7.7	1.1	17.6	4.8	5.5	34.6	37.3	15.5
↪ + SPEC-RL	70	143.6	2.41×	9.4	1.2	19.8	5.1	5.8	36.1	37.2	16.4
GRPO	80	495.3	1.00×	7.0	0.8	17.6	4.0	3.9	33.3	37.2	14.8
↪ + SPEC-RL	80	152.8	2.52×	7.2	1.0	19.4	2.9	3.9	35.6	42.7	16.1
GRPO	90	553.9	1.00×	7.5	1.4	19.2	3.3	4.9	33.1	37.0	15.2
↪ + SPEC-RL	90	162.5	2.60×	8.8	1.7	19.4	1.8	5.0	34.5	37.2	15.5

Table 18: Intermediate training results of **LLaMA-3.2-1B-Instruct** on DeepMath-6K with **GRPO** and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with GRPO and its SPEC-RL variant interleaved.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	45.2	9.1	68.6	28.3	33.5	68.3	43.6	42.4
GRPO	10	64.8	1.00×	57.3	13.3	80.2	38.2	42.8	68.3	45.5	49.4
↪ + SPEC-RL	10	50.5	1.21×	58.3	13.6	80.4	40.8	44.3	69.8	45.7	50.4
GRPO	20	126.9	1.00×	59.4	14.0	80.0	41.5	45.6	69.1	47.3	51.0
↪ + SPEC-RL	20	75.9	1.50×	62.3	16.2	81.8	43.8	44.6	69.7	49.4	52.5
GRPO	30	189.7	1.00×	60.3	15.7	82.8	41.2	46.1	72.1	50.5	52.7
↪ + SPEC-RL	30	99.6	1.66×	63.1	17.1	83.8	43.4	45.5	71.1	51.0	53.6
GRPO	40	253.8	1.00×	62.7	15.9	82.6	39.0	45.5	74.3	46.0	52.3
↪ + SPEC-RL	40	122.7	1.78×	64.7	18.1	84.0	44.5	47.7	72.8	53.6	55.1
GRPO	50	319.1	1.00×	63.0	16.4	84.0	41.2	46.4	76.3	45.8	53.3
↪ + SPEC-RL	50	144.4	1.88×	64.8	20.0	84.0	44.9	48.6	77.0	51.0	55.8
GRPO	60	385.3	1.00×	61.9	17.9	85.0	42.6	48.0	79.0	48.4	54.7
↪ + SPEC-RL	60	172.8	1.89×	68.1	22.5	85.6	44.9	49.3	79.0	51.0	57.2
GRPO	70	452.2	1.00×	65.8	20.1	84.4	44.5	47.1	81.4	49.4	56.1
↪ + SPEC-RL	70	193.7	1.97×	66.2	21.8	85.0	40.8	49.2	82.0	53.0	56.9
GRPO	80	519.4	1.00×	65.0	19.6	83.2	43.8	49.0	83.3	49.2	56.2
↪ + SPEC-RL	80	212.9	2.04×	68.4	26.1	87.6	43.8	51.4	83.5	51.0	58.8
GRPO	90	587.6	1.00×	66.2	21.1	87.2	43.4	49.5	84.7	49.7	57.4
↪ + SPEC-RL	90	234.4	2.09×	67.8	26.2	86.8	46.3	50.7	84.9	54.9	59.7

Table 19: Intermediate training results of **Qwen-3-14B-Base** on DeepMath-6K with **GRPO** and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with GRPO and its SPEC-RL variant interleaved.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
PPO	10	66.6	1.00×	27.8	3.4	54.8	19.9	21.3	42.6	19.2	27.0
↪ + SPEC-RL	10	46.5	1.34×	30.6	4.5	56.8	17.6	23.3	42.9	19.4	27.9
PPO	20	129.2	1.00×	30.8	4.9	60.4	22.8	25.0	46.6	21.8	30.3
↪ + SPEC-RL	20	80.0	1.44×	32.3	5.5	58.0	23.2	23.4	46.9	20.9	30.0
PPO	30	191.5	1.00×	32.3	5.3	59.4	22.8	26.5	47.8	19.6	30.5
↪ + SPEC-RL	30	106.2	1.56×	34.1	5.6	62.6	23.5	25.6	49.8	22.7	32.0
PPO	40	253.6	1.00×	32.7	6.4	61.4	23.5	25.3	50.7	22.7	31.8
↪ + SPEC-RL	40	126.2	1.69×	33.1	6.6	63.2	22.4	27.3	51.3	23.7	32.5
PPO	50	315.7	1.00×	34.7	6.8	61.8	26.8	25.6	51.5	21.8	32.7
↪ + SPEC-RL	50	157.3	1.68×	37.3	5.9	64.4	26.1	25.9	54.2	26.6	34.3
PPO	60	377.7	1.00×	35.0	7.3	63.8	29.4	26.8	53.9	23.3	34.2
↪ + SPEC-RL	60	172.0	1.79×	35.5	6.7	64.2	23.5	27.0	53.8	24.4	33.6
PPO	70	440.0	1.00×	35.3	6.5	60.6	25.7	26.7	55.0	22.9	33.2
↪ + SPEC-RL	70	194.7	1.83×	35.8	6.7	65.8	27.6	26.8	55.1	25.9	34.8
PPO	80	503.0	1.00×	36.6	6.9	63.8	25.4	29.3	58.6	23.7	34.9
↪ + SPEC-RL	80	207.2	1.93×	38.8	6.2	63.6	29.8	28.0	54.3	25.7	35.2
PPO	90	565.1	1.00×	35.3	8.3	63.0	26.8	25.3	59.4	25.5	34.8
↪ + SPEC-RL	90	230.8	1.94×	36.9	7.7	64.8	25.4	25.9	58.6	25.9	35.0

Table 20: Intermediate training results of **Qwen-3-1.7B-Base** on DeepMath-6K with **PPO** and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with GRPO and its SPEC-RL variant interleaved.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	40.2	11.5	67.4	27.2	34.1	60.4	29.9	38.7
PPO	10	73.4	1.00×	54.7	13.9	75.4	33.5	43.4	63.2	32.2	45.2
↪ + SPEC-RL	10	51.9	1.30×	51.4	13.8	79.6	32.0	42.4	62.6	36.0	45.4
PPO	20	144.8	1.00×	59.7	18.8	81.0	39.3	45.3	67.0	35.1	49.5
↪ + SPEC-RL	20	85.6	1.48×	60.8	19.7	82.0	39.7	45.5	67.4	37.9	50.4
PPO	30	241.0	1.00×	66.4	23.0	82.8	38.6	46.7	77.6	37.3	53.2
↪ + SPEC-RL	30	115.6	1.75×	64.4	22.0	82.2	40.4	48.1	74.6	42.7	53.5
PPO	40	359.0	1.00×	66.2	23.0	83.6	41.9	49.3	79.5	36.6	54.3
↪ + SPEC-RL	40	159.4	1.87×	66.6	23.6	84.0	39.3	49.5	77.6	43.3	54.8
PPO	50	484.2	1.00×	67.8	24.0	86.0	41.2	51.6	82.1	39.7	56.1
↪ + SPEC-RL	50	197.9	1.98×	69.1	23.8	84.2	42.6	49.3	81.9	41.8	56.1
PPO	60	609.9	1.00×	69.5	23.6	85.4	42.6	49.9	82.9	42.0	56.6
↪ + SPEC-RL	60	251.2	1.95×	68.8	25.7	84.6	43.8	52.0	81.7	43.6	57.2
PPO	70	735.0	1.00×	68.4	24.8	84.4	44.1	51.1	83.6	42.9	57.0
↪ + SPEC-RL	70	307.9	1.91×	66.9	25.0	84.6	42.6	50.8	84.0	44.7	56.9
PPO	80	859.9	1.00×	69.5	26.5	85.6	42.3	51.9	83.3	40.9	57.1
↪ + SPEC-RL	80	358.1	1.90×	69.2	24.7	83.4	44.1	50.2	84.4	43.1	57.0
PPO	90	984.0	1.00×	68.9	26.4	85.8	43.0	51.6	83.8	41.6	57.3
↪ + SPEC-RL	90	400.1	1.94×	70.5	25.1	85.2	43.4	50.8	84.4	41.0	57.2

Table 21: Intermediate training results of **Qwen-3-8B-Base** on DeepMath-6K with **PPO** and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with PPO and its SPEC-RL variant interleaved.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	7.5	0.6	14.2	4.0	2.8	32.6	37.0	14.1
PPO	10	65.5	1.00×	5.5	0.8	11.4	4.0	3.6	33.0	42.0	14.3
↪ + SPEC-RL	10	50.2	1.26×	4.4	0.9	14.0	2.6	3.7	32.5	41.0	14.2
PPO	20	131.1	1.00×	8.1	1.0	16.2	3.7	4.7	33.9	35.9	14.8
↪ + SPEC-RL	20	89.0	1.35×	8.0	0.6	16.2	2.9	5.0	34.1	41.2	15.4
PPO	30	192.0	1.00×	6.7	0.7	17.2	3.7	4.1	35.3	38.6	15.2
↪ + SPEC-RL	30	118.3	1.47×	8.9	0.6	17.4	4.4	6.4	34.2	38.8	15.8
PPO	40	250.3	1.00×	8.3	1.0	19.6	2.9	4.4	35.1	39.0	15.8
↪ + SPEC-RL	40	134.8	1.63×	8.8	1.0	18.6	3.3	6.1	33.2	39.9	15.8
PPO	50	306.8	1.00×	7.5	1.0	19.2	4.8	4.6	32.3	40.1	15.6
↪ + SPEC-RL	50	147.2	1.78×	9.1	1.5	19.4	5.1	5.0	35.3	40.3	16.5
PPO	60	361.6	1.00×	9.4	1.0	18.4	5.1	4.3	35.1	41.6	16.4
↪ + SPEC-RL	60	160.5	1.89×	8.4	1.5	19.2	4.8	5.5	34.4	39.9	16.2
PPO	70	415.4	1.00×	7.5	1.2	17.8	3.7	6.1	35.1	40.1	15.9
↪ + SPEC-RL	70	175.8	1.95×	10.0	0.9	19.6	3.3	5.5	34.9	39.9	16.3
PPO	80	469.0	1.00×	9.5	1.0	19.6	4.8	4.1	34.8	41.6	16.5
↪ + SPEC-RL	80	188.8	2.02×	9.7	1.4	19.0	5.5	6.4	36.1	40.3	16.9
PPO	90	521.5	1.00×	7.8	0.8	20.8	4.0	6.4	34.3	42.7	16.7
↪ + SPEC-RL	90	210.6	2.01×	9.2	1.4	20.2	5.5	5.0	35.3	40.7	16.8

Table 22: Intermediate training results of **LLaMA-3.2-1B-Instruct** on DeepMath-6K with **PPO** and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with PPO and its SPEC-RL variant interleaved.

Algorithm	Step	Rollout Efficiency		Math Reasoning					OOD		AVG
		Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	-	-	45.2	9.1	68.6	28.3	33.5	68.3	43.6	42.4
PPO	10	65.2	1.00×	55.3	14.1	78.6	38.2	42.2	70.6	41.4	48.6
↪ + SPEC-RL	10	50.0	1.28×	55.9	13.1	78.2	36.0	41.9	70.0	42.7	48.3
PPO	20	126.8	1.00×	60.5	15.3	79.0	40.4	45.9	69.6	44.2	50.7
↪ + SPEC-RL	20	77.1	1.58×	60.0	14.2	80.0	38.6	44.7	70.2	47.3	50.7
PPO	30	188.0	1.00×	58.4	15.1	79.8	40.4	45.3	72.2	47.0	51.2
↪ + SPEC-RL	30	97.9	1.78×	61.1	17.4	82.0	39.7	45.5	72.4	48.6	52.4
PPO	40	249.6	1.00×	63.0	17.6	83.6	40.4	45.6	73.0	47.9	53.0
↪ + SPEC-RL	40	126.4	1.81×	69.2	23.5	86.2	43.0	49.5	74.8	47.7	56.3
PPO	50	312.8	1.00×	63.6	17.4	84.4	41.9	48.0	77.1	47.7	54.3
↪ + SPEC-RL	50	164.0	1.75×	66.2	24.4	84.0	43.4	51.3	76.9	47.5	56.2
PPO	60	378.2	1.00×	66.6	18.2	84.8	44.1	47.6	81.2	48.4	55.8
↪ + SPEC-RL	60	197.0	1.75×	69.4	23.5	85.8	47.1	49.5	78.4	45.5	57.0
PPO	70	445.0	1.00×	64.5	18.2	85.6	41.2	48.6	84.1	47.5	55.7
↪ + SPEC-RL	70	236.1	1.71×	71.6	27.0	87.0	43.4	53.3	80.6	46.2	58.4
PPO	80	511.7	1.00×	65.3	18.1	87.0	44.9	51.1	85.0	48.8	57.2
↪ + SPEC-RL	80	272.4	1.69×	72.8	27.9	88.6	44.5	52.3	83.6	48.1	59.7
PPO	90	579.2	1.00×	67.5	22.1	86.6	41.9	50.2	87.1	49.5	57.8
↪ + SPEC-RL	90	310.1	1.66×	75.5	29.1	86.2	41.9	53.6	82.3	47.1	59.4

Table 23: Intermediate training results of **Qwen-3-14B-Base** on DeepMath-6K with **PPO** and SPEC-RL. We report rollout efficiency and accuracy every 10 training steps, with PPO and its SPEC-RL variant interleaved.

Algorithm	Step	Gen-Step	Rollout Efficiency		Math Reasoning					OOD		AVG
			Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	0	-	-	21.9	2.5	45.0	12.5	16.7	39.3	17.9	22.3
DAPO	5	10	65.2	1.00×	28.6	3.4	50.0	17.6	20.0	41.5	20.1	25.9
↪ + SPEC-RL	5	10	46.7	1.25×	24.2	3.1	53.4	20.2	21.0	43.0	17.7	26.1
DAPO	10	20	127.4	1.00×	27.3	4.1	58.4	19.1	22.4	42.9	19.6	27.7
↪ + SPEC-RL	10	20	70.6	1.52×	31.4	5.0	56.8	16.5	20.7	44.1	21.6	28.0
DAPO	15	30	187.7	1.00×	29.5	4.8	56.4	19.9	22.7	45.8	20.7	28.5
↪ + SPEC-RL	15	30	95.5	1.60×	32.2	4.5	58.4	23.9	24.9	45.0	26.1	30.7
DAPO	20	40	247.6	1.00×	34.1	4.5	54.8	23.2	21.8	46.8	23.3	29.8
↪ + SPEC-RL	20	40	109.5	1.74×	35.0	5.1	57.2	25.0	24.9	47.7	24.6	31.4
DAPO	25	50	307.5	1.00×	32.0	5.0	59.4	20.6	25.9	47.2	19.4	29.9
↪ + SPEC-RL	25	50	124.9	1.84×	33.9	5.0	59.0	22.8	23.0	49.2	25.7	31.2
DAPO	30	60	367.0	1.00×	31.2	5.4	60.6	25.0	24.3	48.0	22.2	31.0
↪ + SPEC-RL	30	60	137.7	1.94×	30.9	6.1	60.4	24.6	25.0	50.4	26.2	31.9
DAPO	35	70	425.6	1.00×	34.1	5.0	59.8	27.9	24.3	49.7	22.7	31.9
↪ + SPEC-RL	35	70	149.1	2.02×	33.9	5.6	62.2	25.0	25.3	51.6	25.0	32.7
DAPO	40	80	484.6	1.00×	35.5	4.9	61.6	24.6	25.0	50.8	22.7	32.2
↪ + SPEC-RL	40	80	160.2	2.10×	34.7	5.3	60.2	25.4	26.5	53.7	27.4	33.3
DAPO	45	90	543.1	1.00×	33.1	4.9	60.8	24.6	23.0	52.2	24.8	31.9
↪ + SPEC-RL	45	90	171.6	2.17×	33.1	4.9	60.0	25.7	25.5	53.5	27.0	32.8

Table 24: Intermediate training results of **Qwen-3-1.7B-Base** on DeepMath-6K with **DAPO** and SPEC-RL. Since DAPO adopts *Dynamic Sampling*, one training step may correspond to multiple generation steps; thus we additionally report the **Gen-Step** column to indicate how many rollout batches the model has consumed.

Algorithm	Step	Gen-Step	Rollout Efficiency		Math Reasoning					OOD		AVG
			Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	0	-	-	40.2	11.5	67.4	27.2	34.1	60.4	29.9	38.7
DAPO	5	10	75.0	1.00×	49.7	12.9	73.6	27.9	39.1	60.5	32.0	42.2
↪ + SPEC-RL	5	10	59.0	1.20×	51.1	13.3	75.0	32.7	38.8	63.3	34.0	44.0
DAPO	10	20	148.8	1.00×	58.0	17.1	78.6	36.8	43.4	64.0	36.0	47.7
↪ + SPEC-RL	10	20	90.3	1.45×	56.1	14.5	79.0	37.1	40.7	63.0	34.2	46.4
DAPO	15	30	235.5	1.00×	61.4	19.0	80.6	39.7	47.4	70.6	38.1	51.0
↪ + SPEC-RL	15	30	116.8	1.73×	61.4	21.8	81.8	42.6	47.9	69.6	39.0	52.0
DAPO	20	40	354.9	1.00×	65.9	23.0	84.6	41.2	46.1	77.5	36.4	53.5
↪ + SPEC-RL	20	40	152.2	2.00×	63.4	24.7	83.8	39.0	49.8	75.4	38.1	53.5
DAPO	25	51	509.1	1.00×	64.7	22.2	83.4	39.3	49.6	79.8	38.8	54.0
↪ + SPEC-RL	25	50	199.8	2.19×	64.4	22.9	85.4	41.5	47.9	78.7	42.1	54.7
DAPO	30	63	685.0	1.00×	63.9	25.4	83.8	44.5	48.9	81.1	39.2	55.3
↪ + SPEC-RL	30	60	239.9	2.48×	67.8	24.5	84.0	39.7	48.7	80.2	40.3	55.0
DAPO	35	75	867.6	1.00×	68.8	22.7	82.8	40.8	49.5	81.9	38.1	54.9
↪ + SPEC-RL	35	70	278.9	2.73×	68.6	25.5	84.8	40.4	49.9	80.8	44.4	56.3
DAPO	40	87	1052.2	1.00×	65.2	24.0	84.8	40.1	48.6	82.4	39.6	55.0
↪ + SPEC-RL	40	82	326.2	2.88×	69.7	26.4	84.4	43.8	50.4	82.2	44.4	57.3

Table 25: Intermediate training results of **Qwen-3-8B-Base** on DeepMath-6K with **DAPO** and SPEC-RL. Since DAPO adopts *Dynamic Sampling*, one training step may correspond to multiple generation steps; thus we additionally report the **Gen-Step** column to indicate how many rollout batches the model has consumed.

Algorithm	Step	Gen-Step	Rollout Efficiency		Math Reasoning					OOD		AVG
			Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	0	-	-	7.5	0.6	14.2	4.0	2.8	32.6	37.0	14.1
DAPO	5	15	105.6	1.00×	5.3	0.5	14.4	2.9	3.3	32.6	38.8	14.0
↪ + SPEC-RL	5	15	52.4	1.95×	5.8	1.1	14.4	2.6	3.0	32.7	38.3	14.0
DAPO	10	27	179.8	1.00×	6.1	0.6	14.6	2.6	4.1	34.8	40.3	14.7
↪ + SPEC-RL	10	28	69.2	2.16×	7.2	1.1	16.6	2.6	4.9	33.7	39.2	15.0
DAPO	15	38	239.8	1.00×	8.1	0.9	18.4	4.4	4.4	33.9	37.3	15.3
↪ + SPEC-RL	15	39	79.2	2.19×	9.8	0.9	16.4	6.2	3.4	33.9	37.5	15.4
DAPO	20	53	322.1	1.00×	8.4	0.6	18.8	2.9	5.6	34.6	38.8	15.7
↪ + SPEC-RL	20	53	92.4	2.31×	11.1	1.7	19.6	5.9	5.5	34.2	38.1	16.6
DAPO	25	68	402.9	1.00×	10.0	1.0	19.8	3.3	4.6	34.5	38.3	15.9
↪ + SPEC-RL	25	68	105.3	2.43×	10.9	2.1	19.8	4.0	6.1	35.5	35.5	16.3
DAPO	30	83	482.6	1.00×	9.2	0.9	19.2	4.0	5.5	33.0	38.6	15.8
↪ + SPEC-RL	30	83	123.1	2.48×	11.6	2.0	20.2	4.0	5.5	35.5	38.4	16.7

Table 26: Intermediate training results of **LLaMA-3.2-1B-Instruct** on DeepMath-6K with **DAPO** and SPEC-RL. Since DAPO adopts *Dynamic Sampling*, one training step may correspond to multiple generation steps; thus we additionally report the **Gen-Step** column to indicate how many rollout batches the model has consumed.

Algorithm	Step	Gen-Step	Rollout Efficiency		Math Reasoning					OOD		AVG
			Tokens (M)	Speedup	AMC23	AIME24	MATH 500	Minerva Math	Olympiad Bench	MMLU STEM	IFEval	
Base Model	0	0	-	-	45.2	9.1	68.6	28.3	33.5	68.3	43.6	42.4
DAPO	5	10	68.2	1.00×	55.0	13.0	77.4	39.0	40.4	69.0	44.9	48.4
↪ + SPEC-RL	5	10	51.6	1.28×	55.9	13.5	76.8	37.1	39.3	69.5	41.6	47.7
DAPO	10	20	136.1	1.00×	58.6	13.1	81.2	38.2	43.7	68.8	44.4	49.7
↪ + SPEC-RL	10	20	82.6	1.52×	59.1	14.2	79.4	39.0	44.0	67.3	46.0	49.9
DAPO	15	31	212.0	1.00×	58.3	14.1	81.0	37.9	44.9	69.6	46.2	50.3
↪ + SPEC-RL	15	30	109.0	1.78×	62.5	14.7	81.4	42.3	45.2	67.9	50.1	52.0
DAPO	20	44	303.9	1.00×	59.4	14.4	81.6	38.2	44.3	69.9	47.3	50.7
↪ + SPEC-RL	20	40	129.4	2.11×	62.3	15.6	81.2	40.8	42.7	69.4	50.6	51.8
DAPO	25	59	412.7	1.00×	60.3	15.0	82.0	40.8	44.6	71.6	45.5	51.4
↪ + SPEC-RL	25	53	157.1	2.33×	61.1	16.0	82.8	42.3	45.3	70.0	48.4	52.3
DAPO	30	74	525.3	1.00×	60.0	16.0	84.0	42.6	44.9	72.9	48.1	52.6
↪ + SPEC-RL	30	68	199.8	2.40×	64.4	19.3	83.6	43.4	48.1	70.7	50.3	54.3
DAPO	35	89	641.1	1.00×	62.7	16.4	84.6	41.9	45.2	72.9	49.9	53.4
↪ + SPEC-RL	35	83	242.8	2.46×	65.6	19.9	85.2	42.6	48.4	72.4	51.4	55.1

Table 27: Intermediate training results of **Qwen-3-14B-Base** on DeepMath-6K with **DAPO** and **SPEC-RL**. Since DAPO adopts *Dynamic Sampling*, one training step may correspond to multiple generation steps; thus we additionally report the **Gen-Step** column to indicate how many rollout batches the model has consumed.

Prompt

system
You are a helpful assistant.
user
Kim's birthday was 200 days ago. Today is Wednesday. On what day of the week did his birthday fall?
Please reason step by step, and put your final answer within \boxed{}.
assistant

Old Rollout from Epoch$_{t-1}$

To determine the day of the week Kim's birthday fell on, we can follow these steps:

1. Identify the current day and the number of days since Kim's birthday.
   - Today is Wednesday.
   - Kim's birthday was 200 days ago.

2. Determine the day of the week 200 days ago from Wednesday.
   - We need to find the remainder when 200 is divided by 7, because there are 7 days in a week. This will tell us how many days forward or backward from Wednesday we need to count.
   - Calculate $200 \div 7$:

$$200 \div 7 = 28 \text{ rem}ainder\ 4$$

This means that 200 days is equivalent to 28 weeks and 4 days. So, 200 days ago from Wednesday is the same as 4 days before Wednesday.

3. Count backward 4 days from Wednesday.
   - Monday is 1 day before Wednesday.
   - Sunday is 2 days before Wednesday.
   - Saturday is 3 days before Wednesday.
   - Friday is 4 days before Wednesday.

Therefore, 200 days ago from Wednesday was a Friday.
The final answer is **\boxed{Friday}**.

New Rollout from Epoch$_{t}$

To determine the day of the week Kim's birthday fell on, we can follow these steps:

1. Identify the current day and the number of days since Kim's birthday.
   - Today is Wednesday.
   - Kim's birthday was 200 days ago.

2. Determine the day of the week 200 days ago from Wednesday.
   - We need to find the remainder when 200 is divided by 7, because there are 7 days in a week. This will tell us how many days forward or backward from Wednesday we need to count.
   - Calculate the remainder:

$$200 \div 7 = 28 \text{ rem}ainder\ 4$$

   - This means that 200 days ago was 4 days before Wednesday.

3. Count backwards from Wednesday by 4 days.
   - Wednesday - 1 day = Tuesday
   - Tuesday - 1 day = Monday
   - Monday - 1 day = Sunday
   - Sunday - 1 day = Saturday

Therefore, Kim's birthday fell on a **\boxed{\text{Saturday}}**.

Figure 12: Case study comparing rollouts from previous and current training steps. The prompt denotes the model input. The old rollout and new rollout are generated by the respective model from corresponding epochs. Tokens highlighted in yellow indicate the verified speculative prefix. The red box marks incorrect reasoning steps, whereas the green box highlights correct reasoning steps.

Prompt

system
You are a helpful assistant.
user
The arithmetic mean of these five expressions is 24. What is the value of x?

$$x+8 \qquad 15 \qquad 2x \qquad 13 \qquad 2x+4$$

Please reason step by step, and put your final answer within \boxed{}.
assistant

Old Rollout from Epoch$_{t-1}$

To find the value of x, we start by understanding the arithmetic mean of the given expressions and setting it equal to 24. The expressions are:
1. $x+8$
2. 15
3. $2x$
4. 13
5. $2x+4$

The arithmetic mean of these expressions is given by:

$$\frac{(x+8)+15+2x+13+(2x+4)}{5}=24$$

First, we simplify and sum the expressions in the numerator:

$$\frac{(x+8)+15+2x+13+(2x+4)}{5}$$
$$=\frac{x+8+15+2x+13+2x+4}{5}$$
$$=\frac{5x+30}{5}$$

Setting this equal to 24, we get:

$$\frac{5x+30}{5}=24$$

Next, we eliminate the fraction by multiplying both sides of the equation by 5:

$$5x+30=120$$

Then, we isolate x by subtracting 30 from both sides:

$$5x=90$$

Finally, we solve for x by dividing both sides by 5:

$$x=\frac{90}{5}=18$$

So, the value of x is **\boxed{18}**.

New Rollout from Epoch$_t$

To find the value of x, we start by understanding the arithmetic mean of the given expressions and setting it equal to 24. The expressions are:
1. $x+8$
2. 15
3. $2x$
4. 13
5. $2x+4$

The arithmetic mean of these expressions is given by:

$$\frac{(x+8)+15+2x+13+(2x+4)}{5}=24$$

First, we simplify the numerator:

$$(x+8)+15+2x+13+(2x+4)$$
$$=x+8+15+2x+13+2x+4$$
$$=5x+40$$

So the equation becomes:

$$\frac{5x+40}{5}=24$$

Next, we eliminate the denominator by multiplying both sides of the equation by 5:

$$5x+40=120$$

Then, we solve for x by isolating it on one side of the equation. First, subtract 40 from both sides:

$$5x=80$$

Finally, divide both sides by 5:

$$x=16$$

Thus, the value of x is **\boxed{16}**.

Figure 13: Case study comparing rollouts from previous and current training steps. The prompt denotes the model input. The old rollout and new rollout are generated by the respective model from corresponding epochs. Tokens highlighted in yellow indicate the verified prefix. The red box marks incorrect reasoning steps, whereas the green box highlights correct reasoning steps.

Prompt

system
You are a helpful assistant.
user
Find the units digit of 9^{8^7}.
Please reason step by step, and put your final answer within \boxed{}.
assistant

Old Rollout from Epoch$_{t-1}$

To find the units digit of 9^{8^7}, we need to observe the pattern in the units digit of powers of 9.

Let's start by computing the first few powers of 9 and noting their units digits:

$$9^1 = 9\ (units\ digit\ is\ 9)$$
$$9^2 = 81\ (units\ digit\ is\ 1)$$
$$9^3 = 729\ (units\ digit\ is\ 9)$$
$$9^4 = 6561\ (units\ digit\ is\ 1)$$

From the above calculations, we see that the units digits repeat every 2 powers: 9, 1, 9, 1, ...

So, the units digit of 9^n is 9 if n is odd, and 1 if n is even.

Now, we need to determine if 8^7 is odd or even.

Since 8^7 is an odd power of 8, it will be odd as well (because any positive power of an even number is even).

Therefore, the units digit of 9^{8^7} will be the same as the units digit of 9^1, which is 9.

Thus, the units digit of 9^{8^7} is **\boxed{9}**.

New Rollout from Epoch$_t$

To find the units digit of 9^{8^7}, we need to observe the pattern in the units digit of powers of 9.

Let's start by computing the first few powers of 9 and noting their units digits:

$$9^1 = 9\ (units\ digit\ is\ 9)$$
$$9^2 = 81\ (units\ digit\ is\ 1)$$
$$9^3 = 729\ (units\ digit\ is\ 9)$$
$$9^4 = 6561\ (units\ digit\ is\ 1)$$

From the above, we can see that the units digits repeat every 2 powers: 9, 1, 9, 1, ...

So, the units digit of 9^n is 9 if n is odd, and 1 if n is even.

Next, we need to determine if 8^7 is odd or even. Since 8 is even, any power of 8 will also be even. Therefore, 8^7 is an even number.

Since 8^7 is even, the units digit of 9^{8^7} will be 1.

Thus, the units digit of 9^{8^7} is **\boxed{1}**.

Figure 14: Case study comparing rollouts from previous and current training steps. The prompt denotes the model input. The old rollout and new rollout are generated by the respective model from corresponding epochs. Tokens highlighted in yellow indicate the verified speculative prefix. The red box marks incorrect reasoning steps, whereas the green box highlights correct reasoning steps.

Prompt

system
You are a helpful assistant.
user
If $-3 \leq x + 5 \leq 8$ and x is an integer, what is the sum of all the possible solutions?
Please reason step by step, and put your final answer within \boxed{}.
assistant

Old Rollout from Epoch$_{t-1}$

To solve the given inequality $-3 \leq x + 5 \leq 8$, we need to isolate x on one side of the inequality. We can do this by subtracting 5 from all parts of the inequality:

$$-3 - 5 \leq x + 5 - 5 \leq 8 - 5$$

Simplifying each part:

$$-8 \leq x \leq 3$$

Since x is an integer, the possible values for x are:

$$\{-8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3\}$$

Next, we sum these possible values:

$-8 + (-7) + (-6) + (-5) + (-4) + (-3) + (-2) + (-1) + 0 + 1 + 2 + 3$

We can group the positive and negative terms:

$(-8 + 3) + (-7 + 2) + (-6 + 1) + (-5 + 4) - 10 = -5 - 5 - 5 - 10 = -25$

Thus, the sum of all the possible solutions is:
\boxed{-25}.

New Rollout from Epoch$_{t}$

To solve the given inequality $-3 \leq x + 5 \leq 8$, we need to isolate x on one side of the inequality. We can do this by subtracting 5 from all parts of the inequality:

$$-3 - 5 \leq x + 5 - 5 \leq 8 - 5$$

Simplifying each part:

$$-8 \leq x \leq 3$$

Since x is an integer, the possible values for x are:

$$\{-8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3\}$$

Next, we need to find the sum of all these possible values of x. We can sum them directly:

$-8 + (-7) + (-6) + (-5) + (-4) + (-3) + (-2) + (-1) + 0 + 1 + 2 + 3$

We can simplify this by grouping the terms to make the addition easier:

$(-8 + 3) + (-7 + 2) + (-6 + 1) + (-5 + 0) + (-4 + (-2)) + (-3 + (-1)) = -5 + (-5) + (-5) + (-5) + (-6) + (-4) = -30$

Thus, the sum of all the possible solutions is:
\boxed{-30}.

Figure 15: Case study comparing rollouts from previous and current training steps. The prompt denotes the model input. The old rollout and new rollout are generated by the respective model from corresponding epochs. Tokens highlighted in yellow indicate the verified speculative prefix. The red box marks incorrect reasoning steps, whereas the green box highlights correct reasoning steps.