SPINNINGRL: EFFICIENT RLVR VIA THEORETICAL REWARD GAIN MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) has proven effective in enhancing the reasoning capabilities of Large Language Models (LLMs), yet it remains painfully inefficient. Uniform sampling wastes computation on non-informative prompts and aggravates the zero-advantage problem in GRPO-like methods. Prior schedulers often select medium-success-rate prompts via learnability-style metrics (e.g., $p(1-p)$ or $|p-0.5|$), which optimize for *gradient magnitude* rather than *policy improvement*. We instead propose a more direct metric: the KL-constrained *potential reward gain* $\widehat{G}$ that is estimable from existing rollouts, and introduce SPINNINGRL, a non-stationary bandit framework that continuously prioritizes prompts with the highest expected gain under the current policy. Concentrating sampling in this subspace improves sample efficiency and reduces baseline variance, mitigating zero-advantage issues. Experiments show consistent gains on mathematics and logical reasoning under the same runtime budget, with 17%–50% fewer rollouts than uniform sampling at matched performance. The code is anonymously available at this URL.

1 INTRODUCTION

Reinforcement Learning with Verifiable Rewards (RLVR) has achieved remarkable success in improving the reasoning capabilities of Large Language Models (LLMs) (Guo et al., 2025; OpenAI et al., 2026; Team et al., 2025) , yet it remains highly inefficient (Sheng et al., 2025b; Zheng et al., 2025). The primary computational bottleneck is the rollout phase, which typically consumes 50%–80% of the total training time (Sheng et al., 2025b; Zheng et al., 2025; He et al., 2025a; Qu et al., 2025). Typically, during the rollout phase, RL algorithms uniformly sample questions from the entire training set and generate responses for each question.

This uniform sampling strategy is inherently inefficient: question difficulty in the dataset is unevenly distributed, while the policy's capability evolves dynamically throughout training. Consequently, a large portion of computation is wasted on questions that are either too easy or too difficult for the current policy, yielding minimal learning signals (Zheng et al., 2025; Qu et al., 2025). Group-based RLVR methods such as GRPO (Shao et al., 2024) compute advantages based on relative rewards. These trivial or unsolvable questions therefore contribute zero gradient and amplify computational waste (Yu et al., 2025; Le et al., 2026; Feng et al., 2025).

To alleviate this issue and improve the sample efficiency of GRPO-style algorithms, prior works mainly fall into four categories, each with notable limitations: **(1) Offline Filtering**: pre-filters the dataset before RL (An et al., 2025; He et al., 2025b; Li et al., 2025), but ignores the evolving policy and may discard valuable samples. **(2) Dynamic Sampling / Online Filtering**: retains non-zero-advantage questions during training (Yu et al., 2025; Bae et al., 2026), yet often increases rollout cost despite asynchronous generation. **(3) Learnability-based Sampling**: selects prompts by medium success rate under binary rewards (e.g., $p(1-p)$, $|p-0.5|$, or pass-rate bands), including MoPPS(Qu et al., 2025), LILO(Foster et al., 2026), LearnAlign(Li et al., 2026), Bae et al.(Bae et al., 2026), as well as predictive-based variants such as DPS(Mao et al., 2026) and GPS(Qu et al., 2026). **(4) Sequential Sampling**: multi-stage schedules such as Speed-RL(Zhang et al., 2026) adapt budgets on the fly, but add latency and GPU idle time.

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

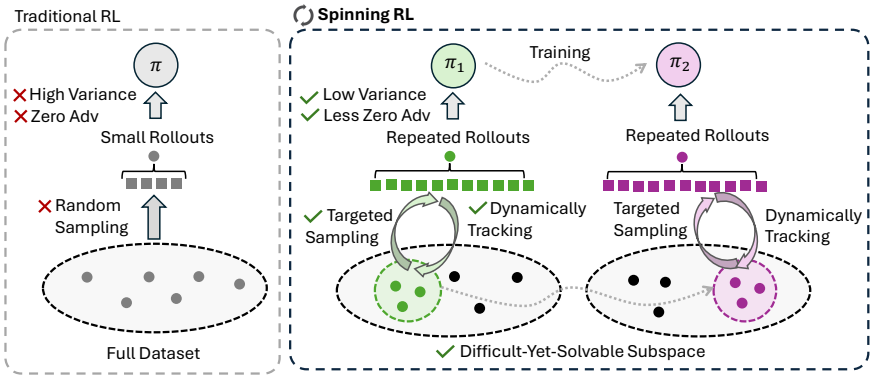


Figure 1: Overview of SPINNINGRL sampling framework: Potential reward gain modeling estimates the utility of each question; SW-UCB uses this utility to dynamically allocate rollout budget under non-stationarity; repeated sampling of high-utility questions then yields more reliable reward statistics, enabling lower-variance baseline estimation.

The limitations of existing approaches stem from two fundamental challenges: **(C1)** how to define a *principled* prompt-value metric for scheduling that generalizes beyond binary success-rate heuristics, and **(C2)** how to sample under a non-stationary, evolving policy.

Motivated by these challenges, we introduce SPINNINGRL, a sample-efficient RLVR framework that combines KL-based theoretical reward gain estimation, sliding-window UCB scheduling, and more accurate baseline estimation for revisited samples. Our contributions include:

- A theoretically grounded **selector metric**—the KL-constrained potential reward gain $\widehat{G}$—derived from a policy-improvement objective. It recovers conventional learnability $p(1-p)$ in the weak-update regime and favors difficult-yet-solvable prompts under finite KL budgets, addressing **(C1)**.
- A lightweight and adaptive **SW-UCB sampling strategy** that formulates prompt selection as a non-stationary multi-armed bandit problem, addressing **(C2)** by continuously prioritizing high-value questions for the evolving policy under a conditional dynamic-regret guarantee, and transferring across models and tasks with a unified hyperparameter setting.
- An **efficient RLVR training paradigm** that consistently improves mathematics, logical reasoning and coding task performance under the same runtime budget, while reducing rollout counts by 17% to 50% compared with uniform sampling at matched performance levels.

2 METHOD

2.1 THEORETICAL REWARD GAIN MODELING

KL-Constrained Improvement and Reward Gain. Let π_θ be the current policy and D the prompt distribution. The expected reward is $J(\pi_\theta) = \mathbb{E}_{q\sim D,\, o\sim\pi_\theta(\cdot|q)}[r(q,o)]$. Under a KL budget δ that limits deviation from π_θ, the optimal policy is the one that maximizes expected reward:

$$\pi_{\theta^*} = \arg\max_{\pi}\ J(\pi) \quad \text{s.t.} \quad \mathbb{E}_{q\sim D}\Big[\mathrm{KL}\big(\pi(\cdot\mid q)\,\|\,\pi_\theta(\cdot\mid q)\big)\Big] \le \delta, \tag{1}$$

Using Lagrange multiplier, the optimal update admits a Boltzmann tilt (Peters & Schaal, 2007):

$$\pi_{\theta^*}(o\mid q) = \frac{1}{Z(q)}\,\pi_\theta(o\mid q)\,\exp\Big(\frac{r(q,o)}{\beta}\Big), \tag{2}$$

where $Z(q) = \mathbb{E}_{o\sim\pi_\theta(\cdot|q)}\big[\exp(r(q,o)/\beta)\big]$ is the normalizing partition function and temperature $\beta > 0$ is determined by δ. We define the *reward gain* of a question as the expected improvement after applying this update,

$$G(q_i) = \mathbb{E}_{o\sim\pi_{\theta^*}}[r(q_i,o)] - \mathbb{E}_{o\sim\pi_\theta}[r(q_i,o)]. \tag{3}$$

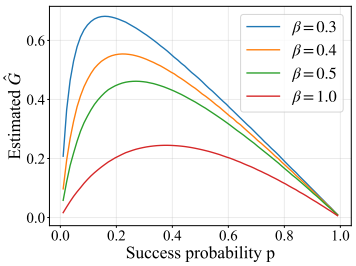


Figure 2: **Theoretical reward gain** distribution under binary reward setting.

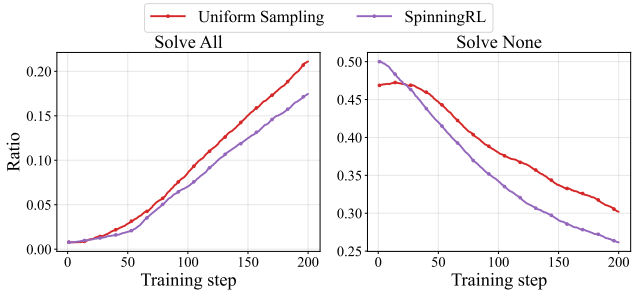


Figure 3: High **zero advantage ratio** in training Qwen3-1.7B, Math. SPINNINGRL addresses the issue by sampling on gain-efficient region.

	Learnability-based	SpinningRL
Optimization objective	$\max \|\nabla_\theta J\|$ i.e. **gradient magnitude**	$\max \big(J(\pi_{\theta^*}) - J(\pi_\theta)\big)$ i.e. **policy improvement**
Sampling region	**medium-difficulty** prompts	**difficult-yet-solvable** prompts
Scheduling metric	$\lvert p-0.5\rvert$ (MoPPS); $T_{\text{low}} \le p \le T_{\text{high}}$ (SPEED); $p(1-p)$ (LILO, LearnAlign)	$\widehat{G}(q)$ (Eq. 4)

Table 1: Learnability-based scheduling v.s. SpinningRL.

Practical Gain Signal for Scheduling. Given N on-policy rollouts $\{o_j, r_{i,j}\}_{j=1}^N$ for q_i, we adopt a Monte Carlo approximation that assigns equal probability $1/N$ to each sampled rollout as the empirical support of Eq. 2. The tilted weights are then reduced to a reward softmax:

$$w_j^* = \frac{\exp(r_{i,j}/\beta)}{\sum_{\ell=1}^N \exp(r_{i,\ell}/\beta)}, \qquad \bar{r}_i = \frac{1}{N}\sum_{j=1}^N r_{i,j},$$

The reward gain in Eq. 3 can be approximated by a practical scheduling signal,

$$\widehat{G}(q_i) = \sum_{j=1}^N w_j^* \, r_{i,j} - \bar{r}_i = \sum_{j=1}^N \Big(w_j^* - \tfrac{1}{N}\Big) r_{i,j}. \tag{4}$$

It is reasonable to assume that even without an explicit KL term in the training objective, each parameter update is sufficiently localized (e.g., due to ratio clipping and limited step sizes) that the updated policy remains in a local trust region of π_θ. Under this assumption, $\hat{G}(q)$ quantifies the expected reward improvement attainable by a constrained local update, and provides a principled criterion for allocating rollout budget across prompts.

2.2 INTERPRETATIONS OF THE GAIN MODEL

Comparison with Learnability-based Methods. Prior sampling methods, regardless of specific mechanisms, mostly prefer prompts of *medium difficulty*. Representative methods include MoPPS (Qu et al., 2025) (rank prompts by $\min |p - 0.5|$), Speed-RL (Zhang et al., 2026) (extra rollouts on groups with $T_{\text{low}} \le p \le T_{\text{high}}$), as well as LILO (Foster et al., 2026), LearnAlign (Li et al., 2026) and Bae et al. (2026). These methods typically use the success-rate variance $p(1-p)$ as a *"learnability"* signal for scheduling. Under binary rewards $r \in \{0, 1\}$, this choice is motivated by the score-function gradient factorization for prompt ξ:

$$\nabla_\theta J(\theta;\xi) = p(1-p)\,\Big(\mathbb{E}\big[\nabla_\theta \log \pi_\theta(o \mid \xi) \mid r{=}1\big] - \mathbb{E}\big[\nabla_\theta \log \pi_\theta(o \mid \xi) \mid r{=}0\big]\Big) = p(1-p)\,\mathbf{d}(\xi,\theta), \tag{5}$$

Therefore, under binary rewards, $p(1-p)$ provides a prompt-level proxy for gradient magnitude $||\nabla_\theta J(\theta;\xi)||$, up to the prompt-dependent direction factor $\mathbf{d}(\xi,\theta)$.

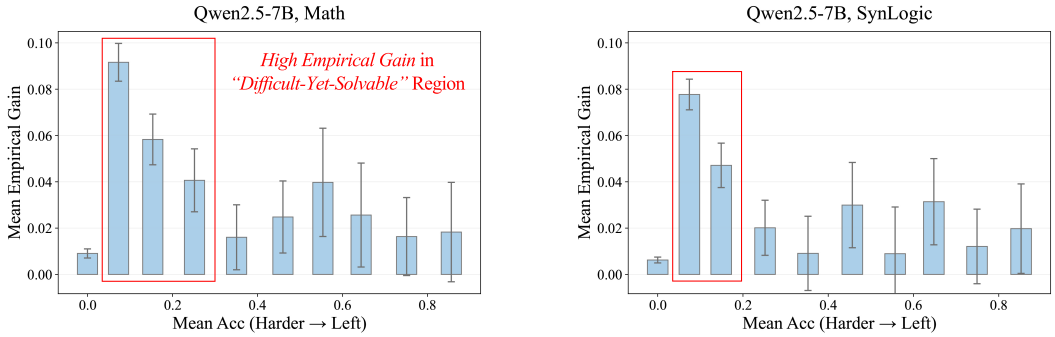


Figure 4: **Future reward growth by prompt difficulty** over 30 additional training steps. Same number of samples are drawn from each difficulty range; horizontal lines denote 95% confidence intervals. **"Difficult-Yet-Solvable"** problems get the highest improvement.

Table 1 summarizes the distinction between the two approaches. Since SPINNINGRL's optimization objective is directly aligned with local-constrained policy improvement, it can be viewed as a **reward-space generalization** of variance-based learnability. We formalize this connection below: $G_\beta(p) \approx p(1-p)/\beta$ in the weak-update regime, while finite KL budgets induce an asymmetric preference for difficult-yet-solvable prompts. We conduct comprehensive ablation studies in Sec. 3.3.

Binary Reward Closed Form. Under binary rewards $r \sim \text{Bernoulli}(p)$ and the KL tilt in Eq. 2, let $a = e^{1/\beta}$. The improved success probability is $p^* = pa/(1-p+pa)$, hence the exact reward gain admits the closed form

$$G_\beta(p) = p^* - p = \frac{p(1-p)\,(e^{1/\beta}-1)}{1-p+p\,e^{1/\beta}}. \tag{6}$$

Maximizing over $p \in (0,1)$ yields the unique maximizer

$$p^*_\beta = \frac{1}{1+e^{1/(2\beta)}} \in (0, 1/2), \tag{7}$$

with $p^*_\beta \to 1/2$ as $\beta \to \infty$ and $p^*_\beta \to 0$ as $\beta \to 0$. Thus β controls *curriculum aggressiveness*: large β and corresponding small KL budget recovers near-medium difficulty, whereas small β shifts preference toward low-success but still solvable prompts, as shown in Figure 2.

Connection to Learnability. In the weak-update regime $\beta \to \infty$, $e^{1/\beta} - 1 = 1/\beta + O(\beta^{-2})$, so

$$G_\beta(p) = \frac{p(1-p)}{\beta} + O(\beta^{-2}) = \frac{\text{Var}(r)}{\beta} + O(\beta^{-2}). \tag{8}$$

Hence $p(1-p)$ is a *local approximation* of G_β, meaning that learnability-based scheduling is a special case of SPINNINGRL under infinitesimal updates, whereas finite KL budgets extend the criterion asymmetrically toward *"difficult-yet-solvable"* prompts. A detailed derivation of Eqs. 6–8 is in Appendix A.

Empirical Gain Distribution. To empirically validate the gain of prompts, we periodically sample 1,000 recent instances every 100 training steps and measure their rewards after an additional 30 steps of training. We then define the average reward increase over this interval as the *empirical gain* of each sample, which serves as a diagnostic of the relationship between prompt difficulty and learning progress. As shown in Figure 4, "difficult-yet-solvable" problems consistently exhibit the largest gain, while the gain decreases as the samples become easier. By contrast, samples that are initially beyond the model's capability remain largely unapproachable after further training. We further assess the predictive validity of $\widehat{G}_\beta$ against learnability-based scores in Appendix L.

2.3 ADAPTIVE SAMPLING AS NON-STATIONARY BANDITS

The estimated potential reward gain in Eq. 4 is highly non-uniform across questions and evolves as the policy model gets updated. As shown in Figure 3, when training Qwen3-1.7B on DAPO-Math-

17k, after 200 steps, approximately 50% of randomly sampled questions are either too difficult or too easy for the current policy, implying negligible gain and limited utility for further sampling.

Bandit Formulation. We cast adaptive sampling as a non-stationary bandit over prompts, with latent means given by the time-varying gains $G_t(q_i)$ and empirical feedback $\widehat{G}_t(q_i)$. Under a bounded-variation abstraction of $\{G_t\}$, this bandit formulation admits a *conditional dynamic-regret bound*. We schedule with Sliding-Window UCB (SW-UCB) (Garivier & Moulines, 2011), a non-stationary variant of UCB1 (Auer et al., 2002) that estimates each arm from observations in a recent window of size W. The formal setup, assumptions, and an asymptotic upper bound on the conditional dynamic regret are given in Appendix B.

UCB Score Composition. Let $N_t(q_i)$ denote the number of times arm q_i has been selected within the current window W. The SW-UCB score of arm q_i at time t is defined as

$$\mathrm{UCB}_t(q_i) = \hat{G}_t(q_i) + \lambda \sqrt{\frac{2 \log W}{N_t(q_i) + \epsilon}}.$$

The first term is the *exploitation term*, which captures the estimated reward gain of the prompt. The second term is the *exploration term*, which models uncertainty through the selection count and assigns a larger bonus to under-sampled prompts. The hyperparameter λ controls the exploration–exploitation trade-off, and $\epsilon > 0$ prevents numerical instability.

2.4 Baseline Estimation with Lower Variance

GRPO estimates the baseline for each sample within a single update step, computing the advantage as $(r - \mu_t)/(\sigma_t + \epsilon)$, where μ_t and σ_t denote the mean and standard deviation of grouped rewards at step t. However, since the total number of rollouts per step is limited (e.g., 8 or 16), such one-shot estimation may have high variance and produce zero advantages which leads to wasted computation.

History-Aware Baseline. Under SW-UCB, samples with high potential gain may be revisited within a recent window. Whenever q is sampled at step t, we form the step-wise second moment

$$M_{2,t}(q) = \sigma_t^2(q) + \mu_t^2(q),$$

and update a running baseline by exponential smoothing:

$$\begin{aligned} \bar{\mu}_t(q) &= (1 - \alpha)\, \bar{\mu}_{t-1}(q) + \alpha\, \mu_t(q), \\ \bar{M}_{2,t}(q) &= (1 - \alpha)\, \bar{M}_{2,t-1}(q) + \alpha\, M_{2,t}(q), \\ \bar{\sigma}_t^2(q) &= \bar{M}_{2,t}(q) - \bar{\mu}_t(q)^2. \end{aligned}$$

Rewards are normalized by $(r - \bar{\mu}_t(q))/(\bar{\sigma}_t(q) + \epsilon)$. α controls how quickly past revisit steps are forgotten. This moving average reduces the variance of prompt-level reward statistics under repeated sampling. A detailed derivation is in Appendix C.

2.5 SpinningRL

Based on the methods proposed above, we introduce SpinningRL, an efficient RL scheduling algorithm that maximizes estimated theoretical reward gain through adaptive UCB sampling. A visual overview is given in Figure 1. The method offers three advantages:

- It concentrates sampling on a difficult-yet-solvable subspace, where the potential reward gain is theoretically largest.
- It employs a sliding-window UCB mechanism that dynamically tracks this subspace under non-stationarity, balancing exploitation and exploration to minimize regret.
- It constructs a low-variance baseline estimator for high-value questions within this subspace that are revisited, reducing the chance of zero advantage.

The name reflects that informative samples *spin* through this region repeatedly, focusing compute where the expected learning gain is highest. The full algorithm is detailed in Algorithm 1.

Algorithm 1 SPINNINGRL: SW-UCB Sampling with Theoretical Potential Reward Gain Estimation

Input: current policy π_{θ_t}, candidate set $\mathcal{Q}_t$, rollout budget K, window size W, exploration coefficient λ, smoothing coefficient α
State: reward buffers $\{\mathcal{R}_q\}$, selection counts $\{N_t(q)\}$, smoothed baseline statistics $(\overline{\mu}_{t-1}, \overline{\sigma}_{t-1})$
for each training step $t = 1, \ldots, T$ **do**
  Compute gain score for each $q \in \mathcal{Q}_t$:

$$s_t(q) \leftarrow \hat{G}_t(q) + \lambda \sqrt{\frac{2 \log W}{N_t(q) + \epsilon}}$$

  Initialize selected batch $\mathcal{B}_t \leftarrow \emptyset$
  while $|\mathcal{B}_t| < K$ **do**
    Select $q^\star \leftarrow \arg\max_{q \in \mathcal{Q}_t} s_t(q)$
    Add $q^\star$ to $\mathcal{B}_t$ and set $s_t(q^\star) \leftarrow -\infty$
  end while
  Sample rollouts for each $q \in \mathcal{B}_t$ and update $\{\mathcal{R}_q\}$ and $\{N_t(q)\}$
  Compute batch statistics (μ_t, σ_t) from collected rewards
  Update smoothed statistics $(\overline{\mu}_t, \overline{\sigma}_t)$ using exponential moving averages
  Update π_{θ_t} with GRPO using normalized advantages
end for

3 EXPERIMENTS

3.1 SETTINGS

Tasks. We conduct experiments on three tasks:

- **Mathematics:** Training is performed on DAPO-Math-17K, a large-scale mathematics dataset collected from web sources and official competition websites (Yu et al., 2025). We conduct evaluation on MATH500 (Lightman et al., 2024) and challenging math competitions including AMC23, AIME24, AIME25, and AIME26.
- **Logical Reasoning:** We use SynLogic, a large-scale synthetic dataset designed to improve LLMs' logical reasoning abilities (Liu et al., 2025a) for both training and validation. It comprises 35 diverse reasoning tasks, with all examples verifiable via simple rules.
- **Coding:** We train on DeepCoder-Preview Luo et al. (2025) with sandbox-verified unit-test rewards, and evaluate on LiveCodeBench v5 and Codeforces. Detailed settings are in Appendix F.

Models. We use Qwen3-1.7B, Qwen3-4B (Yang et al., 2025), and Qwen2.5-7B (Qwen et al., 2025) to verify the effectiveness and generalizability of our method.

Baselines. We compare SPINNINGRL with the following sampling strategies: (1) **Uniform Sampling (US)**: randomly samples prompts with equal probability from the entire dataset. (2) **Dynamic Sampling (DS)**: follows the implementation in Yu et al. (2025), where rollouts are first collected from a larger generation batch; samples with non-zero advantages are then filtered until a sufficient training batch is formed. (3) **MoPPS** (Qu et al., 2025): a heuristic-based sampling method that models each sample as a bandit arm and prioritizes samples whose empirical success rates are closest to 0.5. (4) **Speed-RL** (Zhang et al., 2026): a sequential sampling strategy that first performs a small number of rollouts (e.g., 4) for each sample, and then conducts additional rollouts on groups with moderate difficulty (i.e., non-zero advantages). We additionally compare against solving-state-predictive method DPS (Mao et al., 2026) in Appendix H.

Training and evaluation settings. We follow the standard GRPO training setup for all methods. Qwen2.5-7B is trained with a context length of 4K, while Qwen3-1.7B is trained with 8K. For SPINNINGRL, a unified hyperparameter configuration with sliding window size W = 50, rollout group size N = 8, β = 0.4, λ = 0.3, ϵ = 0.5 and α = 0.8 consistently performs well across different models and tasks. The candidate set $\mathcal{Q}_t$ at each step is drawn uniformly at random from the training dataset, with $|\mathcal{Q}_t| = |D|/8$. Detailed settings are in Appendix E.

Table 2: Mean@32 evaluation on Math.

Method	MATH	AMC23	AIME24	AIME25	AIME26	Avg.↑	Rollouts↓	Runtime↓
Qwen3-1.7B	52.8	28.2	7.6	5.7	5.2	19.9	–	–
+US	85.6	63.3	34.0	24.9	23.2	46.2	204k	21h
+DS	**87.1**	66.3	37.5	27.9	25.8	48.9	307k	22h
+MoPPS	85.6	65.4	34.8	25.9	23.2	47.0	204k	22h
+SPEED	83.7	60.4	29.3	24.7	20.6	43.7	234k	21h
+SpinningRL	86.8	**66.5**	**39.7**	**28.1**	**28.4**	**49.9**	204k	21h
Qwen3-4B	36.9	37.8	24.0	23.6	26.6	29.8	–	–
+US	92.7	82.3	54.9	46.8	**51.6**	65.7	153k	37h
+DS	92.1	80.3	57.6	44.9	48.7	64.7	340k	38h
+MoPPS	**92.9**	82.6	54.4	48.0	49.5	65.5	153k	37h
+SPEED	91.0	81.6	47.6	39.5	44.9	60.9	235k	44h
+SpinningRL	92.3	**84.9**	**58.2**	**50.0**	51.1	**67.3**	153k	37h
Qwen2.5-7B	6.5	2.1	0.3	0.7	0.3	2.0	–	–
+US	77.1	50.4	14.6	11.5	10.5	32.8	204k	7h
+DS	74.6	**53.0**	**17.5**	**12.4**	10.1	33.5	384k	7h
+MoPPS	76.8	52.2	14.9	9.8	9.7	32.7	204k	8h
+SPEED	75.5	51.7	16.4	8.4	9.6	32.3	204k	7h
+SpinningRL	**78.3**	50.2	15.0	12.1	**13.1**	**33.7**	204k	7h

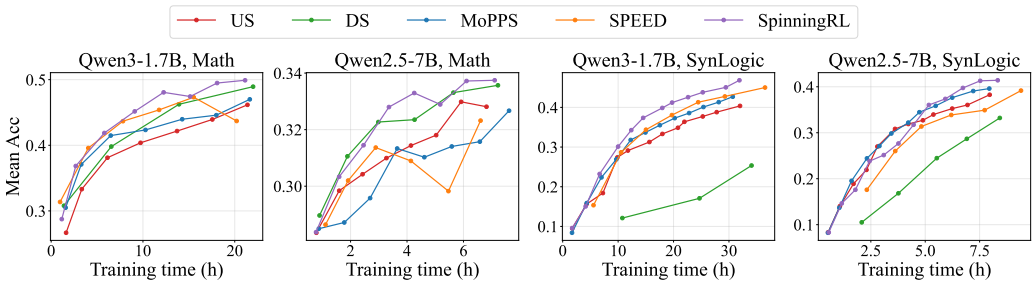


Figure 5: Training-time efficiency comparison of different methods.

3.2 Main Results

Table 2, Table 3 and Table 6 report the *endpoint* accuracy of different methods under a comparable total training hours. Figure 5 shows the mean reward trajectories throughout training. Overall, SpinningRL achieves stronger final performance, converges faster, and exhibits more stable training dynamics.

Benchmark performance. On mathematics, SpinningRL improves the unweighted average accuracy over uniform sampling (US) by 3.7 on Qwen3-1.7B and 0.9 on Qwen2.5-7B under the same rollout budget, while heuristic-based methods such as MoPPS and Speed-RL show little or no improvement. A three-seed evaluation in Appendix E.3 shows limited run-to-run variability and is consistent with the primary-run comparison. Comparing to DS, SpinningRL shows significant improvement on several challenging benchmarks on Qwen3-1.7B (+2.2 on AIME24, +2.6 on AIME26), while using ∼33% fewer rollouts. On Qwen2.5-7B, SpinningRL matches DS on unweighted average (+0.2) while using roughly half the rollouts (204k vs. 384k). On SynLogic, SpinningRL shows clearer and more consistent gains: +6.4 / +3.1 over US, +4.1 / +1.8 over MoPPS, and +1.8 / +2.2 over Speed-RL.

Training dynamics. In Figure 5, SpinningRL shows faster and steadier learning curves on both models and tasks. Speed-RL can rise quickly in early training but plateaus or degrades later on math task. DS is competitive at matched wall-clock on some math runs, but its learning curve is rollout-heavy and less efficient on SynLogic. When aligning by cumulative rollout counts (Figure 7), SpinningRL remains on the Pareto frontier.

Learning behaviors. As shown in Figure 6, SpinningRL leads to a faster increase in response length while maintaining a consistently lower clip ratio on both tasks. This suggests that our gain-

Table 3: Mean@16 evaluation on SynLogic.

Method	Acc.↑	Rollouts↓	Runtime↓
Qwen3-1.7B	9.5	–	–
+US	40.4	307k	32h
+DS	25.3	457k	34h
+MoPPS	42.7	307k	31h
+SPEED	45.0	287k	37h
+SPINNINGRL	**46.8**	307k	31h
Qwen2.5-7B	8.3	–	–
+US	38.3	307k	8h
+DS	33.2	786k	8h
+MoPPS	39.6	307k	8h
+SPEED	39.2	304k	9h
+SPINNINGRL	**41.4**	307k	8h

Table 4: Time breakdown of a training step.

Process	Time (s)	Ratio (%)
Rollout	343.72	70.81
Update Actor	83.87	17.30
Reward Cal.	17.82	3.67
SW-UCB Sel.	1.02	0.21

Table 5: β ablation on Qwen3-1.7B Math.

Method	β	Avg.
SPINNINGRL	0.3	49.5
	0.4	**49.9**
	0.5	49.0

Table 6: Mean@8 evaluation on Code.

Model	Method	LCBv5	Codeforces	Avg.↑	Rollouts↓	Runtime↓
Qwen3-1.7B	Base	18.7	4.2	11.5	–	–
	+US	33.7	12.5	23.1	153K	43h
	+MoPPS	34.8	12.6	23.7	153K	45h
	+SPINNINGRL	**38.7**	**15.4**	**27.1**	153K	42h
Qwen3-4B	Base	32.1	11.1	21.6	–	–
	+US	50.8	22.2	36.5	102K	36h
	+MoPPS	51.4	23.2	37.3	102K	38h
	+SPINNINGRL	**53.0**	**25.6**	**39.3**	102K	37h

aware sampling strategy provides stronger and more stable learning signals, allowing the policy to explore longer reasoning trajectories without causing unstable updates.

By jointly using estimated reward gain and uncertainty in the SW-UCB score, SPINNINGRL naturally balances exploration and exploitation and forms an *adaptive curriculum* over difficult-yet-solvable prompts, as visualized in Figure 8 in Appendix J.

Efficiency comparison. The reward gain estimation, SW-UCB scheduling and history-aware advantage computation introduce negligible overhead in the training process. We achieve this efficiency via lazy updates that refresh prompt-level statistics only when needed, as detailed in Appendix D. Table 4 reports the average time breakdown over 200 training steps on Qwen3-1.7B Math task, showing that the additional computation introduced by our sampling method is minimal.

3.3 ABLATION STUDIES

Different βs. In the reward gain model, a smaller β means a higher KL-divergence budget and encouraged exploration on the relatively more challenging questions. We test different β values on Qwen3-1.7B Math, normalizing the scheduling signal by the theoretical maximum $G_\beta(p^*_\beta)$ under each β while keeping all other hyperparameters fixed. Results in Table 5 show that SPINNINGRL prefers moderately small β values (0.3-0.4), which is consistent with our observations in Figure 2.

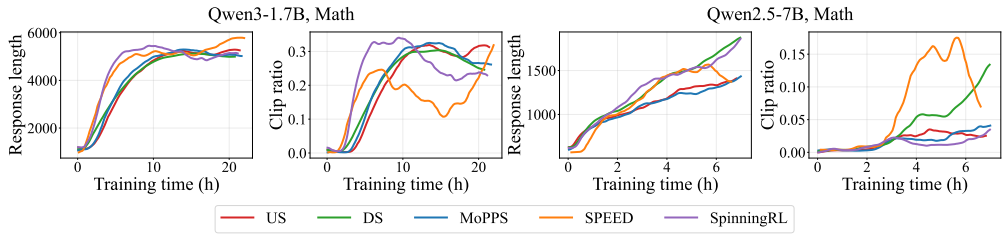


Figure 6: Response length and clip ratio comparison.

Table 7: Mean@32 evaluation on Math with Qwen3-1.7B.

Method	MATH	AMC23	AIME24	AIME25	AIME26	Avg.↑	#Rollouts
US	85.6	63.3	34.0	24.9	23.2	46.2	204k
$\max p(1-p)$	<u>86.2</u>	65.6	36.7	26.7	24.5	47.9	204k
$\min \lvert p-0.5 \rvert$	85.4	65.1	35.8	24.9	23.6	47.0	204k
DS + Filtering $0.2 < p < 0.8$	**86.8**	<u>66.3</u>	<u>38.9</u>	**28.6**	<u>25.6</u>	<u>49.2</u>	359k
SPINNINGRL ($\widehat{G}$)	**86.8**	**66.5**	**39.7**	<u>28.1</u>	**28.4**	**49.9**	204k

Comparison with learnability-based selectors. To compare our reward-gain-induced selector metric against online learnability-based scoring, we evaluate LILO/LearnAlign's $p(1-p)$ ranking and MoPPS's $|p-0.5|$ selector under the same SW-UCB scheduler, as well as Bae-style pass-rate filtering on Dynamic Sampling, on Qwen3-1.7B Math. Table 7 shows that reward-gain scheduling consistently outperforms these alternatives under matched rollout budgets.

We additional conducted ablation studies on history-aware baseline estimation in Appendix K and evaluated on non-binary-reward task KK in Appendix G.

4 Related Works

RLVR Efficiency. Recent work has improved the efficiency of reinforcement learning from different perspectives. Some works stabilize training for GRPO-style RLVR targets better advantage normalization (Yu et al., 2025; Liu et al., 2025c; He et al., 2025c). Some work accelerate rollout generation phase by leveraging quantization (Liu et al., 2025b; Xi et al., 2026) or speculative decoding (Chen et al., 2025; Liu et al., 2026; Qin et al., 2026; Shao et al., 2025). Other works focus on asynchronous settings (Fu et al., 2026; Sheng et al., 2025a), improving GPU utilization and reduce generation bottlenecks caused by long-tail rollouts.

Sample Selection for RLVR. Recent methods improve RLVR efficiency by selectively allocating rollout computation to informative prompts. Learnability-based approaches such as MoPPS (Qu et al., 2025), Speed-RL (Zhang et al., 2026), LILO Foster et al. (2026), LearnAlign Li et al. (2026), and Bae et al. (Bae et al., 2026) prioritize medium-difficulty prompts using success-rate-derived signals such as $p(1-p)$, $|p-0.5|$ or pass-rate bands. DPS (Mao et al., 2026) models each prompt's solving-state dynamics from historical rollout rewards and predicts partially solved prompts, while GPS (Qu et al., 2026) learns a prompt-generalizable predictor of success rate or difficulty and additionally promotes diversity during selection. GRESO (Zheng et al., 2025) skips historically zero-variance prompts to reduce rollout cost. Sequential samplers (Xiong et al., 2025; Liao et al., 2025; Zhang et al., 2026) estimate difficulty with a first-stage rollout and then allocate more budget, but incur long-tail latency. These approaches primarily target medium-difficulty prompts without modeling a fine-grained continuum of prompt utility. In contrast, SPINNINGRL aligns the selection signal with policy-improvement objective and provides a fine-grained estimate of prompt utility.

5 Conclusion

We present SPINNINGRL, a gain-aware RLVR framework that combines theoretically grounded reward gain estimation, adaptive SW-UCB scheduling, and history-aware baseline smoothing for more sample-efficient and stable training. Experiments on mathematics, symbolic reasoning and coding benchmarks demonstrate better performance and faster convergence, with negligible additional computational overhead.

6 Limitations

Despite the promising performance, our method still has several limitations. First, the proposed gain does not exactly equal to a single GRPO update and is instead a KL-constrained surrogate, therefore a more systematic study of when this surrogate remains reliable is an important direction for future work. Moreover, our experiments focus on verifiable math, logic, and coding rewards. Extending our non-binary modeling to complex agentic tasks with richer reward structures is a promising direction that remains to be explored.

AI USE STATEMENT

In this work, we used generative AI tools for assisting with translation and implementing methods. We have not used generative AI tools for other tasks such as the development of theoretical models or conceptual frameworks, formulation of mathematical claims, etc. Additionally, we used generative AI tools to create or modify scientific figures for demonstration, and refine the writing of this paper to improve readability. We have reviewed all AI-assisted work. We have double-checked LLM-refined writing for correctness and alignment with the original content.

REFERENCES

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, Jeongyeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.30. URL https://aclanthology.org/2026.eacl-long.30/.

Qiaoling Chen, Zijun Liu, Peng Sun, Shenggui Li, Guoteng Wang, Ziming Liu, Yonggang Wen, Siyuan Feng, and Tianwei Zhang. Respec: Towards optimizing speculative decoding in reinforcement learning systems, 2025. URL https://arxiv.org/abs/2510.26475.

Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Learning to optimize under non-stationarity. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1079–1087. PMLR, 2019. URL http://proceedings.mlr.press/v89/cheung19b.html.

Yunzhen Feng, Parag Jain, Anthony Hartshorn, Yaqi Duan, and Julia Kempe. Don't waste mistakes: Leveraging negative rl-groups via confidence reweighting, 2025. URL https://arxiv.org/abs/2510.08696.

Thomas Foster, Anya Sims, Johannes Forkel, Mattie Fellows, and Jakob Foerster. Learning to reason at the frontier of learnability, 2026. URL https://arxiv.org/abs/2502.12272.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. Areal: A large-scale asynchronous reinforcement learning system for language reasoning, 2026. URL https://arxiv.org/abs/2505.24298.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *Algorithmic Learning Theory (ALT)*, pp. 174–188. Springer, 2011. URL https://arxiv.org/abs/0805.3415.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan

Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Jingkai He, Tianjian Li, Erhu Feng, Dong Du, Qian Liu, Tao Liu, Yubin Xia, and Haibo Chen. History rhymes: Accelerating llm reinforcement learning with rhymerl, 2025a. URL https://arxiv.org/abs/2508.18588.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report, 2025b. URL https://arxiv.org/abs/2505.22312.

Zhiyuan He, Xufang Luo, Yike Zhang, Yuqing Yang, and Lili Qiu. Vl norm: Rethink loss aggregation in rlvr, 2025c. URL https://arxiv.org/abs/2509.07558.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfBRcrXek.

Thanh-Long V. Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in llm reinforcement learning via entropy-guided advantage shaping, 2026. URL https://arxiv.org/abs/2509.21880.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023. URL https://arxiv.org/abs/2312.14852.

Shipeng Li, Zhiqin Yang, Shikun Li, Xiaobo Xia, Hengyu Liu, Xinghua Zhang, Gaode Chen, Dong Fang, Ying Tai, and Zhe Peng. LearnAlign: Data selection for LLM reinforcement learning with improved gradient alignment. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40417–40434, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.2009. URL https://aclanthology.org/2026.findings-acl.2009/.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling, 2025. URL https://arxiv.org/abs/2502.11886.

Mengqi Liao, Xiangyu Xi, Ruinian Chen, Jia Leng, Yangen Hu, Ke Zeng, Shuai Liu, and Huaiyu Wan. Enhancing efficiency and exploration in reinforcement learning for llms, 2025. URL https://arxiv.org/abs/2505.18573.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Bingshuai Liu, Ante Wang, Zijun Min, Liang Yao, Haibo Zhang, Yang Liu, Xu Han, Peng Li, Anxiang Zeng, and Jinsong Su. Spec-rl: Accelerating on-policy reinforcement learning with speculative rollouts, 2026. URL https://arxiv.org/abs/2509.23232.

Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yongyi Hu, Chi Zhang, Yiqi Shi, Shitong Weng, Aili Chen, Shiqi Chen, Yunan Huang, Mozhi Zhang, Pengyu Zhao, Junjie Yan, and Junxian He. Synlogic: Synthesizing verifiable reasoning data at scale for learning logical reasoning and beyond. *CoRR*, abs/2505.19641, 2025a. doi: 10.48550/ARXIV.2505.19641. URL https://doi.org/10.48550/arXiv.2505.19641.

Liyuan Liu, Feng Yao, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Flashrl: 8bit rollouts, full power rl, August 2025b. URL https://fengyao.notion.site/flash-rl.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025c. URL https://arxiv.org/abs/2503.20783.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51, 2025. Notion Blog.

Yixiu Mao, Yun Qu, Qi Wang, Heming Zou, and Xiangyang Ji. Dynamics-predictive sampling for active RL finetuning of large reasoning models. *CoRR*, abs/2603.10887, 2026. doi: 10.48550/ARXIV.2603.10887. URL https://doi.org/10.48550/arXiv.2603.10887.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia

Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wenting Zhan, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2026. URL https://arxiv.org/abs/2412.16720.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning (ICML)*, pp. 745–750, 2007.

Ruoyu Qin, Weiran He, Weixiao Huang, Yangkun Zhang, Yikai Zhao, Bo Pang, Xinran Xu, Yingdi Shan, Yongwei Wu, and Mingxing Zhang. Seer: Online context learning for fast synchronous llm reinforcement learning, 2026. URL https://arxiv.org/abs/2511.14617.

Yun Qu, Qi Wang, Yixiu Mao, Vincent Tao Hu, Björn Ommer, and Xiangyang Ji. Can prompt difficulty be online predicted for accelerating RL finetuning of reasoning models? *CoRR*, abs/2507.04632, 2025. doi: 10.48550/ARXIV.2507.04632. URL https://doi.org/10.48550/arXiv.2507.04632.

Yun Qu, Qi Wang, Yixiu Mao, Heming Zou, Yuhang Jiang, Weijie Liu, Clive Bai, Kai Yang, Yangkun Chen, Saiyong Yang, and Xiangyang Ji. Small generalizable prompt predictive models can steer efficient RL post-training of large reasoning models. *CoRR*, abs/2602.01970, 2026. doi: 10.48550/ARXIV.2602.01970. URL https://doi.org/10.48550/arXiv.2602.01970.

Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, Binyuan Hui, Zheng Yuan, Junyang Lin, Fei Huang, and Jingren Zhou. Codeelo: Benchmarking competition-level code generation of llms with human-comparable elo ratings. *arXiv preprint arXiv:2501.01257*, 2025. URL https://arxiv.org/abs/2501.01257.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Zelei Shao, Vikranth Srivatsa, Sanjana Srivastava, Qingyang Wu, Alpay Ariyak, Xiaoxia Wu, Ameen Patel, Jue Wang, Percy Liang, Tri Dao, Ce Zhang, Yiying Zhang, Ben Athiwaratkun, Chenfeng Xu, and Junxiong Wang. Beat the long tail: Distribution-aware speculative decoding for rl training, 2025. URL https://arxiv.org/abs/2511.13841.

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

Guangming Sheng, Yuxuan Tong, Borui Wan, Wang Zhang, Chaobo Jia, Xibin Wu, Yuqi Wu, Xiang Li, Chi Zhang, Yanghua Peng, Haibin Lin, Xin Liu, and Chuan Wu. Laminar: A scalable asynchronous rl post-training framework, 2025a. URL https://arxiv.org/abs/2510.12633.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297. ACM, March 2025b. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Weixin Xu, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, Zonghan Yang, and Zongyu Lin. Kimi k1.5: Scaling reinforcement learning with llms, 2025. URL https://arxiv.org/abs/2501.12599.

Haocheng Xi, Charlie Ruan, Peiyuan Liao, Yujun Lin, Han Cai, Yilong Zhao, Shuo Yang, Kurt Keutzer, Song Han, and Ligeng Zhu. Jet-rl: Enabling on-policy fp8 reinforcement learning with unified training and rollout precision flow, 2026. URL https://arxiv.org/abs/2601.14243.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning, 2025. URL https://arxiv.org/abs/2502.14768.

Wei Xiong, Chenlu Ye, Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, Jiang Bian, Nan Jiang, and Tong Zhang. Reinforce-ada: An adaptive sampling framework for reinforce-style llm training, 2025. URL https://arxiv.org/abs/2510.04996.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Ruiqi Zhang, Daman Arora, Song Mei, and Andrea Zanette. Speed-rl: Faster training of reasoning models via online curriculum learning, 2026. URL https://arxiv.org/abs/2506.09016.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts, 2025. URL https://arxiv.org/abs/2506.02177.

A PROOFS FOR REWARD-GAIN ANALYSIS

Binary closed form and maximizer. Consider binary rewards $r \sim \mathrm{Bernoulli}(p)$ under the current policy, and the KL tilt $\pi^*(o \mid q) \propto \pi_\theta(o \mid q) \exp(r/\beta)$. Under π^*, each response is reweighted by the Boltzmann factor $e^{r/\beta}$ relative to π_θ. Let $a = e^{1/\beta} > 1$. Correct responses (r=1) are therefore reweighted by a and incorrect ones (r=0) by 1, so the improved success probability is

$$p^* = \frac{pa}{pa + (1-p)} = \frac{pa}{1 + p(a-1)}.$$

The reward gain is therefore

$$G_\beta(p) = p^* - p = \frac{pa}{1 + p(a-1)} - p = \frac{p(1-p)(a-1)}{1 + p(a-1)}.$$

In particular, $G_\beta(0) = G_\beta(1) = 0$ and $G_\beta(p) > 0$ for all $p \in (0, 1)$.

To locate the maximizer, write $G = (a-1)\, u/v$ with $u = p(1-p)$ and $v = 1 + p(a-1)$. Then

$$\frac{\partial G}{\partial p} = (a-1) \cdot \frac{u'v - uv'}{v^2}, \qquad u' = 1 - 2p, \quad v' = a - 1.$$

The numerator of the fraction simplifies as

$$(1-2p)\big(1 + p(a-1)\big) - p(1-p)(a-1) = 1 - 2p - p^2(a-1).$$

Setting $\partial G/\partial p = 0$ (and using $a > 1$) yields the quadratic $p^2(a-1) + 2p - 1 = 0$. The feasible root in $(0, 1)$ is

$$p^*_\beta = \frac{-2 + \sqrt{4 + 4(a-1)}}{2(a-1)} = \frac{\sqrt{a} - 1}{a - 1} = \frac{1}{\sqrt{a} + 1} = \frac{1}{1 + e^{1/(2\beta)}}.$$

Since $\sqrt{a} > 1$, $p^*_\beta \in (0, 1/2)$. As $\beta \to \infty$ (vanishing KL budget δ), $p^*_\beta \to 1/2$, recovering near-medium difficulty prompts; as $\beta \to 0$ (large KL budget), $p^*_\beta \to 0$, so the criterion prefers increasingly hard but still solvable prompts.

Finally, substitute p^*_β into G_β. Using $p^*_\beta(a-1) = \sqrt{a} - 1$ gives

$$\max_{p \in [0,1]} G_\beta(p) = \frac{\sqrt{a} - 1}{\sqrt{a} + 1} = \tanh\Big(\frac{1}{4\beta}\Big).$$

Asymptotic connection to learnability. In the weak-update regime $\beta \to \infty$, expand $a = e^{1/\beta} = 1 + 1/\beta + O(\beta^{-2})$, so $a - 1 = 1/\beta + O(\beta^{-2})$. For any fixed $p \in (0, 1)$, the denominator satisfies $1 + p(a-1) = 1 + O(\beta^{-1})$, and therefore

$$\begin{aligned} G_\beta(p) &= \frac{p(1-p)\,(a-1)}{1 + p(a-1)} = p(1-p)\,(a-1)\,\big(1 - p(a-1) + O(\beta^{-2})\big) \\ &= \frac{p(1-p)}{\beta} + O(\beta^{-2}). \end{aligned}$$

Since $\mathrm{Var}(r) = p(1-p)$ for Bernoulli rewards, the reward gain could be reduced to

$$G_\beta(p) = \frac{\mathrm{Var}(r)}{\beta} + O(\beta^{-2}).$$

Hence maximizing G_β reduces to maximizing $p(1-p)$ when updates are infinitesimal, meaning that learnability-based scheduling is the weak-update special case of SpinningRL. For finite KL budgets (moderate or small β), the factor $(a-1)/(1 + p(a-1))$ breaks the symmetry about $p = 1/2$ and shifts the peak to $p^*_\beta < 1/2$, recovering the difficult-yet-solvable preference observed in Figure 2.

B Conditional Regret Bound of SW-UCB Sampling

Following the non-stationary bandit formulation in Sec. 2.3, let

$$\mu_t(q_i) = G_t(q_i)$$

denote the latent expected reward gain of question q_i at training step t.

We provide a *conditional* guarantee of the regret upper bound of sampling process based on the dynamic-regret analysis of sliding-window UCB for non-stationary stochastic bandits (Garivier & Moulines, 2011; Cheung et al., 2019).

We abstract each RLVR update as $\theta_{t+1} = F(\theta_t, q_t, \text{rollout})$ and make the following assumptions:

Assumption 1 (Conditional concentration) *Conditional on θ_t, gain observations $\hat{G}_t(q)$ are sub-Gaussian around $G_t(q)$.*

Assumption 2 (Bounded temporal variation) *The variation budget*

$$V_T = \sum_{t=1}^{T-1} \sup_{q_i \in \mathcal{Q}} |\mu_{t+1}(q_i) - \mu_t(q_i)|$$

is finite. A larger VT indicates faster-changing (more non-stationary) gains.

Assumption 3 (Candidate-set regularity) *At each step the candidate set $\mathcal{Q}_t \subseteq \mathcal{D}$ is drawn uniformly at random, independently of the gain process, with $|\mathcal{Q}_t| = M$ and $\rho := M/|\mathcal{D}|$. We take $M \gg K$, so that each step scores a large random subsample of the pool: every prompt enters $\mathcal{Q}_t$ with probability ρ at each step, the probability of being excluded for k consecutive steps decays as $(1-\rho)^k$, and no arm is persistently excluded.*

Recall that the dynamic regret is

$$R_T = \sum_{t=1}^{T} \left(\mu_t(q_t^*) - \mu_t(q_t)\right), \quad q_t^* = \arg\max_{q_i \in \mathcal{Q}} \mu_t(q_i).$$

Under Assumptions 1–3, applying SW-UCB (Garivier & Moulines, 2011) with window size W to the prompt-gain process yields the standard dynamic-regret bound of non-stationary stochastic bandits:

$$R_T = O\left(T\sqrt{\frac{M \log W}{W}} + W V_T\right),$$

Subsampling the candidate set costs at most an additional $(1-\rho)\,T\,\Delta_{\max}$ in expectation, where $\Delta_{\max} \le \tanh(1/4\beta)$ bounds the range of the gain. This term is negligible under the high-coverage setting of Assumption 3 and vanishes when $\mathcal{Q}_t = \mathcal{D}$.

Balancing the two terms yields the optimal window size

$$W^* = \Theta\left(\left(\frac{T\sqrt{M \log T}}{V_T}\right)^{2/3}\right),$$

which further gives the regret rate

$$R_T = \tilde{O}\left((M \log T)^{1/3} V_T^{1/3} T^{2/3}\right), \tag{9}$$

where $\tilde{O}(\cdot)$ hides logarithmic factors.

Eq. 9 characterizes the growth rate of the conditional dynamic regret under the stated assumptions, and could be read as an *asymptotic* upper bound on R_T (in the big-O sense). When the policy evolves slowly (smaller V_T), a larger sliding window accumulates more historical information. Otherwise, under rapidly changing dynamics (larger V_T), a smaller window is preferred to reduce stale estimates.

C Derivation of History-Aware Baseline Estimation

We justify the moving average mean and variance computation in Sec. 2.4 under the assumption that whenever a prompt is sampled, its rollout group size is fixed as N.

C.1 Step-wise Statistics

Fix a prompt q. At a revisit step t where q is sampled, let $\{r_{t,j}\}_{j=1}^{N}$ denote the rollout rewards. Define the step mean, variance, and second moment:

$$\mu_t = \frac{1}{N}\sum_{j=1}^{N} r_{t,j}, \tag{10}$$

$$\sigma_t^2 = \frac{1}{N}\sum_{j=1}^{N}(r_{t,j}-\mu_t)^2, \tag{11}$$

$$M_{2,t} = \frac{1}{N}\sum_{j=1}^{N} r_{t,j}^2 = \sigma_t^2 + \mu_t^2. \tag{12}$$

The identity in Eq. 12 follows from $r_{t,j}^2 = (r_{t,j}-\mu_t)^2 + 2(r_{t,j}-\mu_t)\mu_t + \mu_t^2$ and $\sum_j (r_{t,j}-\mu_t) = 0$.

Thus each revisit step is fully summarized by the pair $(\mu_t, M_{2,t})$. Our goal is to combine these step-wise summaries across revisit steps into a single baseline $(\bar{\mu}_t, \bar{\sigma}_t^2)$.

C.2 Running Variance via Second Moments

We obtain the running variance by maintaining an exponentially smoothed second moment and applying the identity $\sigma^2 = M_2 - \mu^2$.

Identity. For any weighted sample, if

$$\mu = \mathbb{E}[r], \qquad M_2 = \mathbb{E}[r^2],$$

then the variance is

$$\sigma^2 = \mathbb{E}[(r-\mu)^2] = \mathbb{E}[r^2] - \mu^2 = M_2 - \mu^2. \tag{13}$$

This holds whenever μ and M_2 are computed under the *same* weights.

Two-step merge. Consider two revisit steps with statistics (μ_1, σ_1^2) and (μ_2, σ_2^2), each based on N rollouts. Pooling the $2N$ rewards gives

$$\mu = \frac{\mu_1+\mu_2}{2}, M_2 = \frac{M_{2,1}+M_{2,2}}{2}, \sigma^2 = M_2 - \mu^2.$$

On the contrary, directly averaging variances by $\frac{1}{2}(\sigma_1^2+\sigma_2^2)$ is incorrect because it ignores the shift between μ_1 and μ_2. The correct pooled variance must be computed from the pooled second moment.

Exponential smoothing across steps. We apply exponential smoothing across rollout steps:

$$\bar{\mu}_t = (1-\alpha)\bar{\mu}_{t-1} + \alpha\mu_t, \tag{14}$$

$$\bar{M}_{2,t} = (1-\alpha)\bar{M}_{2,t-1} + \alpha M_{2,t}. \tag{15}$$

Using the same α for both quantities ensures that $\bar{\mu}_t$ and $\bar{M}_{2,t}$ are weighted averages of past step statistics under identical weights. Therefore, by Eq. 13,

$$\bar{\sigma}_t^2 = \bar{M}_{2,t} - \bar{\mu}_t^2 \tag{16}$$

derives the correct running variance.

Implementation. In implementation, the running statistics are maintained at the *training-step* level instead of individual reward level. Each time prompt q is sampled, its N rollouts are first summarized by $(\mu_t, M_{2,t})$ within step, and these step-wise statistics are then smoothed across revisit history.

Since only steps with rollouts contribute new statistics, in implementation, we iterate over all slots in the SW-UCB sliding window: empty slots skip mean/variance updates but decay the accumulated weight by $1 - \alpha$, so older baselines are gradually forgotten when q is not revisited.

C.3 Variance Reduction

One-step GRPO uses only the current group, so $\mathrm{Var}(\mu_t) = O(\sigma^2/N)$. The history-aware baseline combines means from multiple revisit steps:

$$\bar{\mu}_t = \sum_{k=1}^{t} w_k \mu_k, \qquad \sum_k w_k = 1,$$

where w_k are the effective weights induced by repeated EMA updates. When the same prompt is revisited several times within the SW-UCB window, $\bar{\mu}_t$ aggregates more evidence than a single step, reducing baseline variance. The same logic applies to $\bar{\sigma}_t^2$ because it is derived consistently from $\bar{M}_{2,t}$ and $\bar{\mu}_t$.

D Lazy Updates for Efficient Scheduling

Recomputing potential reward gains and history-aware baselines for every question at every training step is unnecessary and would introduce avoidable overhead. We therefore adopt lazy updates that refresh statistics only when the corresponding sliding-window state changes, while still allowing unused arms to gradually forget stale information.

Lazy update of reward gain. The estimated gain $\widehat{G}_t(q_i)$ depends only on rollout rewards observed inside the current SW-UCB window of size W. Consequently, we recompute $\widehat{G}_t(q_i)$ only when the effective window contents of arm q_i change, i.e., when a new observation for q_i is appended or an old observation expires from the window. Otherwise, the previously computed gain is reused for UCB scoring. The estimated gain used in UCB scoring therefore inherits the same lazy-update schedule whenever the window contents of q_i are unchanged.

Lazy update of the history-aware baseline. The history-aware baseline of Sec. 2.4 is likewise maintained at the prompt level. Whenever prompt q is sampled, we summarize its N rollouts by the step statistics $(\mu_t, M_{2,t})$ and apply one EMA update to $(\bar{\mu}_t, \bar{M}_{2,t})$. If q is not sampled at the current step, we do *not* invent a new mean/variance estimate; instead, while scanning slots in the SW-UCB window, empty slots skip statistic updates but still decay the accumulated EMA weight by $1 - \alpha$. This soft forgetting ensures that baselines of rarely revisited prompts gradually discount outdated history without paying the cost of fresh rollouts.

Negligible overhead. Together, the two lazy updates keep the extra cost of gain estimation, SW-UCB scoring, and history-aware advantage computation negligible relative to rollout generation and policy optimization, as reported in Table 4.

E Experimental Settings

E.1 Training Setup

For a fair comparison, baseline methods are configured to match approximately the same per-step training budget except DS. In particular, the rollout batch size is set to 128 for US, MoPPS, DS. For DS, we follow the original implementation and use a generation batch size of 128×3 with at most 10 generation rounds. For Speed-RL, to ensure that the total training batch size remains consistent across methods at each step, we proportionally rescale the batch sizes of Speed-RL while preserving

its original sampling schedule. We use a first-round generation batch size of 168 with $N_1 = 4$ and a second-round batch size of 42 with $N_2 = 20$.

All experiments use DAPO's token-mean loss aggregation and "clip-higher" setting. The remaining optimization hyperparameters are summarized in Table 8.

Hyperparameter	**Value**
Optimizer	AdamW
Learning Rate	1×10^{-6}
Weight Decay	0.01
Gradient Clip Norm	1.0
PPO Mini-batch Size	32
Rollout Group Size N	8
Training Batch Size	128
Validation Sampling Number	32
Max Prompt Length	2048
Max Response Length	4096 (Qwen2.5-7B) 8192 (Qwen3-1.7B) 8192 (Qwen3-4B)
Sampling Temperature	1.0
Training Top-p	1.0
Validation Top-p	0.7
Clip Ratio (Low)	0.2
Clip Ratio (High)	0.28
KL Loss Coefficient	0.0
Entropy Coefficient	0.0
Loss Aggregation	Token Mean

Table 8: Common training hyperparameters shared across all methods.

E.2 Implementation and Hardware

We implement all methods on top of `verl` (Sheng et al., 2025b) [1]. All experiments are conducted on NVIDIA A100 GPUs.

E.3 Statistical Robustness across Random Seeds

To assess sensitivity to training stochasticity, we conduct a three-seed evaluation on Qwen3-1.7B trained on DAPO-Math-17k. We independently repeat training for different methods using the same set of random seeds and identical training budgets, and report the mean and standard deviation of the unweighted average accuracy over the three runs.

As shown in Table 9, SpinningRL remains competitive across the three random seeds and preserves the same ranking as in the primary run. The relatively small standard deviations indicate limited run-to-run variability across the three random seeds. These results are consistent with the primary-run comparison. Per-benchmark scores of the primary run are reported in Table 2.

Method	**Avg.↑**
US	46.3±0.3
DS	48.7±0.5
MoPPS	46.9±0.2
SpinningRL	**49.7±0.3**

Table 9: Mean and standard deviation over three independent random seeds on Qwen3-1.7B Math.

[1] `https://github.com/verl-project/verl`

F EVALUATION ON CODING TASK

To examine whether the gains of SPINNINGRL transfer beyond mathematics and logical reasoning, we further evaluate on competitive coding with Qwen3-1.7B and Qwen3-4B under the same GRPO training stack as in Sec. E. Results are summarized in Table 6.

Training dataset. We train on the publicly released **DeepCoder-Preview** dataset (Luo et al., 2025), which contains about 24K competitive-programming problems paired with executable unit tests: 7.5K TACO-Verified problems (Li et al., 2023), 16K verified problems from PrimeIntellect's SYNTHETIC-1[2], and 600 LiveCodeBench (v5) problems dated between May 1, 2023 and July 31, 2024. Each prompt provides fully verifiable tests (at least five per problem), so that every rollout yields a binary reward $r \in \{0, 1\}$ after sandbox execution.

Evaluation datasets. Following the DeepCoder-Preview evaluation protocol, we report performance on: (i) **LiveCodeBench v5 (LCBv5)** (Jain et al., 2025) problems dated between August 1, 2024 and February 1, 2025 (held out from the training window); and (ii) **Codeforces** problems from CodeElo (Quan et al., 2025). For each method we use the same decoding protocol and report mean pass rates on both suites, together with cumulative rollouts and wall-clock runtime (Table 6).

Sandbox-based code verification. Unlike math/logic rewards that can be checked by symbolic matching, coding rewards require *executing* untrusted model-generated programs. Both training-time reward computation and offline evaluation therefore run all candidate solutions inside an *isolated sandbox*, rather than on the training host. Following the code-verification backend supported by `verl` (Sheng et al., 2025b), each submission is compiled/interpreted in a containerized environment with hard limits on CPU time and memory per test case. A solution receives reward 1 only if it passes *all* tests within these limits; compilation failures, runtime errors, wrong outputs, and timeouts all yield reward 0. This sandboxing provides (i) safety against side effects from generated code and (ii) reproducible binary rewards for GRPO-style advantage estimation and for SPINNINGRL's gain-based scheduling.

G COMPARISON WITH LEARNABILITY-BASED METHODS

To provide a comprehensive comparison of SPINNINGRL and learnability-based methods, we summarize the key conceptual differences below.

As summarized in Table 1, MoPPS (Qu et al., 2025) and Speed-RL (Zhang et al., 2026) are two representative learnability-style schedulers; closely related methods such as LearnAlign (Li et al., 2026), LILO (Foster et al., 2026), and Bae et al. (2026) likewise prioritize **medium-difficulty** prompts via **success-rate-derived** signals. Concretely, LILO and LearnAlign score prompts by learnability $p(1-p)$; MoPPS ranks arms by $|p - 0.5|$ in a bandit formulation; and Bae et al. as well as Speed-RL keep prompts (or rollout groups) whose empirical pass rates lie in a medium band $[T_{\text{Low}}, T_{\text{High}}]$ (or an equivalent moderate-difficulty filter). LearnAlign applies this criterion **offline** after warmup, whereas MoPPS, Speed-RL, LILO, and Bae et al. apply it **online** during training. SPINNINGRL is also online, but schedules by the KL-constrained potential reward gain $\widehat{G}(q)$. As shown in Sec. 2.2, this criterion recovers $p(1-p)$ in the weak-update regime and extends it under finite KL budgets.

1. SpinningRL generalizes—rather than opposes—learnability. Learnability-based methods select prompts with high $p(1-p)$ because it reflects the **magnitude of the policy gradient**. Following LearnAlign (Li et al., 2026) and LILO (Foster et al., 2026), the policy gradient for prompt ξ can be written as, the policy gradient for prompt ξ can be written as

$$\nabla_\theta \mathcal{J}(\theta; \xi) = p(1-p)\, \mathbf{d}(\xi, \theta), \tag{17}$$

where $\mathbf{d}(\xi, \theta)$ is a direction vector. Bae et al. (2026) likewise filter medium-difficulty prompts to maximize **policy update magnitude**. SpinningRL instead starts from the KL-constrained improvement $G_\beta(q) = J_q(\pi^*) - J_q(\pi_\theta)$. In the weak-update limit, $G_\beta(p) \approx p(1-p)/\beta$ (Eq. 8), so

[2] `https://huggingface.co/datasets/PrimeIntellect/SYNTHETIC-1`

variance-based learnability is a local special case; finite KL budgets then shift the peak to $p^*_\beta < 0.5$, favoring difficult-yet-solvable prompts. Thus G_β should be viewed as a reward-space soft-update criterion that recovers learnability locally and becomes asymmetric only under finite KL budgets, rather than as a claim that natural-gradient geometry alone prefers difficult-yet-solvable prompts.

2. SpinningRL extends beyond binary-reward settings. Learnability-based methods (Li et al., 2026; Foster et al., 2026; Bae et al., 2026) are derived under the binary-reward assumption. In contrast, SpinningRL's reward-gain model applies to general verifiable rewards. On the non-binary task KK (Xie et al., 2025), with reward range $[-3, 3]$ and Qwen2.5-3B, SpinningRL outperforms existing scheduling baselines under the same rollout budget (Table 10).

Table 10: Mean@16 evaluation on KK (3–6 ppl) with Qwen2.5-3B.

Method	Mean@16	#Rollouts↓
US	1.48	307k
DS	2.07	2273k
MoPPS	1.19	307k
SPINNINGRL	**2.49**	307k

3. Empirical comparison with learnability-based scheduling. On Qwen3-1.7B Math, we compare SpinningRL against online learnability-based baselines, including LILO-style $p(1-p)$ ranking, a MoPPS-style $|p - 0.5|$ selector under the same SW-UCB scheduler, and Bae et al.'s difficulty filtering on Dynamic Sampling. The results are reported in the main text (Table 7).

H COMPARISON WITH DPS

H.1 OVERVIEW OF DPS

Dynamics-Predictive Sampling (Mao et al., 2026) is a concurrent online prompt-selection method that aims to identify **partially solved** problems before costly rollouts. Concretely, DPS discretizes each prompt's solving progress into three stages—**unsolved**, **partially solved**, and **solved**—and treats the stage as a latent state of a dynamical system. It models state transitions with a hidden Markov model (HMM) and a transition matrix estimated from historical reward signals, then uses online Bayesian inference to predict the current solvability of candidate prompts and preferentially samples those predicted to be partially solved.

H.2 THEORETICAL DISTINCTION

Although both DPS and SPINNINGRL perform online sample selection for RLVR, their objectives and state representations differ substantially.

Different optimization objectives. DPS's fundamental state is a discrete *solving stage* of a prompt (unsolved / partially solved / solved). Its optimization goal is to select prompts that are currently in the partially solved stage, so that policy updates receive non-zero advantages without wasting rollouts on trivial or unsolvable items. In contrast, SPINNINGRL's fundamental scalar is the continuous *potential reward gain* $G_\beta(q)$ (estimated by $\widehat{G}(q)$), i.e., the expected improvement of the KL-constrained soft policy update on that prompt.

Coarse vs. fine-grained discrimination. The three-stage representation maps a continuum of success rates into a small number of solving states. As a result, prompts with different empirical success rates may be mapped to the same partially-solved state, limiting the granularity of the state representation. In contrast, SPINNINGRL assigns a continuous gain value $G_\beta(q)$, allowing finer discrimination among prompts within the difficult-yet-solvable region.

H.3 EMPIRICAL COMPARISON

We compare DPS and SPINNINGRL on Qwen3-4B Math under the same evaluation protocol. Results are reported in Table 11. As shown, SPINNINGRL consistently outperforms DPS under the same rollout budget.

Table 11: Mean@32 evaluation on Math with Qwen3-4B.

Method	MATH	AMC23	AIME24	AIME25	AIME26	Avg.↑	#Rollouts
US	**92.7**	82.3	54.9	46.8	**51.6**	65.7	153k
DPS	92.5	83.5	56.5	48.5	51.5	66.5	153k
SPINNINGRL	92.3	**84.9**	**58.2**	**50.0**	51.1	**67.3**	153k

I SAMPLE-EFFICIENCY COMPARISON

To further evaluate sample efficiency, we compare different methods under the same rollout budget. Figure 7 plot the performance curves against the cumulative rollout count on two tasks. Across both settings, SPINNINGRL consistently achieves the best performance under the same number of rollouts, demonstrating substantially higher sample efficiency than existing sampling strategies.

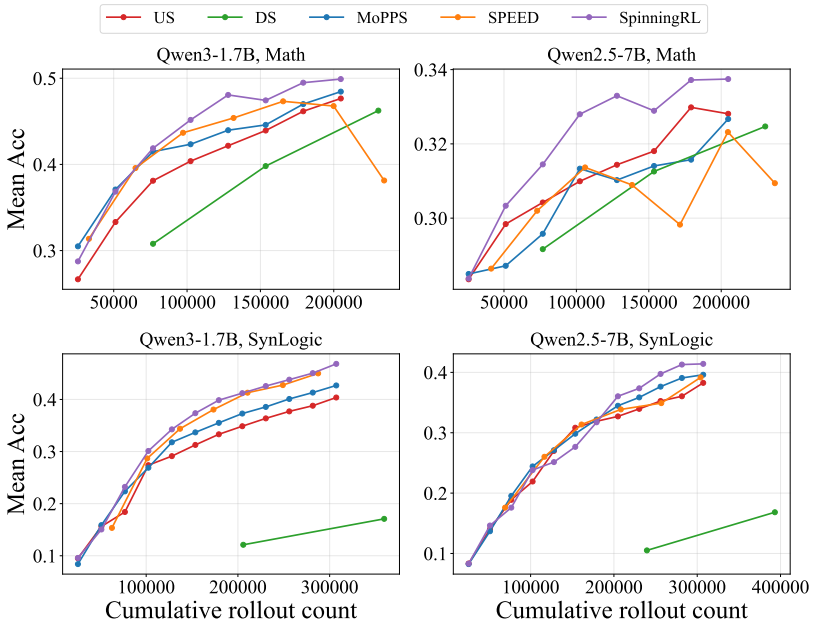


Figure 7: Sample-efficiency comparison under matched rollout budgets.

J AUTO-CURRICULUM LEARNING

According to the UCB score in Sec. 2.3, samples are selected based on both estimated reward gain and uncertainty. We refer to questions with high estimated gain as *exploitation samples*, and questions selected primarily for uncertainty as *exploration samples*. To better characterize the training dynamics, Figure 8 plots the average reward of these two groups at each training step[3]. Both groups improve over time, but exploitation samples consistently exhibit a smaller growth slope and remain

[3]Exploitation samples are identified by $N_t(q_i) > 0$, i.e., samples with historical rewards in the recent window.

at a relatively stable difficulty level, indicating that SPINNINGRL automatically maintains focus on the current gain-efficient region and forms an adaptive curriculum that matches evolving policy.

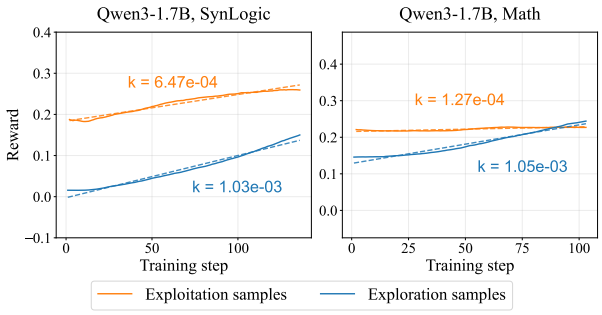
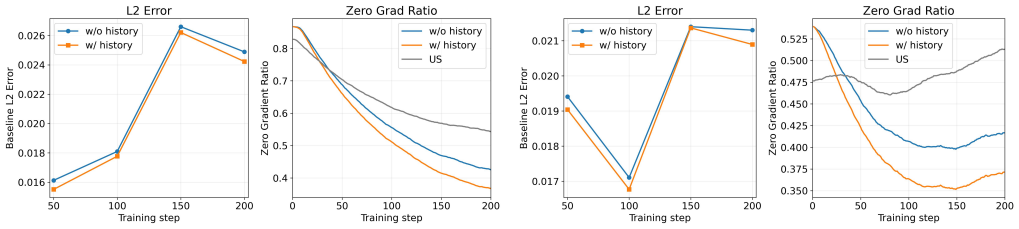


Figure 8: Mean reward of Exploration v.s. Exploitation samples.

K MORE RESULTS ON HISTORY-AWARE BASELINE

To validate the history-aware baseline estimation introduced in Sec. 2.4, we construct a stronger reference baseline by performing $N = 32$ rollouts on each sample in the training batch and use the resulting estimate as a proxy for ground truth. We then measure the ℓ_2 error of baseline estimation with and without the history-based moving average. As shown in Figure 9, the history-aware baseline consistently reduces estimation error and lowers the zero-advantage ratio throughout training, reflecting a variance–bias trade-off under policy drift.



(a) Qwen2.5-7B, Math.

(b) Qwen3-1.7B, Math.

Figure 9: Reduced variance in baseline estimation on Math task.

We also ablate the history-aware moving average by removing it from SPINNINGRL and by adding it to Uniform Sampling. Table 12 shows that this mechanism improves performance under both settings.

Table 12: Ablation of history-aware baseline on Qwen3-1.7B Math.

Method	History	Avg. Acc
US	×	46.2
	✓	47.2
SPINNINGRL	×	<u>49.2</u>
	✓	**49.9**

L PREDICTIVE VALIDITY OF THEORETICAL REWARD GAIN

To further examine whether the proposed potential reward gain is predictive of actual learning progress, we evaluate the correlation between the estimated gain $\widehat{G}_\beta(q)$ at the current policy and

the future reward improvement of the same prompt. Following the diagnostic setting in Sec. 2.2, we periodically sample 1,000 prompts every 100 training steps and measure their reward increase over the subsequent 30 training steps. Specifically, for a prompt q evaluated at training step t, we define its future reward gain as

$$\Delta R_{30,t}(q) = \bar{r}_{t+30}(q) - \bar{r}_t(q), \tag{18}$$

where $\bar{r}_t(q)$ denotes the empirical mean reward of prompt q at step t. Importantly, $\widehat{G}_{\beta,t}(q)$ is computed using only rollout information available up to step t, while $\Delta R_{30,t}(q)$ is evaluated exclusively from future rollouts. Thus, the analysis does not use any future information when computing the acquisition score.

We compare the predictive capability of the proposed reward-gain signal against representative learnability-based criteria, including the variance-based score $p(q)(1-p(q))$ and the distance-to-medium-difficulty score $|p(q) - 0.5|$. For each checkpoint, we compute the Pearson correlation between each acquisition score and the subsequent reward improvement $\Delta R_{30,t}(q)$. We additionally report Spearman rank correlation to measure whether the scores preserve the relative ordering of prompts by their future learning gain:

$$\rho_{\text{Pearson}} = \text{Corr}\big(s_t(q),\, \Delta R_{30,t}(q)\big), \tag{19}$$

where $s_t(q)$ denotes one of the acquisition scores. The correlations are computed over the same diagnostic prompt pool for all methods.

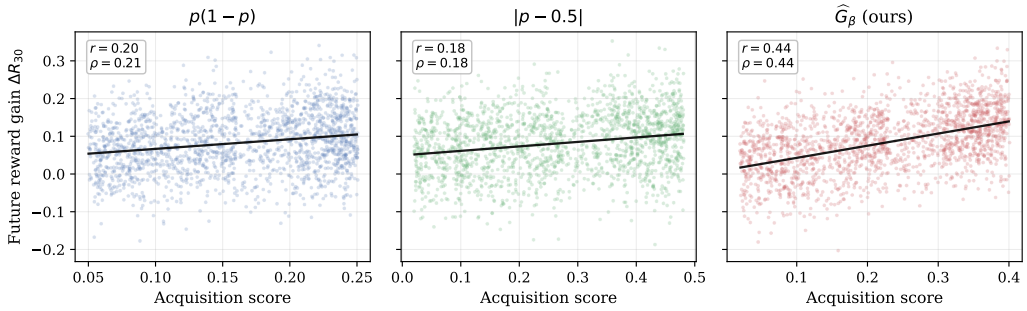


Figure 10: Predictive validity of different prompt acquisition scores. Each point corresponds to a diagnostic prompt sampled at a training checkpoint. The horizontal axis shows the acquisition score computed using information available at the current checkpoint, while the vertical axis shows the reward improvement over the subsequent 30 training steps. We compare (a) $p(1-p)$, (b) $|p-0.5|$, and (c) the proposed $\widehat{G}_\beta$. The Pearson correlation coefficient and the corresponding Spearman rank correlation are reported in each panel. The proposed reward-gain signal exhibits a stronger correlation with future reward improvement, indicating that it provides a more informative estimate of prompt-level learning potential.

Table 13 summarizes the correlation results across all evaluated checkpoints. In addition to the overall correlation computed by pooling all diagnostic prompts, we report the mean and standard deviation of the correlation coefficients across individual checkpoints.

Table 13: Correlation between acquisition scores and future 30-step reward improvement. Numbers are mean $\pm$ std across checkpoints.

Acquisition score	Pearson $r\uparrow$	Spearman $\rho\uparrow$
$p(1-p)$	0.20 ± 0.06	0.21 ± 0.06
$\lvert p-0.5\rvert$	0.18 ± 0.07	0.18 ± 0.07
$\widehat{G}_\beta$ (ours)	$\mathbf{0.44\pm0.04}$	$\mathbf{0.44\pm0.04}$

Overall, these results provide empirical evidence that the theoretical reward-gain signal is not merely a reparameterization of prompt difficulty, but also serves as a useful predictor of subsequent reward improvement under the evolving policy.