SPPO: Sequence-Level PPO for Long-Horizon Reasoning Tasks

Tianyi Wang[1,3*], **Yixia Li**[1*], **Long Li**[2], **Yibiao Chen**[3],
Shaohan Huang[4], **Yun Chen**[5], **Peng Li**[6], **Yang Liu**[6], **Guanhua Chen**[1†]
[1]Southern University of Science and Technology, [2]INFLY TECH
[3]Beijing University of Posts and Telecommunications, [4]Microsoft Research Asia
[5]Shanghai University of Finance and Economics, [6]Tsinghua University

bimu@bupt.edu.cn

Abstract

Proximal Policy Optimization (PPO) was central to aligning Large Language Models (LLMs) in reasoning tasks with verifiable rewards. However, standard token-level PPO struggles in this setting due to the instability of temporal credit assignment over long Chain-of-Thought (CoT) horizons and the prohibitive memory cost of the value model. While critic-free alternatives like GRPO mitigate these issues, they incur significant computational overhead by requiring multiple samples for baseline estimation, severely limiting training throughput. In this paper, we introduce **Sequence-Level PPO (SPPO)**, a scalable algorithm that harmonizes the sample efficiency of PPO with the stability of outcome-based updates. SPPO reformulates the reasoning process as a **Sequence-Level Contextual Bandit** problem, employing a decoupled scalar value function to derive low-variance advantage signals without multi-sampling. Extensive experiments on mathematical benchmarks demonstrate that SPPO significantly surpasses standard PPO and matches the performance of computation-heavy group-based methods, offering a resource-efficient framework for aligning reasoning LLMs.

1 Introduction

Large Language Models (LLMs) (Yang et al., 2026) and Multimodal Large Language Models (MLLMs) have significantly advanced in complex reasoning (Li et al., 2025b; Zhu et al., 2025; Li et al., 2024; Wang et al., 2026a; Ruan et al., 2025b), empowered by long Chain-of-Thought (CoT) prompting (Lightman et al., 2023). To further align these models with logical correctness, Reinforcement Learning (RL) has proven indispensable (Ma et al., 2026; Wang et al., 2026b), particularly in Reinforcement Learning with Verifiable Rewards (RLVR) tasks like mathematical problem-solving (Luo et al., 2025; Zhang et al., 2026; Ruan et al., 2025a; Zhu et al., 2026), SQL tasks (Li et al., 2026b; Yang et al., 2025) and Agent ability (Li et al., 2026d; He et al., 2025; Fang et al., 2025; Li et al., 2026e). Proximal Policy Optimization (PPO, Schulman et al. (2017)) typically relies on a token-level Critic and Generalized Advantage Estimation (GAE) for credit assignment (Guo et al., 2025). However, this framework faces structural incompatibility in long CoT tasks with sparse rewards. The delayed reward forces GAE to propagate signals across thousands of tokens, inducing high bias (Yuan et al., 2025). Furthermore, the Critic tends to "overfit" semantic cues at the sequence tail (See Figure 1), causing the advantage signal to vanish precisely when needed. Consequently, standard PPO often proves unstable for reasoning tasks (Kazemnejad et al., 2025).

In response, Group Relative Policy Optimization (GRPO, Shao et al. (2024)) eliminates the learned Critic in favor of group-based statistical baselines. We posit that GRPO's success stems from implicitly remodeling reasoning as a **Sequence-Level Contextual Bandit** problem, treating the entire response as an atomic action to bypass token-level noise. However, this approach faces a fundamental **Bias-Variance Trade-off**: while it removes the high bias inherent in token-level value estimation, the reliance on Monte Carlo outcomes introduces **high variance** in the gradient signal (Wang et al., 2025a). To mitigate this variance and stabilize training, GRPO incurs a prohibitive computational cost, as it necessitates sampling multiple responses (N) per prompt to construct a valid baseline, significantly bottlenecking training throughput (Lin et al., 2025; Li et al., 2025a).

Despite the empirical success of critic-free methods like GRPO, a critical misconception exists regarding their efficacy; our core contribution is understanding GRPO from a novel perspective: its success stems from implicitly remodeling reasoning as a Sequence-Level Contextual

[*]Equal Contribution. [†]Corresponding Author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7831–7853
July 2-7, 2026

Bandit problem—treating the entire response as an atomic action to bypass token-level noise—rather than a multi-step Markov Decision Process (MDP). We posit that a stable, sequence-level baseline—underpinned by a generalizable scalar value model—is structurally more robust for long-horizon RLVR tasks. Crucially, this explicitly modeled approach not only secures optimization stability but also circumvents the prohibitive computational latency associated with extensive group sampling (Fang et al., 2026a).

Drawing on these insights, we introduce **Sequence-Level PPO (SPPO)**, a novel algorithm that resolves the Bias-Variance dilemma in reasoning alignment. SPPO fundamentally reformulates the reasoning process from a token-level Markov Decision Process (MDP) to a **Sequence-Level Contextual Bandit** problem. In this view, the prompt serves as the static context and the *entire reasoning chain* is treated as a single atomic action. This formulation effectively collapses the time horizon, eliminating the **high bias** of token-level credit assignment inherent to standard PPO. Simultaneously, SPPO employs a learned scalar value function to curb the **high variance** of group-relative baselines, thereby achieving optimization stability without the need for multi-sampling ($N > 1$).

Crucially, SPPO addresses computational bottlenecks through this resource-efficient architecture. Unlike GRPO, which requires expensive multi-sampling for empirical baselines to reduce variance, SPPO leverages its learned scalar value function to enable high-throughput single-sample updates ($N = 1$). Furthermore, we validate a **Decoupled Critic** strategy—using a lightweight critic (e.g., 1.5B) to align a larger policy (e.g., 7B)—which leverages the reduced complexity of value estimation to cut memory usage by **12.8%** (Figure 6). Extensive evaluations on AIME24/25, AMC23, MATH, and Minerva demonstrate that SPPO resolves the value collapse of standard PPO and outperforms computation-heavy baselines. Notably, SPPO matches GRPO's peak performance with a **5.9× training speedup** and superior convergence, offering a scalable paradigm for sparse-reward reasoning tasks.[1]

[1]Our code is available at https://github.com/sustech-nlp/SPPO.

2 Background

2.1 PPO and Credit Assignment in Reasoning

PPO optimizes the policy by maximizing a clipped surrogate objective $J_{\text{PPO}}(\theta)$:

$$J_{\text{PPO}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big]$$

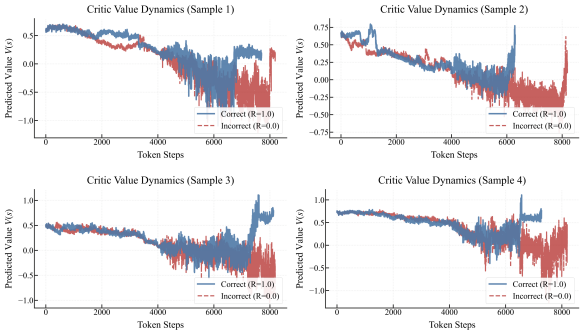

Figure 1: **Analysis of the "Tail Effect".** We visualize Critic value dynamics $V(s_t)$ to diagnose inefficiencies. Blue and red lines denote correct and incorrect trajectories, respectively. The Critic discriminates only near the sequence tail. For correct paths, $V(s_t)$ rises late, causing $\hat{A}_t$ to vanish; for incorrect ones, it fails to penalize intermediate steps. This indicates credit assignment based on token position rather than semantic contribution. The Critic was trained under 8192 context window. Additional randomly sampled visualizations are provided in Appendix B.

where $r_t(\theta)$ is the probability ratio and $\hat{A}_t$ is the advantage. Typically, $\hat{A}_t$ is computed via Generalized Advantage Estimation (GAE) using a learned token-level Critic $V(s_t)$. The TD error is defined as $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, and the advantage is the discounted sum of errors:

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

In sparse-reward reasoning tasks (e.g., CoT), it is standard to set $\gamma = \lambda = 1$ to propagate terminal rewards. Under this setting, GAE simplifies to the difference between the Monte Carlo return G_t and the value estimate:

$$\hat{A}_t^{\text{GAE}} = G_t - V(s_t)$$

However, this mechanism is unstable in long-horizon tasks (Figure 1). As generation approaches the answer, the Critic $V(s_t)$ often "overfits" semantic cues. For correct trajectories, $V(s_t)$ converges to the reward early, causing $\hat{A}_t$ to vanish; for incorrect ones, it underestimates significantly. This "tail effect" bases credit on position rather than contribution, undermining optimization.

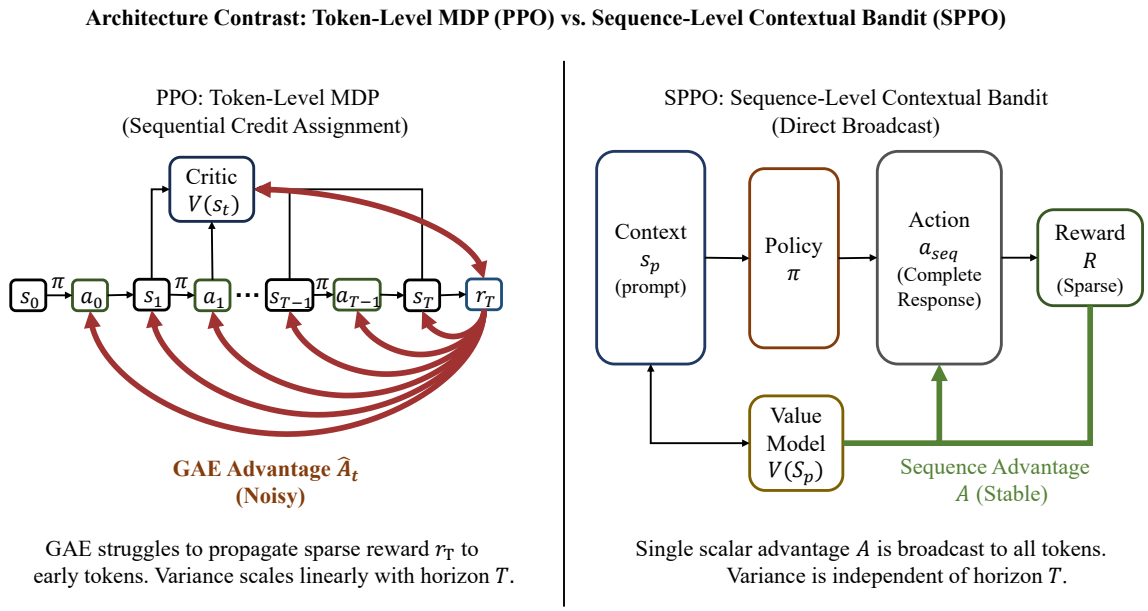

Figure 2: **Overview of SPPO.** Motivated by the implicit bandit behavior of GRPO, SPPO explicitly reformulates reasoning as a Sequence-Level Contextual Bandit, utilizing a scalar value function $V(s_p)$.

2.2 Optimization Mechanics of GRPO

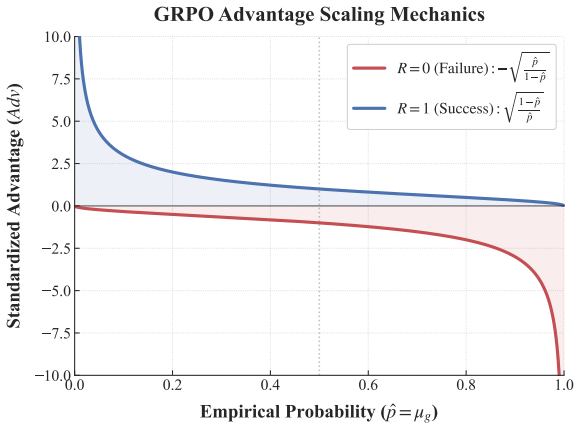

Figure 3: **Visualization of the GRPO Advantage Function.** derived under the Bernoulli assumption (see Appendix A). The plot illustrates how GRPO implicitly models the reasoning task as a **Contextual Bandit**: instead of a static reward, the advantage is dynamically scaled based on the prompt's estimated difficulty $\hat{p}(s_p)$, contrasting success (Blue) against failure (Red).

While standard PPO operates within a token-level MDP, GRPO implicitly shifts the optimization paradigm. It eliminates the step-wise Critic by sampling N outputs per prompt and computing the advantage via group normalization:

$$Adv(s_p, a) = \frac{R - \mu_g}{\sigma_g}.$$

Modeling the sampling process as Bernoulli trials (derivation provided in Appendix A), the advantage function simplifies to:

$$Adv(s_p, a) = \begin{cases} \sqrt{\frac{1-\hat{p}(s_p)}{\hat{p}(s_p)}} & \text{if } R = 1 \\ -\sqrt{\frac{\hat{p}(s_p)}{1-\hat{p}(s_p)}} & \text{if } R = 0 \end{cases}$$

As visualized in Figure 3, GRPO dynamically scales rewards based on prompt difficulty $\hat{p}(s_p)$. Crucially, this mechanism evaluates the *entire response* as an atomic unit against a prompt-dependent baseline. Thus, although GRPO does not explicitly redefine the environment, its advantage formulation implicitly models the reasoning task as a Contextual Bandit rather than a multi-step MDP.

This observation prompts a critical question: *Does GRPO's success imply that PPO's instability stems from the token-level MDP decomposition, rather than the fundamental intractability of value estimation?*

3 Method

3.1 Formulation: Sequence-Level Contextual Bandit

To answer this and formalize the implicit insight, we explicitly reformulate the reasoning process from a token-level MDP to a **Sequence-Level Contextual Bandit (SL-CB)**. By conceptually collapsing the time horizon ($H = 1$), we map the reasoning task to the tuple $(\mathcal{S}, \mathcal{A}, r)$, where the

context $\mathcal{S}$ is defined strictly by the static prompt s_p, and the action $\mathcal{A}$ treats the *entire response sequence* $a_{seq} = (y_1, \ldots, y_T)$ as a single atomic unit. Accordingly, the reward $r(s_p, a_{seq})$ evaluates the **holistic correctness** of the generated chain.

This formulation fundamentally circumvents the credit assignment ambiguity inherent to MDPs. Rather than forcing a Critic to decompose sparse outcomes into noisy token-level signals, we optimize the expected sequence reward conditioned strictly on the prompt. Consequently, the value function $V(s_p)$ simplifies to estimating the scalar **solvability** of the problem, aligning directly with the objective of sparse-reward reasoning.

3.2 SPPO: Sequence-Level Proximal Policy Optimization

Building upon the theoretical insights of the sequence-level bandit formulation, we propose **SPPO**, an algorithm designed to strictly align the optimization objective with the sparse, outcome-oriented nature of reasoning tasks.

Value Function and Advantage Estimation First, we redefine the role of the critic. Unlike the token-level value function in standard PPO, which attempts to predict future returns from arbitrary intermediate states, we train a value model $V_\phi(s_p)$ to estimate the **scalar probability of success** for a given prompt s_p.

To construct the advantage, we treat the single sample outcome as a realization of a Bernoulli trial with probability $V_\phi(s_p)$. We adopt a standardized advantage formulation to stabilize training:

$$A(s_p, a) = R - V_\phi(s_p) \tag{1}$$

where $R \in \{0, 1\}$ is the binary reward. This formulation naturally amplifies the signal when the model is confident but wrong, and suppresses noise when the model is uncertain ($V \approx 0.5$).

To ensure $V_\phi(s_p)$ serves as a calibrated baseline, we optimize it using the Binary Cross-Entropy (BCE) loss:

$$L_V(\phi) = -\mathbb{E}\Big[R \log V_\phi(s_p) + (1 - R) \log(1 - V_\phi(s_p))\Big] \tag{2}$$

Sequence-Level Policy Optimization With the advantage established, we formulate the policy optimization objective. SPPO adapts the clipped surrogate objective of PPO but fundamentally alters the scope of the advantage term. The objective function is defined as:

$$J_{\text{SPPO}}(\theta) = \mathbb{E}_{s_p \sim \mathcal{D},\, a \sim \pi_{\theta_k},\, t \in a}\Big[\min\big(r_t(\theta)A(s_p, a), \text{clip}\big(r_t(\theta), 1 - \epsilon, 1 + \epsilon\big)A(s_p, a)\big)\Big] \tag{3}$$

Here, $r_t(\theta) = \frac{\pi_\theta(a_t|s_p, a_{<t})}{\pi_{\theta_k}(a_t|s_p, a_{<t})}$ represents the probability ratio between the current policy π_θ and the behavior policy π_{θ_k} for the token at timestep t, and ϵ is the standard clipping hyperparameter to constrain the policy update.

Crucially, unlike standard PPO where each token t is assigned a unique, time-dependent advantage $\hat{A}_t$ via GAE, SPPO propagates the single sequence-level advantage $A(s_p, a)$ uniformly to all constituent tokens t in the sequence a. This mechanism ensures that if a reasoning chain leads to a correct answer ($A > 0$), every step in that chain is reinforced equally; conversely, if the chain fails ($A < 0$), every step is penalized. By decoupling the advantage signal from the sequence length, SPPO effectively solves the temporal credit assignment problem that hinders standard PPO in Long Chain-of-Thought tasks.

4 Experiments

4.1 Experimental Setup

We evaluate **DeepSeek-R1-Distill-Qwen-1.5B** and **DeepSeek-R1-Distill-Qwen-7B** models, fine-tuned on DeepScaleR (Luo et al., 2025) and DAPO-17K (Yu et al., 2025) respectively. We evaluate performance using Average@16 accuracy across five held-out benchmarks: **AIME24**(Art of Problem Solving, 2025a), **AIME25**(Art of Problem Solving, 2025a), **AMC23**(Art of Problem Solving, 2025b), **MATH500**(Hendrycks et al., 2021), and **Minerva Math**(Lewkowycz et al., 2022). Comprehensive links to all models and datasets are provided in Appendix D.

Baselines & Implementation We benchmark SPPO against: (1) **Base Model**; (2) **Standard PPO** (token-level); and (3) Sequence-level methods including **ReMax**, **RLOO**, and **GRPO** ($N = 8$). All algorithms are implemented via `verl` (Sheng et al., 2025) using outcome-based rewards (+1 for correct boxed answers and 0 for the incorrect answers). We utilized the precise reward function implemented in `Reasoning360` (Cheng et al., 2025) to conduct both training and evaluation. **Hyperparameters:** We set $\beta_{KL} = 0$ to encourage explo-

ration. Global batch sizes are configured at 256 for the 1.5B model and 512 for the 7B model. Learning rates are set to 1e-6 for Actors and 5e-6 for Critics. Standard PPO employs $\gamma = 1, \lambda = 1$ to propagate sparse rewards. The hyperparameters for all baselines generally follow the official recommended examples provided by the `verl` library. All experiments were conducted on $4 \times$ A100 (1.5B) and $4 \times$ H100 (7B) GPUs. For complete reproducibility, we provide the exact execution scripts and configuration commands for both SPPO and the baselines in Appendix E.

4.2 Main Results

Table 1 presents the performance comparison. Standard PPO struggles to consistently improve over the base model, confirming the instability of GAE in sparse-reward settings. While sequence-level baselines (ReMax, RLOO) improve stability, they generally lag behind group-based approaches.

SPPO achieves the highest overall performance, surpassing GRPO ($N = 8$) on most benchmarks (Avg 48.06 vs. 47.08 on 1.5B). Crucially, SPPO achieves this with single-sample efficiency ($N = 1$), effectively eliminating the "Tail Effect" (Figure 1) without the computational bottleneck of multi-sampling.

Method	A24	A25	AMC	MATH	Minerva	Avg
DeepSeek-R1-Distill-Qwen-1.5B						
Base Model	27.50	21.67	71.56	83.73	20.35	44.96
PPO	27.50	20.83	70.63	81.38	19.89	44.06
ReMax	31.67	25.42	71.88	**84.38**	20.40	46.74
RLOO	30.42	21.67	72.81	84.10	21.73	46.15
GRPO ($N = 8$)	30.00	**26.25**	73.13	83.88	**22.15**	47.08
SPPO (Ours)	**34.17**	25.83	**74.38**	83.78	**22.15**	**48.06**
DeepSeek-R1-Distill-Qwen-7B						
Base Model	41.25	26.67	79.38	87.20	27.94	52.49
PPO	45.20	**35.42**	85.31	88.48	27.80	56.44
ReMax	49.38	31.25	86.56	90.28	27.99	57.09
RLOO	46.67	32.50	86.88	**90.35**	28.72	57.02
GRPO ($N = 8$)	47.08	35.00	86.25	90.15	28.74	57.44
SPPO (Ours)	50.83	35.00	86.25	90.13	28.35	58.11
w/ Small Critic	**52.29**	34.58	**87.19**	89.88	**28.86**	**58.56**

Table 1: Performance comparison on 1.5B and 7B scales. SPPO consistently outperforms baselines. The *Small Critic* variant (1.5B Critic aligning 7B Policy) achieves the top average score.

Critic Decoupling We hypothesize that scalar solvability estimation is significantly simpler than generative reasoning, permitting a smaller Value Function. To test this, we trained the 7B Policy using a **1.5B Critic**. As shown in Table 1 (*w/ Small Critic*), this configuration not only retains effectiveness but achieves the highest average score (58.56). This validates that a lightweight critic can effectively align a large policy, significantly reducing the memory footprint of RLVR training (Figure 6).

4.3 Ablation Study: Impact of Loss Function

To isolate the source of SPPO's performance gains, we ablated the architectural contribution by applying the BCE loss to the standard token-level PPO framework (**PPO + BCE**).

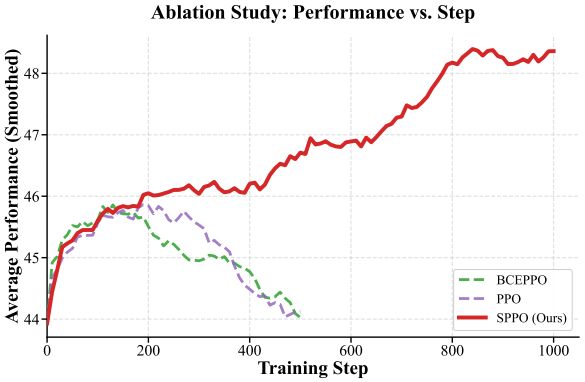

Figure 4: **Ablation Analysis of the Optimization Objective.** We compare SPPO against Standard PPO and a control baseline ("PPO + BCE") that integrates the BCE loss into the token-level framework. The failure of the control baseline demonstrates that the performance gains do not stem from the loss function itself, but from the **Sequence-Level Contextual Bandit formulation**, which propagates a unified advantage signal to resolve credit assignment ambiguity.

As shown in Figure 4, **PPO + BCE** fails to reproduce the success of SPPO, exhibiting the same instability as the standard baseline. Notably, we terminated both PPO-based runs early at 500 steps due to observed performance collapse and degrading scores. This empirical evidence validates that SPPO's efficacy derives fundamentally from its **Sequence-Level Contextual Bandit formulation**—specifically the propagation of a unified advantage signal $A = R - V(s_p)$—rather than the adoption of the BCE loss in isolation.

5 Analysis

5.1 Scalability and Computational Efficiency

To evaluate the scalability of SPPO and its efficiency in larger parameter regimes, we extended our experiments to the **DeepSeek-R1-Distill-Qwen-7B** model. For this analysis, we utilized the DAPO-17K dataset (Luo et al., 2025), a curated subset of high-quality mathematical reasoning problems. We compared SPPO against the baseline algorithms, tracking performance on the

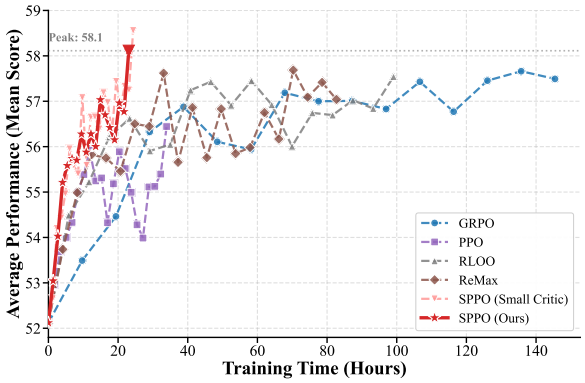

Figure 5: **Training Efficiency on Deepseek-R1-Distill-Qwen-7B (Performance vs. Wall-clock Time).** The plot compares the trajectory of SPPO against strong baselines (GRPO, PPO, RLOO, ReMax) on the DAPO-17k dataset(Yu et al., 2025). **Solid Red:** SPPO with a matched 7B Critic. **Dashed Pink:** SPPO with a decoupled, smaller 1.5B Critic (Deepseek-R1-Distill-Qwen-1.5B). The y-axis noted as the Avg@8 score evaluated on **AIME24**, **AIME25**, **AMC23**, **MATH500**, and **Minerva Math**. SPPO achieves optimal performance significantly faster than group-based methods, and the decoupled critic maintains performance while reducing memory overhead.

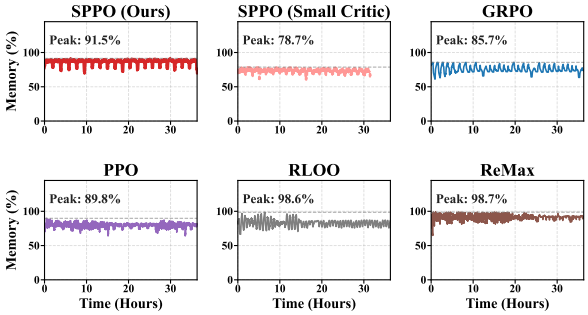

Figure 6: **GPU Memory Allocation Analysis.** Comparison of normalized peak VRAM usage during the training of a 7B policy. The "Decoupled Critic" (7B+1.5B) approach, combined with the system-level optimizations in `verl`, significantly reduces memory bottlenecks compared to symmetric actor-critic setups (7B+7B), making efficient RL alignment accessible on consumer-grade hardware.

hold-out validation set against wall-clock training time.

Training Efficiency As illustrated in Figure 5, SPPO demonstrates superior training efficiency compared to group-based alternatives. GRPO ($N = 8$) and RLOO exhibit a slower "time-to-convergence" primarily due to the computational bottleneck of generating multiple samples per prompt to estimate the baseline. In contrast, SPPO, which operates with single-sample efficiency ($N = 1$), updates the policy more frequently within the same time window. Consequently, SPPO reaches peak performance (mean score ≈ 58) in approximately 22 hours, whereas baselines require significantly longer to reach comparable levels or plateau at lower scores (e.g., standard PPO).

Resource Efficiency and VRAM Optimization Beyond computational throughput, the "Small Critic" configuration offers a decisive advantage in hardware accessibility. Standard RLHF typically requires loading a Critic of equal size to the Policy, effectively doubling the parameter memory footprint. By decoupling the critic size (1.5B) from the policy (7B), SPPO significantly alleviates this bottleneck. Furthermore, by leveraging the advanced memory management and sharding techniques provided by the `verl` library (Sheng et al., 2025), our implementation minimizes redundant memory allocation.

As visualized in Figure 6, the SPPO framework with a decoupled critic maintains a low memory profile. This confirms that SPPO is not only algorithmically stable but also resource-efficient, enabling the alignment of large reasoning models even under constrained GPU budgets.

5.2 Value Model Analysis: Calibration and Correlation

The efficacy of SPPO relies heavily on the quality of the sequence-level value function, $V(s_p)$. In our framework, $V(s_p)$ serves as the baseline for advantage estimation ($A = R - V_\phi(s_p)$). Theoretically, an ideal value model should accurately capture the intrinsic difficulty of a prompt, approximating the expected success rate of the current policy. To validate this, we conducted a correlation analysis between the critic's predictions and the empirical ground truth on a held-out validation set.

Setup We randomly sampled a diverse set of $N = 200$ prompts and executed the policy multiple times for each to compute the empirical pass rate (AVG@k), which serves as the ground truth label for difficulty. We then compared these empirical values against the predicted probabilities output by our Value Model.

Calibration and Distribution Analysis As illustrated in Figure 7, our analysis reveals a distinct positive linear correlation between the predicted values and the empirical results. With a Pearson correlation coefficient of $\mathbf{0.642}$ and a Spearman rank correlation of $\mathbf{0.664}$, we find strong statistical evidence that the Value Model has successfully learned to capture the relative difficulty of prompts, effectively cutting through the inherent stochasticity of LLM generation. Beyond these

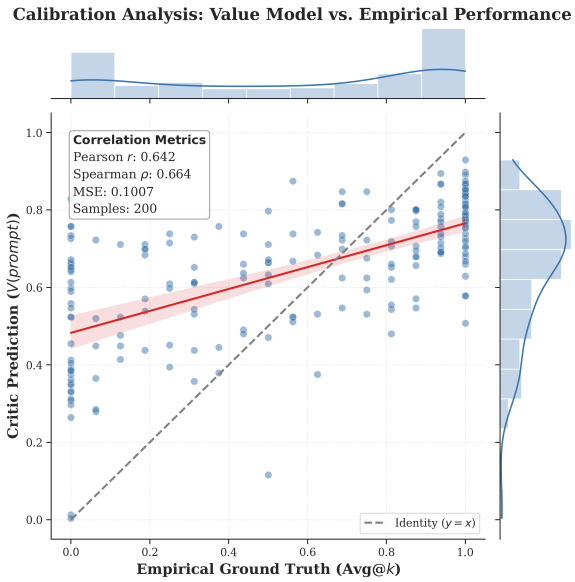

Figure 7: Correlation analysis between the Critic's predicted difficulty (y-axis) and the empirical AVG@k rate (x-axis) (Where k = 64). The plot reveals a clear positive correlation (Pearson $r = 0.642$), indicating the Critic successfully distinguishes between hard and easy tasks. The marginal histograms (top and right) contrast the bimodal distribution of real task difficulty with the more conservative, quasi-normal distribution of the Critic's predictions.

correlation metrics, the marginal histograms provide a critical insight into the model's behavioral tendencies: while the empirical difficulty (top histogram) follows a **bimodal distribution**—where tasks are typically either completely unsolvable (0.0) or consistently solvable (1.0)—the critic's predictions (right histogram) exhibit a **unimodal, quasi-normal distribution** centered around 0.6-0.7. This distributional shift indicates that the critic adopts a conservative prediction strategy, aggregating uncertainty rather than overfitting to the binary extremes of the empirical data.

Implications for SPPO The distributional discrepancy suggests that the Critic tends to be *conservative*, exhibiting a "regression to the mean" behavior rather than predicting extreme probabilities (0 or 1). However, the regression trend (red line) maintains a clear positive slope. This confirms that $V(s_p)$ serves as a valid, variance-reducing baseline.

Specifically, for hard prompts (Avg@$k \approx 0$), the Critic predicts lower values (≈ 0.5), ensuring that a rare success yields a strong positive advantage. For easy prompts (Avg@$k \approx 1$), the Critic predicts higher values (≈ 0.8), ensuring that a failure yields a significant negative penalty.

5.3 Controlled Analysis: The RLVR Benchmark

To strictly disentangle the algorithmic efficacy of SPPO from system-level optimizations inherent to the `verl` framework, and to rigorously validate its robustness in isolation, we extend our evaluation to a suite of **five** representative control environments: *Precision CartPole*, *MountainCar*, *Hopper* (MuJoCo), *LunarLander*, and *Pendulum*. We reconfigure these classic control tasks into a **Reinforcement Learning with Verifiable Rewards (RLVR)** framework. By strictly enforcing structural constraints—specifically long time horizons, deterministic transitions, and sparse binary outcome feedback—we construct a minimalist testbed that mimics the optimization landscape of LLM reasoning without the confounding variables of large-scale distributed training.

Experimental Protocol To mirror the LLM alignment lifecycle, we implement a rigorous three-stage pipeline. First, **Expert Synthesis** trains policies using dense, shaped rewards (e.g., velocity bonuses in *MountainCar*, upright incentives in *Pendulum*) to mimic pre-training supervision. Subsequently, **Supervised Fine-Tuning (SFT)** applies behavior cloning to filtered successful trajectories (e.g., $r > 0.5$ or state $x > 1.0$ in *Hopper*), initializing a model with non-zero but imperfect solvability. Finally, **RL Fine-tuning** introduces the core sparse reward challenge: agents receive a strictly binary terminal reward $r_H \in \{0, 1\}$ with zero intermediate feedback ($r_t = 0$ for $t < H$) and a discount factor $\gamma = 1.0$, compelling the algorithm to bridge the full temporal horizon.

Task Configurations and Hyperparameters To ensure a fair comparison of optimization objectives, we align core PPO hyperparameters (e.g., $\epsilon = 0.2$, learning rates) across algorithms while adapting batch sizes to the distinct exploration dynamics of each domain. Specifically, we utilize a strictly aligned batch size of 16 trajectories per update for *LunarLander* and *Pendulum*; a reduced batch size of 8 for exploration-heavy tasks (*MountainCar*, *Hopper*); and a batch size of 64 for the rapid-dynamics *CartPole*. Success in the RL phase is determined by rigorous outcome criteria: *Precision CartPole* ($H = 200$) requires a final angle $|\theta| \leq 0.5°$; *MountainCar* ($H = 1000$) requires reaching the flag ($x \geq 0.45$); *Hopper* ($H = 1000$) demands survival with forward progress ($x > 1.0$

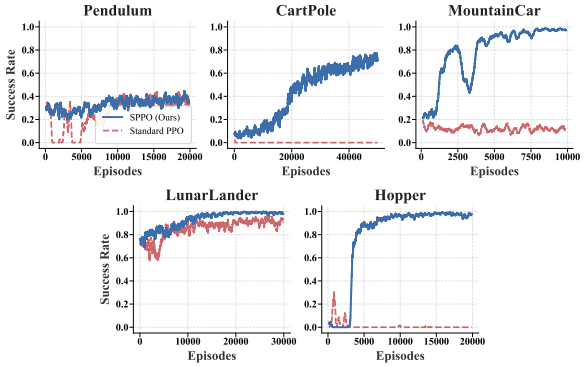

Figure 8: **RLVR Benchmark Results.** Comparison of SPPO (Blue Solid) and Standard PPO (Red Dashed) across five control tasks with sparse outcome rewards ($\gamma = 1.0$). SPPO demonstrates robust convergence in complex control tasks where Standard PPO exhibits instability or failure (e.g., *Hopper*, *MountainCar*), while achieving superior sample efficiency in precision tasks like *CartPole*.

m); *LunarLander* ($H = 1000$) necessitates stable leg contact (> 0.5) within the landing pad ($|x| < 0.4$); and *Pendulum* ($H = 1000$) requires an upright final position ($\cos\theta > 0.8$).

Results and Analysis As illustrated in Figure 8, SPPO consistently matches or outperforms Standard PPO across all domains, confirming its structural superiority in sparse-reward settings. In long-horizon tasks like *Hopper* ($H = 1000$) and *MountainCar*, where the SFT initialization provides a weak prior, Standard PPO flatlines near zero (or stays at a low success rate) as the token-level critic $V(s_t)$ fails to propagate the sparse signal effectively; conversely, SPPO successfully solves these tasks by estimating the sequence-level solvability $V(s_0)$. Furthermore, in *LunarLander*, SPPO maintains monotonic improvement, avoiding the instability observed in the Standard PPO baseline. Finally, SPPO demonstrates superior precision alignment in *Precision CartPole*, rapidly converging to high-precision behaviors where step-level attribution struggles to differentiate between "good" and "perfect" trajectories given binary feedback.

6 Related Work

6.1 Reinforcement Learning Algorithms in LLMs

Proximal Policy Optimization (PPO) (Schulman et al., 2017) aligns LLMs using a dense, token-level value function. However, in sparse-reward reasoning tasks (RLVR), this approach struggles as GAE fails to effectively assign credit over long Chain-of-Thought horizons (Lightman et al., 2023). Group-based methods like GRPO (Shao et al., 2024) mitigate this by estimating baselines via multi-sampling ($N > 1$). By normalizing rewards against the group mean, GRPO implicitly adopts a sequence-level objective, bypassing temporal credit assignment issues.

While recent variants like DAPO (Luo et al., 2025), DPH-RL (Li et al., 2026a), DyJR (Li et al., 2026c), APO (Wang et al., 2026c), Dr.GRPO (Liu et al., 2025) and (Fang et al., 2026b) propose strategies to refine gradient dynamics (e.g., dynamic sampling), they remain fundamentally bound to the computationally expensive multi-sampling paradigm. **In this work, we exclude such orthogonal optimizations.** Our primary objective is to isolate and validate the effectiveness of the **Sequence-Level Contextual Bandit formulation** itself, rather than remedying the inherent instabilities of group-relative baselines. Consequently, SPPO replaces the empirical baseline with a learned scalar value function $V(s_p)$. This enables stable, on-policy learning with single-sample efficiency ($N = 1$), harmonizing PPO's throughput with the structural stability of sequence-level modeling.

6.2 Sequence-Level Exploration

Prior research has extensively explored sequence-level Reinforcement Learning (RL) algorithms. The RLOO (REINFORCE Leave-One-Out) algorithm (Ahmadian et al., 2024) posits that token-level modeling is often superfluous, criticizing the token-level optimization inherent in PPO. However, RLOO is established upon the REINFORCE (Sutton et al., 1999) algorithm. Recent work (Wang et al., 2025b) has shown that the clipping term plays a crucial role in learning stability through a Token Masking mechanism. Furthermore, RLOO's rolling mechanism increases computational requirements as CoT trajectories lengthen.

Moreover, GSPO (Group Sequence Policy Optimization, Zheng et al. (2025)) and GMPO (Geometric-Mean Policy Optimization, Zhao et al. (2025)) have argued that a sequence-level reward is incongruent with PPO's token-level design. However, since these methods explicitly position their core contribution as addressing the routing instability inherent to Mixture-of-Experts (MoE) architectures, we exclude them from our baselines to maintain a focus on general reasoning alignment.

7 Conclusion

To resolve the trade-off between standard PPO's high-bias credit assignment and GRPO's high-variance inefficiency, we introduce SPPO, which reformulates reasoning as a Sequence-Level Contextual Bandit. By employing a scalar critic for advantage estimation, SPPO secures optimization stability with high-throughput single-sample efficiency, offering a scalable paradigm for sparse-reward tasks.

Acknowledgements

This project was supported by National Natural Science Foundation of China (No. 62306132), Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515011564), Natural Science Foundation of Shanghai (No. 25ZR1402136). We thank the anonymous reviewers for their insightful feedback on this work. This work was done during Tianyi's internship at SUSTech.

Limitations

In this work, we primarily focus on RLVR, showing that SPPO effectively harmonizes sample efficiency with structural stability in sparse-reward settings. However, our approach is explicitly tailored for tasks with verifiable outcomes to estimate prompt solvability. As a result, extending this sequence-level bandit formulation to open-ended generation tasks, which lack objective ground-truth verifiers, remains a direction for future research.

Ethical Considerations

Our study is conducted in controlled, text-only benchmark environments and does not involve human subjects or the collection of personal data. As with other agentic and world-modeling capabilities, misuse (e.g., enabling harmful or deceptive behavior) and bias propagation are possible; we encourage responsible deployment with appropriate safeguards and oversight.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *Preprint*, arXiv:2402.14740.

Art of Problem Solving. 2025a. Aime problems and solutions. Accessed: 2026-01-04.

Art of Problem Solving. 2025b. Amc problems and solutions. Accessed: 2026-01-04.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Haonan Li, Taylor W. Killian, Mikhail Yurochkin, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. 2025. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. *Preprint*, arXiv:2506.14965.

Tianqing Fang, Zhisong Zhang, Xiaoyang Wang, Rui Wang, Can Qin, Yuxuan Wan, Jun-Yu Ma, Ce Zhang, Jiaqi Chen, Xiyun Li, Hongming Zhang, Haitao Mi, and Dong Yu. 2025. Cognitive kernel-pro: A framework for deep research agents and agent foundation models training. *Preprint*, arXiv:2508.00414.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chaowen Hu, Lu Pan, Ke Zeng, and Xunliang Cai. 2026a. How to allocate, how to learn? dynamic rollout allocation and advantage modulation for policy optimization. *arXiv preprint arXiv:2602.19208*.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chang Liu, and Peilin Zhao. 2026b. Proximity-based multi-turn optimization: Practical credit assignment for llm agent training. *arXiv preprint arXiv:2602.19225*.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. 2025. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *Preprint*, arXiv:2505.23564.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. 2025. OpenWebVoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27545–27564, Vienna, Austria. Association for Computational Linguistics.

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

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. 2025. Vineppo: Refining credit assignment in rl training of llms. *Preprint*, arXiv:2410.01679.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy

Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *Preprint*, arXiv:2206.14858.

Chen Li, Nazhou Liu, and Kai Yang. 2025a. Adaptive group policy optimization: Towards stable training and token-efficient reasoning. *Preprint*, arXiv:2503.15952.

Long Li, Xuzheng He, Haozhe Wang, Linlin Wang, and Liang He. 2024. How do humans write code? large models do it the same way too. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, page 4638–4649. Association for Computational Linguistics.

Long Li, Zhijian Zhou, Jiaran Hao, Jason Klein Liu, Yanting Miao, Wei Pang, Xiaoyu Tan, Wei Chu, Zhe Wang, Shirui Pan, Chao Qu, and Yuan Qi. 2026a. The choice of divergence: A neglected key to mitigating diversity collapse in reinforcement learning with verifiable reward. *Preprint*, arXiv:2509.07430.

Long Li, Zhijian Zhou, Jiangxuan Long, Peiyang Liu, Weidi Xu, Zhe Wang, Shirui Pan, and Chao Qu. 2026b. Sql-astra: Alleviating sparse feedback in agentic sql via column-set matching and trajectory aggregation. *Preprint*, arXiv:2603.16161.

Long Li, Zhijian Zhou, Tianyi Wang, Weidi Xu, Zuming Huang, Wei Chu, Zhe Wang, Shirui Pan, Chao Qu, and Yuan Qi. 2026c. Dyjr: Preserving diversity in reinforcement learning with verifiable rewards via dynamic jensen-shannon replay. *Preprint*, arXiv:2603.16157.

Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. 2026d. Verified critical step optimization for llm agents. *Preprint*, arXiv:2602.03412.

Xiaozhe Li, Xinyu Fang, Shengyuan Ding, Linyang Li, Haodong Duan, Qingwen Liu, and Kai Chen. 2025b. Np-engine: Empowering optimization reasoning in large language models with verifiable synthetic np problems. *Preprint*, arXiv:2510.16476.

Zhicong Li, Lingjie Jiang, Yulan Hu, Xingchen Zeng, Yixia Li, Xiangwen Zhang, Guanhua Chen, Zheng Pan, Xin Li, and Yong Liu. 2026e. No more stale feedback: Co-evolving critics for open-world agent learning. *Preprint*, arXiv:2601.06794.

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

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

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Weijian Ma, Shizhao Sun, Tianyu Yu, Ruiyu Wang, Tat-Seng Chua, and Jiang Bian. 2026. Thinking with blueprints: Assisting vision-language models in spatial reasoning via structured object representation. *Preprint*, arXiv:2601.01984.

Zhiwen Ruan, Yun Chen, Yutao Hou, Peng Li, Yang Liu, and Guanhua Chen. 2025a. Unveiling overmemorization in finetuning llms for reasoning tasks. *Preprint*, arXiv:2508.04117.

Zhiwen Ruan, Yixia Li, He Zhu, Yun Chen, Peng Li, Yang Liu, and Guanhua Chen. 2025b. Enhancing large language model reasoning via selective critical token fine-tuning. *Preprint*, arXiv:2510.10974.

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

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297. ACM.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Chengbing Wang, Yang Zhang, Wenjie Wang, Xiaoyan Zhao, Fuli Feng, Xiangnan He, and Tat-Seng Chua. 2026a. Think-while-generating: On-the-fly reasoning for personalized long-form generation. *Preprint*, arXiv:2512.06690.

Chengbing Wang, Wuqiang Zheng, Yang Zhang, Fengbin Zhu, Junyi Cheng, Yi Xie, Wenjie Wang, and Fuli Feng. 2026b. Perm: Psychology-grounded empathetic reward modeling for large language models. *Preprint*, arXiv:2601.10532.

Hu Wang, Congbo Ma, Ian Reid, and Mohammad Yaqub. 2025a. Kalman filter enhanced grpo for reinforcement learning-based language model reasoning. *Preprint*, arXiv:2505.07527.

Jiakang Wang, Runze Liu, Lei Lin, Wenping Hu, Xiu Li, Fuzheng Zhang, Guorui Zhou, and Kun Gai. 2025b. Aspo: Asymmetric importance sampling policy optimization. *Preprint*, arXiv:2510.06062.

Tianyi Wang, Long Li, Hongcan Guo, Yibiao Chen, Yixia Li, Yong Wang, Yun Chen, and Guanhua Chen. 2026c. Anchored policy optimization: Mitigating exploration collapse via support-constrained rectification. *Preprint*, arXiv:2602.05717.

Cehua Yang, Dongyu Xiao, Junming Lin, Yuyang Song, Hanxu Yan, Shawn Guo, Wei Zhang, Jian Yang, Mingjie Tang, and Bryan Dai. 2025. Agro-sql: Agentic group-relative optimization with high-fidelity data synthesis. *Preprint*, arXiv:2512.23366.

Jian Yang, Wei Zhang, Jiajun Wu, Junhang Cheng, Tuney Zheng, Fanglin Xu, Weicheng Gu, Lin Jing, Yaxin Du, Joseph Li, Yizhi Li, Yan Xing, Chuan Hao, Ran Tao, Ruihao Gong, Aishan Liu, Zhoujun Li, Mingjie Tang, Chenghua Lin, Siheng Chen, Wayne Xin Zhao, Xianglong Liu, Ming Zhou, Bryan Dai, and Weifeng Lv. 2026. Incoder-32b-thinking: Industrial code world model for thinking. *Preprint*, arXiv:2604.03144.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. 2025. Dapo: An open-source llm reinforcement learning system at scale. *Preprint*, arXiv:2503.14476.

Yufeng Yuan, Yu Yue, Ruofei Zhu, Tiantian Fan, and Lin Yan. 2025. What's behind ppo's collapse in long-cot? value optimization holds the secret. *Preprint*, arXiv:2503.01491.

Xuan Zhang, Ruixiao Li, Zhijian Zhou, Long Li, Yulei Qin, Ke Li, Xing Sun, Xiaoyu Tan, Chao Qu, and Yuan Qi. 2026. Count counts: Motivating exploration in LLM reasoning with count-based intrinsic rewards. In *The Fourteenth International Conference on Learning Representations*.

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. *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. *Preprint*, arXiv:2507.18071.

He Zhu, Junyou Su, Peng Lai, Ren Ma, Wenjia Zhang, Linyi Yang, and Guanhua Chen. 2026. Anchored supervised fine-tuning. In *The Fourteenth International Conference on Learning Representations*.

Wenjie Zhu, Yabin Zhang, Xin Jin, Wenjun Zeng, and Lei Zhang. 2025. Ants: Adaptive negative textual space shaping for ood detection via test-time mllm understanding and reasoning. *arXiv preprint arXiv:2509.03951*.

A Derivation and Analysis of GRPO Advantage

We model the N sampled responses for a prompt s_p as independent Bernoulli trials with success probability $p = P(R = 1|s_p)$. The GRPO advantage is defined as $Adv = (R_i - \mu_g)/\sigma_g$.

For binary rewards $R_i \in \{0, 1\}$, the sample mean corresponds to the empirical success rate $\mu_g = \hat{p}$. Using standard results for Bernoulli distributions, the unbiased sample standard deviation σ_g converges to the population standard deviation for large N:

$$\sigma_g = \sqrt{\frac{N}{N-1}\hat{p}(1-\hat{p})} \approx \sqrt{\hat{p}(1-\hat{p})} \quad (4)$$

Substituting these into the advantage formulation, we obtain the standardized residual:

$$Adv(s_p, R) \approx \frac{R - \hat{p}}{\sqrt{\hat{p}(1-\hat{p})}} \quad (5)$$

Evaluation for the two possible outcomes $R \in \{0, 1\}$ yields:

$$Adv(s_p, a) = \begin{cases} \sqrt{\frac{1-\hat{p}}{\hat{p}}} & \text{if } R = 1 \text{ (Success)} \\ -\sqrt{\frac{\hat{p}}{1-\hat{p}}} & \text{if } R = 0 \text{ (Failure)} \end{cases} \quad (6)$$

B Extended Visualization of Critic Dynamics

In this section, we present a comprehensive visualization containing ten randomly sampled problems from the DeepScaleR dataset. Figure 9 plots the Critic's value estimates $V(s_t)$ over time for both correct and incorrect responses.

These samples consistently exhibit the "Tail Effect" analyzed in Section 1. Across diverse reasoning tasks, the token-level Critic fails to distinguish between correct (Blue) and incorrect (Red) trajectories during the intermediate reasoning process. The value curves typically remain entangled until the final few tokens, confirming that the standard PPO Critic struggles to assign precise temporal credit in long-horizon reasoning tasks.

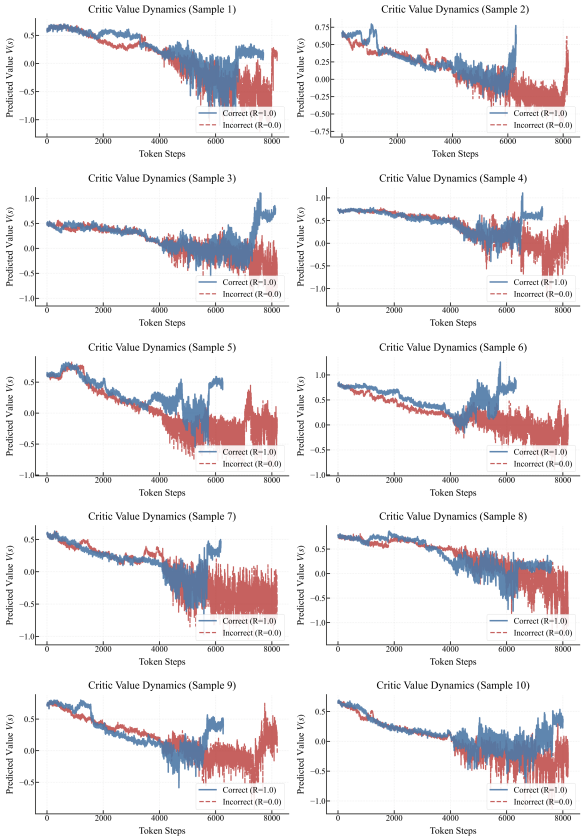

Figure 9: **Extended Analysis of Critic Value Dynamics (10 Random Samples).** Each subplot represents a distinct mathematical problem sampled from the validation set. **Blue Lines:** Value estimates for correct trajectories ($R = 1$). **Red Lines:** Value estimates for incorrect trajectories ($R = 0$). The consistent overlap of value curves until the sequence tail demonstrates the systematic failure of token-level value estimation in distinguishing intermediate reasoning quality.

C Risks

Our proposed SPPO algorithm relies on the assumption of verifiable outcome rewards to estimate prompt solvability. A potential risk involves the overgeneralization of this method to tasks lacking objective ground truths, such as ethical decision-making or subjective content generation. Applying sequence-level optimization in these areas without robust reward modeling may amplify biases present in the base model or lead to the generation of plausible but factually incorrect reasoning chains (hallucination). Furthermore, as SPPO lowers the computational barrier for training strong reasoning models, there is a need for continued monitoring to ensure these accessible capabilities are not deployed for generating harmful content or automating malicious tasks.

D Resources and Implementation Details

To facilitate reproducibility, we summarize the models, datasets, and benchmarks used in our experiments in Table 2. All resources are accessible via HuggingFace.

Category	Resource / Link	License
Models	DeepSeek-R1-Distill-Qwen-1.5B	MIT
	DeepSeek-R1-Distill-Qwen-7B	MIT
Training Datasets	DAPO-Math-17k	Apache 2.0
	DeepScaleR-Preview	MIT
Benchmarks	AIME 2024, AIME 2025	–
	AMC 23, MATH-500	–
	Minerva Math	–

Table 2: Summary of resources and licenses used in this work. Click on the resource names to visit their HuggingFace pages.

E Implementation Details and Execution Commands

In this section, we provide snapshots of the exact execution commands used to reproduce the experimental results. The images are rendered from the actual training scripts.

E.1 SPPO (Ours)

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=sequence_level_adv \
    algorithm.use_kl_in_reward=False \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=256 \
    data.shuffle=True \
    data.trust_remote_code=True \
    data.max_prompt_length=1024 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=True \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.actor.optim.total_training_steps=1000 \
    actor_rollout_ref.actor.optim.warmup_style=constant \
    actor_rollout_ref.actor.optim.weight_decay=0.01 \
    actor_rollout_ref.actor.optim.lr_warmup_steps_ratio=0 \
    actor_rollout_ref.actor.use_torch_compile=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=False \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=False \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.actor.use_dynamic_bsz=False \
    actor_rollout_ref.actor.ppo_mini_batch_size=128 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.rollout.log_prob_use_dynamic_bsz=False \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.rollout.do_sample=True \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.val_kwargs.do_sample=True \
    actor_rollout_ref.rollout.val_kwargs.temperature=0.6 \
    actor_rollout_ref.rollout.max_num_batched_tokens=24576 \
    actor_rollout_ref.rollout.max_model_len=9216 \
    actor_rollout_ref.rollout.enforce_eager=True \
    actor_rollout_ref.rollout.tensor_model_parallel_size=1 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    critic.enable=True \
    critic.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B \
    critic.optim.lr=5e-6 \
    critic.model.use_remove_padding=False \
    critic.model.enable_gradient_checkpointing=True \
    critic.model.fsdp_config.param_offload=False \
    critic.model.fsdp_config.optimizer_offload=False \
    critic.use_dynamic_bsz=False \
    critic.ppo_mini_batch_size=128 \
    critic.ppo_micro_batch_size_per_gpu=1 \
    trainer.critic_warmup=0 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='ds1.5B_PPO_SEQUENCE' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=2 \
    trainer.test_freq=10 \
    trainer.val_before_train=True \
    trainer.total_training_steps=1000 $@
```

Figure 10: Execution Command: SPPO 1.5B (Symmetric)

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=sequence_level_adv \
    algorithm.use_kl_in_reward=False \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=512 \
    data.shuffle=True \
    data.trust_remote_code=True \
    data.max_prompt_length=2048 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=False \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-7B \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.actor.optim.total_training_steps=200 \
    actor_rollout_ref.actor.optim.warmup_style=constant \
    actor_rollout_ref.actor.optim.weight_decay=0.1 \
    actor_rollout_ref.actor.optim.lr_warmup_steps_ratio=0 \
    actor_rollout_ref.actor.use_torch_compile=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=True \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=True \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.actor.use_dynamic_bsz=False \
    actor_rollout_ref.actor.ppo_mini_batch_size=32 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.log_prob_use_dynamic_bsz=False \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.do_sample=True \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.val_kwargs.do_sample=True \
    actor_rollout_ref.rollout.val_kwargs.temperature=0.6 \
    actor_rollout_ref.rollout.max_model_len=20480 \
    actor_rollout_ref.rollout.max_num_batched_tokens=32768 \
    actor_rollout_ref.rollout.enforce_eager=True \
    actor_rollout_ref.rollout.tensor_model_parallel_size=4 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    critic.enable=True \
    critic.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-7B \
    critic.optim.lr=1e-5 \
    critic.model.use_remove_padding=False \
    critic.model.enable_gradient_checkpointing=True \
    critic.model.fsdp_config.param_offload=False \
    critic.model.fsdp_config.optimizer_offload=False \
    critic.use_dynamic_bsz=False \
    critic.ppo_mini_batch_size=32 \
    critic.ppo_micro_batch_size_per_gpu=1 \
    trainer.critic_warmup=0 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='DAPO-R1-7B-Seq' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=10 \
    trainer.test_freq=2 \
    trainer.val_before_train=True \
    trainer.total_training_steps=200 $@
```

Figure 11: Execution Command: SPPO 7B (Symmetric)

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=sequence_level_adv \
    algorithm.use_kl_in_reward=False \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=512 \
    data.shuffle=True \
    data.trust_remote_code=True \
    data.max_prompt_length=2048 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=False \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-7B \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.actor.optim.total_training_steps=200 \
    actor_rollout_ref.actor.optim.warmup_style=constant \
    actor_rollout_ref.actor.optim.weight_decay=0.1 \
    actor_rollout_ref.actor.optim.lr_warmup_steps_ratio=0 \
    actor_rollout_ref.actor.use_torch_compile=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=True \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=True \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.actor.use_dynamic_bsz=False \
    actor_rollout_ref.actor.ppo_mini_batch_size=32 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.log_prob_use_dynamic_bsz=False \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.do_sample=True \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.val_kwargs.do_sample=True \
    actor_rollout_ref.rollout.val_kwargs.temperature=0.6 \
    actor_rollout_ref.rollout.max_model_len=20480 \
    actor_rollout_ref.rollout.max_num_batched_tokens=32768 \
    actor_rollout_ref.rollout.enforce_eager=True \
    actor_rollout_ref.rollout.tensor_model_parallel_size=4 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    critic.enable=True \
    critic.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B \
    critic.optim.lr=1e-5 \
    critic.model.use_remove_padding=False \
    critic.model.enable_gradient_checkpointing=True \
    critic.model.fsdp_config.param_offload=False \
    critic.model.fsdp_config.optimizer_offload=False \
    critic.use_dynamic_bsz=False \
    critic.ppo_mini_batch_size=32 \
    critic.ppo_micro_batch_size_per_gpu=1 \
    trainer.critic_warmup=0 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='DAPO-R1-7B-Seq_small_critic' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=10 \
    trainer.test_freq=2 \
    trainer.val_before_train=True \
    trainer.total_training_steps=200 $@
```

Figure 12: Execution Command: SPPO 7B (Decoupled / Small Critic)

E.2 Group Relative Policy Optimization (GRPO)

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=grpo \
    algorithm.use_kl_in_reward=False \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=256 \
    data.shuffle=True \
    data.trust_remote_code=True \
    data.max_prompt_length=1024 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=True \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.actor.optim.total_training_steps=1000 \
    actor_rollout_ref.actor.optim.warmup_style=constant \
    actor_rollout_ref.actor.optim.weight_decay=0.01 \
    actor_rollout_ref.actor.optim.lr_warmup_steps_ratio=0 \
    actor_rollout_ref.actor.use_torch_compile=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=False \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=False \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.actor.use_dynamic_bsz=False \
    actor_rollout_ref.actor.ppo_mini_batch_size=128 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.rollout.log_prob_use_dynamic_bsz=False \
    actor_rollout_ref.rollout.n=8 \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.rollout.do_sample=True \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.val_kwargs.do_sample=True \
    actor_rollout_ref.rollout.val_kwargs.temperature=0.6 \
    actor_rollout_ref.rollout.max_num_batched_tokens=24576 \
    actor_rollout_ref.rollout.max_model_len=9216 \
    actor_rollout_ref.rollout.enforce_eager=True \
    actor_rollout_ref.rollout.tensor_model_parallel_size=1 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='ds1.5B_GRPO' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=2 \
    trainer.test_freq=10 \
    trainer.val_before_train=True \
    trainer.total_training_steps=1000 $@
```

Figure 13: Execution Command: GRPO 1.5B

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=grpo \
    algorithm.use_kl_in_reward=False \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=512 \
    data.shuffle=True \
    data.trust_remote_code=True \
    data.max_prompt_length=2048 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=False \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-7B  \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.actor.optim.total_training_steps=200 \
    actor_rollout_ref.actor.optim.warmup_style=constant \
    actor_rollout_ref.actor.optim.weight_decay=0.1 \
    actor_rollout_ref.actor.optim.lr_warmup_steps_ratio=0 \
    actor_rollout_ref.actor.use_torch_compile=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=True \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=True \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.actor.use_dynamic_bsz=False \
    actor_rollout_ref.rollout.n=8 \
    actor_rollout_ref.actor.ppo_mini_batch_size=32 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.log_prob_use_dynamic_bsz=False \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.do_sample=True \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.val_kwargs.do_sample=True \
    actor_rollout_ref.rollout.val_kwargs.temperature=0.6 \
    actor_rollout_ref.rollout.max_num_batched_tokens=16384 \
    actor_rollout_ref.rollout.enforce_eager=True \
    actor_rollout_ref.rollout.tensor_model_parallel_size=4 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    trainer.critic_warmup=0 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='GRPO-R1-7B_DAPO-17k' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=10 \
    trainer.test_freq=2 \
    trainer.val_before_train=True \
    trainer.total_training_steps=200 $@z
```

Figure 14: Execution Command: GRPO 7B

E.3 Standard PPO

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=gae \
    algorithm.use_kl_in_reward=False \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=256 \
    data.shuffle=True \
    data.trust_remote_code=True \
    data.max_prompt_length=1024 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=True \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.actor.optim.total_training_steps=1000 \
    actor_rollout_ref.actor.optim.warmup_style=constant \
    actor_rollout_ref.actor.optim.weight_decay=0.01 \
    actor_rollout_ref.actor.optim.lr_warmup_steps_ratio=0 \
    actor_rollout_ref.actor.use_torch_compile=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=False \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=False \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.actor.use_dynamic_bsz=False \
    actor_rollout_ref.actor.ppo_mini_batch_size=128 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.rollout.log_prob_use_dynamic_bsz=False \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.rollout.do_sample=True \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.val_kwargs.do_sample=True \
    actor_rollout_ref.rollout.val_kwargs.temperature=0.6 \
    actor_rollout_ref.rollout.max_num_batched_tokens=24576 \
    actor_rollout_ref.rollout.max_model_len=9216 \
    actor_rollout_ref.rollout.enforce_eager=True \
    actor_rollout_ref.rollout.tensor_model_parallel_size=1 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    critic.enable=True \
    critic.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B \
    critic.optim.lr=1e-5 \
    critic.model.use_remove_padding=False \
    critic.model.enable_gradient_checkpointing=True \
    critic.model.fsdp_config.param_offload=False \
    critic.model.fsdp_config.optimizer_offload=False \
    critic.use_dynamic_bsz=False \
    critic.ppo_mini_batch_size=256 \
    critic.ppo_micro_batch_size_per_gpu=1 \
    trainer.critic_warmup=50 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='ds1.5B_PPO' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=2 \
    trainer.test_freq=10 \
    trainer.val_before_train=True \
    trainer.total_training_steps=1000 $@z
```

Figure 15: Execution Command: Standard PPO 1.5B

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=gae \
    algorithm.use_kl_in_reward=False \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=512 \
    data.shuffle=True \
    data.trust_remote_code=True \
    data.max_prompt_length=2048 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=False \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-7B  \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.actor.optim.total_training_steps=200 \
    actor_rollout_ref.actor.optim.warmup_style=constant \
    actor_rollout_ref.actor.optim.weight_decay=0.1 \
    actor_rollout_ref.actor.optim.lr_warmup_steps_ratio=0 \
    actor_rollout_ref.actor.use_torch_compile=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=True \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=True \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.actor.use_dynamic_bsz=False \
    actor_rollout_ref.actor.ppo_mini_batch_size=32 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.log_prob_use_dynamic_bsz=False \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.do_sample=True \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.val_kwargs.do_sample=True \
    actor_rollout_ref.rollout.val_kwargs.temperature=0.6 \
    actor_rollout_ref.rollout.max_num_batched_tokens=32768 \
    actor_rollout_ref.rollout.max_model_len=20480 \
    actor_rollout_ref.rollout.enforce_eager=True \
    actor_rollout_ref.rollout.tensor_model_parallel_size=4 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    critic.enable=True \
    critic.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-7B \
    critic.optim.lr=1e-5 \
    critic.model.use_remove_padding=False \
    critic.model.enable_gradient_checkpointing=True \
    critic.model.fsdp_config.param_offload=False \
    critic.model.fsdp_config.optimizer_offload=False \
    critic.use_dynamic_bsz=False \
    critic.ppo_mini_batch_size=32 \
    critic.ppo_micro_batch_size_per_gpu=1 \
    trainer.critic_warmup=50 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='PPO-R1_DAPO-17k' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=10 \
    trainer.test_freq=2 \
    trainer.val_before_train=True \
    trainer.total_training_steps=200 $@
```

Figure 16: Execution Command: Standard PPO 7B

E.4 RLOO Baselines

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=rloo \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=256 \
    data.max_prompt_length=1024 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=True \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.actor.ppo_mini_batch_size=128 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=False \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=False \
    actor_rollout_ref.rollout.max_num_batched_tokens=24576 \
    actor_rollout_ref.rollout.max_model_len=9216 \
    actor_rollout_ref.rollout.enforce_eager=True \
    data.trust_remote_code=True \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.rollout.tensor_model_parallel_size=2 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    actor_rollout_ref.rollout.n=5 \
    actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=2 \
    actor_rollout_ref.ref.fsdp_config.param_offload=True \
    algorithm.use_kl_in_reward=True \
    algorithm.kl_penalty=kl \
    algorithm.kl_ctrl.kl_coef=0.001 \
    trainer.critic_warmup=0 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.default_local_dir='./trainer.project_name/{trainer.experiment_name}' \
    trainer.experiment_name='ds1.5B_RLOO_deepscaler' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=2 \
    trainer.test_freq=10 \
    trainer.total_training_steps=1000 $@
```

Figure 17: Execution Command: RLOO 1.5B

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=rloo \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=512 \
    data.max_prompt_length=2048 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=True \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-7B  \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.actor.ppo_mini_batch_size=32 \
    actor_rollout_ref.actor.ppo_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=False \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=False \
    actor_rollout_ref.rollout.max_num_batched_tokens=32768 \
    actor_rollout_ref.rollout.max_model_len=20480 \
    actor_rollout_ref.rollout.enforce_eager=True \
    data.trust_remote_code=True \
    actor_rollout_ref.rollout.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.rollout.tensor_model_parallel_size=4 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    actor_rollout_ref.rollout.n=5 \
    actor_rollout_ref.ref.log_prob_micro_batch_size_per_gpu=1 \
    actor_rollout_ref.ref.fsdp_config.param_offload=True \
    algorithm.use_kl_in_reward=True \
    algorithm.kl_penalty=kl \
    algorithm.kl_ctrl.kl_coef=0.001 \
    actor_rollout_ref.rollout.temperature=0.6 \
    trainer.critic_warmup=0 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.experiment_name='R1-7B_RLOO_DAPO-17k' \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=10 \
    trainer.test_freq=2 \
    trainer.total_training_steps=200 $@
```

Figure 18: Execution Command: RLOO 7B

E.5 ReMax Baselines

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=remax \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=256 \
    data.max_prompt_length=1024 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=True \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.rollout.max_num_batched_tokens=24576 \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.max_model_len=9216 \
    actor_rollout_ref.rollout.enforce_eager=True \
    data.trust_remote_code=True \
    actor_rollout_ref.actor.ppo_mini_batch_size=128 \
    actor_rollout_ref.actor.use_dynamic_bsz=True \
    actor_rollout_ref.actor.ppo_max_token_len_per_gpu=24000 \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=False \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=False \
    actor_rollout_ref.rollout.tensor_model_parallel_size=1 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    actor_rollout_ref.ref.fsdp_config.param_offload=True \
    algorithm.use_kl_in_reward=True \
    algorithm.kl_penalty=kl \
    algorithm.kl_ctrl.kl_coef=0.001 \
    trainer.critic_warmup=0 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='ds1.5B_REMAX_deepscaler' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.val_before_train=False \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=2 \
    trainer.test_freq=10 \
    trainer.total_training_steps=1000 $@
```

Figure 19: Execution Command: ReMax 1.5B

```
python3 -m verl.trainer.main_ppo \
    algorithm.adv_estimator=remax \
    data.train_files="$train_files" \
    data.val_files="$test_files" \
    data.train_batch_size=512 \
    data.max_prompt_length=2048 \
    data.max_response_length=8192 \
    data.filter_overlong_prompts=True \
    data.truncation='left' \
    actor_rollout_ref.model.path=deepseek-ai/DeepSeek-R1-Distill-Qwen-7B \
    actor_rollout_ref.actor.optim.lr=1e-6 \
    actor_rollout_ref.model.use_remove_padding=True \
    actor_rollout_ref.rollout.temperature=0.6 \
    actor_rollout_ref.rollout.val_kwargs.temperature=0.6 \
    actor_rollout_ref.rollout.max_num_batched_tokens=32768 \
    actor_rollout_ref.rollout.max_model_len=20480 \
    actor_rollout_ref.rollout.enforce_eager=True \
    data.trust_remote_code=True \
    actor_rollout_ref.actor.ppo_mini_batch_size=32 \
    actor_rollout_ref.actor.use_dynamic_bsz=True \
    actor_rollout_ref.actor.ppo_max_token_len_per_gpu=24000 \
    actor_rollout_ref.actor.use_kl_loss=False \
    actor_rollout_ref.model.enable_gradient_checkpointing=True \
    actor_rollout_ref.actor.fsdp_config.param_offload=False \
    actor_rollout_ref.actor.fsdp_config.optimizer_offload=False \
    actor_rollout_ref.rollout.tensor_model_parallel_size=4 \
    actor_rollout_ref.rollout.name=vllm \
    actor_rollout_ref.rollout.gpu_memory_utilization=0.8 \
    actor_rollout_ref.ref.fsdp_config.param_offload=True \
    algorithm.use_kl_in_reward=True \
    algorithm.kl_penalty=kl \
    algorithm.kl_ctrl.kl_coef=0.001 \
    trainer.critic_warmup=0 \
    trainer.logger='["console","wandb"]' \
    trainer.project_name='SPPO' \
    trainer.experiment_name='R1-7B_REMAX_DAPO' \
    trainer.default_local_dir='./${trainer.experiment_name}' \
    trainer.val_before_train=False \
    trainer.n_gpus_per_node=4 \
    trainer.nnodes=1 \
    trainer.save_freq=10 \
    trainer.test_freq=2 \
    trainer.total_training_steps=200 $@
```

Figure 20: Execution Command: ReMax 7B