

Skip-Connected Rollouts for Mid-Trajectory Rewards in GRPO

Fengwei Teng^{1,*} Jinyi Bai^{2,*} Xinhao Yao²
Demi Wang³ Jiahao Zhao² Zhijiang Guo⁴

¹Mohamed bin Zayed University of Artificial Intelligence

²Renmin University of China ³Carnegie Mellon University

⁴The Hong Kong University of Science and Technology (Guangzhou)

fengwei.teng@mbzuai.ac.ae

*Equal contribution

Abstract

Group Relative Policy Optimization (GRPO) and its variants are effective for reinforcement learning with verifiable outcome rewards, but incorporating intermediate credit remains difficult. We show that, under practical sampling budgets, token-level Monte Carlo estimates at early reasoning positions have high variance; group-relative normalization turns this noise into frequent advantage-sign errors and wrong-direction gradients. Later estimates instead approach the terminal reward and add little information. We propose SCRoll, a rollout-side module that adds randomized single-boundary segment rewards without changing the downstream optimizer. An upstream stream generates an early-stopped segment s , and a downstream stream samples G continuations conditioned on $[s, q]$, placing the segment before the original problem. The shared downstream group supplies both terminal outcomes for any group-relative objective and the upstream segment’s Monte Carlo reward, avoiding a separate rollout group; a single-stream objective optimizes the upstream segment. Across Qwen2.5-Math-7B and Llama-3.2-3B-Instruct, SCRoll improves most pairings with GRPO, GSPO, DAPO, and CISPO, with the clearest gains for CISPO. A decoupled two-process baseline reaches comparable performance only with substantially more generated tokens, while SCRoll performs better under a matched token budget.

1 Introduction

Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024; Hu et al., 2025; Ahmadian et al., 2024; Li et al., 2024) has emerged as the dominant paradigm for enhancing LLM reasoning, with Group Relative Policy Optimization (GRPO) and its variants establishing a now-standard rollout interface: sample G responses per prompt and compute group-relative advantages from outcome rewards. This interface is stable and inexpensive,

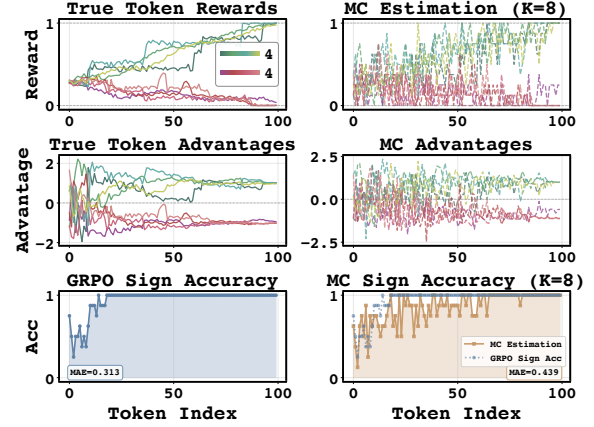


Figure 1: Monte Carlo estimation of token-level rewards under a practical sampling budget $K = 8$. We show a balanced 4/8-success trial selected deterministically as the closest to the component-wise median sign accuracy and MAE among 94 balanced trials from seeds 0–999. Top: true rewards and MC estimates; middle: the corresponding advantages; bottom: sign accuracy against the true token advantages. Averaged over tokens, outcome-only GRPO versus MC attains 0.940 versus 0.859 sign accuracy and 0.313 versus 0.439 MAE. Additional cases from 0/8 to 8/8 are in Section A.3.

yet it offers no slot for intermediate credit at training time. Approaches to finer-grained credit alter this interface in distinct ways: **① optimizer or advantage redesign**, including sequence- or segment-level PPO value estimation and critic-free oracle-conditioned recursion (Wang et al., 2026; Gong et al., 2026; Li et al., 2026); **② reward redesign**, including implicit online rewards from outcome-labeled rollouts, analytic conversion of outcomes into process supervision, deterministic process verification, and learned PRM scores (Cui et al., 2025; Yao et al., 2026; Pronesti et al., 2026; Ding et al., 2026; Zou et al., 2025); or **③ search-oriented training**, where a dense reward is learned jointly for RL and search, while $\max@k$ directly aligns the policy with Best-of- N sampling (Jin et al., 2025; Bagirov et al., 2025). These alternatives make dif-

ferent trade-offs rather than sharing a single reward-model pipeline. We ask a narrower interface-level question: **can intermediate credit be added on top of the existing group-relative rollout without replacing the downstream optimizer or invoking per-token branching?**

The most direct way to add intermediate credit on top of this interface is token-level Monte Carlo: from each intermediate position, sample additional continuations and aggregate their outcome rewards to estimate a token-level reward (Yue et al., 2025). Probing this naive route reveals not a uniform failure but a sharp asymmetry (Figure 1). Even under substantial sampling budgets, MC estimates at **early** reasoning positions suffer from high variance. The downstream effect is more pernicious than symmetric noise: once these noisy estimates are normalized into group-relative advantages, frequent sign flips translate the variance into systematically wrong-direction policy gradients (Section A.3). In the balanced regime, this yields lower sign accuracy and higher MAE than the outcome-only GRPO baseline. At **late** positions, by contrast, the same MC estimates naturally converge to the trajectory’s outcome reward, since the remaining uncertainty leaves little room for divergent continuations. The implication is structural rather than budgetary: the early-position regime is where group-relative MC breaks down, while the late-position regime is exactly where outcome-only GRPO already provides a reliable posterior, pointing at an asymmetric design space the existing interface has not exploited.

Most fine-grained methods are not designed around this positional finite-sample asymmetry. Multi-stage methods structure reasoning into explicit attempts or stages (Gandhi et al., 2025), while other approaches change reward, advantage, or objective construction. Our target is narrower: preserve the downstream group-relative objective and concentrate extra Monte Carlo sampling at a single intermediate boundary. Full token-level Monte Carlo would require prohibitive sampling to make early positions reliable. The asymmetry instead suggests that the early-position failure is *localized*: at late positions, MC estimates already coincide with the outcome reward that group-relative optimizers naturally consume. **This invites a design that exploits the asymmetry directly: route different parts of the trajectory to different optimizers.**

Acting on this insight, we propose SCRoll: we

instantiate Monte Carlo sampling at a *single* intermediate position, namely the boundary between an upstream reasoning segment and G downstream continuations, and assign each part to the optimizer suited to its estimation regime. **① Upstream:** the early-stopped segment s is scored by the Monte Carlo aggregation of the downstream outcomes and trained with a sample-efficient single-stream optimizer (Xu and Ding, 2025), sidestepping group-relative estimation in the regime where it would fail. **② Downstream:** G continuations are sampled from a **skip connection** $[s, q]$ that places the segment before the original problem. This differs deliberately from pure continuation from the prefix (q, s) : reintroducing q after s keeps the upstream segment influential while preserving access to the original problem, reducing the homogeneous-continuation regime in which downstream outcomes collapse and group-relative advantages vanish. Two properties follow from this rollout design. **Loss-side compatibility** falls out structurally: the downstream slot is just an ordinary group-relative loss. **Sampling reuse** also follows: the same G downstream continuations serve as both the downstream group and the upstream MC aggregation, avoiding the per-token branching cost of full token-level Monte Carlo. **③ Efficient realization** requires a custom rollout wrapper on top of verl: it enforces block-aligned split points, routes the downstream requests with identical $[s, q]$ prompts to vLLM with prefix caching enabled, and requires no vLLM engine-side modification. Our contributions are:

- While probing token-level Monte Carlo as a route to fine-grained credit on group-relative rollouts, we observe a sharp asymmetry under practical sampling budgets: at early positions, the unbiased MC reward estimator suffers finite-sample variance large enough that the resulting advantage signals fail to improve over outcome-only GRPO (Figure 1); at late positions, MC estimates already coincide with the outcome reward that group-relative optimizers naturally use. This empirically observed asymmetry motivates a structural alternative to simply increasing the sampling budget.
- We realize this structural alternative as SCRoll, a rollout-side module that augments group-relative policy optimizers with skip-conditioned segment rewards, via a skip-connected upstream/downstream rollout schema and a sample-efficient up-

stream optimizer that leaves the downstream loss untouched.

- Experiments across four representative group-relative optimizers (GRPO, GSPO, DAPO, CISPO) show that SCRoll improves most pairings on in-domain mathematics, with the strongest and most consistent gains on the stronger optimizers (CISPO, DAPO); on out-of-domain reasoning and code, CISPO + SCRoll retains gains across both base models and DAPO + SCRoll retains them on Qwen2.5-Math-7B, while adaptation quality varies with the choice of base optimizer.

2 Related Work

2.1 Group-Based Policy Optimization

Group Relative Policy Optimization (GRPO) has emerged as a dominant paradigm for reinforcement learning from verifiable rewards (Shao et al., 2024; Guo et al., 2025; Yu et al., 2025b; Zheng et al., 2025; Wu et al., 2025; Yao et al., 2025; Jain et al., 2025b). By sampling multiple responses within a group and computing relative advantages, GRPO eliminates the need for explicit value models while effectively guiding policy updates. Recent research has systematically explored and optimized various components within GRPO (Khatri et al., 2025; He et al., 2025b).

Variants of GRPO explore different combinations of design choices within the framework. ❶ *Reward and advantage computation* encompasses methods that improve sample efficiency through decoupled reward modeling and modified advantage computation, while also modeling the influence of entropy mechanisms and step-level fine-grained information on training dynamics (Ahmadian et al., 2024; Liu et al., 2025b; Xiong et al., 2025; Li et al., 2025a; Jain et al., 2025b; He et al., 2025c,a; Chen et al., 2025c). ❷ *Policy ratio design and aggregation* includes clipping mechanisms that enhance training stability through dynamic token-level bounds and gradient-preserving designs (Yu et al., 2025b; Yang et al., 2025b), as well as normalization strategies that stabilize policy updates through ratio normalization and geometric mean aggregation, or reduce sensitivity to outliers through group expectation sampling (Zhao et al., 2025; Zhang et al., 2025a; Sheng et al., 2025). ❸ *Alternative baseline estimation* addresses the limitation that group-relative baselines require multiple concurrent samples per prompt. SPO (Xu and Ding,

2025) replaces the *spatial* baseline (group mean from concurrent rollouts) with a *temporal* baseline via per-prompt value trackers, enabling optimization when only a single sample is available, which motivates our adoption of temporal baselines for upstream optimization where only a single segment per prompt enters the loss after median-NLL selection. Among these design choices, we are particularly interested in the construction of step-level information.

2.2 Fine-Grained Reward/Advantage

A well-known characteristic of vanilla GRPO is its uniform credit assignment: the same advantage is distributed equally across all tokens in a response, which may lead to length bias and inefficiency (Liu et al., 2025a). Recent work obtains finer-grained credit through several mechanisms. ❶ *Algorithm-side methods* redistribute or reweight trajectory feedback through Bellman or likelihood decompositions, entropy signals, reward redistribution, length control, or selection of high-entropy and pivotal tokens (Wen et al., 2024; Tan et al., 2025; Xie et al., 2025; Deng et al., 2025; Li et al., 2025b; Liu et al., 2025a; Lee and Tong, 2025; Wang et al., 2025b,a; Yu et al., 2025a). Process signals themselves also come from different sources: PRIME updates implicit rewards online from policy rollouts and outcome labels, SPRO derives them from the policy, and GRPO can be interpreted as inducing an implicit PRM under token-overlap assumptions (Cui et al., 2025; Fei et al., 2025; Sullivan and Koller, 2025). Explicit learned evaluators instead include discriminative token-level reward models, outcome-gated reasoning-process RMs such as Posterior-GRPO, and MCTS-trained step-level reward models (Chen et al., 2025a; Fan et al., 2025; Ma et al., 2025). ❷ *Sampling-based methods* estimate prefix values from branched continuations or fit advantages from separately sampled trajectories (Kazemnejad et al., 2024; Brantley et al., 2025); TEPO instead maps group rewards to token-level aggregation through Markov likelihood (Lin et al., 2025). These mechanisms trade modeling and estimation structure against computation, and controlled zero-reward experiments show that dense-credit baselines can still stall when no successful rollout is sampled (Prakash and Buvanesh, 2025).

2.3 Multi-Stage Pipeline Designs

GRPO-based methods also integrate with pipeline designs to enhance training effectiveness (Yan et al., 2025; MiniMax, 2025). Beyond single-response optimization discussed above, multi-stage pipelines involve multiple responses or attempts to solve a task. ❶ *Structured decomposition* exploits multi-response generation to improve credit assignment, organizing multiple rollouts into search trees to track retrospective credit and identify pivotal nodes, assigning credit to high-level thought templates across hierarchical reasoning traces, or extending Monte Carlo sampling with tree-based trajectory organization for unbiased value estimation over multiple attempts (Zhang et al., 2025b; Yang et al., 2025a; Kazemnejad et al., 2024; Brantley et al., 2025). ❷ *Internal feedback* constructs multi-turn refinement loops where models learn from their own previous outputs, employing multi-stage RL training with reward shaping to prevent behavior collapse, using layered generation processes where later layers explicitly correct earlier outputs under implicit process rewards, or introducing verify-then-revise mechanisms that alternate between policy and generative verifier roles (Kumar et al., 2025; Ding et al., 2025; Jiang et al., 2025; Ekbote et al., 2025; Li et al., 2025c). ❸ *External signals* guide multi-stage reasoning through curriculum or data generation, utilizing challenger-solver co-evolution patterns that generate questions at the solver’s capability frontier, formulating curriculum selection as non-stationary multi-armed bandits for adaptive difficulty scheduling, updating model parameters during inference using pseudo-labels from majority voting, or enabling self-play through questioner-responder-verifier role alternation (Huang et al., 2025; Chen et al., 2025b; Zuo et al., 2025; Yang et al., 2025c).

Our work differs from these approaches in that, while adopting an upstream-downstream two-stage generation pipeline during training rollouts, it introduces no external verifier or process reward model to guide the policy model. Meanwhile, we still adopt a sampling-based approach to obtain segment-level rewards, but by instantiating upstream-downstream sampling at a single position and reusing the same G downstream continuations for both downstream group-relative optimization and upstream Monte Carlo aggregation, we avoid the per-token branching cost of full token-level Monte Carlo while achieving stage-wise informa-

tion density.

3 Methodology

Figure 2 illustrates the overall SCRoll module. Section 3.1 presents the skip-connected upstream/downstream rollout schema; Section 3.2 formalizes the asymmetric two-stream optimization in which the upstream stream is trained with a sample-efficient single-stream optimizer and the downstream stream is left to any group-relative algorithm the user adopts, making SCRoll a loss-side plug-in.

3.1 Monte Carlo Trajectory Sampling

Ideal Token-Level Reward Estimation for Analysis. To analyze fine-grained credit for reasoning step quality, we consider token-level Monte Carlo reward estimation through trajectory sampling. For any token position t in a reasoning trajectory $o_{1:T}$, we estimate token-level credit by sampling K continuations from the prefix $o_{1:t}$ and averaging their outcome rewards:

$$\mathcal{R}_{\text{MC}}(o_t, a) = \frac{1}{K} \sum_{k=1}^K \mathcal{R}_{\text{score}}(o^{(k)}, a), \quad (1)$$

where $o^{(k)} \sim \pi_{\theta}(\cdot | o_{1:t})$,

and $\mathcal{R}_{\text{score}}(o, a)$ is a simple scoring function based on outcome correctness. In this idealized formulation, each token receives a distinct reward signal reflecting how well its prefix facilitates correct completions. While theoretically appealing for offering token-level feedback at every reasoning step, this ideal token-level Monte Carlo approach faces severe practical challenges when combined with group-relative advantage computation. Computing group-relative advantages at intermediate positions requires forming stable comparison groups through extensive continuation sampling at each token location. For early reasoning tokens, accurately estimating advantage signs demands prohibitively large K values, and insufficient sampling causes token-level rewards to paradoxically underperform outcome-level baselines that trade fine-grained discrimination for stability.

Upstream-Downstream Instantiation: A Cost-Effective Step Beyond. To make intermediate credit computationally feasible, we instantiate Monte Carlo sampling at a single segment boundary through an upstream-downstream architecture.

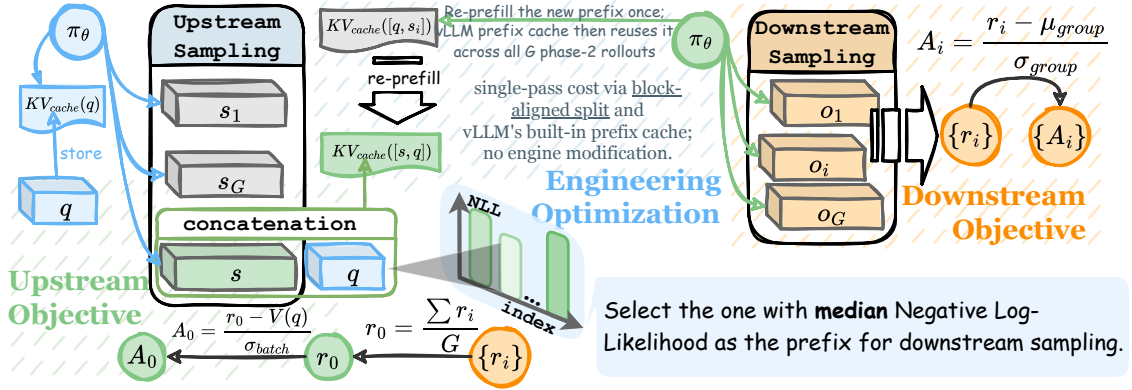


Figure 2: Overview of SCRoll. Three components: upstream sampling and advantage computation; downstream sampling and advantage computation; and an engineering-optimization path that reduces two-phase rollout overhead via block-aligned splits and vLLM’s built-in prefix cache without modifying the vLLM engine.

In the upstream phase, the policy model π_θ generates G parallel early-stopped reasoning segments $\{s_j\}_{j=1}^G$ from q , all terminated at a shared segmentation position t_q drawn uniformly at random within a per-prompt response-length window rather than fixed by formatting rules. Specifically, for each prompt q , we sample $t_q \sim \mathcal{U}[0, L_q]$, where L_q is the per-prompt average response length tracked adaptively via the same KL-adaptive mechanism as the upstream value tracker (see Section 3.2). For sequences that terminate before t_q (or have actual length shorter than t_q), we use half of the actual response length as the split position. We then select one segment $s = s_{k^*}$ via *median-NLL ranking*: the candidate whose per-token negative log-likelihood under π_θ sits at the median of $\{\text{NLL}(s_j)\}_{j=1}^G$, which avoids both the highest-likelihood (collapsed) and lowest-likelihood (off-policy) extremes. The remaining $G - 1$ candidates are discarded. We then sample G continuations from the augmented input $[s, q]$, where the selected segment is placed before the original problem, to obtain complete responses $\{o_i\}_{i=1}^G$ in the downstream phase. This skip connection $[s, q]$ serves dual functions: ❶ *it preserves the upstream segment’s influence, so the downstream outcomes define a skip-conditioned value for s* ; ❷ *it restores direct access to the original problem, avoiding the homogeneous pure-continuation regime in which downstream samples are forced along the same partial trajectory and group-relative advantages col-*

lapse toward zero. Formally:

$$\begin{aligned} \text{I : } & \{s_j\}_{j=1}^G \sim \pi_\theta(\cdot|q), \\ & s = \text{Select}_{\text{medNLL}}(\{s_j\}_{j=1}^G), \\ \text{II : } & \{o_i\}_{i=1}^G \sim \pi_\theta(\cdot|[s, q]). \end{aligned} \quad (2)$$

where $q \in \mathcal{Q}$ denotes the problem, $s \in \mathcal{S}$ the selected reasoning segment, and $o \in \mathcal{O}$ the final output. We define reward signals accordingly: downstream reward $\mathcal{R}_{\text{II}}(o, a) = 2 \cdot \mathbb{I}[o = a] - 1$ measures final correctness, while upstream reward $\mathcal{R}_{\text{I}}(s, a)$ implements the skip-conditioned Monte Carlo aggregation:

$$\mathcal{R}_{\text{I}}(s, a) = \frac{1}{G} \sum_{i=1}^G \mathcal{R}_{\text{II}}(o_i, a), \quad (3)$$

where $\{o_i\}_{i=1}^G$ are the downstream samples from Equation (2). This design achieves a favorable tradeoff: whereas GRPO uses G complete responses per problem with no intermediate feedback, and full token-level Monte Carlo would require K continuations for each token across all G responses, our approach reuses the same G downstream samples to provide both the downstream group and one segment-level reward for the upstream stream.

Engineering Optimization. A naive two-phase implementation would dispatch the upstream rollouts (the G parallel segments from q , each terminated at t_q) and the downstream rollouts (continuations from $[s, q]$) as two sequential GPU batches; the dispatch overhead and repeated prefill can make per-step time substantially larger than a single-batch group-relative optimizer (Figure 9, Section B). Our implementation therefore uses a

custom rollout wrapper on top of verl while leaving the vLLM engine unchanged. Concretely: ❶ in the first phase the G parallel rollouts from q share an identical prompt, so vLLM can reuse the prefix representation of q across upstream sequences; ❷ we constrain t_q so that the re-prefilled prefix $[s, q]$ ends on a KV cache block boundary (block size $B=16$ by default), reducing partial-block re-computation when the prefix is reused in the second phase; ❸ once the median-NLL segment s is selected, the wrapper dispatches the G downstream continuations as standard generate requests all prompted with the identical $[s, q]$. Since the $[s, q]$ prefix is shared across these requests, vLLM’s prefix cache can materialize $KV([s, q])$ once and reuse it across the continuations, leaving only per-rollout suffix tokens to be computed when cache entries are retained. We therefore treat efficiency as an empirical property of the rollout wrapper rather than a strict token-count identity; exact wall-clock behavior depends on prefix-cache hits, scheduling, cache capacity, and completion lengths, so we report time-per-step measurements in Figure 9.

3.2 Asymmetric Optimization Strategy

Given a dataset $\mathcal{D}$ with ground-truth answers a , the skip-connected optimization objective maximizes the expected advantages at both stages:

$$\max_{\theta} \mathbb{E}_{(q,a) \sim \mathcal{D}} [\mathcal{J}_I(\theta; q, a) + \mathcal{J}_{II}(\theta; q, s, a)], \quad (4)$$

where $\mathcal{J}_I$ and $\mathcal{J}_{II}$ are the upstream and downstream policy gradient objectives defined using advantages computed from rewards $\mathcal{R}_I$ and $\mathcal{R}_{II}$ respectively.

Given that the upstream response density is multiplicatively lower than that of the downstream, we employ an asymmetric two-stream optimization strategy: a sample-efficient single-stream optimizer for the upstream signal, and the user’s choice of group-relative policy optimizer for the downstream stream, as detailed below.

Downstream Optimization: Plug-in Group-Relative Optimizer. The downstream stream is optimized by any group-relative algorithm that maintains a G -rollout group structure, leaving SCRoll agnostic to the specific loss form. For each prompt q_m , let s_m^* denote the median-NLL-selected segment, distinct from the $G-1$ discarded segments during parallel generation, and define the augmented input $\tilde{q}_m = [s_m^*; q_m]$. We sample a downstream group $\{o_{m,j}\}_{j=1}^G$, compute group-relative advantages $\hat{A}_{m,j} = (R_{m,j} - \bar{R}_m)/\sigma_{R_m}$

from rewards $\mathcal{R}_{II}(o_{m,j}, a_m)$, and update the policy through a generic group-relative objective:

$$\mathcal{J}_{II}(\theta) = \mathbb{E}_{(q_m, a_m) \sim \mathcal{D}, \{o_{m,j}\}_{j=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | \tilde{q}_m)} [\mathcal{L}_{\text{GR}}(\pi_{\theta}; \{o_{m,j}\}_{j=1}^G, \{\hat{A}_{m,j}\}_{j=1}^G)], \quad (5)$$

where $\mathcal{L}_{\text{GR}}$ denotes any group-relative loss functional. The specific instantiations evaluated in our experiments and their per-algorithm hyperparameters are documented in Section C.

Upstream Optimization: Single-stream Policy Optimization. Since only one selected segment per problem participates in upstream optimization (with the remaining $G-1$ segments discarded after median-NLL selection), group-relative estimation across the upstream candidates is not the right object to optimize. We adopt SPO (Xu and Ding, 2025), which replaces the concurrent-group baseline with a per-prompt temporal return tracker $\hat{V}(q)$. The return tracker and the response-length tracker L_q have independent states and discounted counts C_q^V and C_q^L ; they share only the clipped forgetting factor $\rho = \text{clip}(2^{-D_{\text{KL}}/\tau_{1/2}}, \rho_{\min}, \rho_{\max})$. For $x \in \{V, L\}$, their updates are

$$C_q^x \leftarrow \rho C_q^x + 1, \quad x_q \leftarrow x_q + \frac{z_x - x_q}{C_q^x}, \quad (6)$$

where $z_V = \mathcal{R}_I(s, a)$ and z_L is the observed response length. Thus, reward observations do not directly change L_q or the split distribution. We initialize $\hat{V}(q) = 0$ and $L_q = 1024$, update both trackers on the first observation, sample $t_q \sim \mathcal{U}[0, L_q]$, and use half the realized length after early termination. KV-cache block alignment is then applied separately to the realized $[s, q]$ prefix.¹ The advantage $A_i = \mathcal{R}_I(s_i^*, a_i) - \hat{V}_{-1}(q_i)$ uses the pre-update return baseline, and we normalize it across the batch as $\tilde{A}_i = (A_i - \mu_B)/\sigma_B$. Let $s_i^* = \text{Select}_{\text{medNLL}}(\{s_{i,j}\}_{j=1}^G)$. Because s_i^* is an order statistic, the upstream loss is a selected-segment surrogate rather than an unbiased policy-gradient estimator under the raw segment distribution. The objective is:

$$\mathcal{J}_I(\theta) = \mathbb{E}_{(q_i, a_i) \sim \mathcal{D}, \{s_{i,j}\}_{j=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | q_i)} \left[\frac{1}{|s_i^*|} \sum_{t=1}^{|s_i^*|} \min(r_{i,t}(\theta), \tilde{A}_i, \hat{r}_{i,t}, \tilde{A}_i) \right], \quad (7)$$

¹While the original SPO formulation assumes rewards in $[0, 1]$, our rewards in $[-1, +1]$ are affinely mapped to $[0, 1]$ for tracker updates and mapped back when retrieving baselines.

where $r_{i,t}(\theta) = \pi_{\theta}(s_{i,t}^* \mid q_i, s_{i,<t}^*) / \pi_{\theta_{\text{old}}}(s_{i,t}^* \mid q_i, s_{i,<t}^*)$ and $\hat{r}_{i,t} = \text{clip}(r_{i,t}, 1 - \varepsilon, 1 + \varepsilon)$.

4 Experiments

Training Setup. We train SCRoll on the dapomath-17k dataset (Yu et al., 2025b), a curated mathematical reasoning dataset containing 17k problems spanning various difficulty levels. All models are trained for 500 optimization steps with a batch size of 128 prompts per step. We set the maximum prompt length to 1024 tokens and the maximum completion length to 3072 tokens for both training and evaluation. For fair comparison, all methods using group-relative advantage estimation adopt $G = 8$ rollouts per prompt; SCRoll generates $G = 8$ upstream segments in parallel, selects one via median-NLL ranking, and dispatches G downstream responses from identical skip-connected prompts $[s, q]$ with prefix-cache reuse. The skip-connected rollout is used only during RL training data construction: all reported benchmark evaluations use the standard evaluation prompt q without prepending selected upstream segments or running a two-stage test-time procedure. For SPO which does not use group-relative estimation, we scale the batch size by 8 to ensure comparable sample count; we also omit the warm-up phase in our SPO reproduction to avoid introducing a cost-misaligned initialization stage. We use Qwen2.5-Math-7B and Llama-3.2-3B-Instruct as base models. Complete hyperparameters and implementation details are provided in Section C.

4.1 Main Results

We evaluate SCRoll as a plug-in across four representative group-relative policy optimizers on In-Domain mathematical reasoning and Out-of-Domain general/code reasoning. We use MMLU-Pro (Wang et al., 2024) and LiveCodeBench (Jain et al., 2025a) for the latter. Evaluations are repeated 32 times for AIME24/25 and 3 times for all other benchmarks. Each row in Table 1 is the mean evaluation score of a fixed checkpoint from one training run; repeated decoding measures evaluation-time variability rather than training-seed uncertainty. For each optimizer in {GRPO (Shao et al., 2024), GSPO (Zheng et al., 2025), DAPO (Yu et al., 2025b), CISPO (MiniMax, 2025)}, we report performance with and without SCRoll in the downstream slot. We also include SPO (Xu and Ding, 2025), PRIME (Cui et al., 2025), and SAPO (Gao

et al., 2025) as non-paired references because they do not expose the same reusable G -rollout interface. Complete settings are in Section C.

As shown in Table 1, the clearest gains occur with CISPO, with smaller positive average gains for GRPO and DAPO and marginal gains for GSPO. On MMLU-Pro and LiveCodeBench, CISPO + SCRoll improves both base models, while the other pairings transfer less consistently. Compute controls in Table 3 show that the decoupled two-process baseline reaches a similar score only when given substantially more generated tokens; under the matched 0.58B-token budget, SCRoll performs better. In a Qwen sensitivity run, the paired average gain changes only from +1.9 at $G = 8$ to +1.8 at $G = 16$, although broader group-size scaling remains untested.

4.2 Ablation Studies

We conduct ablation studies to validate the effectiveness of key components in SCRoll. We compare SCRoll against three variants: **❶ w/o Skip:** The upstream segment s is directly used as a prefix for downstream generation without the skip connection to the original problem. **❷ w/o MC Estimation:** The upstream stage receives rewards from its own complete generation without aggregating downstream outcomes. **❸ w/o NLL Selection:** The upstream segment s is picked uniformly at random from the G parallel candidates instead of by median-NLL ranking. The results are summarized in Table 2.

All three components contribute comparable margins (Table 2). **w/o Skip** forces downstream responses along a constrained continuation path, raising the probability of homogeneous outcomes and advantage collapse. **w/o MC Estimation** disconnects the upstream reward from downstream solving, so the upstream stream cannot learn patterns that specifically facilitate downstream completions. **w/o NLL Selection** lets collapsed and off-policy candidates enter upstream optimization with equal probability, whereas median-NLL ranking favors on-policy yet non-degenerate segments.

5 Conclusion

We asked whether intermediate credit can be added on top of the existing G -rollout interface without replacing the downstream optimizer. Probing token-level Monte Carlo revealed a sharp asymmetry under practical sampling budgets: at early posi-

Table 1: Accuracy (%) on In-Domain math and Out-of-Domain general/code benchmarks for **Qwen2.5-Math-7B** and **Llama-3.2-3B-Instruct**. We report absolute scores in every row. The best and second-best values within each model are **bolded** and underlined. Each average is computed over the benchmark columns shown for that model; Minerva was not part of the retained Llama evaluation suite. Evaluation repetitions are 32 for AIME24/25 and 3 for all other benchmarks.

	Qwen2.5-Math-7B									Llama-3.2-3B-Instruct								
Method	In-Domain (Math)						Out-of-Domain			In-Domain (Math)						Out-of-Domain		
	AIME24	AIME25	AMC	MATH	Minerva	Olympiad	MMLU	LiveCode	Avg.	AIME24	AIME25	AMC	MATH	Olympiad	MMLU	LiveCode	Avg.	
Base	12.1	5.4	43.4	58.1	20.6	34.1	18.6	13.8	25.8	3.4	0.4	20.1	36.3	18.1	29.5	12.4	17.2	
SPO	28.8	13.6	56.8	65.9	29.2	40.1	24.1	15.7	34.3	5.8	0.4	26.1	39.2	20.3	29.8	12.9	19.2	
PRIME	30.1	15.7	63.9	71.3	30.8	43.9	25.0	18.9	37.5	5.4	<u>0.8</u>	29.7	34.7	20.8	30.1	13.2	19.2	
SAPO	32.0	14.2	64.7	76.1	<u>33.8</u>	46.5	28.7	16.5	39.1	7.9	3.3	38.1	42.0	21.2	30.4	12.6	22.2	
GRPO	22.4	10.1	50.2	63.8	23.5	35.8	21.3	14.6	30.2	4.7	0.3	24.8	39.6	19.8	30.2	13.1	18.9	
+ SCRoll	26.2	8.9	55.4	65.2	25.1	37.9	21.9	15.5	32.0	7.3	0.4	29.1	39.2	21.1	30.4	12.9	20.1	
GSPO	31.1	14.8	60.3	74.2	28.1	43.6	25.2	16.2	36.7	6.9	0.5	30.4	42.1	22.3	<u>30.5</u>	<u>13.8</u>	21.0	
+ SCRoll	29.3	16.9	62.1	74.6	28.8	42.7	25.5	16.8	37.1	6.2	0.7	31.6	42.4	23.3	30.4	13.7	21.2	
DAPO	33.5	15.9	69.2	75.8	30.4	47.2	27.1	17.0	39.5	13.8	0.4	36.5	41.1	24.3	30.2	12.7	22.7	
+ SCRoll	<u>35.6</u>	<u>17.8</u>	<u>70.2</u>	76.6	32.0	49.0	27.5	<u>17.6</u>	<u>40.8</u>	<u>14.8</u>	0.4	37.1	<u>42.5</u>	<u>25.0</u>	30.0	12.4	<u>23.2</u>	
CISPO	35.0	17.4	67.5	<u>78.5</u>	31.8	48.6	27.2	16.9	40.4	11.0	0.6	25.0	35.3	23.9	29.7	11.9	19.6	
+ SCRoll	36.2	19.1	70.5	80.0	34.0	50.4	<u>28.0</u>	18.9	42.1	15.2	0.4	<u>37.5</u>	44.0	25.6	30.6	14.0	23.9	

Table 2: Ablation study on In-Domain benchmarks comparing different structural settings. We report accuracy (%) for Qwen2.5-Math-7B. The best results are **bolded**.

Setting	AIME24	AIME25	AMC	MATH	Minerva	Olympiad
w/o Skip	33.2	15.6	67.4	74.0	29.6	46.1
w/o MC Estimation	34.1	17.5	68.3	77.8	32.5	48.6
w/o NLL Selection	33.5	16.4	66.9	75.4	30.4	47.1
Ours	36.2	19.1	70.5	80.0	34.0	50.4

tions, high MC variance survives group-relative normalization as sign errors that misdirect gradients below the outcome-only baseline, while at late positions MC estimates already coincide with the outcome reward. We exploit this asymmetry by routing different parts of a rollout to different optimizers, yielding SCRoll: a rollout-side module that pairs a sample-efficient single-stream upstream optimizer with the user’s group-relative downstream optimizer, joined by a skip connection $[s, q]$. Across GRPO, GSPO, DAPO, and CISPO, SCRoll improves most configurations on in-domain mathematics, with CISPO + SCRoll and DAPO + SCRoll achieving the strongest in-domain results; on out-of-domain reasoning and code, CISPO + SCRoll retains gains across both base models while DAPO + SCRoll retains them on Qwen2.5-Math-7B. Future work can extend the schema to multi-point upstream evaluation and to multi-stage RL for agentic tasks.

Limitations

Model scale and architecture. Our experiments are conducted on Qwen2.5-Math-7B and Llama-3.2-3B-Instruct, two base models at the 3B–7B

scale that are tractable under our compute budget. We do not yet have direct evidence that the benefits of SCRoll transfer to substantially larger models (e.g., 30B+ or 70B+), to mixture-of-experts architectures, or to models without strong math pretraining. The relative gain of the skip-connected structure may shrink when the base policy is already capable enough to recover from a poor upstream segment without an explicit shortcut.

Training data and domain. SCRoll is trained exclusively on dapomath-17k, a mathematical reasoning corpus with verifiable numerical answers. Our out-of-domain evaluation on MMLU-Pro and LiveCodeBench provides preliminary evidence that the learned hierarchical reasoning pattern can transfer beyond math, but the training signal itself still relies on a verifiable outcome reward. Although dapomath-17k is deduplicated before training, competition-style math corpora can contain paraphrased or structurally similar problems across sources; we therefore do not claim that our benchmark suite is free of all semantic overlap with the training distribution. Extending SCRoll to open-ended tasks such as long-form generation, dialogue, or instruction following, where reward verification is noisier or requires a learned reward model, is left to future work.

Training horizon and seeds. We train each method for 500 optimization steps, which does not characterize behavior under much longer runs where collapse or saturation may appear. Due to compute constraints, the main results in Table 1

use a single training run per (method, base model) configuration; repeated evaluation ($32\times$ for AIME, $3\times$ for others) measures decoding variability but not training-seed uncertainty. Main experiments use $G = 8$, and broader group-size scaling remains untested.

Engineering complexity. The efficient realization of upstream and downstream sampling requires a custom rollout wrapper on top of verl and relies on vLLM’s built-in prefix caching together with a block-aligned split point. This does not require vLLM engine-side modifications, but the wall-clock benefit depends on prefix-cache hits, scheduling, and cache capacity; the block-alignment constraint is currently tied to vLLM’s KV cache layout, and porting to other rollout engines (e.g., SGLang) requires verifying that the cache-reuse semantics are equivalent. Median-NLL selection over the G parallel upstream segments is one specific choice; beyond the uniform-random ablation reported in Section 4.2, we have not exhaustively compared against further selection rules.

Potential risks. As with other work that improves the reasoning capability of large language models, more capable reasoners can be misused, for example to automate the production of harmful or deceptive content. SCROLL does not introduce new attack vectors beyond those of the underlying base models, but we encourage downstream practitioners to pair any capability gains with appropriate safety evaluation and deployment safeguards.

References

- Arash Ahmadian, Chris Cremer, Matthias Gall , Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet  st n, and Sara Hooker. 2024. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12248–12267.
- Farid Bagirov, Mikhail Arkhipov, Ksenia Sycheva, Evgeniy Glukhov, and Egor Bogomolov. 2025. [The best of N worlds: Aligning reinforcement learning with best-of-N sampling via max@k optimisation](#). *arXiv preprint arXiv:2510.23393*.
- Kiant  Brantley, Mingyu Chen, Zhaolin Gao, Jason D. Lee, Wen Sun, Wenhao Zhan, and Xuezhou Zhang. 2025. [Accelerating RL for LLM reasoning with optimal advantage regression](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Hongzhan Chen, Tao Yang, Shiping Gao, Ruijun Chen, Xiaojun Quan, Hongtao Tian, and Ting Yao. 2025a. [Discriminative policy optimization for token-level reward models](#). In *Forty-second International Conference on Machine Learning*.
- Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Pich , Nicolas Gontier, Yoshua Bengio, and Ehsan Kamaloo. 2025b. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*.
- Xiwen Chen, Wenhui Zhu, Peijie Qiu, Xuanzhao Dong, Hao Wang, Haiyu Wu, Huayu Li, Aristeidis Sotiras, Yalin Wang, and Abolfazl Razi. 2025c. DRA-GRPO: Your GRPO needs to know diverse reasoning paths for mathematical reasoning. *arXiv preprint arXiv:2505.09655*.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, and 6 others. 2025. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*.
- Wenlong Deng, Yi Ren, Danica J. Sutherland, Christos Thrampoulidis, and Xiaoxiao Li. 2025. [Token hidden reward: Steering exploration-exploitation in GRPO training](#). In *2nd AI for Math Workshop @ ICML 2025*.
- Fei Ding, Baiqiao Wang, Zijian Zeng, and Youwei Wang. 2025. Multi-layer grpo: Enhancing reasoning and self-correction in large language models. *arXiv preprint arXiv:2506.04746*.
- Ruiyi Ding, Yongxuan Lv, Xianhui Meng, Jiahe Song, Chao Wang, Chen Jiang, and Yuan Cheng. 2026. [PRPO: Aligning process reward with outcome reward in policy optimization](#). *arXiv preprint arXiv:2601.07182*.
- Chanakya Ekbote, Vijay Lingam, Sujay Sanghavi, Jun Huan, Behrooz Omidvar-Tehrani, Anoop Deoras, and Stefano Soatto. 2025. MURPHY: Feedback-aware GRPO with retrospective credit assignment for multi-turn code generation. *arXiv preprint arXiv:2511.07833*.
- Lishui Fan, Yu Zhang, Mouxian Chen, and Zhongxin Liu. 2025. [Posterior-GRPO: Rewarding reasoning processes in code generation](#). *arXiv preprint arXiv:2508.05170v1*.
- Wu Fei, Shuxian Liang, Yibo Yang, Yang Lin, Jing Tang, Lei Chen, Xiansheng Hua, and Hao Kong. 2025. Self-guided process reward optimization with redefined step-wise advantage for process reinforcement learning. *arXiv preprint arXiv:2507.01551*.
- Kanishk Gandhi, Ayush K Chakravarthy, Anikait Singh, Nathan Lile, and Noah Goodman. 2025. [Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars](#). In *Second Conference on Language Modeling*.

- Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. 2025. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*.
- Xue Gong, Qi Yi, Ziyuan Nan, Guanhua Huang, Ke-jiao Li, Yuhao Jiang, Ruibin Xiong, Zenan Xu, Jiaming Guo, Shaohui Peng, and Bo Zhou. 2026. [Segmental advantage estimation: Enhancing PPO for long-context LLM training](#). *arXiv preprint arXiv:2601.07320*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shiron Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, and 175 others. 2025. [Deepseek-r1 incentivizes reasoning in llms through reinforcement learning](#). *Nature*, 645(8081):633–638.
- Andre Wang He, Daniel Fried, and Sean Welleck. 2025a. [Rewarding the unlikely: Lifting GRPO beyond distribution sharpening](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25548–25560, Suzhou, China. Association for Computational Linguistics.
- Bingxiang He, Zekai Qu, Zeyuan Liu, Yinghao Chen, Yuxin Zuo, Cheng Qian, Kaiyan Zhang, Weize Chen, Chaojun Xiao, Ganqu Cui, Ning Ding, and Zhiyuan Liu. 2025b. [Justrl: Scaling a 1.5 b llm with a simple rl recipe](#). *arXiv preprint arXiv:2512.16649*.
- Shenghua He, Tian Xia, Xuan Zhou, and Hui Wei. 2025c. Response-level rewards are all you need for online reinforcement learning in llms: A mathematical perspective. *arXiv preprint arXiv:2506.02553*.
- Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*.
- Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2025. R-zero: Self-evolving reasoning llm from zero data. *arXiv preprint arXiv:2508.05004*.
- Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025a. [Live-codebench: Holistic and contamination free evaluation of large language models for code](#). In *The Thirteenth International Conference on Learning Representations*.
- Samyak Jain, Ayush Agrawal, and Navin Goyal. 2025b. [Towards understanding the optimization landscape of GRPO and its variants](#). In *First Workshop on Foundations of Reasoning in Language Models*.
- Yuhua Jiang, Yuwen Xiong, Yufeng Yuan, Chao Xin, Wenyan Xu, Yu Yue, Qianchuan Zhao, and Lin Yan. 2025. Pag: Multi-turn reinforced llm self-correction with policy as generative verifier. *arXiv preprint arXiv:2506.10406*.
- Can Jin, Yang Zhou, Qixin Zhang, Hongwu Peng, Di Zhang, Zihan Dong, Marco Pavone, Ligong Han, Zhang-Wei Hong, Tong Che, and Dimitris N. Metaxas. 2025. [Your reward function for RL is your best PRM for search: Unifying RL and search-based TTS](#). *arXiv preprint arXiv:2508.14313v3*.
- Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2024. Vineppo: Refining credit assignment in rl training of llms. *arXiv preprint arXiv:2410.01679*.
- Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. 2025. The art of scaling reinforcement learning compute for llms. *arXiv preprint arXiv:2510.13786*.
- Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. 2025. [Training language models to self-correct via reinforcement learning](#). In *The Thirteenth International Conference on Learning Representations*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Alan Lee and Harry Tong. 2025. Token-efficient rl for llm reasoning. *arXiv preprint arXiv:2504.20834*.
- Gang Li, Yan Chen, Ming Lin, and Tianbao Yang. 2025a. [Drpo: Efficient reasoning via decoupled reward policy optimization](#). *arXiv preprint arXiv:2510.04474*.
- Jiahui Li, Lin Li, Tai-Wei Chang, Kun Kuang, Long Chen, Jun Zhou, and Cheng Yang. 2025b. [RED: Unleashing token-level rewards from holistic feedback via reward redistribution](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 4993–5022, Suzhou, China. Association for Computational Linguistics.
- Lianrui Li, Dakuan Lu, Jiawei Shao, and Xuelong Li. 2025c. [Scrpo: From errors to insights](#). *arXiv preprint arXiv:2511.06065*.
- Yu Li, Rui Miao, Tian Lan, and Zhengling Qi. 2026. [OPPO: Bayesian value recursion for token-level credit assignment in LLM reasoning](#). *arXiv preprint arXiv:2605.21851*.
- Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. 2024. [Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models](#). In *Forty-first International Conference on Machine Learning*.

- Xingyu Lin, Yilin Wen, En Wang, Du Su, Wenbin Liu, Chenfu Bao, and Zhonghou Lv. 2025. Token-level policy optimization: Linking group-level rewards to token-level aggregation via markov likelihood. *arXiv preprint arXiv:2510.09369*.
- Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. 2025a. Dler: Doing length penalty right-incentivizing more intelligence per token via reinforcement learning. *arXiv preprint arXiv:2510.15110*.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. [Understanding rl-zero-like training: A critical perspective](#). In *Second Conference on Language Modeling*.
- Yiran Ma, Zui Chen, Tianqiao Liu, Mi Tian, Zhuo Liu, Zitao Liu, and Weiqi Luo. 2025. [What are step-level reward models rewarding? counterintuitive findings from MCTS-boosted mathematical reasoning](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):24812–24820.
- MiniMax. 2025. MiniMax-M1: Scaling test-time compute efficiently with Lightning Attention. *arXiv preprint arXiv:2506.13585*.
- Jatin Prakash and Anirudh Buvanesh. 2025. What can you do when you have zero rewards during rl? *arXiv preprint arXiv:2510.03971*.
- Massimiliano Pronesti, Anya Belz, and Yufang Hou. 2026. [Beyond outcome verification: Verifiable process reward models for structured reasoning](#). *arXiv preprint arXiv:2601.17223*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Yuepeng Sheng, Yuwei Huang, Shuman Liu, Anxiang Zeng, and Haibo Zhang. 2025. Espo: Entropy importance sampling policy optimization. *arXiv preprint arXiv:2512.00499*.
- Michael Sullivan and Alexander Koller. 2025. Grpo is secretly a process reward model. *arXiv preprint arXiv:2509.21154*.
- Hongze Tan, Zihan Wang, Jianfei Pan, Jinghao Lin, Hao Wang, Yifan Wu, Tao Chen, Zhihang Zheng, Zhihao Tang, and Haihua Yang. 2025. Gtpo and grpo-s: Token and sequence-level reward shaping with policy entropy. *arXiv preprint arXiv:2508.04349*.
- Haozhe Wang, Qixin Xu, Che Liu, Junhong Wu, Fangzhen Lin, and Wenhua Chen. 2025a. Emergent hierarchical reasoning in llms through reinforcement learning. *arXiv preprint arXiv:2509.03646*.
- 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. 2025b. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.
- Tianyi Wang, Yixia Li, Long Li, Yibiao Chen, Shao-han Huang, Yun Chen, Peng Li, Yang Liu, and Guanhua Chen. 2026. [SPPO: Sequence-level PPO for long-horizon reasoning tasks](#). *arXiv preprint arXiv:2604.08865*.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhua Chen. 2024. [MMLU-pro: A more robust and challenging multi-task language understanding benchmark](#). In *Advances in Neural Information Processing Systems* 37.
- Muning Wen, Junwei Liao, Cheng Deng, Jun Wang, Weinan Zhang, and Ying Wen. 2024. Entropy-regularized token-level policy optimization for language agent reinforcement. *arXiv preprint arXiv:2402.06700*.
- Yihong Wu, Liheng Ma, Lei Ding, Muzhi Li, Xinyu Wang, Kejia Chen, Zhan Su, Zhanguang Zhang, Chenyang Huang, Yingxue Zhang, Mark Coates, and Jian-Yun Nie. 2025. It takes two: Your grpo is secretly dpo. *arXiv preprint arXiv:2510.00977*.
- Guofu Xie, Yunsheng Shi, Hongtao Tian, Ting Yao, and Xiao Zhang. 2025. Capo: Towards enhancing llm reasoning through generative credit assignment. *arXiv preprint arXiv:2508.02298*.
- Jian Xiong, Jingbo Zhou, Jingyong Ye, Qiang Huang, and Dejing Dou. 2025. AAPPO: Enhancing the reasoning capabilities of LLMs with advantage margin. *arXiv preprint arXiv:2505.14264*.
- Zhongwen Xu and Zihan Ding. 2025. Single-stream policy optimization. *arXiv preprint arXiv:2509.13232*.
- Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. [Learning to reason under off-policy guidance](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025a. Reasonflux: Hierarchical llm reasoning via scaling thought templates. *arXiv preprint arXiv:2502.06772*.
- Shihui Yang, Chengfeng Dou, Peidong Guo, Kai Lu, Qiang Ju, Fei Deng, and Rihui Xin. 2025b. Dcipo: Dynamic clipping policy optimization. *arXiv preprint arXiv:2509.02333*.

Ziyi Yang, Weizhou Shen, Chenliang Li, Ruijun Chen, Fanqi Wan, Ming Yan, Xiaojun Quan, and Fei Huang. 2025c. Spell: Self-play reinforcement learning for evolving long-context language models. *arXiv preprint arXiv:2509.23863*.

Chaorui Yao, Yanxi Chen, Yuchang Sun, Yushuo Chen, Wenhao Zhang, Xuchen Pan, Yaliang Li, and Bolin Ding. 2025. Group-relative reinforce is secretly an off-policy algorithm: Demystifying some myths about grpo and its friends. *arXiv preprint arXiv:2509.24203*.

Jiarui Yao, Ruida Wang, and Tong Zhang. 2026. PRL: Process reward learning improves LLMs’ reasoning ability and broadens the reasoning boundary. *arXiv preprint arXiv:2601.10201v1*.

Jiahao Yu, Zelei Cheng, Xian Wu, and Xinyu Xing. 2025a. GPO: Learning from critical steps to improve LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gao-hong Liu, Juncan Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, and 17 others. 2025b. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chuhuai Yue, Chengqi Dong, Yinan Gao, Hang He, Jiajun Chai, Guojun Yin, and Wei Lin. 2025. Promoting efficient reasoning with verifiable stepwise reward. *arXiv preprint arXiv:2508.10293*.

Han Zhang, Ruibin Zheng, Zexuan Yi, Zhuo Zhang, Hanyang Peng, Hui Wang, Zike Yuan, Cai Ke, Shiwei Chen, Jiacheng Yang, Yangning Li, Xiang Li, Jiangyue Yan, Yaoqi Liu, Liwen Jing, Jiayin Qi, Ruifeng Xu, Binxiang Fang, and Yue Yu. 2025a. Gepo: Group expectation policy optimization for stable heterogeneous reinforcement learning. *arXiv preprint arXiv:2508.17850*.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Yuanheng Zhu, Qichao Zhang, Zijie Zhao, and Dongbin Zhao. 2025b. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. *arXiv preprint arXiv:2511.12159*.

Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, Fang Wan, and Furu Wei. 2025. Geometric-mean policy optimization. *arXiv preprint arXiv:2507.20673*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Jiaru Zou, Ling Yang, Jingwen Gu, Jiahao Qiu, Ke Shen, Jingrui He, and Mengdi Wang. 2025. ReasonFlux-PRM: Trajectory-aware PRMs for long chain-of-thought reasoning in LLMs. *arXiv preprint arXiv:2506.18896*.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Yuchen Zhang, Xinwei Long, Ermo Hua, Biqing Qi, Youbang Sun, Zhiyuan Ma, Lifan Yuan, Ning Ding, and Bowen Zhou. 2025. Ttrl: Test-time reinforcement learning. *arXiv preprint arXiv:2504.16084*.

A Why Fine-Grained Rewards Fail with Group-Relative Estimation

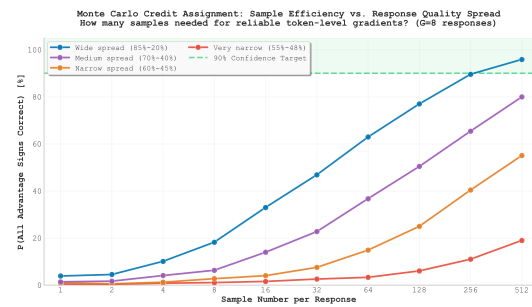
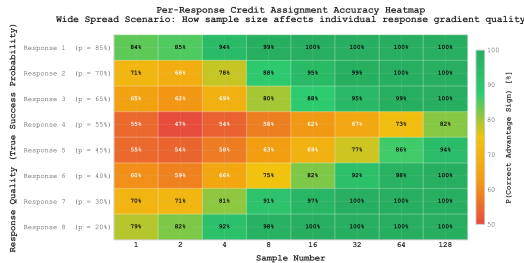


Figure 3: Probability of correct advantage signs for all $G = 8$ samples across varying group spreads. Reliable credit assignment for narrow spreads requires prohibitive sample counts ($K > 512$).

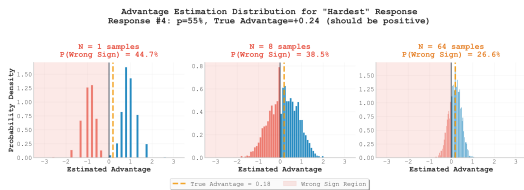
As claimed in Section 1, directly integrating fine-grained Monte Carlo credit signals with group-relative advantage estimation runs into a high-variance regime under practical sampling budgets: although the per-prefix Monte Carlo reward estimator is itself unbiased, its high variance survives the group-relative normalization $(R_i - \bar{R})/\sigma_R$ as frequent **sign errors** in the resulting advantages, producing systematically wrong-direction gradients that reinforce wrong behaviors or penalize correct ones. This appendix provides simulation evidence quantifying this phenomenon. The severity of this problem explains why existing methods *circumvent* rather than solve it. For example, PRIME (Cui et al., 2025) aggregates token rewards to response-level before computing group-relative advantages, while VSRM (Yue et al., 2025) abandons group-relative estimation entirely in favor of PPO. SCROLL takes a different approach: *structurally partitioning* the problem so that upstream optimization avoids group-relative computation while downstream optimization has sufficient samples for stable estimation.

A.1 Position-Dependent Variance

The effectiveness of group-relative advantage estimation varies dramatically with trajectory position. For **very early tokens** (problem restatement and initial setup), the reward is largely determined by the model’s own policy distribution rather than the specific tokens generated, as the reasoning space has not yet been meaningfully pruned. Trajectory sampling from these positions yields high variance, but this variance is *symmetric* across methods. **Intermediate tokens** (the decisive reasoning steps) are where methods diverge. Quality differences emerge as some reasoning paths make productive progress while others take detours or make errors. However, Monte Carlo estimation at these positions suffers from prohibitive variance: distinguishing a 55% success probability prefix from a 60% one requires hundreds of samples. This is precisely where SCROLL’s upstream-downstream structure provides stable optimization signals for intermediate segments. For **terminal tokens** (final computation and answer extraction), the outcome is semantically determined; sampling variance is minimal and single samples provide reliable gradients, so all methods perform comparably.



(a) Credit assignment accuracy heatmap. Prefixes near the group mean are most susceptible to sign errors.



(b) Distribution of advantage estimates for a prefix with $p = 0.55$. Even with $K = 8$, substantial overlap between positive and negative estimates remains.

Figure 4: Monte Carlo variance analysis. (a) Classification accuracy for individual samples. (b) Distribution shift and sign volatility under sampling noise.

A.2 Simulation Setup and Results

We simulate a group of $G = 8$ trajectories where each prefix has a true success probability p_i . Advan-

tages are estimated by sampling K continuations per prefix and applying group-relative normalization. An estimate is considered correct if the signs of all G advantages match their ground-truth signs. Simulation results in Figures 3 and 4 demonstrate that sample efficiency degrades rapidly as group quality spread narrows. When reasoning steps provide only incremental improvements, even $K = 512$ samples fail to yield 50% probability of correct sign assignment for the entire group. Individual prefix analysis in Figure 4a confirms that prefixes near the group mean are most vulnerable to noise-induced sign flips, which are precisely the prefixes that represent incremental but genuine reasoning progress.

A.3 Correctness Distribution Analysis

To further illustrate the sign inconsistency problem, we systematically examine how advantage estimation accuracy varies across different correctness distributions within a group of $G = 8$ responses. We sample random trajectories and categorize them by the number of correct responses (from 0/8 to 8/8), visualizing the true token rewards, advantages, and estimation accuracy for each distribution type. All nine cases are shown in Figure 5.

The degenerate 0/8 and 8/8 groups should not be interpreted as successful credit assignment: their outcome standard deviation is zero, so outcome-only GRPO assigns zero advantage to every response. Sign agreement with nonzero token-level advantages is therefore not a meaningful success criterion at these endpoints. We instead use the balanced regime for a controlled comparison. Across the 94 balanced trials obtained from fixed seeds 0–999, outcome-only GRPO versus $K = 8$ MC attains mean token-level sign accuracy of 0.939 versus 0.861 and mean MAE of 0.315 versus 0.437; all 94 trials favor GRPO on both metrics. The median-representative trial used in Figure 1 follows a deterministic, outcome-agnostic selection rule.

The informative mixed cases (1/8 through 7/8) exhibit persistent sign inconsistency problems:

- **Early tokens:** Sign accuracy frequently drops below 50%, meaning random guessing would perform equally well.
- **Trajectory crossings:** When reward trajectories of different-outcome responses cross (common in early-to-mid reasoning), MC estimation cannot

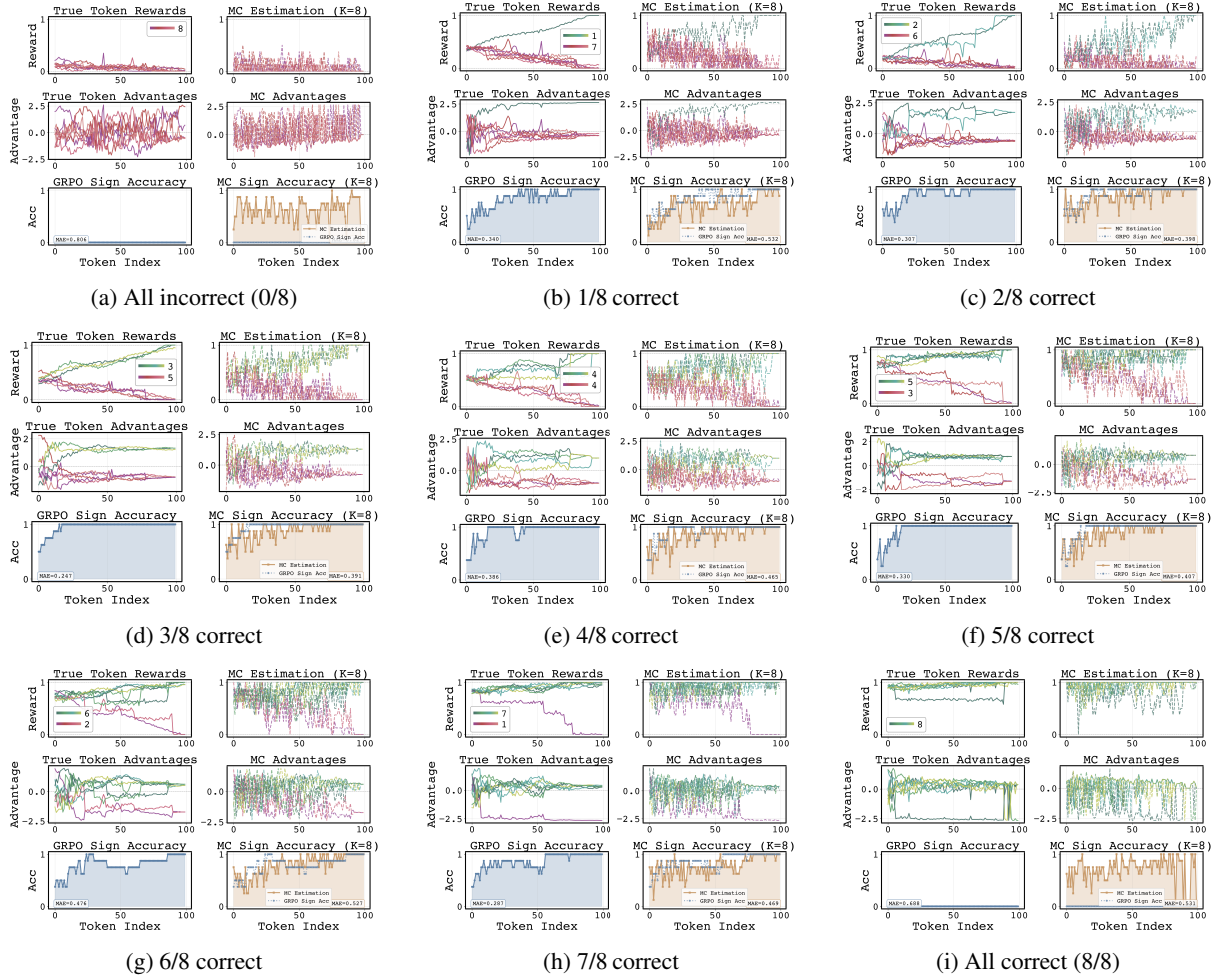


Figure 5: Illustrative advantage-estimation trajectories for every correctness count from 0/8 through 8/8. At the degenerate endpoints, outcome-only GRPO has zero group variance and therefore no meaningful within-group advantage signal. Mixed groups avoid this degeneracy but exhibit substantial finite- K sign errors, particularly at early positions.

reliably distinguish which response is on a better path.

- **Late convergence:** As the remaining continuation uncertainty vanishes, both estimators approach the terminal-outcome ordering and their sign accuracy improves.

However, as shown in Figure 6, MC estimation eventually surpasses GRPO with sufficient sampling. As K increases from 8 to 8192, token-averaged sign accuracy rises from 0.859 to 0.996 and MAE falls from 0.439 to 0.028; at $K = 8192$, both metrics improve over GRPO’s 0.940 sign accuracy and 0.313 MAE. Thus fine-grained MC credit can recover the token-level signal asymptotically, but full token-level branching requires an impractical sampling budget.

This analysis supports a narrower claim: in informative balanced groups, token-level Monte Carlo

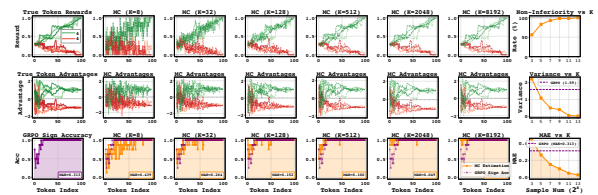


Figure 6: MC estimation accuracy across different sampling budgets ($K = 8$ to $K = 8192$). Left: true token rewards and advantages. Middle: MC estimates at each K . Right: pointwise non-inferiority rate, variance, and MAE. By $K = 512$, MC exceeds GRPO in token-averaged sign accuracy and has lower advantage MAE, but this requires up to $1000\times$ the $K = 8$ sampling budget.

at practical sampling budgets remains less reliable than the outcome-only baseline precisely in the early regime where fine-grained credit would be most useful.

A.4 How SCRoll Structurally Partitions the Problem

These findings explain why SCRoll’s architectural design is effective. In the **upstream stage**, although G parallel candidate segments are generated and one is selected via median-NLL ranking, only this single selected segment enters the upstream loss; group-relative normalization across the G candidates would still be vulnerable to the sign-error regime characterized above. SCRoll therefore uses single-stream policy optimization (SPO) with a temporal baseline $\hat{V}(q)$ on this one selected segment. This removes the fragile same-batch rank comparison among upstream candidates: the sign of the upstream advantage is measured against a prompt-specific historical return that becomes more informative as the prompt receives repeated observations, while the pre-update baseline prevents the current reward from being absorbed into its own reference value. In the **downstream stage**, with G rollouts conditioned on the same $[s, q]$ prefix, group-relative estimation becomes reliable: downstream responses are closer to terminal tokens where outcome variance is naturally lower, and the group size provides sufficient samples for stable advantage computation. This structural partitioning enables SCRoll to optimize segment-level behavior without relying on group-relative normalization over early upstream candidates.

A.5 Step-by-Step Data Flow Example

The following illustrative trace complements the pipeline overview in Figure 2. It shows how one prompt contributes to both objectives without introducing a second downstream rollout group.

1. **Read the prompt state and choose a boundary.** For a sampled question q , read its response-length state L_q and discounted count C_q^L , sample $t_q \sim \mathcal{U}[0, L_q]$, and apply the separate KV-cache block-alignment constraint.
2. **Generate candidate segments.** Sample $G = 8$ early-stopped candidates $s_1, \dots, s_8 \sim \pi_\theta(\cdot \mid q)$ at the shared boundary and compute each candidate’s per-token NLL.
3. **Select one segment.** Let $s = s_{k^*}$ be the candidate with median per-token NLL. Only s enters

the upstream loss; the other seven candidates are discarded after selection.

4. **Construct the skip-conditioned input.** Prepend the selected reasoning segment to the original question to obtain $\tilde{q} = [s, q]$.
5. **Sample one shared downstream group.** Draw $o_1, \dots, o_8 \sim \pi_\theta(\cdot \mid \tilde{q})$ and score their terminal answers. For example, an illustrative reward vector $(+1, +1, -1, +1, -1, +1, +1, -1)$ has mean $\bar{R} = 0.25$.
6. **Reuse the same outcomes in two objectives.** The upstream segment receives $\mathcal{R}_1(s, a) = \bar{R} = 0.25$ and is compared with the pre-update return tracker $\hat{V}_{-1}(q)$, whose discounted count is C_q^V . The downstream objective instead standardizes the eight individual rewards into group-relative advantages. Thus the same eight terminal outcomes supply one segment-level return and one ordinary group-relative training group; upstream and downstream token masks ensure that each loss is evaluated under its generation context.

The numeric reward vector is a didactic example rather than a reported experimental observation.

B Training Dynamics Analysis

We analyze the optimization behavior across different training configurations, examining convergence patterns, policy stability indicators, and computational efficiency metrics throughout the training process. All training curves presented in this section are from experiments on the **Qwen2.5-Math-7B** base model. For visual clarity, we select a representative subset of methods in each figure rather than displaying all baselines.

Critique-GRPO baseline. Critique-GRPO is our self-critique implementation in which the policy itself generates seven direct rollouts and one self-reflection rollout per prompt; it does not use an external stronger model. Because the direct and self-reflection samples arise from two different rollout processes, they are not an exchangeable group under a single conditioning process and cannot directly serve as SCRoll’s downstream group. We include Critique-GRPO only as a training-curve reference.

B.1 Performance Metrics

Figure 7 shows the mean accuracy (Mean@32) across training, which reflects the average qual-

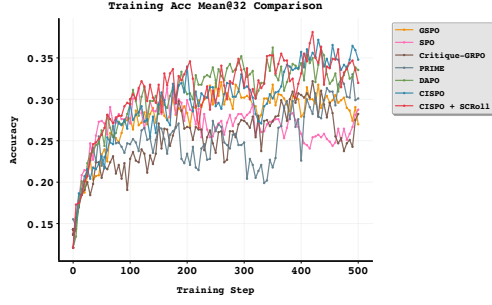


Figure 7: Mean accuracy at 32 samples across training steps.

ity across all sampled responses, serving as an indicator of consistent behavior and sample efficiency. SCroll demonstrates superior convergence compared to baseline methods, reaching higher asymptotic performance while maintaining stable training trajectories. Furthermore, SCroll exhibits faster initial improvement, which we attribute to the segment-level reward signals provided by averaging downstream outcomes for upstream segment optimization; traditional outcome-based methods receive sparse feedback only at trajectory endpoints, while SCroll’s segment-level rewards provide intermediate reward signals that accelerate policy learning in the critical early training phase.

B.2 Optimization Stability



Figure 8: Actor policy entropy throughout training.

Entropy Analysis. Figure 8 shows the actor policy entropy across training. Most methods exhibit decreasing entropy over the 500-step training schedule, with DAPO being the exception due to its explicit entropy regularization. This entropy reduction is a common behavior as policies converge toward higher-reward solutions. During our exploration of various algorithmic components, we also designed and experimented with some entropy-increasing mechanisms. Unfortunately, we found no clear correlation between entropy trends and

final performance, suggesting that our method may not be well-suited for improvements in this direction.

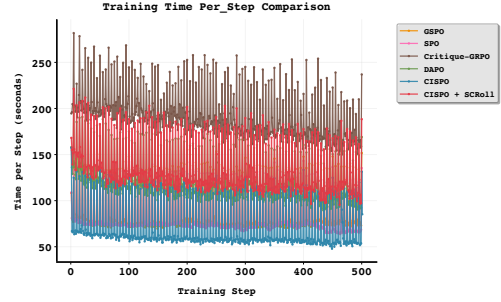


Figure 9: Training time per step across different methods. Despite the two-phase upstream-downstream architecture, SCroll remains close to single-phase baselines in our implementation: a custom verl rollout wrapper chooses a block-aligned split point and sends the downstream batch through vLLM with prefix caching enabled, without modifying the vLLM engine.

Training Time. Figure 9 compares the wall-clock training time per optimization step across methods. A naive two-batch implementation of SCroll would first generate upstream segments and then dispatch a separate batch for downstream continuations, incurring substantial overhead from sequential processing. Our implementation uses a custom rollout wrapper on top of verl: it constrains the split position t_q so that the reordered prefix $[s, q]$ lands on a vLLM KV cache block boundary, then dispatches the phase-2 continuations through the unmodified vLLM rollout path with prefix caching enabled. The realized wall-clock cost depends on prefix-cache hits, scheduling, cache capacity, and completion lengths; for this reason, we report empirical time-per-step rather than relying on a strict generated-token equivalence claim.

B.3 Compute Control

Decoupled two-process control. The two-process baseline trains one shared policy with two independent rollout groups: an actual-prefix MC branch samples $K = 8$ continuations from $[q, s]$ for the upstream segment return, while a vanilla branch samples $G = 8$ complete responses from q for the downstream CISPO objective. Unlike SCroll, neither group is reused for the other objective. Table 3 reports both the checkpoint matched to SCroll’s 0.58B generated-token budget and the final checkpoint at the same 500 optimization steps.

At matched generated tokens, SCRoll scores 48.4 versus 46.4; at the same number of steps, the decoupled baseline reaches 48.9 but consumes 1.04B tokens, approximately $1.79\times$ SCRoll’s generation budget. This comparison supports sample efficiency from rollout reuse, not an inherent advantage over an independently sampled baseline given additional compute.

C Experimental Setup and Hyperparameters

This section provides comprehensive details on the experimental configuration, training procedures, and evaluation protocols used throughout this work. All experiments are designed to ensure fair comparison across methods and reproducibility of results.

Dataset. We train all models on a curated mathematical reasoning dataset containing 17,398 problems after deduplication (dapo-math-17k). The training data consists of diverse mathematical problems spanning arithmetic, algebra, geometry, number theory, and competition mathematics. We evaluate on standard benchmarks including AIME 2024/2025 (American Invitational Mathematics Examination), AMC (American Mathematics Competition), MATH, Minerva Math, and Olympiad-level problems. For out-of-domain evaluation, we use MMLU-Pro (general reasoning across multiple subjects) and LiveCodeBench (code generation tasks). Benchmark evaluation is separated from the SCRoll training rollout: models are prompted with the original evaluation question q only, without selected upstream segments or two-stage test-time sampling. All training data is formatted with a consistent prompt structure, and responses are truncated to fit within maximum length constraints. We rely on the deduplicated dapo-math-17k release and do not claim exhaustive removal of paraphrastic overlap between public competition-style training problems and evaluation benchmarks.

Model Architectures. We conduct experiments on two base models: **Qwen2.5-Math-7B**, a 7-billion parameter language model specialized for mathematical reasoning, and **Llama-3.2-3B-Instruct**, a 3-billion parameter instruction-tuned model. Both models use the same tokenizer as their respective base model families. For SCRoll, the upstream and downstream phases share the same model parameters, with only the input conditioning (original problem q vs. augmented input $[s, q]$)

differing between phases.

Optimization Hyperparameters. We train all models for **500 optimization steps** with a **batch size of 128 prompts** per step. The learning rate is set to 1×10^{-6} with linear warmup over the first 10 steps and no decay thereafter. We use the AdamW optimizer with weight decay 0.1, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. Gradient norms are clipped at 1.0 to improve training stability. We employ mixed-precision training (bfloat16) to accelerate computation and reduce memory usage. The policy gradient mini-batch size is set to 32 prompts, meaning each optimization step processes 4 mini-batches ($128/32 = 4$). Loss aggregation uses token-mean normalization, where gradients from all tokens contribute equally after normalizing by response length.

SCRoll Configuration. For SCRoll, the upstream phase generates $G = 8$ parallel candidate segments $\{s_i\}_{i=1}^G$ from each prompt q , all terminated at a shared split position $t_q \sim \mathcal{U}[0, L_q]$. The response-length state is initialized to $L_q = 1024$ tokens; maximum segment length is implicitly bounded by the 3072-token response limit. The realized split is rounded down so that the reordered prefix $[s, q]$ ends on a vLLM KV-cache block boundary ($B = 16$ tokens), a separate engineering constraint that does not alter the tracker update. From $\{s_i\}_{i=1}^G$ we select s by median per-token NLL, discard the other $G - 1$ candidates, and issue $G = 8$ downstream requests with the identical augmented prompt $[s, q]$.

The return tracker $\hat{V}(q)$ and response-length tracker L_q have independent states and discounted counts C_q^V and C_q^L ; they share only the forgetting factor

$$\rho = \text{clip}\left(2^{-D_{\text{KL}}/\tau_{1/2}}, \rho_{\min}, \rho_{\max}\right).$$

For $x \in \{V, L\}$, let x_q denote the corresponding state. After observing $z_V = \mathcal{R}_1(s, a)$ and response length z_L , the updates are

$$C_q^x \leftarrow \rho C_q^x + 1, \quad x_q \leftarrow x_q + \frac{z_x - x_q}{C_q^x}.$$

Consequently, reward observations do not directly update L_q or the split distribution. We set $\rho_{\min} = 0.875$, $\rho_{\max} = 0.96$, and $\tau_{1/2} = 8.0$ steps. The return baseline is initialized to zero and both trackers update on their first observation; early-terminated

Table 3: Compute controls on Qwen2.5-Math-7B with CISPO as the downstream objective and $G = K = 8$. The decoupled two-process baseline uses an actual-prefix MC group for the upstream objective and an independent q -conditioned vanilla group for the downstream objective; neither group is reused. The generated-token-matched row is selected at 0.58B cumulative generated tokens, while “same steps” is the 500-step checkpoint.

Setting	AIME24	AIME25	AMC	MATH	Minerva	Olympiad	Avg.	Gen. tokens
CISPO ($G = 8$)	35.0	17.4	67.5	78.5	31.8	48.6	46.5	0.47B
Two-process, generated-token-matched	34.2	17.0	68.0	78.6	32.0	48.5	46.4	0.58B
Two-process, same steps	36.5	18.9	71.2	82.3	33.7	50.9	48.9	1.04B
CISPO + SCRoll ($G = 8$)	36.2	19.1	70.5	80.0	34.0	50.4	48.4	0.58B

candidates use half their realized response length as the split.

Upstream (SPO). We enable prioritized sampling with $\epsilon = 0.05$; after median-NLL selection, the upstream batch contributes $128 \text{ prompts} \times 1$ selected segment = 128 samples to the SPO loss.

Downstream. Per-optimizer losses and hyperparameters for GRPO, GSPO, DAPO, and CISPO are listed in Table 4; every downstream batch contains $128 \text{ prompts} \times 8 \text{ responses} = 1024$ samples. We use $w_{\text{up}} = w_{\text{down}} = 0.5$, batch-normalize upstream advantages as $\tilde{A} = (A - \mu_B)/\sigma_B$, and disable additional SCRoll-specific KL and entropy terms ($\beta = 0$, $\alpha_{\text{ent}} = 0$). Any KL term in Table 4 is therefore prescribed by the downstream optimizer itself.

Baseline Configurations. All baselines use the same training schedule (500 steps), batch size (128 prompts), learning rate (1×10^{-6}), and base model to ensure fair comparison. Since SPO generates only 1 rollout per prompt, its batch size is scaled to 1024 prompts to match the total sample count of other methods with group size 8. Baseline-specific hyperparameters follow their respective original papers.

Compute Infrastructure and Budget. All training and evaluation experiments are conducted on a single node with $8 \times$ NVIDIA H100 (80 GB) GPUs interconnected via NVLink, using bfloat16 mixed-precision training and vLLM (Kwon et al., 2023) as the rollout engine. A single training run (500 optimization steps for one (method, base model) configuration) takes approximately 18 hours of wall-clock time, corresponding to roughly ~ 144 GPU-hours per run. The total compute for the main results in Table 1, namely two base models (Qwen2.5-Math-7B and Llama-3.2-3B-Instruct) $\times$ nine optimization methods (SCRoll and eight baselines), together with the ablation studies, amounts to approximately

$\sim 3,000$ GPU-hours in total.²

Licensing of Artifacts. All models, datasets, and evaluation benchmarks used in this work are publicly released for research use. Qwen2.5-Math-7B is released under the Apache 2.0 license; Llama-3.2-3B-Instruct is released under the Llama 3.2 Community License Agreement. The dap-math-17k training set is released by Yu et al. (2025b) for research use, with constituent problems originating from publicly available mathematical competitions. The evaluation benchmarks (MATH, AIME 2024/2025, AMC, Minerva Math, Olympiad, MMLU-Pro, and LiveCodeBench) are publicly available for academic evaluation. Our use of these artifacts is consistent with their intended research purpose. The trained models and the rollout-side block-aligned split-point selection code produced in this work will be released under an open-source license compatible with the upstream artifacts; the underlying vLLM and verl stacks are used unmodified.

Rollout Implementation. SCRoll is implemented as a custom rollout wrapper on top of verl and vLLM. The wrapper first samples block-aligned upstream candidate segments from the original prompt, selects the median-NLL segment, and then dispatches the downstream requests with identical augmented prompts $[s, q]$ to the same vLLM-backed rollout path with prefix caching enabled. The vLLM engine itself is not modified; the additional logic lives at the rollout/data-construction layer, where upstream and downstream tokens are masked separately so that log-probabilities are computed under the context used to generate each stage.

²Numbers are best estimates from per-step wall-clock measurements (cf. Figure 9); exact totals vary by $\pm 10\%$ across methods due to differences in rollout strategy.

Table 4: Per-optimizer loss formulations and hyperparameters used in the downstream slot of SCRoll and in the standalone baselines. All +SCRoll variants share the same upstream setup; standalone baselines use the corresponding downstream optimizer alone. Across paired comparisons, we keep the batch size (128 prompts), learning rate (1×10^{-6}), group size $G=8$, and dapo-math-17k training data fixed; only the loss form and per-loss hyperparameters differ. Token-level / sequence-level loss aggregation modes follow the original publications. “IS” denotes the importance-sampling ratio $\pi_\theta / \pi_{\theta_{\text{old}}}$.

Optimizer	Loss form (per original paper)	Hyperparameters
GRPO (Shao et al., 2024)	Token-level PPO clip on the IS ratio with group-relative advantages $\hat{A}_i = (R_i - \bar{R}) / \sigma_R$; symmetric clipping range $[1 - \varepsilon, 1 + \varepsilon]$.	$\varepsilon_{\text{low}}=0.2$, $\varepsilon_{\text{high}}=0.2$; loss_agg_mode=token-mean; use_kl_loss=True with $\beta_{\text{KL}}=10^{-3}$ (low-variance KL).
GSPO (Zheng et al., 2025)	Sequence-level importance sampling: clip the geometric-mean per-sequence IS ratio rather than per-token, eliminating per-token noise amplification.	$\varepsilon_{\text{low}}=3 \times 10^{-4}$, $\varepsilon_{\text{high}}=4 \times 10^{-4}$, $\varepsilon_c=10.0$ (dual-clip); loss_agg_mode=seq-mean-token-mean; use_kl_loss=False.
DAPO (Yu et al., 2025b)	Asymmetric clip-higher PPO loss with dynamic resampling: enlarges the upper clip bound so high-advantage updates are not throttled; rejects groups whose reward variance is degenerate; adds an overlong-response shaping penalty.	$\varepsilon_{\text{low}}=0.2$, $\varepsilon_{\text{high}}=0.28$, $\varepsilon_c=10.0$ (dual-clip); token-mean loss aggregation; no KL loss; degenerate-group filtering enabled; overlong buffer $L_{\text{buf}}=512$ tokens with penalty factor 1.0.
CISPO (Mini-Max, 2025)	Truncated-IS REINFORCE-style loss: rather than clipping the gradient via PPO’s $\min(r\hat{A}, \text{clip}(r)\hat{A})$, CISPO clips the IS factor itself (with stop-gradient) and applies it as a per-token weight on the policy log-prob, retaining gradient signal at out-of-trust-region tokens.	$\varepsilon_{\text{IS-low}}=10.0$ (IS upper cap), $\varepsilon_{\text{high}}=0.28$; use_kl_loss=True with $\beta_{\text{KL}}=10^{-3}$ (low-variance KL); loss_agg_mode=token-mean.