PROS: TOWARDS COMPUTE-EFFICIENT RLVR VIA ROLLOUT PREFIX REUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large reasoning models (LRMs) trained with *Reinforcement Learning with Verifiable Rewards* (RLVR) have achieved remarkable progress on complex reasoning tasks. However, RLVR heavily relies on on-policy rollout generation, whose cost grows rapidly with rollout length and model size, eventually becoming the training bottleneck. Our empirical analysis reveals that independent rollouts for the same query often share similar early steps, indicating substantial redundancy. To address this, we propose **PROS** (**P**refix **R**euse for **O**n-policy **S**ampling), a paradigm that reuses promising prefixes of historical rollouts in RLVR training. PROS appends these self-generated partial rollouts to the original queries to form *Augmented Queries*, which are then used as regular training inputs in subsequent iterations, thereby reducing redundant computation. To select training batch from augmented queries, PROS adopts a hierarchical Bayesian model to estimate their pass rates and prioritize those with the highest reward uncertainty. Experiments across diverse settings show that PROS consistently improves training efficiency and achieves higher accuracy than strong baselines. These results highlight PROS as a practical path toward scalable and compute-efficient RLVR. The source code is available in supplementary materials.

1 INTRODUCTION

Large reasoning models (LRMs) have achieved remarkable progress on complex tasks, especially in code generation and mathematical problem solving, in some cases even surpassing average human performance (OpenAI et al., 2024; Shao et al., 2024; DeepSeek-AI et al., 2025; Team et al., 2025a; Lambert et al., 2025). A key driver behind these advances is *Reinforcement Learning with Verifiable Rewards* (RLVR), where an LRM is optimized as a policy that generates chain-of-thoughts (CoTs) as trajectories and receives binary rewards on final answers from deterministic verifiers. This paradigm enables models to improve reasoning abilities via supervision from verifiable outcomes, offering a scalable path for learning from self-exploration.

Despite its promise, the RLVR paradigm is constrained by heavy reliance on on-policy rollout generation. Each training iteration of a RLVR algorithm consists of four steps: *Select* queries from a given dataset as a training batch, *Generate* several rollouts based on selected queries with the policy, *Verify* these rollouts with a deterministic verifier to obtain binary rewards, and *Update* parameters of the policy to increase the likelihood of high-reward actions via policy-gradient methods such as PPO(Schulman et al., 2017) or GRPO(Shao et al., 2024). As models tackle increasingly complex problems, the length of their chain-of-thoughts (CoTs) grows accordingly. Consequently, the cost of online rollout generation escalates and soon dominates the overall training time, making it one of the principal obstacles to further scaling RLVR.

Our empirical analysis reveals that, for a given query, independently sampled reasoning trajectories can be naturally organized into a tree structure like a branching search over the answer space as shown in Figure 1. Although their final answers may diverge, the early steps often share similar lines of reasoning. This observation indicates substantial redundancy: by reusing the prefixes of historical rollouts in the previous iteration, we can avoid repeatedly generating these near-duplicate initial steps and thereby saving a significant amount of computation.

Based on this observation, we introduce **PROS** (**P**refix **R**euse for **O**n-policy **S**ampling), a paradigm designed to make RLVR more compute-efficient for further scaling. In each training iteration, PROS

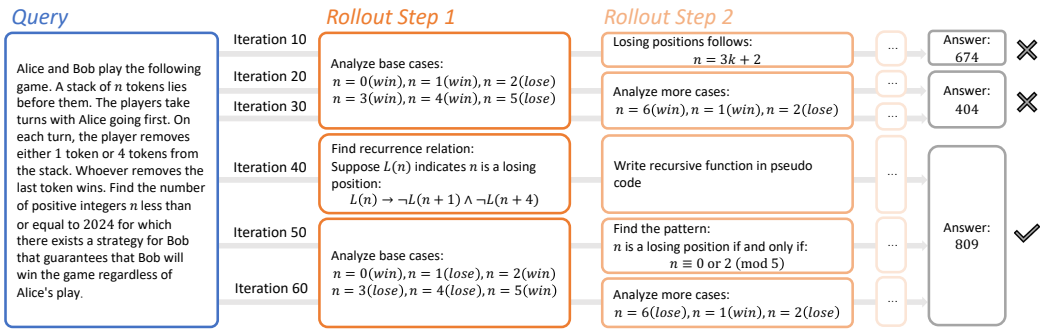

Figure 1: Rollouts for a query from different training iterations can naturally be organized into a tree structure, where their reasoning trajectories are highly similar in early steps and gradually diverge.

identifies high-quality prefixes of generated rollouts using readily available signals such as token-level entropy or value estimation from a critic. These rollout prefixes are then appended to their original queries to construct *Augmented Queries*. An augmented query preserves the task semantics of the original query while adding partial reasoning steps produced by the policy itself. It can be fed back into subsequent training iterations as a prompt, shortening on-policy generated reasoning steps required and thus reducing overall training compute. Beyond efficiency, PROS also enables a form of cross-iteration search pruning: it allows the policy to exploit those high-quality initial steps while avoiding useless early dead ends, thereby steering subsequent reasoning towards promising directions. This reallocates training compute to meaningful exploration in later reasoning steps, enabling the policy to acquire richer reasoning patterns.

As training progresses, more augmented queries are generated, and the dataset expands into a two-layer tree-structured hierarchy: each original query serves as a parent, and its derived augmented queries serve as children. A central challenge is selecting which augmented queries to train on. Bae et al. (2025) give a principle that queries with the highest reward variance exhibit the greatest learnability and provide the most informative training signals. Motivated by this, we instantiate a hierarchical Bayesian model over augmented queries, estimate per-query pass rate from historical reward statistics and prioritize those queries with high reward uncertainty (i.e. pass rate close to 0.5) (Hong et al., 2022a;b).

Essentially, PROS can be seamlessly integrated into most policy-gradient RLVR algorithms as a plugin. With the main algorithm logic unchanged, it only adds two additional steps, augmented query construction (Section 3) and augmented query selection (Section 4), both of which incur negligible computational overhead. In summary, this paper makes the following contributions:

- We propose **PROS**, a simple and general training paradigm that reuses high-value historical prefixes as Augmented Queries, reducing redundant generation and enhance policy exploitation.
- We further introduce a selection mechanism over all augmented queries that targets at the most valuable one for training, which is implemented via pass rate estimation by a hierarchical Bayesian model.
- PROS consistently outperforms strong baselines across diverse settings, achieving the best performance–cost trade-offs. Integrated as a plugin, it can further raise the upperbound of both PPO and GRPO, yielding an average improvements of +3.96 and +6.21 points on AIME24 and AMC23, respectively.

2 TRAINING BOTTLENECK ANALYSIS

RLVR Basics We begin by reviewing the process of *Reinforcement Learning with Verifiable Rewards* (RLVR) for training large reasoning models. For simplicity, we describe the setting with batch size 1, and denote the policy by π_θ. In each training iteration, a query q is selected from a training dataset, and π_θ autoregressively produces a chain of thought $y = [y_1, \ldots, y_T]$ conditioned on q. A deterministic verifier serves as an environment mapping (q, y) to a binary reward

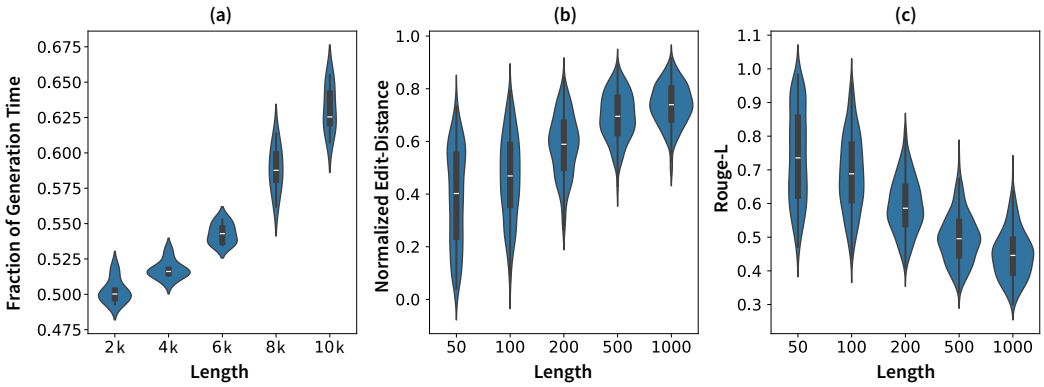

Figure 2: (a) Fraction of rollout generation time versus different max rollout length under fixed batch size and hardware. (b) Pairwise normalized edit distance versus prefix window size averaged over 64 rollouts per query. (c) Self-rouge versus prefix window size averaged over 64 rollouts per query.

$r = R(q, y) \in \{0, 1\}$, indicating whether the final answer matches the ground truth or satisfies a task-specific specification (e.g. passing all unit tests for code generation). And the objective of RLVR finetuning is to optimize the policy parameters θ to maximize the expected rewards, i.e. $\max_\theta \mathbb{E}_{q;y\sim\pi_\theta(\cdot|q)}[R(q, y)]$, which is commonly implemented via policy-gradient methods like PPO or GRPO.

Generation Share. As both the scale of LRMs and the length of CoTs grow, on-policy rollout generation gradually dominates the training time of RLVR algorithms. We perform a preliminary measurement to quantify the generation share (i.e. the fraction of wall-clock time spent in rollout generation) under varying max rollout length $L_{\max}$. As shown in Figure 2, the generation share rises rapidly as the maximum generation length increases. Currently, the scale of LRMs have already approached hundreds of billions, with context lengths more than hundreds of thousands tokens. This trend highlights a scalability issue, underscoring the need for methods to alleviate generation costs.

Redundancy in Early Reasoning Steps. We empirically observe strong similarity at the beginning of different rollouts for the same query across training iterations. Their reasoning trajectories naturally unfold like a branching search: the early steps share similar setup, while later steps gradually diverge for different reasoning directions. A specific case is presented in Figure 1. To further quantify this effect, we sample 64 independent rollouts per query on DAPO-Train dataset (Yu et al., 2025) and measure pairwise similarity on truncated prefixes of different lengths. Concretely, we compute the average of two pairwise similarity metrics: (i) normalized edit distance (i.e. EditDist/TotalLen), and (ii) ROUGE-L (Lin, 2004). As shown in Figure 2, shorter prefixes yield markedly higher similarity, indicating substantial redundancy in repeatedly generating near-duplicate initial reasoning steps.

Wasted Computation in Early Dead Ends. A second source of redundancy also follows from the tree structure of reasoning trajectories. Correct solution trajectories are sparse in the reasoning space. Naive multi-sampling may therefore start on branches that are truly dead ends. Subsequently, their remaining suffix contribute little training value. For example, in Figure 1, rollouts in iterations 10 to 30 make a mistake at the first step and still consume long suffixes. This unnecessary dead-end cost compounds as model scale and sequence length grow.

3 Augmented Query Construction

3.1 Overview

Building on the observations from Section 2, we introduce **PROS** (**P**refix **R**euse for **O**n-policy **S**ampling), a paradigm designed to mitigate redundant generation and wasted computation in RLVR.

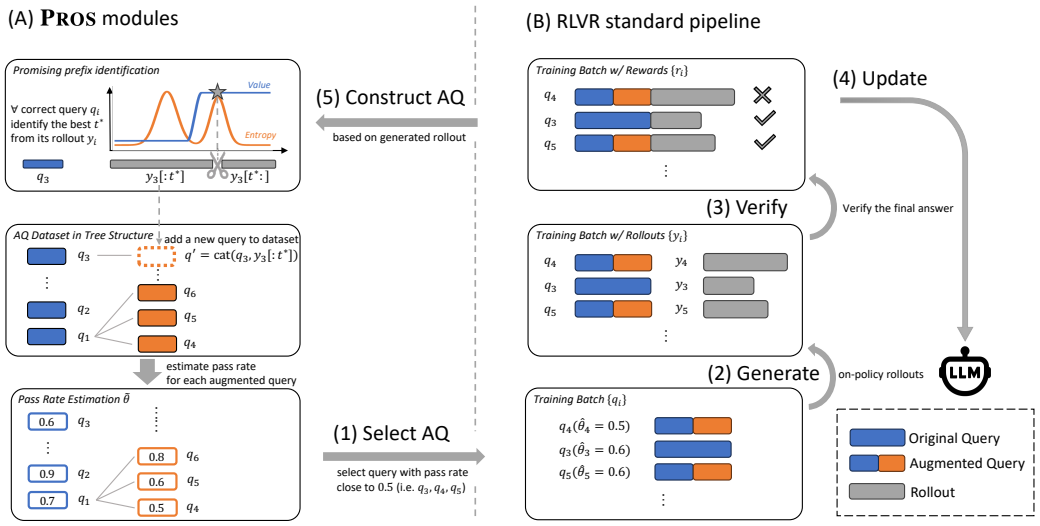

Figure 3: Overview of **PROS**. The right-hand side follows standard RLVR algorithms, while the left-hand side is the augmented query modules introduced by PROS. (1) Estimate the pass rate $\tilde{\theta}$ of each query and select those with highest uncertainty; (2) Generate rollouts $\{y_i\}$ for selected queries $\{q_i\}$; (3) Verify rollouts against ground truths; (4) Update the policy with rollouts and rewards; (5) Construct augmented queries by appending high-quality rollout prefixes to original queries (e.g. $q' = \text{concat}(q_3, y_3[:t^*])$) for future reuse.

Concretely, in each training iteration, after the rollouts are generated and verified, we identify high-quality prefixes within them. We then append these valuable prefixes to their original queries to create new *Augmented Queries* (AQ), which are then added to the training set as new query instances for subsequent training iterations. Over time, this augmentation process expands the training data into a two-layer tree-structured dataset: each original query serves as a parent node, and its derived augmented queries become child nodes.

During training, augmented queries are treated equivalently as normal queries. The policy generates continual reasoning trajectories conditioned on both the historical prefix and the original query. The reinforcement learning update then proceeds over the collected continuations as in standard RLVR. The only difference is that both credit assignment and gradient update are conducted only on the newly generated continuations without the reused historical prefixes. This way, the policy is reinforced for how it continues from the prefix, preventing overfitting to the previously generated reasoning steps. Figure 3 illustrates the overall PROS pipeline.

Notably, an augmented query preserves the task semantics of its original query, only providing unverified partial reasoning steps generated by the policy itself. It does not reveal any final answer and remains the on-policy nature of RLVR algorithm, in contrast to off-policy experience replay methods (Fedus et al., 2020; Schaul et al., 2016; Liang et al., 2021).

3.2 PROMISING PREFIX IDENTIFICATION

A crucial question in PROS is identifying which rollout prefixes are worth reusing. We employ a heuristic strategy based on both policy uncertainty and value signals, along with a length-based constraint, to construct augmented queries that are both informative and efficient.

We consider two signals to estimate a prefix's potential: uncertainty-based signal and value-based signal. As the uncertainty signal, we use the token-level entropy along the rollout, which serves as a proxy for uncertainty. A higher entropy indicates that the policy was less certain about the next steps, suggesting the prefix lies in an unexplored and informative region of the reasoning space. Such prefixes are candidates for reuse because they encourage exploration and mitigate over-confidence (Cui et al., 2025; Wang et al., 2025). Using entropy is almost free without extra computational overhead, since the RLVR algorithm already requires a forward pass to compute the log-probability at every timestep.

Besides entropy, we also leverage the readily available value function learned by critic model for those actor-critic methods. Prefixes with higher predicted value are more likely to lead to a correct final answer. So reusing them can help to exploit and focus computational costs on promising reasoning directions.

In practice, we reuse the prefix $y[:t^*] = [y_0, \ldots, y_{t^*}]$ with highest token-level entropy from each correct rollout y, i.e. $t^* = \arg\max_{t \in [0,T)} H(\pi(\cdot|y_{<t}, q))$. When the underlying RLVR algorithm is actor–critic, we additionally use the value function as a filter, restricting t^* to the top 10% of timesteps with the highest predicted values. In addition to these signals, we also impose simple length constraints to seek a balance between exploitation and exploration. Specifically, we limit the range of $t^* \in [\frac{1}{4}T, \frac{3}{4}T)$. Although coarse, this constraint ensures that the reused prefix yields a non-trivial reduction in generation time, while leaving sufficient room for the policy exploration without giving away too much of the solution.

4 Augmented Query Selection

4.1 Overview

As the augmented query dataset grows continuously across iterations, selecting a valuable training batch from a large amounts of AQs is non-trivial. Prior studies suggest that training is most effective when queries fall into an intermediate difficulty range rather than being trivially easy or impossibly hard (Yu et al., 2025; Razin et al., 2025; Vygotsky, 1978). Furthermore, Bae et al. (2025) theoretically demonstrates that queries with rollout pass rate close to 0.5 exhibit the strongest learnability signal. Both Yu et al. (2025) and Bae et al. (2025) adopt an online filtering mechanism, removing queries whose pass rates are too high or too low *after* on-policy rollout generation. However, such online filtering requires costly rollout generation for every query. To reduce this overhead, we instead estimate pass rates of augmented queries from historical observations on rewards via Bayesian inference, and we prioritize those queries with estimated pass rates near 0.5.

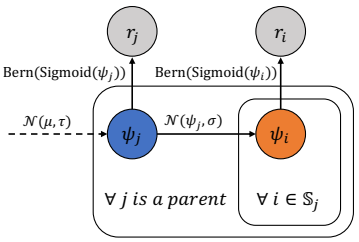

Figure 4: Overview of graphical model. r indicates reward of each query, following a Bernoulli controlled by log odds ψ. ψ_i of each augmented query i is conditionally Gaussian centered on its parent's ψ_j.

As stated before, the augmented query dataset forms a tree-structured hierarchy, where each original query serves as a parent while its derived augmented queries serve as children. To leverage the relationship between original query and its derived augmented queries, we initialize a two-layer logit–normal Bayesian model for pass rate estimation (Hong et al., 2022a;b). Let pass rate $\theta = \text{sigmoid}(\psi) = (1 + \exp(-\psi))^{-1}$, we suppose the log-odds of each parent par follows $\psi_{par} \sim \mathcal{N}(\mu, \tau^2)$ and the log-odds of its children i is conditionally Gaussian, $\psi_i \mid \psi_{par} \sim \mathcal{N}(\psi_{par}, \sigma^2)$. Each time a query is used for generating a rollout, a binary reward $r \sim \text{Bern}(\theta)$ is then observed, indicating whether the rollout passes verification. The graphical model is presented in Figure 4.

This formulation encodes the inductive bias that an augmented query solves the same underlying problem as its parent but begins from an intermediate reasoning state, so a child's pass rate should be correlated with that of its parent. The hierarchical structure enables information sharing between related queries and improves statistical efficiency.

Building on the Bayesian model above, we can infer the posterior distribution of latent log-odds by incorporating historical reward observations. Posterior samples $\{\tilde{\psi}\}$ are then drawn for each query, and the corresponding pass rates $\tilde{\theta} = \sigma(\tilde{\psi})$ are estimated for training batch selection.

4.2 Bayesian Inference with Pólya–Gamma Augmentation

In order to sample from the joint posterior, we derive the full conditional distributions for both parents and children, which allows Gibbs sampling. Specifically at iteration t , let $H_i = (n_i, s_i)$ denote the reward history of query i, where n_i is the number of times the query being selected and s_i the number of success times of its corresponding rollouts. Suppose the parent set of nodes is $\mathbb{F}$, and the children set of node j is $\mathbb{S}_j$. By Bayes' rule, the joint posterior admits a hierarchical chain

factorization:

$$p(\{\psi\}|H) \propto \prod_{j\in\mathbb{F}} \left[\underbrace{p(\psi_j)}_{\text{prior of parent}} \times \underbrace{p(H_j|\psi_j)}_{\text{likelihood of parent}} \times \prod_{i\in\mathbb{S}_j} \underbrace{p(\psi_i|\psi_j)}_{\text{prior of child}} \times \underbrace{p(H_i|\psi_i)}_{\text{likelihood of child}} \right]$$

A key challenge in posterior calculation above is the non-conjugacy of the Bernoulli likelihood and Gaussian prior. We address this by introducing Pólya–Gamma (PG) auxiliary variables following Polson et al. (2013); Dumitrascu et al. (2018), which render the calculation tractable.

We first derive the conditional posterior $p(\psi_i|H_i, \psi_{par})$ of a child i, given its parent ψ_{par} and history $H_i = (s_i, n_i)$,

$$p(\psi_i|H_i,\psi_{par}) \propto p(\psi_i|\psi_{par}) \cdot p(H_i|\psi_i) = p(\psi_i|\psi_{par}) 2^{-n_i} \int_0^\infty \exp(\kappa\psi_i - \frac{\omega_i\psi_i^2}{2}) p_{PG}(\omega_i|n_i,0) d\omega_i$$

where $\kappa = s_i - \frac{n_i}{2}$ and $\omega_i \sim PG(n_i, 0)$. The identity follows directly from Polson et al. (2013, Theorem 1). By introducing an auxiliary variable ω_i, the conditional posterior $p(\psi_i|H_i, \psi_{par}, \omega_i)$ becomes Gaussian and thus tractable, given that the prior $p(\psi_i|\psi_{par})$ is Gaussian. In particular, we have the following proposition:

Proposition 4.1. *Given the log-odds of parent ψ_{par} and the history $H_i = (s_i, n_i)$,*

$$\omega_i|\psi_i, H_i \sim \mathrm{PG}(n_i, \psi_i) \quad \psi_i|\omega_i, H_i, \psi_{par} \sim \mathcal{N}(m, V) \textit{ where } V = \frac{1}{\sigma^{-2}+\omega_i}, m = V \cdot (\sigma^{-2}\psi_{par} + \kappa)$$

Similarly, we introduce a PG variable ω_{par} for each parent ψ_{par} and derive its conditional posterior with the same augmentation method:

Proposition 4.2. *Given the log-odds of all children $\{\psi_k\}_k^{\mathbb{S}_{par}}$ and the history $H_{par} = (s_{par}, n_{par})$,*

$$\omega_{par}|\psi_{par}, H_{par} \sim \mathrm{PG}(n_{par}, \psi_{par}) \qquad \psi_{par}|\omega_{par}, H_{par}, \{\psi_k\}_k^{\mathbb{S}_{par}} \sim \mathcal{N}(m, V)$$

where $V = (\tau^{-2} + \omega_{par} + |\mathbb{S}_{par}|\sigma^{-2})^{-1}$, $m = V \cdot (\tau^{-2}\mu + \sigma^{-2}\sum_k \psi_k + \kappa_{par})$

Proposition 4.1 and 4.2 together define a simple Gibbs sampler for the joint posterior $p(\{\psi\}, \{\omega\}|H)$, from which we can efficiently sample $\tilde{\psi}$ and estimate pass rates $\tilde{\theta}$ for all augmented queries. The overall estimation process is presented in Algorithm 1. Detailed proofs in this section are presented in Appendix B.

Algorithm 1 Pass Rate Estimation for Augmented Query

Input: Log-odds of all queries from previous iteration $\{\psi\}$; History of all queries $\{(n,s)\}$; Number of Gibbs sweeps G; Model Hyperparameters μ, τ, σ

for $t=1$ **to** G **do**
  `// One Gibbs sweep`
  foreach *parent* $par \in \mathbb{F}$ **do**
    foreach *child* $i \in \mathbb{S}_{par}$ **do**
      `// Posterior sampling based on Proposition 4.1`
      sample $\omega_i \sim PG(n_i, \psi_i)$;
      $V \leftarrow (\sigma^{-2}+\omega_i)^{-1}, m \leftarrow V \cdot (\sigma^{-2}\psi_{par}+\kappa)$, sample $\psi_i \sim \mathcal{N}(m, V)$
    end
    `// Posterior sampling based on Proposition 4.2`
    sample $\omega_{par} \sim \mathrm{PG}(n_{par}, \psi_{par})$; $V \leftarrow (\tau^{-2}+\omega_{par}+|\mathbb{S}_{par}|\sigma^{-2})^{-1}, m \leftarrow V \cdot (\tau^{-2}\mu + \sigma^{-2}\sum_k \psi_k + \kappa_{par})$, sample $\psi_{par} \sim \mathcal{N}(m, V)$
  end
end

4.3 Exponential Forgetting for Non-stationary Environment

As training progresses, the policy evolves continuously, which leads to gradual shifts in the pass rate of queries. Therefore, we introduce an exponential forgetting mechanism. At each iteration,

Table 1: Comparison between PROS and other baselines under different settings. `#Tokens` denotes the average number of tokens generated per iteration (in millions), and `#Time` denotes the average GPU wall-clock time per iteration (in minutes).

	DAPO-Train				AIME-Old			
	AIME24	AMC23	#Tokens	#Time	AIME24	AMC23	#Tokens	#Time
PPO	28.23	73.63	10.24	9.17	31.35	57.85	4.39	6.06
w/ Dynamic Sampling	28.85	66.69	11.07	17.62	<u>33.96</u>	**69.21**	9.10	40.04
w/ Experience Replay	30.10	73.02	8.93	8.98	31.35	57.85	3.61	5.88
w/ Priority Sampling	<u>31.87</u>	73.25	11.35	9.76	31.46	60.82	4.15	5.91
w/ PROS-ablation	31.35	<u>73.93</u>	7.92	9.29	31.87	65.09	6.93	8.89
w/ PROS	**33.23**	**78.20**	8.49	9.77	**34.27**	<u>65.62</u>	5.00	8.09
GRPO	29.58	73.40	9.27	6.58	31.04	59.98	3.58	3.77
w/ Dynamic Sampling	<u>31.15</u>	<u>76.68</u>	11.16	19.45	29.58	**68.90**	9.22	45.38
w/ Experience Replay	29.90	74.62	7.60	6.36	31.15	59.98	3.14	4.14
w/ Priority Sampling	30.73	73.70	9.55	6.87	<u>33.44</u>	60.14	3.88	4.33
w/ PROS-ablation	30.52	75.91	7.87	7.00	<u>33.44</u>	63.41	5.88	6.26
w/ PROS	**34.17**	**78.28**	8.46	7.57	**34.38**	<u>67.53</u>	5.84	6.69

we scale down the historical statistics s_i and n_i of every augmented query by a forgetting factor $\lambda \in (0, 1)$ before incorporating new observations. This exponential forgetting ensures that more recent rewards exert greater influence on posterior updates, allowing the sampler to adapt to policy improvements.

5 EXPERIMENTS

5.1 SETTINGS

We conduct main experiments on the Qwen3-8B model, trained using PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024). For evaluation, we report *Pass@1* on two benchmarks: AIME 2024 and AMC 2023. To mitigate variance, we report averages over 32 and 16 independent runs on these datasets, respectively, following Hochlehnert et al. (2025).

Training configuration. We adopt two math reasoning datasets as training corpora: *DAPO-Train* is a large and diverse corpus covering broad domains of math problems(Yu et al., 2025); *AIME-Old* consists of all AIME problems prior to 2024, which is more relevant to the AIME24 benchmark. We train for 400 and 300 iterations on these datasets, respectively. At each iteration, we generate 8 rollouts per query. The batch size is 512 with mini-batch size 64, yielding 8 gradient updates per PPO epoch. The maximum response length is set to 6144 tokens.

Baselines. We compare our proposed **PROS** against several strong baselines: (1) *Vanilla* is the standard PPO/GRPO training algorithm. (2) *Dynamic Sampling* (Yu et al., 2025) constructs training batches by filtering queries whose on-policy rollouts are either all correct or all incorrect. (3) *Priority Sampling* (Team et al., 2025a;b) tracks the historical pass rate $\tilde{\theta}$ of each query and samples proportionally to $1-\tilde{\theta}$. (4) *Experience Replay* augment PPO/GRPO with a replay buffer with replay ratio set to $1/8$. We also apply truncated importance sampling following ACER (Wang et al., 2017). (5) PROS-*ablation*: a variant of PROS with query being randomly sampled, isolating the effect of proposed augmented query selection mechanism. Additional details are provided in Appendix C.

6 MAIN RESULTS

Overall performance. Table 1 summarizes the comparison between **PROS** and other baselines. PROS delivers consistent improvements across all four settings. Under PPO trained on DAPO-Train, it outperforms the vanilla baseline by $+5$ and achieves the best performance on both AIME24 and AMC23. Comparable gains are observed under GRPO. These results validate our hypothesis: reusing promising prefixes enhances exploitation, thereby improving overall performance. In contrast, *Experience Replay* improves efficiency but yields performance comparable to vanilla. *Dynamic Sampling* achieves strong results, particularly on AMC23, but requires 2–4$\times$ more wall-clock time

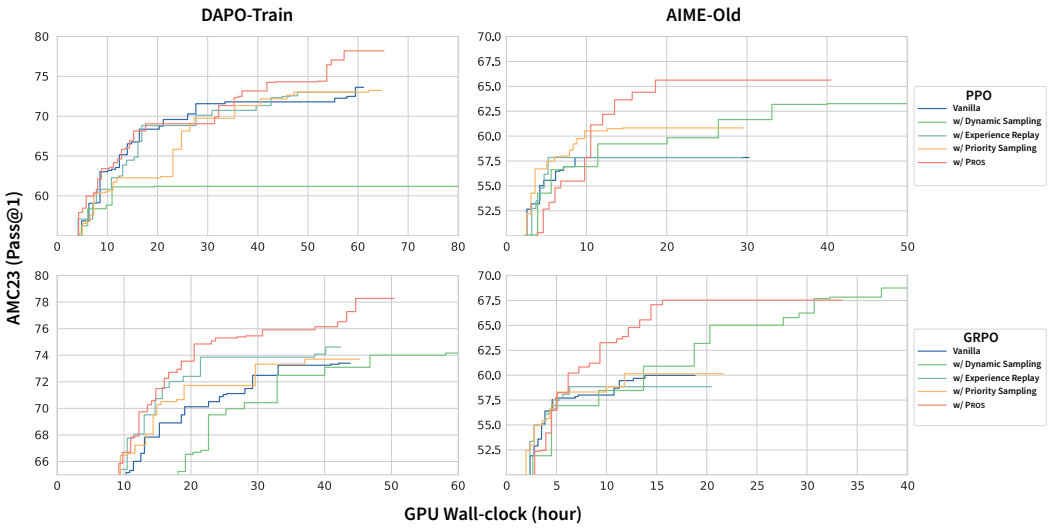

Figure 5: Training efficiency comparison among different methods. The x-axis denotes the training time (GPU hours), and the y-axis reports the best performance achieved up to that time.

Table 2: Comparison between PROS and other baselines on Qwen3-4B trained with AIME-Old.

	AIME24	AMC23	#Time
PPO	21.25	52.59	5.83
w/ dynamic	<u>25.73</u>	<u>61.19</u>	10.83
w/ replay	23.75	53.73	5.81
w/ prior	24.90	60.44	5.86
w/ PROS	**27.40**	**62.12**	6.21

Table 3: Ablation study on hyperparameters by varying σ and λ. PROS consistently surpasses Vanilla (59.98 on AMC23; 31.04 on AIME24).

$\lambda = 0.99$ (default)			$\sigma = 0.3$ (default)		
σ	AMC23	AIME24	λ	AMC23	AIME24
0.10	64.25	34.79	0.95	65.62	**35.94**
0.20	62.42	**36.98**	0.99	**67.53**	34.38
0.30	**67.53**	34.38	0.995	66.77	33.44

than PROS due to repeated rejection sampling. By comparison, PROS attains strong performance with only modest computational overhead. Most of the extra cost comes from its length scaling behavior (see below), which yields longer CoTs. To isolate the influence of augmented query sampling, we compare PROS with the PROS-*ablation* variant. The results show that prefix reuse alone already improves compute efficiency and achieves better performance. Moreover, incorporating the proposed augmented query selection mechanism consistently provides additional gains. We also provide an additional experiments on Qwen3-4B in Table 2, which exhibit similar trends.

Performance-Cost trade-offs. To further assess the compute efficiency of different methods, we analyze the performance growth with respect to GPU wall-clock time in Figure 5. It is shown that PROS consistently exhibits better efficiency under different settings. Compared to dynamic sampling, which achieves competitive performance but at prohibitive cost, PROS provides a significantly more favorable balance between efficiency and performance.

Influence of Hyper-parameters. We conduct an ablation study to evaluate the robustness of **PROS** with respect to key hyperparameters. In the Bayesian model (Section 4.1), the child prior $\psi_i \mid \psi_{\text{par}} \sim \mathcal{N}(\psi_{\text{par}}, \sigma^2)$ introduces a variance parameter σ^2 that controls similarity between parent and child nodes, while the temporal decay factor λ governs the rate of forgetting in pass rate estimation. We train PROS with GRPO on the AIME-Old dataset, varying both σ and λ. Table 3 reports Pass@1 results, and Figure 7 shows the estimation error of pass rates. The results show that PROS consistently brings improvements to vanilla GRPO across all hyperparameter settings, and the pass rate estimation remains robust, becoming increasingly accurate as training proceeds.

Length scaling. A key property of RLVR algorithms is their ability to benefit from increased reasoning lengths, which enables improved test-time scaling (Snell et al., 2024). We present the length scaling trends of different methods in Figure 6. Across all settings, the rollout length of PROS

Figure 6: Length scaling trends of different methods under different settings.

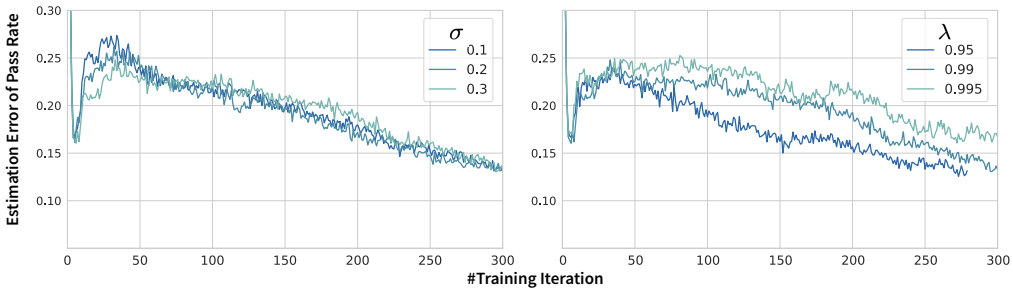

Figure 7: Average pass rate estimation error in PROS with different hyperparameters.

continues to scale up as the training goes. By contrast, vanilla PPO/GRPO, as well as their experience replay and prioritized sampling variants, suffer from length collapse on the AIME-Old dataset, limiting their ability to further exploitation.

7 Related Work

Our proposed method focuses on improving efficiency of rollout generation in RLVR. In traditional reinforcement learning, a common approach to reduce the cost of on-policy rollout generation is experience replay (Fedus et al., 2020; Schaul et al., 2016; Liang et al., 2021). For instance, PPO typically reuses the same batch of rollouts for multiple training epochs. However, He et al. (2025) show that, due to the complexity of large language models, repeatedly reusing the same data quickly leads to overfitting and entropy collapse, which prevents RLVR from scaling effectively in the long run. Another line of related work reduces task difficulty by providing partial reference solutions as hints during training, which requires costly human annotation(Xi et al., 2024; Liu et al., 2025a;b). In contrast, our approach derives partial solutions directly from the model's own past rollouts, removing the need for external reference solutions in complex reasoning settings. Thirdly, our augmented query selection also draws inspiration from curriculum learning (Soviany et al., 2022; Narvekar et al., 2020; Wang et al., 2021), by dynamically selecting training queries whose difficulty best matches the current policy. Several concurrent studies also investigate online data selection in RLVR to improve performance (Sun et al., 2025; Bae et al., 2025; Zheng et al., 2025; Qu et al., 2025). Our method differs in that it is specifically designed for the hierarchical augmented dataset.

8 Conclusion

We presented **PROS**, a prefix-reuse paradigm for RLVR training that constructs augmented queries from historical rollout and leverages a hierarchical Bayesian model for uncertainty-aware selection. Experiments demonstrate its consistent improvements in both efficiency and accuracy compared to strong baselines, highlighting the promise of PROS for compute-efficient training of LRMs.

While effective, our current design of augmented queries is still simple, relying on entropy and value signals that may introduce bias. Future work could investigate richer or more principled criteria for prefix identification, explore adaptive integration with other forms of uncertainty estimation, and extend prefix reuse to broader domains beyond mathematical reasoning.

Ethics Statement

This work focuses on improving the compute efficiency of reinforcement learning with verifiable rewards (RLVR) for large reasoning models. All experiments are conducted on publicly available datasets (AIME, AMC, DAPO) that do not contain sensitive personal information. Our approach does not introduce new data collection, and we adhere to the original licenses and intended use of these datasets.

We believe this work raises no additional ethical concerns beyond those already inherent in the study of large language models and reinforcement learning. We emphasize that our contributions are intended solely for research, and caution should be exercised when transferring these methods to real-world applications.

Reproducibility Statement

All experiments in this paper are conducted on publicly available datasets (AIME, AMC, and DAPO), which can be readily accessed and downloaded. Complete proofs of the theoretical propositions are provided in Appendix B. Detailed descriptions of experimental settings, including training configurations, hyperparameters, and baseline implementations, are provided in Appendix C to facilitate replication. The full source code repository together with reproducibility scripts is included in the supplementary materials.

References

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online Difficulty Filtering for Reasoning Oriented Reinforcement Learning, April 2025. URL http://arxiv.org/abs/2504.03380. arXiv:2504.03380 [cs].

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The Entropy Mechanism of Reinforcement Learning for Reasoning Language Models, May 2025. URL http://arxiv.org/abs/2505.22617. arXiv:2505.22617 [cs].

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, 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, Yanhong Xu, 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: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, January 2025. URL http://arxiv.org/abs/2501.12948. arXiv:2501.12948 [cs].

Bianca Dumitrascu, Karen Feng, and Barbara E Engelhardt. Pg-ts: Improved thompson sampling for logistic contextual bandits, 2018. URL https://arxiv.org/abs/1805.07458.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay, 2020. URL https://arxiv.org/abs/2007.06700.

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, 2025. URL https://arxiv.org/abs/2505.22312.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A Sober Look at Progress in Language Model Reasoning: Pitfalls and Paths to Reproducibility, April 2025. URL http://arxiv.org/abs/2504.07086. arXiv:2504.07086 [cs] version: 1.

Joey Hong, Branislav Kveton, Sumeet Katariya, Manzil Zaheer, and Mohammad Ghavamzadeh. Deep hierarchy in bandits, 2022a. URL https://arxiv.org/abs/2202.01454.

Joey Hong, Branislav Kveton, Manzil Zaheer, and Mohammad Ghavamzadeh. Hierarchical bayesian bandits, 2022b. URL https://arxiv.org/abs/2111.06929.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Xingxing Liang, Yang Ma, Yanghe Feng, and Zhong Liu. Ptr-ppo: Proximal policy optimization with prioritized trajectory replay, 2021. URL https://arxiv.org/abs/2112.03798.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Mingyang Liu, Gabriele Farina, and Asuman Ozdaglar. Uft: Unifying supervised and reinforcement fine-tuning, 2025a. URL https://arxiv.org/abs/2505.16984.

Ziru Liu, Cheng Gong, Xinyu Fu, Yaofang Liu, Ran Chen, Shoubo Hu, Suiyun Zhang, Rui Liu, Qingfu Zhang, and Dandan Tu. Ghpo: Adaptive guidance for stable and efficient llm reinforcement learning, 2025b. URL https://arxiv.org/abs/2507.10628.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey, 2020. URL https://arxiv.org/abs/2003.04960.

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, 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, 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, 2024. URL https://arxiv.org/abs/2412.16720.

Nicholas G. Polson, James G. Scott, and Jesse Windle. Bayesian inference for logistic models using polya-gamma latent variables, 2013. URL https://arxiv.org/abs/1205.0310.

Yun Qu, Qi Cheems Wang, Yixiu Mao, Vincent Tao Hu, and Xiangyang Ji. Can Prompt Difficulty be Online Predicted for Accelerating RL Finetuning of Reasoning Models?, July 2025. URL http://arxiv.org/abs/2507.04632. arXiv:2507.04632 [cs].

Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D. Lee, and Sanjeev Arora. What makes a reward model a good teacher? an optimization perspective, 2025. URL https://arxiv.org/abs/2503.15477.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016. URL https://arxiv.org/abs/1511.05952.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey, 2022. URL https://arxiv.org/abs/2101.10382.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving Data Efficiency for LLM Reinforcement Fine-tuning Through Difficulty-targeted Online Data Selection and Rollout Replay, June 2025. URL http://arxiv.org/abs/2506.05316. arXiv:2506.05316 [cs].

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, 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, and Zonghan Yang. Kimi k1.5: Scaling Reinforcement Learning with LLMs, January 2025a. URL http://arxiv.org/abs/2501.12599. arXiv:2501.12599 [cs] version: 1.

V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, Aohan Zeng, Baoxu Wang, Bin Chen, Boyan Shi, Changyu Pang, Chenhui Zhang, Da Yin, Fan Yang, Guoqing Chen, Jiazheng Xu, Jiale Zhu, Jiali Chen, Jing Chen, Jinhao Chen, Jinghao Lin, Jinjiang Wang, Junjie Chen, Leqi Lei, Letian Gong, Leyi Pan, Mingdao Liu, Mingde Xu, Mingzhi Zhang, Qinkai Zheng, Sheng Yang, Shi Zhong, Shiyu Huang, Shuyuan Zhao, Siyan Xue, Shangqin Tu, Shengbiao Meng, Tianshu Zhang, Tianwei Luo, Tianxiang Hao, Tianyu Tong, Wenkai Li, Wei Jia, Xiao Liu, Xiaohan Zhang, Xin Lyu, Xinyue Fan, Xuancheng Huang, Yanling Wang, Yadong Xue, Yanfeng Wang, Yanzi Wang, Yifan An, Yifan Du, Yiming Shi, Yiheng Huang, Yilin Niu, Yuan Wang, Yuanchang Yue, Yuchen Li, Yutao Zhang, Yuting Wang, Yu Wang, Yuxuan Zhang, Zhao Xue, Zhenyu Hou, Zhengxiao Du, Zihan Wang, Peng Zhang, Debing Liu, Bin Xu, Juanzi Li, Minlie Huang, Yuxiao Dong, and Jie Tang. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning, 2025b. URL https://arxiv.org/abs/2507.01006.

L. S. Vygotsky. *Mind in Society: Development of Higher Psychological Processes*. Harvard University Press, 1978. ISBN 9780674576285. URL http://www.jstor.org/stable/j.ctvjf9vz4.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 Rule: High-Entropy Minority Tokens Drive Effective Reinforcement Learning for LLM Reasoning, June 2025. URL http://arxiv.org/abs/2506.01939. arXiv:2506.01939 [cs].

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning, 2021. URL https://arxiv.org/abs/2010.13166.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay, 2017. URL https://arxiv.org/abs/1611.01224.

Zhiheng Xi, Wenxiang Chen, Boyang Hong, Senjie Jin, Rui Zheng, Wei He, Yiwen Ding, Shichun Liu, Xin Guo, Junzhe Wang, Honglin Guo, Wei Shen, Xiaoran Fan, Yuhao Zhou, Shihan Dou, Xiao Wang, Xinbo Zhang, Peng Sun, Tao Gui, Qi Zhang, and Xuanjing Huang. Training large language models for reasoning through reverse curriculum reinforcement learning, 2024. URL https://arxiv.org/abs/2402.05808.

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.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. VAPO: Efficient and Reliable Reinforcement Learning for Advanced Reasoning Tasks, April 2025. URL http://arxiv.org/abs/2504.05118. arXiv:2504.05118 [cs].

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, June 2025. URL http://arxiv.org/abs/2506.02177. arXiv:2506.02177 [cs].

A The Use of Large Language Models

Large language models (LLMs) were only used for editing and polishing the text of this paper, in order to improve clarity and fluency of presentation. They were not used for generating ideas, conducting experiments, analyzing results, or writing technical content.

B Proofs of Section 4.2

B.1 Problem Setup

We first review the problem and our proposed Bayesian model. Selection on the AQ tree can be casted as a hierarchical multi-armed bandit: each parent arm corresponds to an original query and its children correspond to its derived augmented queries. Each arm (i.e. query) inherently has a success rate θ. Pulling an arm yields a binary reward $r \sim \mathrm{Bern}(\theta)$, indicating whether one generated rollout from the query is correct.

We propose a two-layer logit–normal Bayesian model for success rate estimation. Let success rate $\theta = \mathrm{sigmoid}(\psi) = (1+\exp(-\psi))^{-1}$. We suppose the log-odds of each parent par follows $\psi_{par} \sim \mathcal{N}(\mu, \tau^2)$ and the log-odds of its children i is conditionally Gaussian, $\psi_i \mid \psi_{par} \sim \mathcal{N}(\psi_{par}, \sigma^2)$.

The objective is to leverage the historical observations on rewards to derive the joint posterior conditioned on history, from which we can sample log-odds parameter $\tilde{\psi}$ of both parents and children and then estimate their success rates $\tilde{\theta} = \mathrm{sigmoid}(\tilde{\psi})$.

Specifically at iteration t , let $H_i = (n_i, s_i)$ denote the reward history of query i, where n_i is the number of the query being selected and s_i the number of successes of its corresponding rollouts. Suppose the father set of nodes is $\mathbb{F}$, and the children set of node j is $\mathbb{S}_j$. By Bayes' rule, the joint posterior admits a hierarchical chain factorization:

$$p(\{\psi\}|H) \propto \prod_{j\in\mathbb{F}} \left[\underbrace{p(\psi_j)}_{\text{prior of parent}} \times \underbrace{p(H_j|\psi_j)}_{\text{likelihood of parent}} \times \prod_{i\in\mathbb{S}_j} \underbrace{p(\psi_i|\psi_j)}_{\text{prior of child}} \times \underbrace{p(H_i|\psi_i)}_{\text{likelihood of child}} \right] \tag{1}$$

In the following, we will apply Pólya–Gamma augmentation to render the posterior sampling tractable by sample from an augmented posterior $p(\{\psi\}, \{\omega\}|H)$, which has a single Gibbs sampler.

B.2 Proofs of Proposition 4.1

Polson et al. (2013) gives the following lemma about PG distribution:

Lemma B.1. *(Pólya-Gamma identity) Let $p(\omega)$ denote the density of the random variable $\omega \sim PG(b, 0)$, $b > 0$. Then the following integral identity holds for all $a \in \mathbb{R}$:*

$$\frac{\exp(\psi)^a}{(1+\exp(\psi))^b} = 2^{-b}\exp((a-b/2)\psi)\int_0^\infty \exp(-\omega\psi^2/2)p(\omega)d\omega$$

We first derive the conditional posterior $p(\psi_i|H_i, \psi_{par})$ of a child i, given its parent ψ_{par} and history $H_i = (s_i, n_i)$,

$$\begin{aligned}
&p(\psi_i|H_i, \psi_{par}) \\
\propto &p(\psi_i|\psi_{par}) \cdot p(H_i|\psi_i) \\
= &p(\psi_i|\psi_{par}) \cdot p(s_i, n_i|\psi_i) \\
= &p(\psi_i|\psi_{par}) \cdot \frac{\exp(\psi_i)^{s_i}}{(1+\exp(\psi_i))^{n_i}} \\
= &p(\psi_i|\psi_{par}) \cdot 2^{-n_i}\int_0^\infty \exp(\kappa\psi_i - \frac{\omega_i\psi_i^2}{2})p_{PG}(\omega_i|n_i, 0)d\omega_i
\end{aligned}$$

where $\kappa = s_i - \frac{n_i}{2}$ and $\omega_i \sim PG(n_i, 0)$. The identity follows directly from Lemma B.1. Now that we have the following identities:

$$p(\psi_i, \omega_i|H_i, \psi_{par}) \propto p(\psi_i|\psi_{par}) \cdot \exp(\kappa\psi_i - \frac{\omega_i\psi_i^2}{2})p_{PG}(\omega_i|n_i, 0)$$

$$p(\omega_i|\psi_i, H_i, \psi_{par}) = \frac{\exp(-\frac{\omega_i\psi_i^2}{2})p_{PG}(\omega_i|n_i, 0)}{\int_0^\infty \exp(-\frac{\omega_i\psi_i^2}{2})p_{PG}(\omega_i|n_i, 0)d\omega_i} = p_{PG}(\omega_i|n_i, \psi_i)$$

The second identity follows from the definition of Pólya-Gamma probability density. The full conditional posterior of ψ_i then follows:

$$p(\psi_i|H_i, \psi_{par}, \omega_i) = p(\psi_i, \omega_i|H_i, \psi_{par})/p(\omega_i|H_i) \propto p(\psi_i|\psi_{par}) \cdot \exp(\kappa\psi_i - \frac{\omega_i\psi_i^2}{2})$$

Given that the prior $\psi_i|\psi_{par} \sim \mathcal{N}(\psi_{par}, \sigma^2)$, ψ_i's full conditional posterior $p(\psi_i|H_i, \psi_{par}, \omega_i)$ is also a Gaussian:

$$\psi_i|H_i, \psi_{par}, \omega_i \sim \mathcal{N}(m, V), \text{ where } V = \frac{1}{\sigma^{-2}+\omega_i},\ m = V \cdot (\sigma^{-2}\psi_{par} + \kappa)$$

And the proof of Proposition 4.1 ends.

B.3 Proofs of Proposition 4.1

Similarly, we can also derive parents' conditional posteriors $p(\psi_{par}|H_{par}, \{\psi_k\}_k^{\mathbb{S}_{par}})$ by introducing PG variables:

$$\begin{aligned}
&p(\psi_{par}|H_{par}, \{\psi_k\}_k^{\mathbb{S}_{par}}) \propto p(\psi_{par}) \cdot p(s_{par}, n_{par}|\psi_{par}) \cdot \prod_{k\in\mathbb{S}_{par}} p(\psi_k|\psi_{par}) \\
= &\left(p(\psi_{par}) \cdot \prod_{k\in\mathbb{S}_{par}} p(\psi_k|\psi_{par})\right) \cdot 2^{-n_{par}}\int_0^\infty \exp(\kappa_{par}\psi_{par} - \frac{\omega_{par}\psi_{par}^2}{2})p_{PG}(\omega_{par}|n_{par}, 0)d\omega_{par}
\end{aligned}$$

where $\kappa_{par} = s_{par} - \frac{n_{par}}{2}$, $\omega_{par} \sim PG(n_{par}, 0)$. The identity also follows from Lemma B.1. Based on this, we can easily derive the following conditional posteriors:

$$p(\psi_{par},\omega_{par}|H_{par},\{\psi_k\}_k^{\mathbb{S}_{par}}) \propto \left(p(\psi_{par}) \cdot \prod_{k\in\mathbb{S}_{par}} p(\psi_k|\psi_{par}) \right) \exp(\kappa_{par}\psi_{par} - \frac{\omega_{par}\psi_{par}^2}{2}) p_{PG}(\omega_{par}|n_{par},0)$$

$$p(\omega_{par}|\psi_{par},H_{par}) = \frac{\exp(-\frac{\omega_{par}\psi_{par}^2}{2}) p_{PG}(\omega_{par}|n_{par},0)}{\int_0^\infty \exp(-\frac{\omega_{par}\psi_{par}^2}{2}) p_{PG}(\omega_{par}|n_{par},0)} = p_{PG}(\omega_{par}|n_{par},\psi_{par})$$

$$p(\psi_{par}|\omega_{par},H_{par},\{\psi_k\}_k^{\mathbb{S}_{par}}) \propto \left(p(\psi_{par}) \cdot \prod_{k\in\mathbb{S}_{par}} p(\psi_k|\psi_{par}) \right) \exp(\kappa_{par}\psi_{par} - \frac{\omega_{par}\psi_{par}^2}{2})$$

Given that prior $\psi_{par} \sim \mathcal{N}(\mu, \tau^2)$, $\psi_k|\psi_{par} \sim \mathcal{N}(\psi_{par}, \sigma^2)$, ψ_{par}'s full conditional posterior is also a Gaussian:

$$\psi_{par}|\omega_{par}, H_{par}, \{\psi_k\}_k^{\mathbb{S}_{par}} \sim \mathcal{N}(m, V)$$

, where $V = \frac{1}{\tau^{-2}+\omega_{par}+|\mathbb{S}_{par}|\sigma^{-2}}, m = V \cdot (\tau^{-2}\mu_0 + \sigma^{-2}\sum_k \psi_k + \kappa_{par})$. And the proof of Proposition 4.2 ends.

C Experimental Settings

The main experiments are conduct on PPO and GRPO with implementation in verl[1]. At each training iteration, we generate 8 rollouts per query with temperature set to 1.0. We use a batch size of 512 and a mini-batch size of 64, yielding 8 gradient updates per PPO epoch. The maximum response length is set to 6144 tokens. We adopt the clip-higher strategy ($\epsilon_{high} = 0.28$) and the overlong reward shaping ($L_{cache} = 1024$) introduced by Yu et al. (2025), but do not apply additional KL regularization or entropy loss. Specifically for PPO, we adopt the decoupled-GAE and length-adaptive GAE proposed in VAPO (Yue et al., 2025). We also conduct value pretraining for twenty iterations following VAPO. For dynamic sampling baseline, we reuse the implementation code in verl. For prioritized sampling, we tracks the number of rollouts being generated for each query and the number of success times within these rollouts to calculate the pass rates. Similar to our proposed PROST, we also adopt a exponential decay of $\lambda = 0.9$. For experience replay, we only reuse the rollouts from the last iteration to avoid large gap. In specific, we use 64×8 rollouts from replay buffer, while the rest of $(512 - 64) \times 8$ rollouts are generated by current policy. The truncation threshold for truncated importance sampling is set to 10 following Wang et al. (2017). For PROST, we adopt $\mu = 0, \tau = 1.5$ to ensure a near uniform prior of pass rate θ. The exponential discounting factor $\lambda = 0.99$ and the variance of children prior $\sigma = 0.3$ by default. We also apply a diversity regularization in the selection stage that each query appears at most one time within any consequential K training iterations. To ensure a fair comparison with respect to training efficiency and GPU wall-clock, we apply identical engineering hyperparameters to all methods, such as `gpu_memory_utilization` for inference engine, `max_token_len_per_gpu` for dynamic batching, etc.

[1] `https://github.com/volcengine/verl`