SINGLE-ROLLOUT ASYNCHRONOUS OPTIMIZATION FOR AGENTIC REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) is becoming increasingly important for post-training large language models (LLMs). Previous RL pipelines for LLMs were mostly synchronous and batch-interleaved, which is inefficient for long-horizon agentic tasks. Recently, asynchronous RL has emerged as a more efficient alternative by updating the model as rollouts arrive. Building on advances in asynchronous RL efficiency and off-policy robustness, we study stable and effective single-rollout optimization for long-horizon agentic tasks. For example, a key challenge is that group-wise sampling in the widely-used GRPO framework does not naturally fit asynchronous agentic training. In this paper, we present Single-rollout Asynchronous Optimization (SAO) to address the stability and off-policy challenges in asynchronous RL. To reduce off-policy effects and improve generalization, we replace group-wise sampling with single-rollout sampling, that is, using one rollout per prompt. We further improve this single-rollout strategy with practical value-model training designs. To improve optimization stability, we introduce a strict double-side token-level clipping strategy. SAO is able to train stably for one thousand steps and consistently outperform GRPO and its variants on agentic coding and reasoning benchmarks, such as SWE-Bench Verified, BeyondAIME, and IMOAnswerBench. We also demonstrate that single-rollout RL is particularly effective in a simulated online learning setting, where the model must adapt to changing evolving environments. To this end, SAO is successfully deployed in the agentic RL pipeline for training the open GLM-5.2 model (750B-A40B).

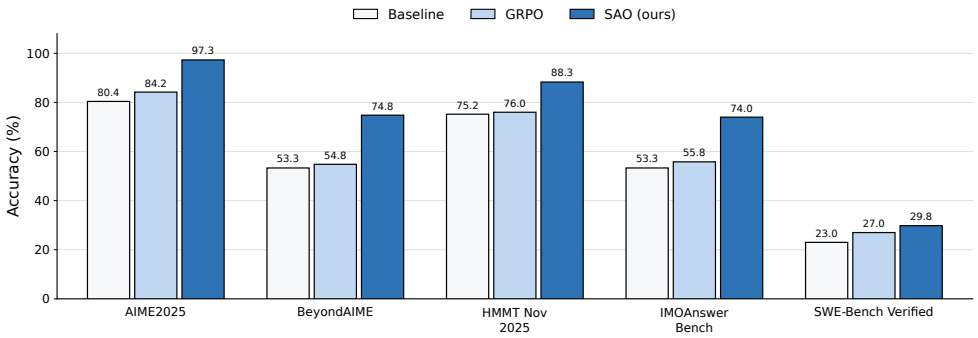


Figure 1: **The performance of SAO on reasoning and coding benchmarks.** The four reasoning benchmarks are evaluated in a reasoning-with-Python-tool setting, where the baseline is the Qwen3-30B-A3B SFT model; SWE-Bench Verified evaluates coding with the Qwen3-30B-A3B baseline. SAO outperforms the corresponding baseline and GRPO across all five benchmarks.

1 INTRODUCTION

Large Language Model (LLM) development is shifting from supervised pre-training toward post-training reinforcement learning (RL). Recent work in Reinforcement Learning has demonstrated that scaling RL compute together with test-time compute is highly effective to improve model intelligence (DeepSeek-AI, 2025; OpenAI, 2024; Cobbe et al., 2021; Lightman et al., 2023). Most LLM RL

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

pipelines remain synchronous: the policy generates a batch of rollouts, and optimization starts only after the entire batch is collected (Ouyang et al., 2022; Noukhovitch et al., 2024).

For agentic and coding workloads, rollout lengths are highly variable, so short trajectories finish quickly while long ones become stragglers; as a result, large portions of the GPU cluster idle while waiting for the slowest rollouts (Fu et al., 2025; Lu et al., 2025). Asynchronous RL mitigates this *imbalanced generation overhead* by consuming rollouts continuously as they arrive, improving utilization and wall-clock efficiency (Mnih et al., 2016; Liang et al., 2018; Hoffman et al., 2020).

However, asynchrony introduces two challenges. First, a trajectory may span multiple rollout-policy versions, increasing policy lag and complicating stable optimization. Prior work advances both asynchronous efficiency and off-policy robustness (Fu et al., 2025; Noukhovitch et al., 2024; Lu et al., 2025); we focus on stable and effective single-rollout optimization for long-horizon agentic RL. Second, group-wise methods such as GRPO (Shao et al., 2024) retain prompt-group dependencies within asynchronous training. GRPO samples a group of responses for each prompt and uses the group-level average for advantage estimation. The group-wise sampling induces latency-driven off-policy behavior because the group has to wait for the slower one to finish before fed into training. In addition, group-wise sampling is incompatible with online or complex agentic settings where the environment often provides only a single trajectory feedback per prompt (Sutton & Barto, 2018; Schulman et al., 2017; Yao et al., 2022; Nakano et al., 2021).

In this paper, we propose Single-rollout Asynchronous Optimization (SAO) for agentic RL. It keeps asynchronous RL training stable and effective under policy lag while preserving the efficiency of asynchrony. Instead of group-wise sampling, such as GRPO, SAO uses one rollout per prompt, allowing completed trajectories to enter training batches without waiting for other responses to the same prompt. Our contribution is the combined asynchronous optimization design and its empirical validation, integrating direct off-policy correction with effective value-model training. Our contributions are as follows:

- To stabilize training under varied policy lag, we directly form the current-to-rollout probability ratio from logged behavior probabilities and apply double-sided hard masking. In contrast to IcePop's separate engine-mismatch calibration and PPO update (Ling Team et al., 2025), this design jointly handles policy drift and engine mismatch without old-policy inference.
- To reduce off-policy effects, we use one single rollout sampling for each prompt instead of group-wise sampling previously populated by GRPO. To further make this setting practical in agentic RL, we improve value-model training. Specifically, we combine more frequent critic updates, following established actor–critic practice (Fujimoto et al., 2018), with frozen-attention value tuning and evaluate their effects under asynchronous training.
- To handle multi-turn agent trajectories with interleaved environment feedback, we integrate observation-skipping token-level GAE (Wang et al., 2025) into single-rollout asynchronous training and evaluate token-level versus step-level credit assignment.

We evaluate SAO on agentic coding and math reasoning benchmarks, including SWE-Bench Verified (Jimenez et al., 2023), AIME2025 (Balunović et al., 2025), BeyondAIME (ByteDance Seed, 2025), HMMT (Balunović et al., 2025), and IMOAnswerBench (Luong et al., 2025). The results demonstrate that our asynchronous RL design can stably train for around one thousand steps and achieves consistently better performance than improved GRPO. In addition, we show that the single-rollout strategy in SAO is uniquely suited for simulated online learning, where it can adapt to dynamic environmental changes.

2 PRELIMINARIES

In reinforcement learning for language models, the model is parameterized by θ as a stochastic policy $\pi_\theta(y|q)$, which generates a response sequence $y = [y_1, \ldots, y_{|y|}]$ given a query q from dataset $\mathcal{D}$. RL optimizes π_θ by maximizing a clipped surrogate objective that encourages stable policy updates. Formally, for a given batch of data, the unified optimization target is defined as:

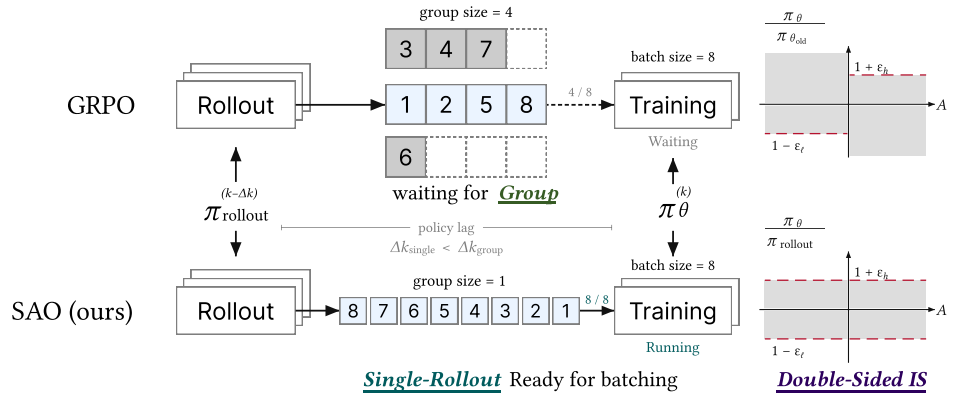


Figure 2: Overview of SAO with single rollout design. The numbers denote the generation order of trajectories. For SAO, each trajectory becomes available for training immediately upon completion. In contrast, GRPO must wait until all trajectories in a group are generated before training can begin.

$$\mathbb{E}\left[\frac{1}{|y|}\sum_{t=1}^{|y|}\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

where $r_t(\theta) = \frac{\pi_\theta(y_t|q,y_{<t})}{\pi_{\theta_{\text{old}}}(y_t|q,y_{<t})}$ is the probability ratio between the current and old policies, ϵ is the clipping hyperparameter. The fundamental distinction between PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) lies in how they estimate the advantage $\hat{A}_t$ and whether they use an auxiliary value network. Appendix B gives the group-relative objective.

Proximal Policy Optimization (PPO). Standard PPO typically adopts an Actor-Critic architecture, requiring the training of a separate value function (Critic) V_ϕ, parameterized by ϕ, to estimate the expected return of the current state. This critic is optimized concurrently with the policy to minimize the value error $\mathcal{L}_\phi^{\text{VF}} = \mathbb{E}[(V_\phi(q, y_{<t}) - R)^2]$, where R denotes the cumulative reward. To balance bias and variance, PPO employs Generalized Advantage Estimation (GAE). The advantage $\hat{A}_t^{\text{GAE}}$ is computed as an exponentially weighted sum of temporal difference errors:

$$\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{|y|-t-1}(\gamma\lambda)^l\delta_{t+l}$$

where $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$. While effective, this approach necessitates maintaining a copy of the model parameters for the value function, essentially doubling the memory footprint during training and increasing computational overhead.

3 Asynchronous Reinforcement Learning with Single Rollout

In this section, we introduce SAO to tackle training instability and off-policy drift in asynchronous RL training. With a simple token-level clipping strategy and single rollout as an alternative to group-wise sampling, we show that asynchronous RL can be stably scaled to thousands of training steps and achieve significant performance improvements. Figure 2 shows the overall design of SAO.

3.1 Stabilizing Asynchronous RL via Direct Double-Sided Importance Sampling

A primary challenge in asynchronous RL is policy lag between rollout generation and policy updates, compounded by training–inference mismatch. We propose Direct Double-Sided Importance Sampling (DIS), which uses the logged per-token rollout probability as the behavior probability and directly compares it with the current training policy:

$$r_t(\theta) = \exp\left(\log \pi_\theta(a_t|s_t) - \log \pi_{\text{rollout}}(a_t|s_t)\right) \tag{1}$$

We apply double-sided hard masking to this ratio: tokens outside $[1-\epsilon_\ell, 1+\epsilon_h]$ contribute no policy gradient, regardless of the advantage sign. The token-level calibration function is

$$f(x;\epsilon_\ell,\epsilon_h) = \begin{cases} x, & \text{if } 1-\epsilon_\ell < x < 1+\epsilon_h \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The resulting policy objective is

$$L(\theta) = \hat{\mathbb{E}}_t \left[\mathrm{sg}(f(r_t(\theta),\epsilon_l,\epsilon_h))\, \hat{A}_t \log \pi_\theta(a_t|s_t) \right] \tag{3}$$

Here L is maximized, sg denotes stop-gradient, and $\hat{A}_t$ is held fixed during the policy update. Gradients flow only through $\log \pi_\theta$; the ratio and its mask are detached.

DIS adds negligible overhead and does not materially affect training efficiency; implementation details are given in Appendix D. Compared with IcePop (Ling Team et al., 2025), which handles engine mismatch and policy updates using separate ratios, DIS masks the direct current-to-rollout ratio. This jointly handles both discrepancies without old-policy inference and supports trajectories spanning rollout-policy versions; a detailed comparison appears in Appendix C.

3.2 Reducing Off-Policy with Single Rollout

In asynchronous RL, an inevitable problem is off-policy. Yet current popular group-wise sampling RL algorithms, e.g., GRPO, could introduce more severe off-policy. Group-wise sampling introduces an "imbalanced generation" bias, and the group data has to wait for the "slowest" sample to finish before being fed into training. One promising solution is to replace group-wise sampling with one rollout per prompt. Each completed trajectory becomes available for a training batch without waiting for other responses to the same prompt; an optimizer step still consumes a batch of trajectories. This targets asynchronous training directly, while also fitting online settings with one feedback trajectory per prompt.

However, single-rollout optimization inherently suffers from high variance in gradient estimation, similar to REINFORCE (Zhang et al., 2021). To reduce variance requires a sufficiently good value model. In this part, we focus on simple strategies to optimize value modeling to ultimately boost the policy's performance.

Faster Value Update than Policy. We identify that the primary source of instability in single-rollout RL is the interdependence between the policy and the value function. If the value model V_ϕ is inaccurate, the advantage estimates $\hat{A}_t$ become noisy, leading to destructive policy updates. To mitigate this, we use faster value updates, following the established practice of updating the critic more frequently than the actor (Fujimoto et al., 2018). Specifically, for every single gradient update applied to the policy π_θ, we enforce K updates to the value network V_ϕ (where $K > 1$). In our experiments, we set $K = 2$ so that the critic tracks current returns more closely, supporting stable advantage estimation. The higher explained variance in Appendix Figure 5(a) is consistent with improved value tracking.

Stabilizing Value Model Training via Parameter Freezing. In our pilot experiments, we find the instability of value model training, where the gradient norms of the value model are significantly larger than the corresponding policy model. Further decomposition shows that this instability originates primarily from the Full Attention layers, whereas the Mixture-of-Experts (MoE) layers remain relatively stable. Based on this observation, we employ a "Frozen-Attention" training strategy for the value model. During the RL training, we freeze the parameters of the attention modules in V_ϕ and optimize the MoE projections. We hypothesize that the pre-trained attention weights already possess sufficient semantic capability to attend to relevant tokens. By restricting optimization to the MoE layers, we effectively regularize the value model.

Skip-Observation Token-level GAE for Agentic Tasks. Agentic trajectories interleave model actions and environment feedback: $T = [a_0, o_0, a_1, o_1, \dots]$. Observation tokens provide context but are not policy actions. We therefore compute GAE over generated action-token positions, bridging the intervening observations rather than treating them as additional action steps.

Table 1: Experimental Results on math reasoning benchmarks(Accuracy %).

Model	AIME2025	BeyondAIME	HMMT Nov 2025	IMOAnswerBench
Claude-Sonnet-4.5	87.0	62.0	81.7	65.8
GPT-5 High	94.6	74.0	89.2	76.0
GLM-4.7	95.7	-	93.5	82.0
Qwen3-30B-A3B				
w/ python	14.6	10.5	17.3	7.8
w/o python	85.0	63.0	76.7	55.3
SFT (w/ python)	80.4	53.3	75.2	53.3
SFT (w/o python)	14.6	46.8	17.3	42.0
GRPO (w/ python)	84.2	54.8	76.0	55.8
Qwen3-30B-A3B				
SAO (ours)	**97.3**	**74.8**	**88.3**	**74.0**
- *SAO (w/ DIS only)*	94.2	71.5	86.7	71.3
- *GRPO (+ DIS)*	93.5	70.8	84.0	70.0

Table 2: Experimental Results on SWE-Bench Verified (Accuracy %).

Model	Accuracy (%)
Qwen3-30B-A3B	23.0
+ GRPO (w/ DIS)	27.0
+ SAO (ours)	29.8

We use "Skip-Observation" GAE to link the current action token to the next generated action token across environment feedback. Formally, let $a_{i,N}$ be the last token of action i, and $a_{i+1,0}$ be the first token of the next action. We define the advantage as:

$$\hat{A}(a_{i,N}) = \delta + \gamma\lambda\hat{A}(a_{i+1,0}) \tag{4}$$

where the temporal difference residual δ is calculated bridging the observation gap:

$$\delta = r_t + \gamma V(a_{i+1,0}) - V(a_{i,N}) \tag{5}$$

Here each value is conditioned on the full preceding context, including observations. VAGEN's Bi-Level GAE (Wang et al., 2025) uses turn-level targets, whereas our experiments favor token-level estimation over the turn-level variants we evaluate; details are provided in Appendix A.2.

Scaling Value Pretraining. Finally, we find it essential to scale the data used for value model pretraining. We construct this corpus from the RL training set and a subset of the SFT training set, using the SFT model to generate 16 samples per prompt. Our experiments demonstrate that the "cold start" problem in value estimation is a major bottleneck. By significantly increasing the scale of the value pretraining corpus, we provide a robust initialization point that promotes the effectiveness of our single-rollout and TTUR mechanisms from the early stages of training.

4 Experiments

4.1 Experimental Setup

Training Details. All algorithmic comparisons use asynchronous rollout and training, including the GRPO, VAPO, and running-mean baselines. For math reasoning with Python, we finetune Qwen3-30B-A3B-Thinking-2507 (Yang et al., 2025a) for 3 epochs on Tool-Integrated Reasoning (TIR) data produced by GPT-OSS-120B (OpenAI, 2025) and use the finetuned model to initialize the

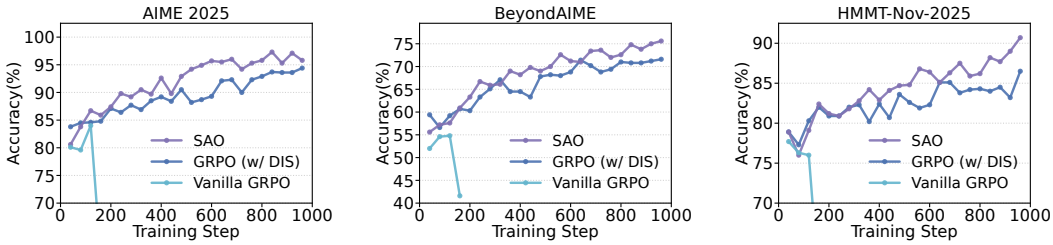


Figure 3: Performance comparison between SAO and GRPO (w/ DIS) during training. It can be observed that SAO almost consistently outperforms the optimized GRPO during the training process on different benchmarks.

policy and value model. TIR requires the model to interleave natural-language math reasoning with Python tool calls.

For RL of agentic reasoning, we employ a batch size of 128, a group size of 1, and a max-length of 128k tokens. The policy is optimized with a learning rate of 1×10^{-6}, with token-masking parameters $\epsilon_\ell = 0.3$ and $\epsilon_h = 4.0$, corresponding to ratio bounds of 0.7 and 5.0. We adopt a length-adaptive GAE (Yue et al., 2025) with $\lambda_{\text{policy}} = 1 - \frac{1}{\alpha l}$ and $\alpha = 1.5$. The value model is trained with a learning rate of 5×10^{-6} and $\lambda_{\text{critic}} = 1$. We set the $K = 2$ for faster value update for the value model, performing two value model updates per batch. For GRPO variants, each training batch contains 16 prompts with 8 rollout samples per prompt, yielding the same batch size of 128. For the RL of coding agent, we directly use Qwen3-30B-A3B-Thinking-2507 for training and keep almost all the hyperparameters the same as TIR, except for $\epsilon_{\text{low}} = 0.8$ and $\epsilon_{\text{high}} = 3.0$. For SWE-Bench Verified, we use OpenHands as the scaffold, with a maximum of 300 turns and a 128k context budget.

Evaluation. We evaluate SAO on four math reasoning benchmarks including AIME2025, BeyondAIME (ByteDance Seed, 2025), HMMT Nov 2025 (Balunović et al., 2025) and IMOAnswerBench (Luong et al., 2025), reporting Pass@1 accuracy. All evaluations use top-$p = 1.0$, temperature 1.0, and a maximum generation length of 128k tokens. Math-reasoning evaluations allow up to 50 turns to support extensive reasoning and tool calls, while SWE-Bench Verified evaluations allow up to 300 OpenHands interaction turns. To reduce variance, we report the mean performance across 16 evaluation runs for AIME2025 / HMMT / IMOAnswerBench and 4 runs for BeyondAIME.

4.2 Main Results

Tables 1 and 2 summarize the performance of baselines and different training strategies. GRPO denotes Async GRPO with DAPO (Yu et al., 2025), using clip-higher and token-level loss; it keeps the latest old policy for importance sampling. GRPO (w/ DIS) retains group-relative advantages and adds DIS. SAO (w/ DIS only) instead uses one rollout per prompt and a learned value model with DIS, without the additional value-training designs, including faster critic updates, frozen attention, and Skip-Observation GAE.

As shown in Tables 1 and 2, SAO consistently outperforms all baselines on both agentic reasoning and coding benchmarks. Standard GRPO suffers from a performance collapse at approximately 160 training steps The scores reported for these models represent their final valid performance before collapsing. Figure 3 illustrates the evaluation performance across training steps of SAO compared to vanilla GRPO and GRPO (w/ DIS). Vanilla GRPO tends to quickly collapse, while GRPO with DIS can achieve stable training, demonstrating the effectiveness of DIS. In addition, SAO and GRPO (w/ DIS) exhibit comparable performance in the initial stage; a distinct performance divergence occurs after approximately 400 training steps, demonstrating the effectiveness and stability of SAO.

The SWE-Bench Verified results use the earlier-generation Qwen3-30B-A3B backbone, a compact MoE with 3B active parameters. Frontier-scale validation on the 750B-total/40B-active GLM-5.2 model is presented separately in Section 4.5. System-efficiency measurements for the asynchronous pipeline appear in Appendix D.

4.3 Ablation Studies

We conduct extensive ablation studies to evaluate the impact of various training configurations on the performance of SAO. The results are shown in Table 4.

- *Effects of faster value update*: To ablate the effects of faster value update than the policy model, we conduct experiments where the value model is updated only once per batch (critic-train-epoch=1), as opposed to the two updates per batch employed in SAO.
- *Full vs. frozen-attention value model.*: To evaluate the impact of attention-freezing during RL, this variant performs full-parameter updates on the value model.
- *Vanilla VAPO and single-rollout with Running-mean baseline.* Standard VAPO (Yue et al., 2025) with length-adaptive GAE and a value-based RL baseline. Besides, a single-rollout baseline that maintains a sliding window of the 8 most recent rewards for each prompt, using their mean as a baseline for advantage estimation to provide a simple alternative to parametric value models.

As shown in Table 4, all examined variants exhibit a performance decline relative to the proposed SAO, validating the necessity of each design choice. Table 3 further summarizes the value-training strategy and critic-update settings behind the main value-model ablations. Regarding update frequency, the results indicate that a single update is insufficient for the critic to accurately track rapid policy shifts, leading to less reliable baseline estimations. The full-parameter value-training variant further suggests that frozen-attention updates help regularize critic optimization in complex reasoning tasks. In addition, the RL with running-mean reward lags far from SAO, demonstrating the advantage and necessity of a well-trained value model. As for the vanilla VAPO, the training also quickly collapses during training, similar to the vanilla GRPO.

Table 3: Ablation results of value model training strategy and critic update frequency. We compare partial parameters, i.e., frozen-attention, with full-parameter value update in RL training, as well as the effectiveness of faster critic updates per policy step. We report Accuracy (%) for all datasets

	Value Training Strategy	Critic Update Frequency	AIME2025	BeyondAIME
SAO	Frozen Attention	2	97.3	74.8
Single-step-update	Frozen Attention	1	95.00	69.75
Full-Parameter Value Training	Full-Parameter	2	90.62	74.50

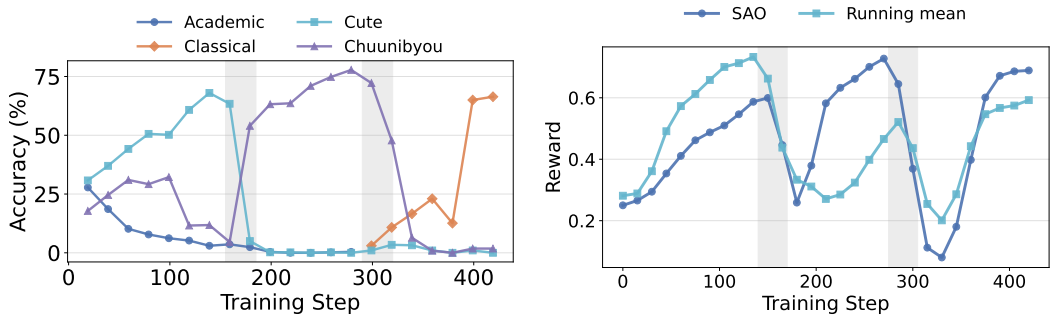
Table 4: Ablation results of value model training strategy and critic update frequency. We compare partial parameters, i.e., frozen-attention, with full-parameter value update in RL training, as well as the effectiveness of faster critic updates per policy step. We report Accuracy (%) for all datasets

	AIME2025	BeyondAIME
SAO	97.3	74.8
SAO w/o Faster value	95.0	69.8
SAO w/o Frozen attention	90.6	74.5
Vanilla VAPO (w/o DIS)	91.3	69.0
Running mean baseline	79.8	55.3

4.4 Online Learning Simulation

Task Design. In real-world online learning environments, feedback is typically restricted to a single trajectory per prompt. This constraint is inherently incompatible with group-based optimization strategies like GRPO, which depend on relative rewards within a sample group for advantage estimation. In contrast, SAO utilizes a value-based critic to provide advantage estimation, allowing for effective policy updates from individual trajectories.

Therefore, we design a simulated online writing task to assess the adaptability of SAO in non-stationary environments. In this setting, the feedback signal is designed as the language tone of user



(a) Held-out accuracy across three writing styles. SAO rapidly adapts to preference shifts (shaded regions).

(b) Training rewards under preference shifts (shaded regions). SAO adapts faster and achieves higher rewards than the running-mean baseline.

Figure 4: Online learning simulation under changing writing-style preferences.

preference. Reward criteria are sequentially adjusted to favor three distinct stylistic archetypes: *cute*, *chuunibyou*, and *classical*.

Dynamic Reward Assignment. For reward signal assignment, we employ GLM-4.7 (Z.ai, 2025) as an LLM-based judge to evaluate two primary dimensions: response quality and stylistic adherence. The final reward $r \in \{0, 1\}$ is computed as: $r = r_{\text{quality}} \times r_{\text{style}}$ where $r_{\text{quality}}, r_{\text{style}} \in \{0, 1\}$ denote binary rewards. In our actual experiments, the candidate stylistic archetype set consists of *Academic, Cute, Chuunibyou* during the first two phases, and *Classical, Cute, Chuunibyou* in the final phase.

Results. As illustrated in Figure 4a, we evaluate the performance of the three candidate linguistic styles of each phase on a held-out test set . SAO demonstrates rapid policy realignment following each reward preference shift, characterized by transitions between stylistic archetypes to maintain adherence to the evolving environmental feedback.

Comparison against Running-Mean Baseline. To better understand the effectiveness of the value model in the online environment, we also adopt the Running Mean Advantage Estimation approach as the baseline. This method approximates the baseline b by tracking a sliding window of the 128 most recent rewards, thereby facilitating advantage computation as $\hat{A} = r - \mathbb{E}[r_{window}]$. By decoupling advantage estimation from intra-prompt sample groups, this setup permits policy optimization in an online, single-rollout context. Figure 4b depicts the evolution of training rewards of SAO and the Running Mean baseline, where the speed and magnitude of reward recovery following stylistic shifts serve as key indicators of algorithmic adaptability. The Running Mean baseline exhibits an adaptation lag as its historical window retains rewards from the preceding distribution. In contrast, SAO's critic tracks reward shifts and achieves faster recovery and higher rewards in this setting. An adaptive window can strengthen a history-based baseline, but introduces change-detection thresholds and window-control rules that may require scenario-specific tuning. SAO learns its conditional baseline directly from feedback, without an additional change-point detector or window-control schedule.

4.5 Frontier-Scale Validation on GLM-5.2

Beyond the Qwen3-30B-A3B experiments, we deploy SAO in the agentic RL pipeline for the open-weight GLM-5.2 model, a 750B-total/40B-active MoE. Starting from GLM-5.2 SFT, SAO-based RL improves all five evaluated agentic coding benchmarks (Table 5). Both the Async GRPO and SAO training runs incorporate compactionRL, which supports context compaction during training rollouts. Relative to SFT, SAO achieves gains of 3.6 points on SWE-Bench Pro, 2.0 on Terminal v2.1, 3.7 on NL2Repo, 12.8 on DeepSWE with miniSWE-agent, and 7.9 on ProgramBench. These results demonstrate the effectiveness of single-rollout asynchronous optimization at frontier model scale.

5 Related Work

5.1 Reinforcement Learning for Language Models

The standard RLHF pipeline trains a reward model from preference data and optimizes the policy with PPO (Ouyang et al., 2022; Schulman et al., 2017). To reduce the overhead and instability of value-

Model	SWE-Bench Pro	Terminal v2.1	NL2Repo	DeepSWE miniSWE-agent	ProgramBench
GLM-5.2 SFT	58.4	76.0	44.4	38.1	55.3
GLM-5.2 SFT + Async GRPO	59.6	77.5	**49.5**	37.2	–
GLM-5.2 SFT + SAO	**62.0**	**78.0**	48.1	**50.9**	**63.2**

Table 5: GLM-5.2 agentic benchmark results (%).

function learning, critic-free objectives such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024; DeepSeek-AI, 2025) and REINFORCE-style baselines (e.g., RLOO) (Ahmadian et al., 2024) have become increasingly popular. GRPO forms advantages by normalizing rewards within a prompt-level group, which improves stability in synchronous training but introduces an implicit synchronization barrier: updates must wait until all group members are generated, exacerbating staleness and off-policy drift under asynchrony.

Recent work further refines GRPO/PPO-style objectives through DAPO's clip-higher and token-level loss (Yu et al., 2025), sequence-level importance weighting (Zheng et al., 2025), dynamic clipping (Yang et al., 2025b), and smooth adaptive weighting (Gao et al., 2025). VAPO (Yue et al., 2025) improves value-based reasoning RL with designs including length-adaptive GAE. IcePop (Ling Team et al., 2025) masks training–inference discrepancies separately from PPO policy-update clipping; SAO instead masks the combined current-to-rollout ratio. More frequent critic updates have precedents in actor–critic learning (Fujimoto et al., 2018), and observation-skipping GAE appears in VAGEN (Wang et al., 2025). We integrate these principles with direct off-policy correction and frozen-attention critic tuning for long-horizon asynchronous agentic RL.

5.2 Synchronous and Asynchronous RL for LLMs

Most large-scale LLM RL implementations remain synchronous: collect a batch of rollouts with a policy snapshot, then run optimizations on that batch (Ouyang et al., 2022). With long-tail lengths in reasoning and tool-use, synchronous barriers cause stragglers and substantial idle time, motivating asynchronous actor–learner designs where rollout generation and learning proceed concurrently (Mnih et al., 2016; Sutton & Barto, 2018). However, asynchrony introduces policy lag and off-policy drift, often requiring staleness-aware training or off-policy corrections (Espeholt et al., 2018).

Several recent systems target asynchronous RL specifically for LLMs. Noukhovitch et al. (2024) study asynchronous RLHF as online-but-off-policy learning and characterize efficiency–robustness tradeoffs. AReaL (Fu et al., 2025) fully decouples rollout from training and incorporates staleness-aware PPO-style updates for reasoning tasks. ROLL Flash provides fine-grained parallelism and rollout–train decoupling for RLVR and agentic training, and evaluates off-policy algorithms under asynchrony (Lu et al., 2025). Complementary to asynchronous systems, MobileRL studies online agentic RL for mobile GUI agents and introduces difficulty-adaptive GRPO variants to improve stability and sample efficiency in multi-turn GUI environments (Xu et al., 2025). Our work complements these advances by focusing on stable and effective single-rollout optimization in long-horizon agentic RL. We compare algorithms within the same asynchronous pipeline to study direct off-policy correction and value learning; the online experiment additionally tests adaptation when only one feedback trajectory is available per prompt.

6 Conclusion

In this work, we explore the optimization of asynchronous RL on the training effectiveness and stability We proposed SAO, a single-rollout asynchronous RL strategy that addresses off-policy and instability. SAO stabilizes training with token-level importance sampling and double-sided clipping/masking, and improves generalization by replacing group-wise sampling with single-rollout enabled by stronger value-model training. On agentic reasoning and coding tasks, SAO shows consistent outperformance over GRPO baselines, and adapts effectively in simulated online learning.

AI USE STATEMENT

We used generative AI tools to assist with language polishing, literature checking, and manuscript review. The authors are responsible for the research design, implementation, experiments, and interpretation of results. All AI-assisted material is subject to author review and verification. We take full responsibility for the accuracy and integrity of the final manuscript, including all content developed with generative AI assistance.

REFERENCES

Arash Ahmadian et al. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *arXiv preprint arXiv:2402.14740*, 2024.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025.

ByteDance Seed. BeyondAIME, 2025. URL https://huggingface.co/datasets/ByteDance-Seed/BeyondAIME. Hugging Face dataset.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Lasse Espeholt et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, 2018.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1587–1596. PMLR, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint arXiv:2511.20347*, 2025.

Matthew W. Hoffman et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pp. 3053–3062, 2018. URL https://proceedings.mlr.press/v80/liang18b.html.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Ling Team, Anqi Shen, Baihui Li, Bin Hu, Bin Jing, Cai Chen, Chao Huang, Chao Zhang, Chaokun Yang, Cheng Lin, et al. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model. *arXiv preprint arXiv:2510.18855*, 2025.

Han Lu, Zichen Liu, Shaopan Xiong, Yancheng He, Wei Gao, Yanan Wu, Weixun Wang, Jiashun Liu, Yang Li, Haizhou Zhao, et al. Part ii: Roll flash–accelerating rlvr and agentic training with asynchrony. *arXiv preprint arXiv:2510.11345*, 2025.

Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu Hoang Trinh, Quoc V Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35418–35442, 2025. doi: 10.18653/v1/2025.emnlp-main.1794. URL https://aclanthology.org/2025.emnlp-main.1794/.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Reiichiro Nakano et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron Courville. Asynchronous rlhf: Faster and more efficient off-policy rl for language models. *arXiv preprint arXiv:2410.18252*, 2024.

OpenAI. Learning to reason with llms. *OpenAI technical blog*, 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL https://arxiv.org/abs/2508.10925.

Long Ouyang et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Fei-Fei Li, Yejin Choi, and Manling Li. VAGEN: Reinforcing world model reasoning for multi-turn VLM agents. *arXiv preprint arXiv:2510.16907*, 2025. URL https://arxiv.org/abs/2510.16907.

Yifan Xu, Xiao Liu, Xinghan Liu, Jiaqi Fu, Hanchen Zhang, Bohao Jing, Shudan Zhang, Yuting Wang, Wenyi Zhao, and Yuxiao Dong. MobileRL: Online agentic reinforcement learning for mobile gui agents. *arXiv preprint arXiv:2509.18119*, 2025. URL https://arxiv.org/abs/2509.18119.

Zhongwen Xu and Zihan Ding. Single-stream policy optimization. *arXiv preprint arXiv:2509.13232*, 2025.

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

Shihui Yang, Chengfeng Dou, Peidong Guo, Kai Lu, Qiang Ju, Fei Deng, and Rihui Xin. Dcpo: Dynamic clipping policy optimization. *arXiv preprint arXiv:2509.02333*, 2025b.

Shunyu Yao et al. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Qiying Yu et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 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, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

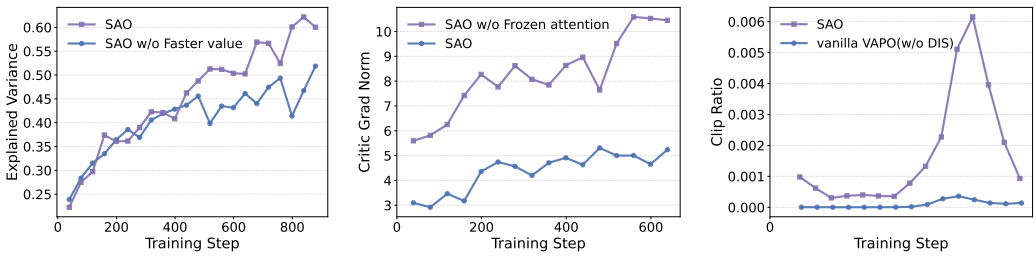
Z.ai. GLM-4.7. Official model card, 2025. URL https://huggingface.co/zai-org/GLM-4.7.

Junzi Zhang, Jongho Kim, Brendan O'Donoghue, and Stephen Boyd. Sample efficient reinforcement learning with reinforce. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12): 10887–10895, 2021.

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

A Additional Experimental Results

A.1 Training Dynamics



(a) (b) (c)

Figure 5: Training dynamics of asynchronous single-rollout RL. (a) Explained Variance for SAO and a single-critic-update baseline. (b) Critic gradient norm during value training under full-parameter optimization and frozen-attention optimization used in SAO. (c) Token-level clip ratio during training for SAO with the proposed DIS and the VAPO baseline.

We analyze the training dynamics of SAO to understand how it facilitates training stability.

Effects of Faster Value Update. Figure 5(a) illustrates Explained Variance comparison between SAO and the single-critic-update baseline during training. Explained Variance assesses the alignment between the predicted values $V(s)$ and the ground-truth returns R, defined as $EV = 1 - \frac{\mathrm{Var}(R - V(s))}{\mathrm{Var}(R)}$. SAO demonstrates significantly higher explained variance after approximately 400 training steps, indicating faster value convergence and better alignment with policy distribution.

Gradient of Critic Models. We examine the impact of freezing attention parameters in value training. As shown in Figure 5(b), full-parameter value training exhibits significantly larger critic gradient norms, suggesting unstable optimization dynamics. In contrast, the frozen-attention strategy maintains lower and smoother gradient norms, implying improved numerical stability.

Clipped Tokens. Figure 5(c) monitors the token-level clip ratio of SAO applying our proposed DIS strategy and the standard VAPO baseline without it. While VAPO maintains a near-zero clip ratio, it fails to effectively gate divergent off-policy updates, leading to a rapid training collapse at approximately 90 steps.

A.2 RL WITH AGENTIC STEP AS ACTION

Relation to VAGEN. VAGEN-Base (Wang et al., 2025) uses the same observation-skipping action-token recursion described in the main text. VAGEN-Full extends this to Bi-Level GAE: it first computes turn-level targets and then propagates them within each turn. SAO instead uses a single token-level recursion with length-adaptive λ (Yue et al., 2025). Our ablation below examines credit-assignment granularity through two step-wise variants, assigning a shared advantage to all tokens within a turn, rather than reproducing VAGEN-Full's hierarchical estimator. Both variants underperform token-level estimation in our setting (Table 6).

To mitigate the high variance inherent in token-level value predictions, we implement a step-wise GAE calculation. We define a step (denoted as S_i) as a single conversation turn and assume that all constituent tokens share a uniform learning signal within each step. We define the step-level value $V(S_i)$ using two primary aggregation methods based on constituent token values predictions $\{v_{i,1}, v_{i,2}, \ldots, v_{i,n}\}$:

- *Step Average.* The step value is the average of all token value predictions within that step, $V(S_i) = \frac{1}{n}\sum_{j=1}^{n} v_{i,j}$. In this setting, the value model is trained on all tokens.
- *Last-Token Prediction.* We define the step value as $V(S_i) = v_{i,n}$, using only the final token of each step, assuming that the final token provides the most accurate value prediction of the step, as it encapsulates the most comprehensive information of the entire unit. During value model training, we apply a loss mask to all intermediate tokens, ensuring that only the last token of each step contributes to the optimization.

To provide a more stable learning signal, we implement a step-wise GAE calculation that shifts the advantage estimation from a token-level to a step-level granularity. We derive a single advantage $\hat{A}_i$ for each step based on the step-level TD error $\delta_i = R_i + \gamma V(S_{i+1}) - V(S_i)$. This advantage is then assigned uniformly to all tokens within the corresponding step, effectively smoothing out the local noise prevalent in auto-regressive generation. Furthermore, the length-adaptive GAE mechanism is modified to scale the decay factor λ based on the total number of steps rather than the raw token length, where

$$\lambda_{\text{policy}} = 1 - \frac{1}{\alpha * \text{step number}}$$

However, as illustrated by the training reward in Figure 6 and the performance results in Table 6, both step-wise approaches underperform token-wise value training. We attribute this failure to the fact that token-level training provides a finer-grained supervision signal for both the critic and the policy, which is essential for accurately capturing the logical transitions within complex reasoning trajectories.

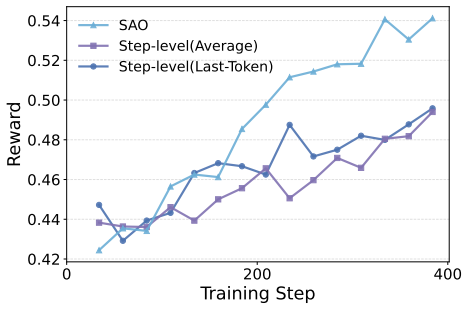


Figure 6: Training reward for token-level SAO training and step-level variants, where token-level shows better training rewards.

A.3 COMPARISON TO OTHER BASELINES OF SINGLE-ROLLOUT STRATEGIES

SPO (Xu & Ding, 2025) or directly using the historical running-mean reward as the baseline for advantage estimation are also feasible ways to achieve RL with a single rollout per prompt. However,

Table 6: The ablation on the action granularity for value and policy model training. *Step-level* denotes that each agent *step* is viewed as an action to calculate the value. *Token-level* refers to each token being viewed as an action. We report the results with the same training steps (400 steps).

	AIME2025	BeyondAIME
Step-level (Average)	85.8	60.5
Step-level (Last-Token)	87.3	62.8
Token-level	89.8	66.8

SPO and running-mean baselines rely on prior information about training-data difficulty and achieve worse performance than SAO, as shown in the experiment section.

B Asynchronous Baseline Objective

For a prompt q, GRPO (Shao et al., 2024) samples G responses with rewards $R_1, \ldots, R_G$ and forms the group-relative advantage

$$\hat{A}_{i,t} = \frac{R_i - \mu_R}{\sigma_R}, \qquad \mu_R = \frac{1}{G}\sum_{j=1}^{G} R_j, \qquad \sigma_R = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(R_j - \mu_R)^2 + \epsilon_{\text{stab}}}. \tag{6}$$

Our asynchronous GRPO baseline uses DAPO's clip-higher and token-level loss (Yu et al., 2025). For a batch of responses $\mathcal{B}$, its policy objective is

$$J_{\text{GRPO}}(\theta) = \frac{1}{\sum_{i\in\mathcal{B}}|y_i|}\sum_{i\in\mathcal{B}}\sum_{t=1}^{|y_i|}\min\Big(u_{i,t}\hat{A}_{i,t}, \text{clip}(u_{i,t}, 1-\epsilon_l, 1+\epsilon_h)\hat{A}_{i,t}\Big), \tag{7}$$

where sums and lengths count generated action tokens, and $u_{i,t} = \pi_\theta(a_{i,t} \mid s_{i,t})/\pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t})$ uses training-engine probabilities from the current policy and the latest old-policy snapshot. GRPO (w/ DIS) retains the group-relative advantage and instead uses the direct current-to-rollout ratio and detached masking objective of DIS. SAO (w/ DIS only) uses the same direct correction with a learned value model and one rollout per prompt.

C Relation to IcePop

IcePop (Ling Team et al., 2025) distinguishes the current training policy π_θ, an old training-engine policy $\pi_{\theta_{\text{old}}}$, and the inference-engine rollout policy π_{rollout}. For a rollout sampled from the old checkpoint, it separates the same-checkpoint training–inference discrepancy from policy staleness:

$$\rho_t = \frac{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}{\pi_{\text{rollout}}(a_t \mid s_t)}, \qquad u_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}. \tag{8}$$

IcePop calibrates the engine ratio ρ_t with a double-sided mask and retains a PPO clipped surrogate for the policy-update ratio u_t. DIS instead applies its detached double-sided mask directly to the combined ratio

$$r_t(\theta) = u_t(\theta)\rho_t = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{rollout}}(a_t \mid s_t)}. \tag{9}$$

Thus, DIS selects tokens according to their overall current-to-behavior discrepancy rather than separately thresholding engine mismatch and clipping policy drift. Recording the actual per-token rollout probability also avoids reconstructing the old training-engine policy when asynchronous trajectories contain tokens generated under different policy versions.

D System Efficiency

DIS overhead. DIS stores one scalar behavior log-probability per generated token, obtained from logits already computed during rollout. The current-policy log-probability is computed in the

standard training forward pass, so DIS requires no additional model forward pass. For a trajectory of T generated tokens, storing behavior log-probabilities requires $2T$ bytes in FP16/BF16 or $4T$ bytes in FP32, excluding serialization overhead. The additional work is limited to extracting and transferring these scalars and applying the token-level mask.

We compare synchronous and asynchronous RL on math tasks with Python tools using Qwen3-30B-A3B and 64 NVIDIA H100 80GB GPUs. Both configurations use a rollout batch size of 512 and a training batch size of 128. Policy weights are synchronized to the rollout engine once every four policy updates. The synchronous pipeline uses 64 GPUs, while the asynchronous pipeline allocates 32 GPUs to training and 32 to rollout generation, overlapping the two stages.

Pipeline	GPU allocation	Total GPUs	Time / training step
Synchronous	64	64	304 s
Asynchronous	32 training + 32 rollout	64	184 s

Table 7: Wall-clock comparison under the same total GPU budget and batch sizes. These measurements characterize asynchronous versus synchronous execution; SAO's algorithmic comparisons study stability and task effectiveness within the asynchronous pipeline.

The asynchronous pipeline achieves a $1.65\times$ speedup, reducing time per training step by 39.5%. Overlapping rollout generation and optimization reduces idle waiting associated with long-tail agentic trajectories, providing an efficient execution setting for SAO.