RESUM: SYNERGIZING LLM REASONING AND SUMMARIZATION WITH REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning with Verifiable Rewards (RLVR) is a central technique for improving long-horizon reasoning in Large Language Models (LLMs). Yet, reward-driven reasoning can produce unnecessarily long rollouts, repeated verification, and accumulated errors that reduce coherence and exhaust the generation budget. Existing long-context organization methods commonly rely on external compressors, cache mechanisms, or auxiliary modules. We propose `ReSum`, an RLVR framework that internalizes self-summarization and related reasoning-control behavior to organize subsequent reasoning within the policy. Our pilot studies reveal that summary-like cues occur around high-uncertainty states, are followed by lower token-level entropy, and can improve recovery from incorrect rollout prefixes. Building on these observations, ReSum constructs adaptive contrastive rollout trees around two complementary events: Natural Points (NPs), where a model-generated cue is masked, and Artifact Points (APs), where a cue is injected at a selected non-cue position. ReSum then applies a dual-reference, cue-aware advantage that evaluates each continuation against cue-containing and cue-free rollout distributions, providing state-local credit assignment for beneficial reasoning-control behavior. Across six mathematical benchmarks, multiple backbone sizes and families, and a multimodal geometry benchmark, ReSum consistently improves reasoning accuracy by an average of 4% while reducing rollout length by 18.6%. These results demonstrate a practical RLVR approach for achieving a stronger accuracy–length trade-off in verifiable long-horizon reasoning.

1 INTRODUCTION

Recently, Reinforcement Learning with Verifiable Rewards (RLVR) Schulman et al. (2017); Guo et al. (2025); Yu et al. (2025); Li et al. (2025c) has substantially improved the long-chain reasoning abilities of Large Language Models (LLMs) across complex tasks, including search Jin et al. (2025); Li et al. (2025c); Wang et al. (2025), coding Mai et al. (2025); Liu et al. (2024a), and GUI interaction Shi et al. (2025); Lu et al. (2026); Yuan et al. (2025); Luo et al. (2025). However, these gains often come with a reward bias toward longer Chain-of-Thought (CoT) sequences Chen et al. (2024b); Zhang et al. (2025c); Liu et al. (2026); Wu et al. (2025b); Srivastava et al. (2025), which makes LLMs prone to overthinking. As reasoning trajectories become longer, models may forget or misremember earlier steps Dong et al. (2025c); Liu et al. (2024b); Chen et al. (2024a), revisit already resolved subproblems, and exhaust their reasoning budget through repeated retries Ling et al. (2025) or self-validation loops.

To address these issues, existing studies have explored a range of long-context organization techniques Koh et al. (2022); Dai et al. (2025); Ming et al. (2024), which compress long interaction histories into summaries through cache monitoring Yang et al. (2024c), autoencoder-based compressors Ge et al. (2023); Dai et al. (2025), or additional agentic modules Zhang et al. (2025a); Wu et al. (2025a). However, these methods typically rely on intricate workflows or auxiliary models to manage long contexts externally, which may introduce additional overhead and raise concerns about the faithfulness Longpre et al. (2021); Ming et al. (2024); Long et al. (2025) of summaries or outputs produced by external modules. This motivates a central question: *can LLMs' own self-summarization serve as an intrinsic mechanism for organizing long reasoning trajectories?* To investigate this question, we conduct two pilot studies, as shown in Figure 1. In Figure 1(a), inspired by recent studies Gao et al. (2025); Dong et al. (2025b) that highlight the importance of high-entropy tokens in LLM reasoning, we analyze token entropy around summarization phrases in rollout trajectories.

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

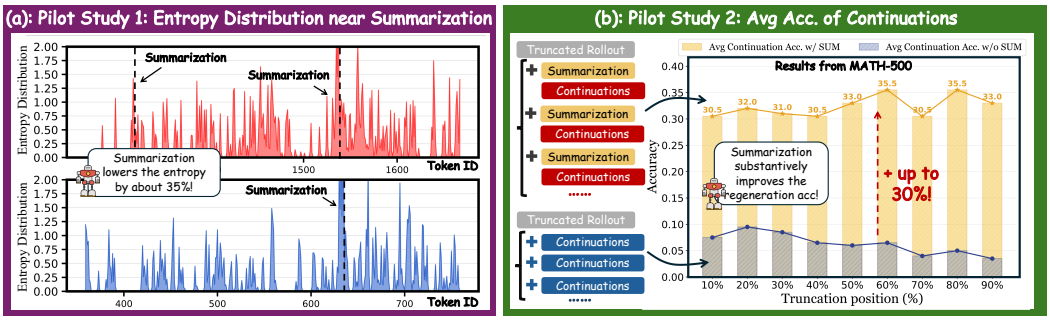

Figure 1: Pilot studies. **(a):** Analysis of token entropy distributions near the summarization phrases. **(b):** Average accuracy of continuations of wrong rollouts truncated from different positions.

We find that positions preceding summarization phrases consistently exhibit high entropy, while the summarization phrase itself is followed by a sharp entropy decrease. In Figure 1(b), we truncate incorrect rollouts at different positions and regenerate 100 continuations with and without an injected summarization phrase. We find that injecting summarization phrases consistently improves regeneration accuracy, with gains of up to 30%. Together, these findings reveal a previously overlooked behavior of LLMs:

> *Self-summarization serves as an intrinsic control mechanism in long-horizon reasoning: it arises when the model reaches a high-entropy state, and reduces generation uncertainty by consolidating ongoing rollouts and reorganizing the wrong prefixes.*

Motivated by these findings, we train LLMs to use self-summarization at appropriate points in the reasoning process, thereby reorganizing their subsequent reasoning and recovering from low-quality rollout prefixes. To this end, ReSum encourages beneficial summary-like reasoning-control events through adaptive branching and fine-grained credit assignment. **Artifact Points (APs)** are sampled from positions without a detected cue, where a cue is injected to estimate whether it improves the subsequent continuation. **Natural Points (NPs)** are positions identified by the cue detector in a model-generated rollout, where the matched cue is masked to test its contribution. The detector deliberately includes both explicit consolidation phrases and a small set of related reflection or repair cues; therefore, NPs are operationally defined reasoning-control events, while "self-summarization" refers to the consolidation subset. This construction lets ReSum learn when such control events are useful while retaining self-summarization as its central motivating behavior.

On the resulting rollout tree, ReSum forms cue-containing and cue-free reference groups and evaluates each rollout against both reference distributions. This dual-reference advantage supplies fine-grained, process-label-free supervision for cue-conditioned continuations. With no supervised fine-tuning, ReSum improves accuracy by an average of 4% while reducing rollout length by 18.6%. Our contributions are threefold:

- We present pilot studies showing that summary-like reasoning-control cues are associated with lower token-level uncertainty and can mitigate error propagation from incorrect rollout prefixes.
- We propose ReSum, a tree-based RL framework that incentivizes beneficial self-summarization and related reasoning-control behavior. ReSum combines two types of contrastive branching with a dual-reference advantage to provide fine-grained supervision over when to introduce or retain a control cue.
- Extensive experiments validate the effectiveness of ReSum on mathematical and multimodal reasoning, improving accuracy by an average of 4% while substantially reducing rollout length.

2 Related Work

Reinforcement Learning with Verifiable Rewards. Beyond prompting and supervised fine-tuning, Reinforcement Learning with Verifiable Rewards (RLVR) Schulman et al. (2017); Rafailov et al. (2023); Guo et al. (2025); Liu et al. (2025) has emerged as a core driver for enhancing the long chain-of-thought reasoning capabilities of LLMs and incentivizing reasoning behaviors such as planning and reflection. Building upon traditional Proximal Policy Optimization (PPO) Schulman

et al. (2017), subsequent research has focused on designing critic-free, lightweight, and efficient policy gradient paradigms, which can be roughly grouped into three lines: **(1) fine-grained objective designs** Guo et al. (2025); Yu et al. (2025); Zheng et al. (2025); Wang et al. (2026), which incorporate finer advantage or loss formulations Guo et al. (2025); Zhao et al. (2025b); Feng et al. (2025); Chu et al. (2026), apply policy-gradient losses at multiple granularities Yu et al. (2025); Zheng et al. (2025), or adopt curriculum scaling Dai et al. (2026); Li et al. (2026). For instance, DAPO Yu et al. (2025) extends the gradient loss from the sequence level in GRPO Guo et al. (2025) to token-level variants, while GPG Chu et al. (2026) returns to the original policy gradient formulation and removes auxiliary clipping or KL-regularization terms. **(2) process-level rewarding** Ma et al. (2026a); Feng et al. (2025), which tackles the credit assignment problem by distributing sequence-level rewards with a discount factor Ji et al. (2025); Xie et al. (2024) or by training Process Reward Models (PRMs) Shao et al. (2024); Zhang et al. (2025b); Wang et al. (2025). **(3) rollout expansion** Ji et al. (2025); Hou et al. (2025); Dong et al. (2025b;a), which assigns distinct advantages and gradient losses to rollout segments of varying importance. For example, TreeRL Hou et al. (2025) selects tokens with the highest uncertainty for branching, while Tree-GRPO Ji et al. (2025) selects thought-action steps and optimizes them with a two-level advantage.

In-context Organization for LLMs. As emerging studies unraveling that *"longer reasoning is not always better"* Wu et al. (2025b); Liu et al. (2026); Srivastava et al. (2025); Chen et al. (2024b), recent work has discovered the in-context organization Li et al. (2025a); Petrov et al. (2025) (like compression Berton et al. (2025), resolve Wang et al. (2023), self-correction Xiong et al. (2025); Zhao et al. (2025a); Paul et al. (2024) and summarization) mechanisms to refine the models' attention of the long in-contexts Hooper et al. (2025). Based on the invoking mode of such techniques, there are ***(1)*** *external methods* Dai et al. (2025); Ge et al. (2023); Guo et al. (2026), where the long-contexts are reframed by cache monitors Ma et al. (2026b), text compressors Tian et al. (2026) or harnessing engineering Zhang et al. (2025a); Ma et al. (2026c). ***(2)*** *internal methods* Wang et al. (2026); Bensal et al. (2025); Li et al. (2025d;b), where models are trained or incentivized to selectively revisit their generated content with distillation, supervised fine-tuning or RLVR. For example, Re^2 Wang et al. (2026) encourages LLMs to perform a "redo" operation with an out-group reward and Test-Time Scaling (TTS) Muennighoff et al. (2025); RLTR Li et al. (2025d) leverages the "tool-call completeness" as the reward to refine the planning ability of agents to mitigate the insufficient annotations in real-world applications. *In alignment with these studies, this paper studies how summarization ability can be internalized for complex reasoning.*

3 METHODOLOGY

3.1 PRELIMINARIES

Notations. We adopt the standard RLVR notation for Large Language Models (LLMs). The LLM π_θ serves as the policy model and autoregressively generates rollouts given an input query $q \in \mathcal{D}$, where $\mathcal{D}$ is the training set. For the rollouts $\{o_m\}_{m=1}^{M}$ generated by the current policy π_θ, we assign a scalar reward r_m using either LLM-based reward models or verifiable reward functions. In the case of mathematical reasoning, $r_m \in \{0, 1\}$ indicates verifiable task reward of mathematical results.

Group Relative Policy Optimization (GRPO). GRPO Guo et al. (2025) is a critic-free policy optimization method that computes relative advantages within a group of responses sampled from the same query. It has inspired numerous subsequent studies aimed at refining the normalization scheme Chu et al. (2026), adjusting levels of gradient optimization Yu et al. (2025), weighting different samples or rollouts Dai et al. (2026), and related aspects. Specifically, GRPO optimizes the following objective to train the policy model π_θ:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) =& \mathbb{E}_{q\sim\mathcal{D},\{o_m\}_{m=1}^{M}\sim\pi_{\text{old}}(\cdot;q)}\Big[\frac{1}{M}\sum_{m=1}^{M}\frac{1}{|o_m|}\sum_{i=1}^{|o_m|}\min\big(\rho_{m,i}(\theta)\hat{A}_{m,i},\\ &\text{clip}(\rho_{m,i}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{m,i}\big) - \beta\mathbb{D}_{\text{KL}}(\pi_\theta(\cdot;q)||\pi_{\text{ref}}(\cdot;q))\Big] \\ \rho_{m,i}(\theta) =& \frac{\pi_\theta(o_{m,i}; o_{m,<i}, q)}{\pi_{\text{old}}(o_{m,i}; o_{m,<i}, q)}, \quad \hat{A}_{m,i} = \frac{r_m - \text{mean}(\{r_m\}_{m=1}^{M})}{\text{std}(\{r_m\}_{m=1}^{M})}\end{aligned} \tag{1}$$

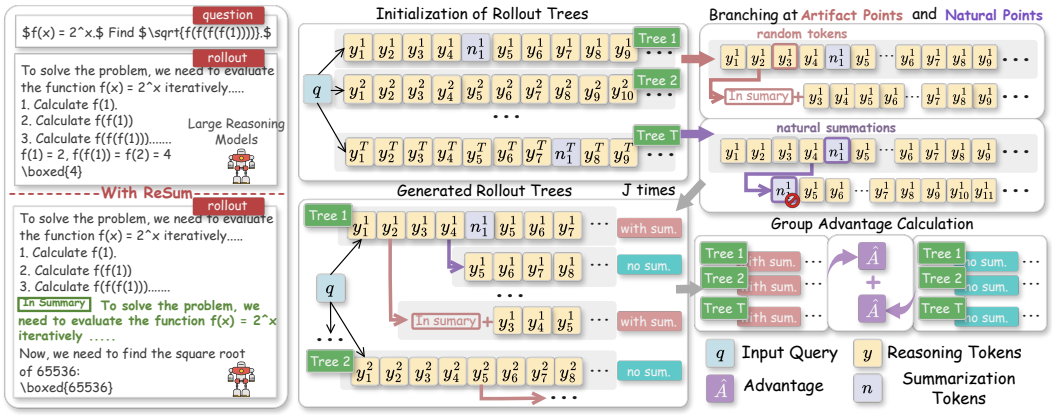

Figure 2: The proposed ReSum. **(left):** ReSum incentivizes self-summarization and related reasoning-control behavior. **(middle & right):** the LLM generates initial rollouts for each query, then branches at Artifact Points (APs) and Natural Points (NPs). Each branch is evaluated against cue-containing and cue-free reference distributions.

Where π_{old} and π_{ref} represent the old policy and reference policy respectively. $\mathbb{D}_{\text{KL}}$ represents the KL-Divergence. ϵ and β are the clipping range and regularization hyper-parameter respectively. $\rho_{m,i}$ represents the importance sampling ratio of rollout o_m at token $o_{m,i}$.

3.2 Summarization-based Rollout Expansion

Beyond binary correctness of the final answer, periodic self-summarization can help mitigate errors caused by repetitive retries or loss of track of earlier reasoning. Static prompting provides a global instruction, whereas ReSum uses RLVR to identify and contrastively evaluate summary-like reasoning-control events at the current rollout state, as shown in Figure 2. (A detailed comparison between prompt refinement and ReSum is provided in Appendix E.)

Inspired by recent work showing that tree-based RLVR, i.e., sampling multiple branches from a single rollout, is equivalent to preference alignment between branches Zhang et al. (2025b); Lai et al. (2024), we devise a tree-based rollout strategy grounded in self-summarization. By dynamically injecting or masking summary-like cues and incorporating branches into reference groups, ReSum learns when these events improve the continuation. Specifically, the tree generation follows:

► **Initialization:** For each query q, we first generate T independent initial rollouts $\{\tau_t^0\}_{t=1}^T$ from the policy model π_θ. Each initial rollout τ_t^0 serves as the root of a rollout tree $\mathcal{T}_t$.

► **Sampling:** In the j-th generation loop, where $j = 1, \ldots, J-1$, for each rollout tree $\mathcal{T}_t$, we select one point from two types of branching points: Artifact Points (APs) $\mathcal{L}_{t,j}^{ap}$ and Natural Points (NPs) $\mathcal{L}_{t,j}^{np}$. APs are randomly sampled from the initial rollout and used to inject a cue. NPs are keyword-matched, naturally occurring summary-like reasoning-control events in the model rollout. We prioritize NPs; APs fill the remaining branch budget.

► **Branching:** In the j-th generation loop, for each rollout tree $\mathcal{T}_t$, we resample branches from either $\mathcal{L}_{t,j}^{ap}$ or $\mathcal{L}_{t,j}^{np}$. **(1)** At an AP, we append a cue sampled from the collection in Appendix D to the truncated prefix and regenerate its continuation. **(2)** At an NP, we mask the matched cue in the prefix and regenerate its continuation. ReSum retains the full preceding history and augments the continuation prompt with these state-specific interventions. The resulting branch is merged into $\mathcal{T}_t$.

► **Termination:** The overall generation loop terminates when the total number of rollouts reaches the budget B, where B is divisible by T. Notably, branches are evenly distributed across trees.

Different combinations of T and J can yield rollout trees of varying shapes. Specifically, a larger T leads to greater diversity in summarizations across different rollouts, while a larger J produces more branches within each individual rollout. We examine this trade-off further in the ablation studies. Notably, all branching operations are performed exclusively on the initial rollouts.

3.3 SUMMARIZATION-AWARE GROUP-RELATIVE POLICY OPTIMIZATION (SGPO)

In tree-based rollouts, branches provide state-local counterfactual supervision. We use a small correctness-gated auxiliary preference for cue-bearing correct rollouts, then normalize each rollout against two reference groups. Specifically, let $\tau_{t,e}$ denote the e-th rollout in $\mathcal{T}_t$. The reward is $R(\tau_{t,e}) = R_A(\tau_{t,e}) + \mathbf{1}[R_A(\tau_{t,e}) = 1]R_F(\tau_{t,e})$, where

$$R_A(\tau_{t,e}) = \begin{cases} 1, & \text{if Correct}(\tau_{t,e}) \\ 0, & \text{otherwise} \end{cases}, R_F(\tau_{t,e}) = \frac{0.2 \times N_{\mathrm{NP}}(\tau_{t,e}) + 0.05 \times N_{\mathrm{AP}}(\tau_{t,e})}{N_{\mathrm{NP}}(\tau_{t,e}) + N_{\mathrm{AP}}(\tau_{t,e}) + \varepsilon_R} \tag{2}$$

where $N_{\mathrm{NP}}(\tau_{t,e})$ and $N_{\mathrm{AP}}(\tau_{t,e})$ denote detected natural and injected artifact cues, respectively, and $\varepsilon_R = 10^{-6}$ is a numerical stabilizer. The correctness gate focuses the auxiliary preference on successful trajectories, and normalization keeps its scale bounded across different numbers of cues. We use the larger coefficient for NPs because they are model-generated events, whereas APs are injected counterfactual probes. Let G_{cue} contain rollouts with a detected natural cue or an AP injection, and let G_{nocue} be its complement. For every rollout, we compute two reference-normalized terms:

$$\hat{A}^g(\tau_{t,e}) = \frac{R(\tau_{t,e}) - \mathrm{mean}_{\tau \in G_g}(\{R(\tau)\})}{\mathrm{std}_{\tau \in G_g}(\{R(\tau)\}) + \xi}, \quad g \in \{\mathrm{cue}, \mathrm{nocue}\} \tag{3}$$

Here $\xi > 0$ is the advantage stabilizer. The two terms are evaluated for every rollout, exactly as in Algorithm 1, forming a dual-reference signal. The rollout-level advantage is $\hat{A}(\tau_{t,e}) = \hat{A}^{\mathrm{cue}}(\tau_{t,e}) + \hat{A}^{\mathrm{nocue}}(\tau_{t,e})$, and is assigned to all generated tokens in $\tau_{t,e}$. For example, a correct cue-bearing branch receives its task reward plus the bounded auxiliary term, then is scored relative to both the cue-bearing and cue-free reward distributions. This gives the update signal a transparent and reproducible interpretation of the full reasoning-control mechanism. The optimization objective is:

$$\begin{aligned} \mathcal{J}_{\mathrm{ReSum}}(\theta) = & \mathbb{E}_{q \sim \mathcal{D}, \{\mathcal{T}_t\}_{t=1}^{T} \sim \pi_{\mathrm{old}}(\cdot;q)} [\frac{1}{T} \sum_{t=1}^{T} \frac{1}{|\mathcal{T}_t|} \sum_{e=1}^{|\mathcal{T}_t|} \frac{1}{|\tau_{t,e}|} \sum_{i=1}^{|\tau_{t,e}|} \min\big(\rho_{t,e,i}(\theta)\hat{A}(\tau_{t,e}), \\ & \mathrm{clip}(\rho_{t,e,i}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}(\tau_{t,e})\big) - \beta \mathbb{D}_{\mathrm{KL}}(\pi_\theta(\cdot;q) || \pi_{\mathrm{ref}}(\cdot;q))] \end{aligned} \tag{4}$$

Where π_{old} and π_{ref} represent the old policy and reference policy respectively. $\mathbb{D}_{\mathrm{KL}}$ represents the KL-Divergence. ϵ and β are the clipping range and regularization hyper-parameter respectively. $\rho_{t,e,i}(\theta) = \pi_\theta(\tau_{t,e,i}; \tau_{t,e,<i}, q)/\pi_{\mathrm{old}}(\tau_{t,e,i}; \tau_{t,e,<i}, q)$ represents the importance sampling ratio.

3.4 THEORETICAL PERSPECTIVE

Inspired by the tree-structured analysis in recent studies Ji et al. (2025); Zhao et al. (2026), we provide a concise perspective on ReSum. ReSum retains the preceding prefix and uses an explicit cue to reorganize the model's subsequent reasoning. The cue creates a localized decision point: one continuation can consolidate the current reasoning state, while the matched branch exposes the value of proceeding without that intervention. Under a fixed rollout budget, this contrast turns trajectory-level correctness into a targeted signal for selecting productive control points and makes the continuation easier to optimize when the cue preserves task-relevant state and redirects redundant reasoning.

Specifically, let $H_{<i}$ denote the prefix before token i, and let $\mathcal{S}(H_{<i})$ denote that prefix followed by a summary-like cue. Let $p^\star$ denote the target continuation distribution. We consider the setting in which the cue preserves task-relevant state, measured by $\Delta_i = \mathbb{D}_{\mathrm{KL}}(p^\star(\cdot \mid q, H_{<i}) \,\|\, p^\star(\cdot \mid q, \mathcal{S}(H_{<i})))$. ReSum branches from the same prefix with and without the cue. Let their expected returns be $R_{\mathrm{cue}}(i)$ and $R_{\mathrm{nocue}}(i)$, and define $G_i = R_{\mathrm{cue}}(i) - R_{\mathrm{nocue}}(i)$. This perspective characterizes continuation quality and state reorganization. Furthermore, we introduce the following proposition:

Proposition 1 (Cue-aware continuation advantage) *Assume that (i) the cue is δ-state-faithful, i.e., $\Delta_i \leq \delta$, and (ii) the reward is locally stable with respect to small changes in the continuation distribution. Then the cue-conditioned branch preserves expected return up to the distributional bound and improves it when it redirects redundant continuation behavior.*

The detailed derivations of the distributional bound are deferred to Appendix A. This conditional analysis explains how ReSum can improve credit assignment and yield more useful training signals under the same rollout budget.

Table 1: Performance comparison between ReSum and others on 6 challenging mathematical reasoning datasets, using Qwen2.5-Math-7B. The best results are in **bold**.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./Δ_{GRPO}
Base Model	12.19	4.79	35.23	48.60	15.07	16.33	22.04
GRPO	20.94	8.44	58.98	72.20	27.76	37.33	37.61
Dr. GRPO	21.04	8.23	58.59	72.05	28.58	35.89	37.40$_{-0.21}$
GPG	21.98	9.06	59.61	72.05	27.21	37.67	37.93$_{+0.32}$
DAPO	21.25	8.75	58.20	72.70	29.50	37.22	37.94$_{+0.33}$
GSPO	19.38	8.33	60.16	73.00	28.12	37.26	37.71$_{+0.10}$
GRPO-AD	21.56	9.48	59.06	73.25	29.14	37.07	38.26$_{+0.65}$
DGPO	23.85	10.21	61.02	74.25	31.07	38.33	39.79$_{+2.18}$
ReSum (ours)	**25.42**	**13.33**	**62.50**	**76.45**	**32.44**	**39.67**	**41.64**$_{+4.03}$

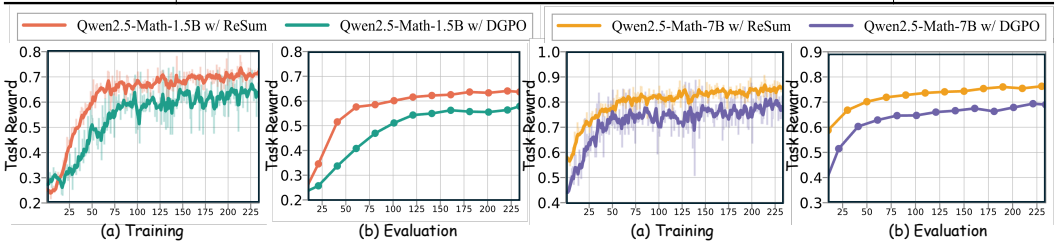

Figure 3: The training and evaluation dynamics of the task reward. We compare ReSum and DGPO with Qwen2.5-Math-1.5B **(left)** and Qwen2.5-Math-7B **(right)**.

4 Experiments

4.1 Experiment Setup

Datasets and Backbones. For training, we employ the MATH Hendrycks et al. (2021) dataset following DGPO Dai et al. (2026), which consists of problems with varying difficulty levels and requires long-chain reasoning. For evaluation, we select six commonly used mathematical benchmarks: AIME24, AIME25, AMC23, Minerva Lewkowycz et al. (2022), MATH500 Hendrycks et al. (2021), and Olympiad He et al. (2024). Results on AIME24, AIME25, and AMC23 are averaged over 32 runs, while results on the remaining benchmarks are averaged over 4 runs. We select Qwen2.5-Math-7B Yang et al. (2024b), Qwen2.5-Math-1.5B Yang et al. (2024b), Qwen2.5-3B Yang et al. (2024a), and DeepSeek-Math-7B Shao et al. (2024) as our backbone models. We also apply ReSum to the multimodal benchmark GEOQA-8K Chen et al. (2025) to further validate its generality, with Qwen2.5-3B-VL-Instruct Bai et al. (1) as the backbone.

Baselines. We select several competitive baselines for comprehensive comparison, including GRPO Guo et al. (2025), Dr. GRPO Liu et al. (2025), GPG Chu et al. (2026), DAPO Yu et al. (2025), GSPO Zheng et al. (2025), GRPO-AD Zhang & Zuo (2025), and DGPO Dai et al. (2026). To ensure a fair comparison, we make the following adjustments to the baselines: **(1)** we disable the resampling design in GPG and DAPO; **(2)** we retain only the advantage-reweighting component from GRPO-Lead Zhang & Zuo (2025), which we refer to as GRPO-AD; **(3)** we disable the Multi-Aspect Reformulation (MQR) in MathForge Dai et al. (2026) and treat only its core loss design, termed Difficulty-aware Group Policy Optimization (DGPO), as our baseline. Details of these baselines are provided in Appendix C.

Implementation Details. Following DGPO Dai et al. (2026), our codebase is built on Open-R1 Face (2025). We use 8 NVIDIA H20 GPUs for all experiments. During training, the response temperature is set to 1.0. During evaluation, the response temperature and the top-p value are set to 0.6 and 0.95, respectively. The maximum completion length is set to 4096 for all stages. Further implementation details are provided in Appendix D. The significance test of ReSum against DGPO is reported in Appendix E.

4.2 Main Results

Comparison with State-of-the-Art Methods. Table 1 summarizes the accuracy of ReSum and several competitive RLVR baselines on six mathematical benchmarks, all using Qwen2.5-Math-7B as

Table 2: Performance comparison between ReSum and others on 6 challenging mathematical reasoning datasets, using various backbones. The best results are in **bold**.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./Δ_{GRPO}
Qwen2.5-Math-1.5B	6.87	3.65	30.94	34.95	8.55	21.93	17.82
+GRPO	11.35	3.96	46.48	64.85	20.13	29.59	29.39
+DGPO	11.25	5.73	49.84	65.45	21.14	30.85	30.71+1.32
+ReSum (ours)	**15.96**	**6.83**	**51.33**	**67.12**	**24.58**	**32.59**	**33.07**+3.68
Qwen2.5-3B	2.81	0.73	22.66	48.65	13.69	19.37	17.99
+GRPO	5.31	1.56	33.28	63.35	22.89	26.41	25.47
+DGPO	6.98	1.56	36.56	65.80	25.28	26.96	27.19+1.72
+ReSum (ours)	**7.50**	**2.29**	**39.06**	**68.40**	**26.69**	**28.93**	**28.81**+3.34
DeepSeek-Math-7B	0.42	0.10	13.28	31.05	9.56	9.00	10.57
+GRPO	0.63	0.10	19.14	41.45	14.71	13.44	14.91
+DGPO	**1.98**	0.42	21.02	41.85	18.93	15.00	16.53+1.62
+ReSum (ours)	1.66	**0.96**	**22.40**	**43.10**	**20.22**	**15.56**	**17.32**+2.31

Table 3: Comparison of synergistic results of DGPO / ReSum with other policy optimization methods using Qwen2.5-Math-7B. The best results are in **bold**.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg. / Δ
GPG	21.98	9.06	59.61	72.05	27.21	37.67	37.93
+DGPO	21.77	10.00	60.00	73.45	30.06	38.26	38.92+0.99
+ReSum (ours)	**27.29**	**10.83**	**61.88**	**74.75**	**30.76**	**39.07**	**40.76**+2.83
DAPO	21.25	8.75	58.20	72.70	29.50	37.22	37.94
+DGPO	24.48	9.79	58.75	74.90	31.99	**39.56**	39.91+1.97
+ReSum (ours)	**25.67**	**10.67**	**62.18**	**75.30**	**32.11**	39.34	**40.88**+2.94
GSPO	19.38	8.33	60.16	73.00	28.12	37.26	37.71
+DGPO	23.33	**10.00**	59.14	74.15	30.88	38.41	39.32+1.61
+ReSum (ours)	**25.10**	**10.00**	**61.04**	**74.50**	**31.38**	**38.98**	**40.17**+2.46

the backbone. ReSum achieves the highest scores on all datasets, reaching an average of 41.64% and performance gain of 4.03% over standard GRPO. This trend is particularly evident on the competitive datasets: For example on AIME24, ReSum outperforms DGPO by 1.57%, and on AIME25 ReSum reaches 13.33% compared to 10.21%. While DGPO refines outcome-level reweighting, ReSum additionally supplies state-local supervision through cue-conditioned branches. The improvements confirm the value of our design on the evaluated mathematical reasoning settings.

Application to Different Backbones. We further examine whether the gains observed with ReSum transfer across model sizes and families. Table 2 reports results with Qwen2.5-Math-1.5B, Qwen2.5-3B, and DeepSeek-Math-7B. ReSum improves the average score beyond both GRPO and DGPO for each evaluated backbone. For the smallest Qwen2.5-Math-1.5B, ReSum outperforms DGPO by 2.36%; for Qwen2.5-3B, the average score increases from 25.47% with GRPO to 28.81% with ReSum. DeepSeek-Math-7B follows the same pattern, reaching 17.32% with ReSum compared with 16.53% for DGPO. The gains across scales indicate that cue-conditioned rollout construction supplies useful learning signals to both compact and larger reasoning models. These results support a scalable, model-agnostic rollout-and-advantage design that organizes reasoning internally.

Compatibility with Existing Methods. Table 3 presents the compatibility results of ReSum with GPG, DAPO, and GSPO. In every combination, ReSum yields a larger average improvement than augmenting the same base method with DGPO. Starting from 37.93% for GPG, adding DGPO lifts the score to 38.92%, whereas adding ReSum achieves 40.76% (+2.83%). The pattern holds for DAPO with a +2.94% gain and for GSPO with a +2.46% gain. These results show that ReSum's cue-conditioned rollout construction and dual-reference normalization can be combined with several policy-optimization objectives in the evaluated setting.

Application to Multi-modal Domain. To study whether ReSum's benefits extend beyond text-only mathematics, we evaluate on the visual question-answering dataset GEOQA-8K with Qwen2.5-3B-VL-Instruct. As shown in Table 4, ReSum reaches an average score of 62.04%, outperforming GRPO by 5.61%, while other methods such as DGPO and GPG deliver only modest improvements.

Table 4: Performance comparison between ReSum and others on multimodal dataset GEOQA-8K with Qwen2.5-3B-VL-Instruct. The best results are in **bold**.

Method	Base Model	GRPO	Dr. GRPO	GPG	DAPO	GSPO	GRPO-AD	DGPO	**ReSum**
Avg. / Δ_{GRPO}	39.79	57.43	57.96+0.53	59.02+1.59	59.02+1.59	57.16-0.27	58.09+0.66	59.95+2.52	**62.04**+5.61

Table 5: Experiment results under different rollout budgets $B = T \times J$ and configurations of tree structures with Qwen2.5-3B. The best results are in **bold**.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./Δ_{GRPO}
Qwen2.5-3B	2.81	0.73	22.66	48.65	13.69	19.37	17.99
GRPO	5.31	1.56	33.28	63.35	22.89	26.41	25.47
Rollout Budget $B = T \times J = 2$							
ReSum ($T = 2, J = 1$)	4.67	1.56	33.06	62.73	22.81	24.96	24.97-0.50
Rollout Budget $B = T \times J = 4$							
ReSum ($T = 2, J = 2$)	5.57	1.86	35.97	66.90	23.81	26.41	26.75+1.28
ReSum ($T = 4, J = 1$)	6.13	2.07	37.50	65.76	25.53	27.79	27.46+1.99
Rollout Budget $B = T \times J = 16$							
ReSum ($T = 2, J = 8$)	6.67	2.10	37.10	67.97	27.05	26.97	27.98+2.51
ReSum ($T = 8, J = 2$)	6.60	**2.84**	38.11	68.17	26.33	**29.25**	28.55+3.08
ReSum ($T = 4, J = 4$)	**7.50**	2.29	**39.06**	**68.40**	**26.69**	28.93	**28.81**+3.34

Although the multimodal setting introduces additional challenges from visual encoding, the model still benefits from ReSum's cue-conditioned reasoning organization. This result extends the evidence from text-only mathematics to multimodal geometry and demonstrates ReSum's applicability to visually grounded verifiable reasoning. ReSum operates within the policy model and uses no external compression module or hand-crafted inference-time summary prompt.

4.3 Further Analysis

Branching Configuration Analysis. We examine how the rollout tree structure affects ReSum under fixed rollout budgets in Table 5, where the rollout budget is defined as $B = TJ$. With a rollout budget of $B = 4$, ReSum with $T = 4, J = 1$ reaches 27.46%, outperforming the configuration with $T = 2, J = 2$ by 0.71%. This result suggests that diverse initial paths provide stronger supervision than deeper branching within fewer trees. At a rollout budget of $B = 16$, ReSum with $T = 8, J = 2$ achieves 28.55%, surpassing the 27.98% obtained by $T = 2, J = 8$. Meanwhile, we find the balanced $T = 4, J = 4$ configuration yields the highest score among all settings. This indicates that combining trajectory diversity with a moderate amount of summarization branching yields the best learning signal. At the smallest rollout budget of $B = 2$, samples are too scarce for the branch structure to realize its full benefit; the stronger configurations show that ReSum gains value from both trajectory diversity and adaptive branching.

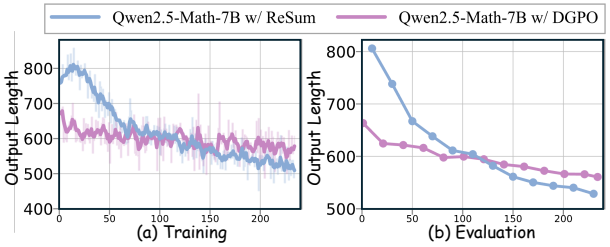

Figure 4: Training and evaluation dynamics of output length with Qwen2.5-Math-7B.

Learning Dynamics. Figure 3 shows the training and evaluation dynamics of the task reward for ReSum and DGPO, tested on Qwen2.5-Math-1.5B and 7B backbones. For both model scales, ReSum consistently outperforms DGPO across training and inference. While the training-stage improvement may partly reflect latent overfitting effects, the consistently higher upper bound (+0.08 approximately) and faster convergence during inference confirm the effectiveness of ReSum. Its summarization-driven design provides robust and generalizable reasoning enhancements.

We also show the average rollout length during training and evaluation with Qwen2.5-Math-7B in Figure 4. Although ReSum initially exhibits an increased rollout length, it maintains a length reduction throughout the remainder of training in both phases, eventually reducing the average rollout length by approximately 18.6% relative to DGPO. Together with the reward gains, this

Table 6: Ablation results of Artifact Points (APs), Natural Points (NPs) and our SGPO. Notably w/o SGPO means to disable the dual-group mechanism and use GRPO. The best results are in **bold**.

Models	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg./Δ_{DGPO}
Qwen2.5-Math-1.5B	6.87	3.65	30.94	34.95	8.55	21.93	17.82
GRPO	11.35	3.96	46.48	64.85	20.13	29.59	29.39
ReSum	**15.96**	**6.83**	**51.33**	**67.12**	**24.58**	**32.59**	**33.07**
w/o APs	12.50	5.10	50.41	65.40	22.43	31.41	31.21
w/o NPs	11.67	4.27	47.26	65.55	21.97	29.93	30.11
w/o SGPO	14.64	6.52	50.59	66.13	24.30	32.15	32.39
Qwen2.5-Math-7B	12.19	4.79	35.23	48.60	15.07	16.33	22.04
GRPO	20.94	8.44	58.98	72.20	27.76	37.33	37.61
ReSum	**25.42**	13.33	**62.50**	**76.45**	**32.44**	**39.67**	**41.64**
w/o APs	24.46	**13.48**	60.39	76.00	31.03	39.01	40.73
w/o NPs	20.32	11.97	58.73	73.02	31.20	38.39	38.94
w/o SGPO	25.11	10.56	60.91	75.59	32.02	38.66	40.48

establishes a favorable accuracy–length trade-off: ReSum retains the full reasoning history while guiding subsequent generation toward concise, task-directed trajectories.

Ablation of Components. Table 6 reports the ablation results. Removing NPs causes the largest degradation across both model scales: the average drops from 33.07% to 30.11% on the 1.5B model and from 41.64% to 38.94% on the 7B model. Excluding APs also reduces performance to 31.21% and 40.73%, respectively. When SGPO is replaced with vanilla GRPO, performance falls to 32.39% and 40.48%. These results quantify complementary contributions from natural cue events, artifact branching, and dual-reference credit assignment. Together, they show that ReSum learns from a broad family of summary-like reasoning-control events, with self-summarization providing the central consolidation behavior.

Analysis of Accuracy across Problem Difficulty Levels. Figure 5 compares ReSum with the DGPO baseline across the five MATH500 difficulty levels. ReSum improves accuracy at every level, with a 3% gain on Level 1 and the largest gain, 6%, on Level 5. This pattern highlights ReSum's strength on problems with longer logical dependencies, where early detours and repeated verification can accumulate across a rollout. DGPO allocates optimization effort according to sample difficulty; ReSum complements this with state-local rollout branches that identify and reinforce effective consolidation or repair events. By reorganizing the active reasoning state at these control points, ReSum helps the policy carry forward task-relevant intermediate results and continue along a more direct solution trajectory. The growing gain on the most difficult problems therefore provides positive evidence that cue-conditioned credit assignment is particularly valuable when reasoning traces become long and structurally demanding.

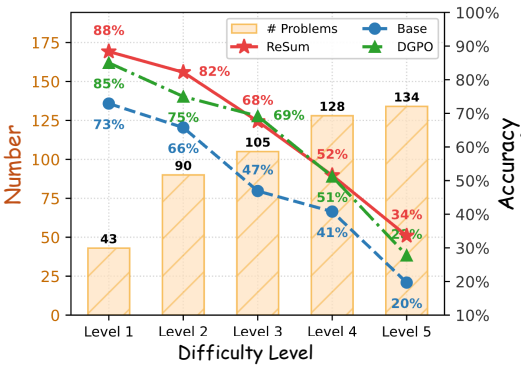

Figure 5: Analysis of accuracy across different difficulty levels of MATH500.

5 Conclusion

We introduced ReSum, a reinforcement learning framework for organizing long reasoning with self-summarization and related reasoning-control cues. By expanding rollouts into a tree around natural and artificial cues and applying a dual-reference advantage, ReSum converts trajectory-level rewards into state-local process supervision. Experiments across mathematical benchmarks, multiple backbones, and multimodal geometry show consistent accuracy gains of 4% and an 18.6% reduction in rollout length. These results establish an effective, externally module-free approach to improving the accuracy–length trade-off in verifiable reasoning.

AI USE STATEMENT

In this work, we used generative AI tools only to edit the paper to improve grammar, wording, and readability. We have not used generative AI tools for any required-disclosure research task, including generating synthetic data, developing theoretical models or conceptual frameworks, formulating mathematical claims, designing or evaluating the methodology or experiments, implementing the method, interpreting results, or performing data analysis. The other recommended-disclosure tasks are not applicable to this work. We have reviewed all AI-assisted edits and take responsibility for the final content of this submission, including all claims, equations, experimental results, figures, and citations.

ETHICS STATEMENT

This work studies reinforcement learning methods for improving the efficiency and organization of long-horizon language-model reasoning. The evaluated tasks use publicly available mathematical and multimodal benchmarks with verifiable answers and do not involve human subjects, personal data, or sensitive user information. ReSum can still produce incorrect reasoning or answers, so outputs should be independently checked in educational, scientific, and other high-stakes applications. We are not aware of ethical risks specific to the proposed training procedure beyond the general risks associated with deploying large language models.

REPRODUCIBILITY STATEMENT

We provide the algorithm, reward definition, dual-reference advantage, cue pool, training configuration, baseline settings, rollout-budget analysis, ablations, efficiency measurements, and evaluation protocol in the main paper and appendices. Experiments use public MATH, AIME24, AIME25, AMC23, MATH500, Minerva, OlympiadBench, and GEOQA-8K benchmarks. The implementation is based on Open-R1 and uses the backbone models and hardware configuration described in Appendix D; all reported results use the evaluation settings specified there. These details support faithful reimplementation and direct comparison with the reported baselines.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 4(5), 1.

Shelly Bensal, Umar Jamil, Christopher Bryant, Melisa Russak, Kiran Kamble, Dmytro Mozolevskyi, Muayad Ali, and Waseem AlShikh. Reflect, retry, reward: Self-improving llms via reinforcement learning. *arXiv preprint arXiv:2505.24726*, 2025.

Gabriele Berton, Jayakrishnan Unnikrishnan, Son Tran, and Mubarak Shah. Compllm: Compression for long context q&a. *arXiv preprint arXiv:2509.19228*, 2025.

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric Xing, and Liang Lin. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 513–523, 2021.

Liang Chen, Lei Li, Haozhe Zhao, and Yifan Song. Vinci. r1-v: Reinforcing super generalization ability in vision-language models with less than $3, 2025.

Longze Chen, Ziqiang Liu, Wanwei He, Yinhe Zheng, Hao Sun, Yunshui Li, Run Luo, and Min Yang. Long context is not long at all: A prospector of long-dependency data for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8222–8234, 2024a.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024b.

Xiangxiang Chu, Hailang Huang, Xiao Zhang, Fei Wei, and Yong Wang. GPG: A simple and strong reinforcement learning baseline for model reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=inccdtfx8x.

Yanqi Dai, Yuxiang Ji, Xiao Zhang, Yong Wang, Xiangxiang Chu, and Zhiwu Lu. Harder is better: Boosting mathematical reasoning via difficulty-aware GRPO and multi-aspect question reformulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=nfURupkdRJ.

Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. Pretraining context compressor for large language models with embedding-based memory. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28715–28732, 2025.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, et al. Agentic entropy-balanced policy optimization. *arXiv preprint arXiv:2510.14545*, 2025a.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025b.

Zican Dong, Junyi Li, Jinhao Jiang, Mingyu Xu, Wayne Xin Zhao, Bingning Wang, and Weipeng Chen. Longred: Mitigating short-text degradation of long-context large language models via restoration distillation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10687–10707, 2025c.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, january 2025. *URL https://github.com/huggingface/open-r1*, 7, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous rl. *arXiv preprint arXiv:2508.07976*, 2025.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Ruishan Guo, Yibing Liu, Guoxin Ma, Yan Wang, Yueyang Zhang, Long Xia, Kecheng Chen, Zhiyuan Sun, and Daiting Shi. When less is more: The llm scaling paradox in context compression. *arXiv preprint arXiv:2602.09789*, 2026.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, Sebastian Zhao, June Paik, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Squeezed attention: Accelerating long context length llm inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32631–32652, 2025.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. Treerl: Llm reinforcement learning with on-policy tree search. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12355–12369, 2025.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for llm agent reinforcement learning. *arXiv preprint arXiv:2509.21240*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. An empirical survey on long document summarization: Datasets, models, and metrics. *ACM computing surveys*, 55(8):1–35, 2022.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*, 2024.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Chengzhengxu Li, Xiaoming Liu, Zhaohan Zhang, Shaochu Zhang, Shengchao Liu, Guoxin Ma, Yu Lan, and Chao Shen. Upfront chain-of-thought: A cooperative framework for chain-of-thought compression. *arXiv preprint arXiv:2510.08647*, 2025a.

Jiaqi Li, Xinyi Dong, Yang Liu, Zhizhuo Yang, Quansen Wang, Xiaobo Wang, Song-Chun Zhu, Zixia Jia, and Zilong Zheng. Reflectevo: Improving meta introspection of small llms by learning self-reflection. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 16948–16966, 2025b.

Renda Li, Hailang Huang, Fei Wei, Feng Xiong, Yong Wang, and Xiangxiang Chu. Adacurl: Adaptive curriculum reinforcement learning with invalid sample mitigation and historical revisiting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23123–23131, 2026.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025c.

Zhiwei Li, Yong Hu, and Wenqing Wang. Encouraging good processes without the need for good answers: Reinforcement learning for llm agent planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1654–1666, 2025d.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

Zhan Ling, Kang Liu, Kai Yan, Yifan Yang, Weijian Lin, Ting-Han Fan, Lingfeng Shen, Zhengyin Du, and Jiecao Chen. Longreason: A synthetic long-context reasoning benchmark via context expansion. *arXiv preprint arXiv:2501.15089*, 2025.

Shaoteng Liu, Haoqi Yuan, Minda Hu, Yanwei Li, Yukang Chen, Shu Liu, Zongqing Lu, and Jiaya Jia. Rl-gpt: Integrating reinforcement learning and code-as-policy. *Advances in Neural Information Processing Systems*, 37:28430–28459, 2024a.

Shuaitong Liu, Renjue Li, Lijia Yu, Lijun Zhang, Zhiming Liu, and Gaojie Jin. Badthink: Triggered overthinking attacks on chain-of-thought reasoning in large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32141–32149, 2026.

Xinyu Liu, Runsong Zhao, Pengcheng Huang, Chunyang Xiao, Bei Li, Jingang Wang, Tong Xiao, and Jingbo Zhu. Forgetting curve: A reliable method for evaluating memorization capability for long-context models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4667–4682, 2024b.

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

Yongchao Long, Xian Wu, Yingying Zhang, Xianbin Wen, Yuxi Zhou, and Shenda Hong. Copy-paste to mitigate large language model hallucinations. *arXiv preprint arXiv:2510.00508*, 2025.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7052–7063, 2021.

Zhengxi Lu, Yuxiang Chai, Yaxuan Guo, Xi Yin, Liang Liu, Hao Wang, Han Xiao, Shuai Ren, Pengxiang Zhao, Guangyi Liu, et al. Ui-r1: Enhancing efficient action prediction of gui agents by reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17608–17616, 2026.

Run Luo, Lu Wang, Wanwei He, Longze Chen, Jiaming Li, and Xiaobo Xia. Gui-r1: A generalist r1-style vision-language action model for gui agents. *arXiv preprint arXiv:2504.10458*, 2025.

Chiyu Ma, Shuo Yang, Kexin Huang, Jinda Lu, Haoming Meng, Shangshang Wang, Bolin Ding, Soroush Vosoughi, Guoyin Wang, and Jingren Zhou. Fipo: Eliciting deep reasoning with future-kl influenced policy optimization. *arXiv preprint arXiv:2603.19835*, 2026a.

Da Ma, Lu Chen, Situo Zhang, Yuxun Miao, Su Zhu, Zhi Chen, Hongshen Xu, Hanqi Li, Shuai Fan, Lei Pan, et al. Compressing kv cache for long-context llm inference with inter-layer attention similarity. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 16407–16411. IEEE, 2026b.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026c.

Xinji Mai, Haotian Xu, Zhong-Zhi Li, Weinong Wang, Jian Hu, Yingying Zhang, Wenqiang Zhang, et al. Agent rl scaling law: Agent rl with spontaneous code execution for mathematical problem solving. *arXiv preprint arXiv:2505.07773*, 2025.

Yifei Ming, Senthil Purushwalkam, Shrey Pandit, Zixuan Ke, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. Faitheval: Can your language model stay faithful to context, even if" the moon is made of marshmallows". *arXiv preprint arXiv:2410.03727*, 2024.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1100–1126, 2024.

Aleksandar Petrov, Mark Sandler, Andrey Zhmoginov, Nolan Miller, and Max Vladymyrov. Long context in-context compression by getting to the gist of gisting. *arXiv preprint arXiv:2504.08934*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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

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

Yucheng Shi, Wenhao Yu, Zaitang Li, Yonglin Wang, Hongming Zhang, Ninghao Liu, Haitao Mi, and Dong Yu. Mobilegui-rl: Advancing mobile gui agent through reinforcement learning in online environment. *arXiv preprint arXiv:2507.05720*, 2025.

Gaurav Srivastava, Aafiya Hussain, Sriram Srinivasan, and Xuan Wang. Do llms overthink basic math reasoning? benchmarking the accuracy-efficiency tradeoff in language models. *arXiv preprint arXiv:2507.04023*, 2025.

Yuxuan Tian, Zihan Wang, Yebo Peng, Aomufei Yuan, Zhiming Wang, Bairen Yi, Xin Liu, Yong Cui, and Tong Yang. Keepkv: Achieving periodic lossless kv cache compression for efficient llm inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33259–33267, 2026.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 2609–2634, 2023.

Pinzheng Wang, Shuli Xu, Juntao Li, Yu Luo, Dong Li, Jianye Hao, and Min Zhang. **Re**^{2}: Unlocking llm reasoning via reinforcement learning with re-solving. *arXiv preprint arXiv:2603.07197*, 2026.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*, 2025.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025a.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. *arXiv preprint arXiv:2502.07266*, 2025b.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning. *arXiv preprint arXiv:2502.19613*, 2025.

A Yang, B Yang, B Zhang, B Hui, B Zheng, B Yu, C Li, D Liu, F Huang, H Wei, et al. Qwen2. 5 technical report. Technical report, 2024a.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024b.

Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3258–3270, 2024c.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Xinbin Yuan, Jian Zhang, Kaixin Li, Zhuoxuan Cai, Lujian Yao, Jie Chen, Enguang Wang, Qibin Hou, Jinwei Chen, Peng-Tao Jiang, et al. Enhancing visual grounding for gui agents via self-evolutionary reinforcement learning. *arXiv preprint arXiv:2505.12370*, 2025.

Jixiao Zhang and Chunsheng Zuo. Grpo-lead: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5642–5665, 2025.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025a.

Wenlin Zhang, Xiangyang Li, Kuicai Dong, Yichao Wang, Pengyue Jia, Xiaopeng Li, Yingyi Zhang, Derong Xu, Zhaocheng Du, Huifeng Guo, et al. Process vs. outcome reward: Which is better for agentic rag reinforcement learning. *arXiv preprint arXiv:2505.14069*, 2025b.

Xinliang Frederick Zhang, Anhad Mohananey, Alexandra Chronopoulou, Pinelopi Papalampidi, Somit Gupta, Tsendsuren Munkhdalai, Lu Wang, and Shyam Upadhyay. Do llms really need 10+ thoughts for" find the time 1000 days later"? towards structural understanding of llm overthinking. *arXiv preprint arXiv:2510.07880*, 2025c.

Xutong Zhao, Tengyu Xu, Xuewei Wang, Zhengxing Chen, Di Jin, Liang Tan, Zishun Yu, Zhuokai Zhao, Yun He, Sinong Wang, et al. Boosting llm reasoning via spontaneous self-correction. *arXiv preprint arXiv:2506.06923*, 2025a.

Yubao Zhao, Weiquan Huang, Sudong Wang, Ruochen Zhao, Chen Chen, Yao Shu, and Chengwei Qin. Training multi-turn search agent via contrastive dynamic branch sampling. *arXiv preprint arXiv:2602.03719*, 2026.

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

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.

Appendix
ReSum: Synergizing LLM Reasoning and Summarization with Reinforcement Learning

The Appendix of the paper is organized as:

- Appendix A: We give the proof of the proposition.
- Appendix B: We introduce combinations of ReSum and other methods.
- Appendix C: We report datasets and baselines.
- Appendix D: We report implementation details.
- Appendix E: We report the further analysis.
- Appendix F: We report the algorithm.
- Appendix G: We report the limitations.

A Proof of Proposition 1

We prove the conditional statement in two steps: first we show that a δ-state-faithful cue induces only a small continuation-distribution shift; then local reward stability bounds the expected-return difference between cue-conditioned and cue-free branches.

According to the assumptions in the proposition, the cue operator $\mathcal{S}$ is δ-state-faithful, i.e., $\Delta_i = \mathbb{D}_{\mathrm{KL}}(p^\star(\cdot \mid q, H_{<i}) \,\|\, p^\star(\cdot \mid q, \mathcal{S}(H_{<i}))) \le \delta$. Let $\pi_{\mathrm{full}}(\cdot) = p^\star(\cdot \mid q, H_{<i})$ and $\pi_{\mathrm{cue}}(\cdot) = p^\star(\cdot \mid q, \mathcal{S}(H_{<i}))$. Then by Pinsker's inequality,

$$D_{\mathrm{TV}}(\pi_{\mathrm{full}}, \pi_{\mathrm{cue}}) \le \sqrt{\frac{1}{2}\mathbb{D}_{\mathrm{KL}}(\pi_{\mathrm{full}} \| \pi_{\mathrm{cue}})} \le \sqrt{\frac{\delta}{2}}.$$

So the cue-conditioned branch remains close to the full-prefix branch in total variation distance. Then, we assume the reward is locally stable with respect to the continuation distribution, meaning that there exists a constant $L > 0$ such that for any two continuation policies π_1, π_2,

$$|\mathcal{R}(\pi_1) - \mathcal{R}(\pi_2)| \le L \cdot D_{\mathrm{TV}}(\pi_1, \pi_2).$$

Applying this to π_{full} and π_{cue}, we obtain

$$|\mathcal{R}(\pi_{\mathrm{full}}) - \mathcal{R}(\pi_{\mathrm{cue}})| \le L \cdot D_{\mathrm{TV}}(\pi_{\mathrm{full}}, \pi_{\mathrm{cue}}) \le L\sqrt{\frac{\delta}{2}},$$

which implies

$$\mathcal{R}(\pi_{\mathrm{cue}}) \ge \mathcal{R}(\pi_{\mathrm{full}}) - L\sqrt{\frac{\delta}{2}}.$$

Now we have $R_{\mathrm{nocue}}(i) = \mathcal{R}(\pi_{\mathrm{full}})$ and $R_{\mathrm{cue}}(i) = \mathcal{R}(\pi_{\mathrm{cue}})$, which yields

$$G_i = R_{\mathrm{cue}}(i) - R_{\mathrm{nocue}}(i) \ge -L\sqrt{\frac{\delta}{2}}.$$

This proves the bounded-degradation part of the proposition. Strict improvement arises when the cue changes the continuation toward a higher-return reasoning trajectory. ReSum estimates this condition empirically through its contrastive branches while retaining the preceding context and evaluating the full family of detected reasoning-control events.

In summary, a δ-state-faithful cue induces only a small distribution shift, and local reward stability bounds its return difference. When it redirects a redundant continuation toward a better trajectory, the cue-conditioned branch can outperform the cue-free branch. This establishes the conditional proposition.

B Details of Combining ReSum and Other Methods

This section introduces how ReSum is integrated with other policy optimization methods. Since ReSum primarily modifies the advantage estimation and introduces a tree-structured rollout, the combination is straightforward: we keep the original loss formulation of each baseline method and replace their advantage terms with our summarization-aware group-relative advantage defined in Eq. equation 3. The following subsections provide the exact optimization objectives for each combination.

B.1 Combination with GPG

GPG removes the clipping and KL-regularization terms from GRPO and directly optimizes the policy gradient. When combined with ReSum, we retain GPG's unclipped objective while replacing its original group-relative advantage with our summarization-aware dual-level advantage. For a batch of queries $\{q_s\}_{s=1}^{S}$ and their corresponding rollout trees $\{\mathcal{T}_{s,t}\}_{t=1}^{T}$, the optimization objective is:

$$\begin{aligned}\mathcal{J}_{\text{GPG+ReSum}}(\theta) =& \mathbb{E}_{\{q_s\}_{s=1}^{S}\sim\mathcal{D},\{\mathcal{T}_{s,t}\}_{t=1}^{T}\sim\pi_{\theta_{\text{old}}}(\cdot|q_s)}\Bigg[\frac{1}{\sum_{s=1}^{S}\sum_{t=1}^{T}|\mathcal{T}_{s,t}|} \\ & \sum_{s=1}^{S}\sum_{t=1}^{T}\sum_{e=1}^{|\mathcal{T}_{s,t}|}\frac{1}{|\tau_{s,t,e}|}\sum_{i=1}^{|\tau_{s,t,e}|}\Big[-\log\pi_\theta(\tau_{s,t,e,i}\mid q_s,\tau_{s,t,e,<i})\,\hat{A}_{s,t,e,i}\Big]\Bigg],\end{aligned} \tag{5}$$

where $\hat{A}_{s,t,e,i}$ is the summarization-aware group-relative advantage computed by Eq. equation 3, combining both $\hat{A}^{\text{sum}}$ and $\hat{A}^{\text{nosum}}$ advantages over the rollout tree.

B.2 Combination with DAPO

DAPO employs a PPO-style clipped objective with asymmetric clipping bounds and a composite reward signal that includes both accuracy and length penalty (i.e., $r_{s,t,e} = r_{\text{acc},s,t,e} + r_{\text{length},s,t,e}$). When integrating ReSum with DAPO, we preserve DAPO's clipped objective, token-level loss computation, and overlong reward shaping, while substituting its advantage estimation with our summarization-aware group-relative advantage. Notably, the composite reward is used for advantage calculation in Eq. equation 2 and Eq. equation 3. The optimization objective is:

$$\begin{aligned}\mathcal{J}_{\text{DAPO+ReSum}}(\theta) =& \mathbb{E}_{\{q_s\}_{s=1}^{S}\sim\mathcal{D},\{\mathcal{T}_{s,t}\}_{t=1}^{T}\sim\pi_{\theta_{\text{old}}}(\cdot|q_s)}\Bigg[\frac{1}{\sum_{s=1}^{S}\sum_{t=1}^{T}|\mathcal{T}_{s,t}|} \\ & \sum_{s=1}^{S}\sum_{t=1}^{T}\sum_{e=1}^{|\mathcal{T}_{s,t}|}\frac{1}{|\tau_{s,t,e}|}\sum_{i=1}^{|\tau_{s,t,e}|}\min\Big(\rho_{s,t,e,i}(\theta)\hat{A}_{s,t,e,i}, \\ & \text{clip}(\rho_{s,t,e,i}(\theta),1-\epsilon_{\text{low}},1+\epsilon_{\text{high}})\hat{A}_{s,t,e,i}\Big)\Bigg],\end{aligned} \tag{6}$$

where $\rho_{s,t,e,i}(\theta)$ is the token-level importance sampling ratio defined in Eq. equation 4, and $\hat{A}_{s,t,e,i}$ is our summarization-aware group-relative advantage. The asymmetric clipping bounds ϵ_{low} and ϵ_{high} follow the original DAPO design.

B.3 Combination with GSPO

GSPO operates at the sequence level, computing a sequence-level importance sampling ratio and applying PPO-style clipping over entire responses. To integrate ReSum with GSPO, we retain GSPO's sequence-level formulation and replace its group-relative advantage with our summarization-aware dual-level advantage. The optimization objective is:

$$\mathcal{J}_{\text{GSPO+ReSum}}(\theta) = \mathbb{E}_{\{q_s\}_{s=1}^{S}\sim\mathcal{D},\{\mathcal{T}_{s,t}\}_{t=1}^{T}\sim\pi_{\theta_{\text{old}}}(\cdot|q_s)} \left[\frac{1}{\sum_{s=1}^{S}\sum_{t=1}^{T}|\mathcal{T}_{s,t}|} \sum_{s=1}^{S}\sum_{t=1}^{T}\sum_{e=1}^{|\mathcal{T}_{s,t}|} \min\left(S_{s,t,e}(\theta)\hat{A}_{s,t,e}, \text{clip}(S_{s,t,e}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{s,t,e} \right) \right], \tag{7}$$

$$S_{s,t,e}(\theta) = \left(\prod_{i=1}^{|\tau_{s,t,e}|} \rho_{s,t,e,i}(\theta) \right)^{1/|\tau_{s,t,e}|}.$$

where $S_{s,t,e}(\theta)$ is the sequence-level importance sampling ratio for the e-th rollout in the t-th tree. Here $\hat{A}_{s,t,e}$ denotes the sequence-level summarization-aware advantage obtained by averaging our token-level dual-group advantage $\hat{A}_{s,t,e,i}$ over the entire rollout. This combination allows GSPO to benefit from ReSum's fine-grained process supervision while maintaining its sequence-level update granularity.

In all the above combinations, the total rollout budget per query remains identical to the original baselines. The tree-structured expansion of ReSum is applied before advantage computation, and the partition into G_{cue} and G_{nocue} follows the procedure described in Section 3.3.

C Datasets and Baselines

The datasets are introduced as follows:

MATH. The MATH dataset Hendrycks et al. (2021) provides 12,500 competition-level problems drawn from high school contests such as AMC-10, AMC-12, and AIME, each labeled with difficulty levels 1-5, and spanning seven subject areas including algebra, geometry, and number theory. Following DGPO Dai et al. (2026), we use the 7,500-problem training split for RLVR training.

AIME24 AIME24 is collected from the two 2024 American Invitational Mathematics Examination contests. It contains 30 open-ended problems with integer answers between 0 and 999. AIME24 has become a widely adopted stress test for advanced mathematical reasoning in LLMs, and we report scores averaged over 32 evaluation runs to ensure reliable comparisons.

AIME25. Similarly, AIME25 is collected from the two 2025 American Invitational Mathematics Examination contests. It features 30 integer-answer problems from AIME-I and AIME-II. Because it is the most recent release, AIME25 serves as a clean evaluation set that guards against potential data contamination in older benchmarks, making it a stringent probe of genuine problem-solving ability. We also average results over 32 runs for this benchmark.

AMC23 AMC23 offers 46 multiple-choice problems covering algebra, geometry, number theory, and combinatorics at the high school level. As the qualifying stage for the AIME, AMC problems are somewhat more accessible than AIME tasks but still require solid mathematical reasoning. We evaluate on AMC23 with 32-run averaging.

MATH500. MATH500 Lightman et al. (2023) is a 500-problem subset of the MATH test set, uniformly sampled to retain the original distribution across difficulty levels and subjects. Introduced by OpenAI for process-supervised reasoning studies, it has become the standard in-distribution benchmark for models trained on MATH. Evaluation follows the convention of exact match on answers extracted from `\boxed{}`, and we report averages over 4 runs.

Minerva. Minerva Lewkowycz et al. (2022) draws its 272 problems primarily from OpenCourseWare materials, spanning university and graduate-level STEM subjects such as astronomy, solid-state chemistry, differential equations, and special relativity. Rather than competition mathematics, Minerva tests the model's ability to apply quantitative reasoning in scientific contexts, complementing the competition-focused benchmarks with a broader probe of scientific problem solving. Results are averaged over 4 runs.

OlympiadBench. OlympiadBench He et al. (2024) assembles 8,476 expert-level problems from international and Chinese Olympiad competitions in mathematics and physics, together with items from

the Chinese college entrance exam (Gaokao). It offers bilingual (English/Chinese) and multimodal problem formats with detailed step-by-step annotations, making it one of the most demanding tests of cross-lingual, multi-domain mathematical reasoning. We report 4-run averages on this benchmark.

GEOQA-8K. GEOQA-8K Chen et al. (2025) is an extended version of the GeoQA geometric question answering corpus Chen et al. (2021). It contains 8,000 problems that pair textual descriptions with visual diagrams, requiring joint reasoning over language and geometry. Each problem includes an annotated solving program that makes the reasoning chain explicit. We use GEOQA-8K to test whether ReSum's self-summarization gains transfer to multimodal settings, where the additional visual modality increases context complexity. Results are averaged over 4 evaluation runs.

The baselines are introduced as follows:

GRPO Guo et al. (2025) is a critic-free reinforcement learning algorithm that eliminates the value network by sampling a group of candidate responses per prompt and computing normalized group-relative advantages as training signals. Originally developed for DeepSeek's mathematical reasoning pipeline, GRPO significantly reduces memory and computational costs compared to PPO while maintaining competitive training stability.

Dr. GRPO Liu et al. (2025) critically examines the statistical biases embedded in the standard GRPO objective, identifying that the per-sample normalization scheme introduces systematic gradient distortions that can impede learning. It proposes a debiased reformulation of the advantage estimator that corrects these biases, offering a theoretically grounded perspective on R1-Zero-like training and improving optimization reliability without additional computational overhead.

GPG Chu et al. (2026) revisits the classical policy gradient mechanism and proposes a minimalist reinforcement learning approach called Group Policy Gradient that directly optimizes the original RL objective without relying on surrogate loss functions or clipping. By stripping away the approximations introduced by PPO-style methods, GPG demonstrates that a clean, unmodified policy gradient formulation can serve as a surprisingly strong and competitive baseline for LLM reasoning tasks.

DAPO Yu et al. (2025) is an open-source, industrial-scale RL training system for LLMs that introduces four targeted algorithmic improvements over GRPO: Clip-Higher for asymmetric clipping to encourage output diversity, dynamic sampling to filter uninformative training instances, token-level policy gradient loss for more granular optimization, and overlong reward shaping to penalize excessively verbose responses. Together, these techniques address the entropy collapse and reward hacking issues that commonly arise in large-scale chain-of-thought RL training.

GSPO Zheng et al. (2025) replaces the token-level importance ratio used in PPO and GRPO with a sequence-level importance ratio, arguing that sequence-level clipping provides a more principled and stable trust region for policy updates in language model training. Developed by the Qwen team, GSPO demonstrates notably superior scalability and training stability compared to token-level methods, and serves as the core RL algorithm underlying the Qwen3 model series.

DGPO Dai et al. (2026) proposes a difficulty-aware extension of GRPO that dynamically adjusts the optimization process based on the estimated difficulty of each training problem. By incorporating difficulty signals into the advantage computation and combining them with a multi-aspect question reformulation strategy, DGPO steers the policy to allocate greater learning effort toward harder problems, thereby improving the efficiency and effectiveness of RL-based mathematical reasoning training.

D Implementation Details

To avoid overfitting to a fixed phrase, we maintain a pool of summary-like reasoning-control cues. In NP detection, a match to any pool element defines a natural cue event. In AP injection, we randomly select a pool element and append it to the truncated prefix. Matching is case-insensitive. The pool combines explicit consolidation phrases with a limited set of reflection, verification, repair, and strategy-switching cues, operationalizing a broad family of reasoning-control events. Within this family, self-summarization denotes the consolidation behavior that motivates ReSum. The full collection is:

"in summary,", "to summarize my progress so far,", "let me summarize what I have so far.", "let me take stock of what I have.", "let me step back and review.", "let me recap what I have done so far.", "putting it all together,", "let me consolidate my work so far.", "to summarize", "let me summarize", "iĺl summarize", "i will summarize", "summarizing", "in summary", "a quick summary", "brief summary", "quick recap", "let me give a summary", "to recap", "let me recap", "to review", "let me review", "reviewing", "let me re-examine", "re-examining", "let me revisit", "revisiting", "let me re-check", "let me double-check", "double-checking", "let me go back", "going back to", "what i have so far", "what we have so far", "so far i have", "so far, i have", "so far we have", "so far, we have", "up to now", "thus far", "at this point, i have", "at this point, we have", "let me take stock", "taking stock of", "let me consolidate"", "in conclusion", "to conclude", "let me conclude", "concluding", "putting it all together", "combining the above", "combining these results", "combining everything", "taking stock", "to wrap up", "wrapping up", "reflecting on", "let me reflect", "let me think about", "let me reconsider", "reconsidering", "stepping back", "let me step back", "let me pause", "wait, let me", "actually, let me", "let me verify", "verifying", "let me check", "checking my work", "let me confirm", "confirming", "let me try a different approach", "let me try another approach", "let me try again", "let me start over", "let me restart", "let me redo", "let me rethink", "rethinking", "let me approach this differently", "a different way to think about", "alternatively,", "on second thought", "wait,", "wait.", "hmm,", "hmm.", "hold on,", "hold on.", "actually,", "to put it all together", "to put it another way", "to rephrase", "in other words, let me".

Our prompt-only comparison uses the following periodic self-summarization instruction:

Please reason step by step. As you work through the problem, periodically summarize your progress and reflect on your approach to make sure you are on the right track. Put your final answer within `\boxed{}`.

Following MathForge Dai et al. (2026), our codebase is built upon the Open-R1 framework Face (2025) and all experiments are conducted on 8 NVIDIA H20 GPUs. For reinforcement learning, responses are generated with a temperature of 1.0 and a maximum completion length of 4096 tokens. During evaluation, we use a temperature of 0.6, a top-p of 0.95, and a maximum generation length of 4096 tokens. The task reward is binary, with a correct final answer yielding 1 and an incorrect answer yielding 0.

For experiments on the MATH dataset, the system prompt instructs the model to reason step by step and enclose the final answer within `\boxed{}`. The maximum prompt length is set to 512 tokens. We generate $G = 16$ responses per query and use a training batch size of 32. For ReSum, the rollout budget $B = TJ$ is set to G, ensuring a fair comparison of rollout budget with GRPO. For ReSum, the format reward weights for NPs and APs in Eq. 2 are set to 0.2 and 0.05, respectively. Branching is only applied to the initial rollouts. β is set to 0, and the clipping range parameter ϵ is set to 0.2. We train for 1 epoch with a learning rate of 1×10^{-6} and gradient accumulation steps of 1, resulting in 230 training steps. When combining ReSum with GPG, DAPO, or GSPO, we retain the respective baseline's original hyperparameters (e.g., DAPO's asymmetric clipping bounds ϵ_{low} and ϵ_{high}, GSPO's sequence-level importance sampling) and only replace their advantage estimation with our summarization-aware group-relative advantage defined in Eq. 3.

For the multimodal experiments, we extend the maximum prompt length to 2048 tokens and adjust the system prompt to instruct the model to provide the final answer without units within `\boxed{}`. The model is trained for 2 epochs with a learning rate of 1×10^{-6}, a batch size of 32, and gradient accumulation steps of 1, resulting in 480 training steps. All other settings remain consistent with the text-only experiments.

E Further Analysis

Comparisons with naive prompt refinement. Table 7 compares the Base Model, Prompt Refinement (an explicit periodic self-summarization instruction), and ReSum across three backbones. Prompt Refinement produces no comparable end-to-end improvement, whereas ReSum consistently improves accuracy. This difference shows the value of contrastive rollout construction and credit

Table 7: Performance comparison between ReSum and prompt refinement on 6 challenging mathematical reasoning datasets. The best results are in **bold**.

Model	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
Qwen2.5-Math-1.5B							
Baseline	6.87	3.65	30.94	34.95	8.55	21.93	17.82
Prompt Refinement	6.67	4.17	34.33	39.85	11.31	23.41	19.96
ReSum	**15.96**	**6.83**	**51.33**	**67.12**	**24.58**	**32.59**	**33.07**
Qwen2.5-3B							
Baseline	2.81	0.73	22.66	48.65	13.69	19.37	17.99
Prompt Refinement	2.81	1.15	22.73	47.25	14.89	17.74	17.76
ReSum	**7.50**	**2.29**	**39.06**	**68.40**	**26.69**	**28.93**	**28.81**
DeepSeek-Math-7B							
Baseline	0.42	0.10	13.28	31.05	9.56	9.00	10.57
Prompt Refinement	0.33	0.10	13.85	30.54	9.75	10.08	10.78
ReSum	**1.66**	**0.96**	**22.40**	**43.10**	**20.22**	**15.56**	**17.32**

Table 8: Significance test (p-value) of ReSum over DGPO on all backbones of each dataset.

Model	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
Qwen2.5-Math-1.5B	0.0080	0.0121	0.0051	0.0018	0.0020	0.0095	0.0064
Qwen2.5-3B	0.0151	0.0098	0.0353	0.0001	0.0190	0.0076	0.0145
DeepSeek-Math-7B	0.0231	0.0049	0.0225	0.0069	0.0028	0.0130	0.0122
Qwen2.5-Math-7B	0.0353	0.0040	0.0077	0.0070	0.0365	0.0008	0.0152

assignment over a static instruction alone. The pilot study measures conditional recovery from selected incorrect prefixes, while Table 7 measures end-to-end accuracy over the full benchmark distribution; their gains therefore use different denominators.

Significance Test. We provide the significance test in Table 8. We find that ReSum consistently achieves statistically significant differences over DGPO across all backbones and benchmarks, with all p-values below 0.05. More specifically, for Qwen2.5-Math-1.5B, the p-values range from 0.0018 to 0.0121, with an average of 0.0064; for Qwen2.5-3B, they range from 0.0001 to 0.0353, with an average of 0.0145. Similar trends are observed on DeepSeek-Math-7B and Qwen2.5-Math-7B, whose average p-values are 0.0122 and 0.0152 respectively. Notably, the significance remains stable not only on in-distribution benchmarks such as MATH500, but also on more challenging competition-style datasets such as AIME24, AIME25, and Olympiad. These results suggest that the gains of ReSum are not caused by random fluctuations from repeated sampling, but reflect a robust and reproducible advantage over DGPO. Combined with the consistent improvements in the main results, the significance test further validates that incentivizing periodic self-summarization indeed provides a reliable benefit for long-chain reasoning.

Efficiency. Table 9 reports the average time cost of DGPO and ReSum under a matched number of completed rollouts per query. ReSum adds approximately 13% training-time overhead, primarily from branch construction, while its evaluation-step time is close to DGPO's. This measured cost profile pairs modest training orchestration overhead with improved end-to-end accuracy and shorter generated rollouts.

Case Study. We provide some cases in this part, which shows that ReSum incentivizes the appropriate summarization ability by summarizing and rethinking generated contents:

Table 9: Efficiency study of ReSum of each dataset.

Backbone	DGPO					ReSum				
	initial rollout	advantage	train/step	eval/step		initial rollout	branching	advantage	train/step	eval/step
Qwen2.5-Math-1.5B	21.50	0.81	22.31	12.5		10.20	14.41	0.68	25.29	12.78

To solve the equation $\sqrt{x+3}+\sqrt{x-2}=5$, we can follow these steps:
1. Isolate one of the square roots. 2. Square both sides to eliminate the square root. 3. Simplify and isolate the remaining square root. 4. Square both sides again to eliminate the remaining square root. 5. Solve the resulting equation for x. 6. Verify the solution by substituting back into the original equation.
Let's implement this step-by-step in Python using SymPy:

```
import sympy as sp

x = sp.symbols('x')
equation = sp.sqrt(x + 3) + sp.sqrt(x - 2) - 5
solution = sp.solve(equation, x)
valid_solutions = [sol for sol in solution if equation.subs(x, sol)
    == 0]
print(valid_solutions)
```

```
[7]
```

The solution to the equation $\sqrt{x+3}+\sqrt{x-2}=5$ is $x=7$. To verify, we can substitute $x=7$ back into the original equation:

$$\sqrt{7+3}+\sqrt{7-2}=\sqrt{10}+\sqrt{5}$$

We need to check if $\sqrt{10}+\sqrt{5}=5$. Let's compute this explicitly:

$$\sqrt{10}\approx 3.162, \quad \sqrt{5}\approx 2.236, \quad \sqrt{10}+\sqrt{5}\approx 5.398\neq 5$$

To summarize, it seems there was an error in the verification. Let's recheck the solution by solving the equation again carefully.
We'll start by isolating one of the square roots and then squaring both sides:

$$\sqrt{x+3}=5-\sqrt{x-2}$$
$$x+3=(5-\sqrt{x-2})^2=25-10\sqrt{x-2}+(x-2)$$
$$3=23-10\sqrt{x-2} \implies 10\sqrt{x-2}=20 \implies \sqrt{x-2}=2$$
$$x-2=4 \implies x=6$$

Let's verify $x=6$ by substituting back into the original equation:

$$\sqrt{6+3}+\sqrt{6-2}=\sqrt{9}+\sqrt{4}=3+2=5\checkmark$$

This verifies that $x=6$ is the correct solution. The previous output of 7 was incorrect due to a verification error.

```
valid_solutions = [sol for sol in solution if equation.subs(x, sol)
    == 0]
print(valid_solutions)
```

```
[6]
```

The final answer is:

$$\boxed{6}$$

Problem: How many 4-digit numbers have digits that sum to 10, where each digit is between 1 and 4 inclusive?

Model Reasoning:

To solve this problem, we need to find the number of solutions to

$$x_1 + x_2 + x_3 + x_4 = 10, \quad 1 \le x_i \le 4.$$

1. Variable substitution. Let $y_i = x_i - 1$. Then $0 \le y_i \le 3$ and the equation becomes

$$y_1 + y_2 + y_3 + y_4 = 6.$$

2. Unconstrained count. By stars and bars, the total number of non-negative integer solutions is

$$\binom{6+4-1}{4-1} = \binom{9}{3} = 84.$$

3. Subtract violating cases. If $y_1 > 3$, let $z_1 = y_1 - 4 \ge 0$. The equation becomes $z_1 + y_2 + y_3 + y_4 = 2$, giving $\binom{5}{3} = 10$ solutions. By symmetry, there are $4 \times 10 = 40$ violating cases.
4. Check double-counted cases. If $y_1 > 3$ and $y_2 > 3$ simultaneously, the equation becomes $z_1 + z_2 + y_3 + y_4 = -2$, which has *no solutions*. No correction needed.

To summarize

Conclusion. Putting it all together, the number of valid solutions is

$$\binom{9}{3} - 4 \cdot \binom{5}{3} = 84 - 40 = \boxed{44}.$$

Code verification:

```
import math
binom_9_3 = math.comb(9, 3)    # 84
binom_5_3 = math.comb(5, 3)    # 10
valid_solutions = binom_9_3 - 4 * binom_5_3
print(valid_solutions)          # Output: 44
```

The final answer is: $\boxed{44}$

To solve the equation $\lfloor x \rfloor + \lfloor 2x \rfloor = 5$, let's break down the problem step-by-step.
1. Let $\lfloor x \rfloor = n$, where n is an integer. This means that $n \leq x < n + 1$. 2. Then, $\lfloor 2x \rfloor$ depends on the value of $2x$. Since $n \leq x < n + 1$, we have $2n \leq 2x < 2n + 2$. Therefore, $\lfloor 2x \rfloor$ can be either $2n$ or $2n + 1$. 3. We need to satisfy the equation $n + \lfloor 2x \rfloor = 5$. 4. This gives us two cases to consider:

- Case 1: $\lfloor 2x \rfloor = 2n$. Then, $n + 2n = 5$, so $3n = 5$, which gives $n = \frac{5}{3}$. Since n must be an integer, this case is not possible.
- Case 2: $\lfloor 2x \rfloor = 2n + 1$. Then, $n + (2n + 1) = 5$, so $3n + 1 = 5$, which gives $3n = 4$, and thus $n = \frac{4}{3}$. Since n must be an integer, this case is also not possible for the same reason.

To summarize both possible cases lead to non-integer values of n. Since $n = \lfloor x \rfloor$ must be an integer, neither case can occur. Therefore, the equation has no real solution.
Let's verify this conclusion by checking a dense grid of real values.

```
import numpy as np

def check_solution(x):
   return np.floor(x) + np.floor(2 * x) == 5

test_points = np.linspace(-10, 10, 10000)
solutions = [x for x in test_points if check_solution(x)]
print(solutions)
```

```
[]
```

The Python check finds no sampled real values satisfying the equation, which is consistent with the case analysis.
The final answer is:

$$\boxed{\text{No real solution}}$$

F ALGORITHM

The algorithm of ReSum is shown in algorithm 1.

G LIMITATIONS

ReSum uses a lightweight keyword detector to identify summary-like reasoning-control events, so the quality and coverage of the cue pool influence the learned behavior. The pool includes consolidation, reflection, verification, repair, and strategy-switching expressions; the reported gains therefore characterize the complete reasoning-control mechanism, with self-summarization as its central motivation. The rollout-tree shape and auxiliary reward weights also require configuration for different model scales and tasks, although the branch-budget study provides practical guidance for this choice. The current evidence covers verifiable mathematical reasoning and multimodal geometry, where final answers are easy to evaluate; extending the same training signal to code generation, tool use, search, and other open-ended long-horizon tasks remains an important direction. ReSum retains the preceding reasoning prefix during cue-based reorganization, so future work can investigate complementary designs that combine its state-local training signal with explicit context replacement or compression.

Algorithm 1: ReSum Training Pipeline

Input : policy π_θ, reference policy π_{ref}, training set $\mathcal{D}$, number of trees T, generation loops J, rollout budget $B = TJ$ (matched to GRPO group size G), summarization phrase pool $\mathcal{P}$, clipping ϵ, KL coefficient β, reward stabilizer $\varepsilon_R > 0$, advantage stabilizer $\xi > 0$

// J: generation loops; j=1 samples initial rollouts; B=TJ total rollouts.

for *each training step* **do**

  Sample query $q \sim \mathcal{D}$ and freeze $\pi_{\text{old}} \leftarrow \pi_\theta$

  for $t = 1$ **to** T **do**

    $\mathcal{T}_t \leftarrow \emptyset$

  // -- Rollout generation: $j = 1, \ldots, J$; one rollout per tree per loop --

  for $j = 1$ **to** J **do**

    for $t = 1$ **to** T **do**

      if $j = 1$ **then**

        $\tau_{t,1} \sim \pi_{\text{old}}(\cdot \mid q)$, $\text{kind}(\tau_{t,1}) \leftarrow$ root

        $\mathcal{N}_t \leftarrow$ ordered NP positions on $\tau_{t,1}$ detected by $\mathcal{P}$

        $\mathcal{A}_t \leftarrow$ non-summary token positions on $\tau_{t,1}$ (excluding $\mathcal{N}_t$ and other phrase matches)

        store $\tau_{t,0} \leftarrow \tau_{t,1}$ as the branching anchor

      // branch only from the initial rollout $\tau_{t,0}$

      if $\mathcal{N}_t \neq \emptyset$ **then**

        pop the next NP position p from $\mathcal{N}_t$

        $\mathcal{F} \leftarrow$ prefix of $\tau_{t,0}$ up to p, with the matched phrase masked

        $\tau_{t,j} \sim \pi_{\text{old}}(\cdot \mid q, \mathcal{F})$, $\text{kind}(\tau_{t,j}) \leftarrow$ np

      else

        sample AP position p uniformly from $\mathcal{A}_t$

        $\mathcal{F} \leftarrow$ prefix of $\tau_{t,0}$ up to p, appended with $s \sim \mathcal{P}$

        $\tau_{t,j} \sim \pi_{\text{old}}(\cdot \mid q, \mathcal{F})$, $\text{kind}(\tau_{t,j}) \leftarrow$ ap

      $\mathcal{T}_t \leftarrow \mathcal{T}_t \cup \{\tau_{t,j}\}$

  // -- Rewards (Eq. equation 2) --

  for $\tau \in \bigcup_{t=1}^{T} \mathcal{T}_t$ **do**

    $R_A(\tau) \leftarrow \mathbf{1}[\text{Correct}(\tau)]$, $n \leftarrow \#\{\text{natural cue matches from } \mathcal{P} \text{ in } \tau\}$, excluding an injected AP cue

    if $\text{kind}(\tau) =$ ap **then**

      $N_{\text{NP}}(\tau) \leftarrow n$; $N_{\text{AP}}(\tau) \leftarrow 1$

    $N_{\text{NP}}(\tau) \leftarrow n$; $N_{\text{AP}}(\tau) \leftarrow 0$

    if $N_{\text{NP}}(\tau) + N_{\text{AP}}(\tau) > 0$ **then**

      $$R_F(\tau) \leftarrow \frac{0.2\, N_{\text{NP}}(\tau) + 0.05\, N_{\text{AP}}(\tau)}{N_{\text{NP}}(\tau) + N_{\text{AP}}(\tau) + \varepsilon_R}$$

    else

      $R_F(\tau) \leftarrow 0$

    $R(\tau) \leftarrow R_A(\tau) + \mathbf{1}[R_A(\tau) = 1] R_F(\tau)$; store $n(\tau) \leftarrow n$

  // -- Dual-reference advantages (Eq. equation 3) --

  $G_{\text{cue}} \leftarrow \{\tau : n(\tau) > 0 \ \vee \ \text{kind}(\tau) = \text{ap}\}$; $G_{\text{nocue}} \leftarrow \{\tau \in \bigcup_t \mathcal{T}_t : \tau \notin G_{\text{cue}}\}$

  if $G_{\text{cue}} \neq \emptyset$ **then**

    $\mu_{\text{cue}} \leftarrow \text{mean}_{\tau \in G_{\text{cue}}} R(\tau)$; $\sigma_{\text{cue}} \leftarrow \text{std}_{\tau \in G_{\text{cue}}} R(\tau)$

  if $G_{\text{nocue}} \neq \emptyset$ **then**

    $\mu_{\text{nocue}} \leftarrow \text{mean}_{\tau \in G_{\text{nocue}}} R(\tau)$; $\sigma_{\text{nocue}} \leftarrow \text{std}_{\tau \in G_{\text{nocue}}} R(\tau)$

  for $\tau \in \bigcup_{t=1}^{T} \mathcal{T}_t$ **do**

    $$\hat{A}^{\text{cue}}(\tau) \leftarrow \begin{cases} \dfrac{R(\tau) - \mu_{\text{cue}}}{\sigma_{\text{cue}} + \xi}, & G_{\text{cue}} \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

    $$\hat{A}^{\text{nocue}}(\tau) \leftarrow \begin{cases} \dfrac{R(\tau) - \mu_{\text{nocue}}}{\sigma_{\text{nocue}} + \xi}, & G_{\text{nocue}} \neq \emptyset \\ 0, & \text{otherwise} \end{cases}$$

    $\hat{A}(\tau) \leftarrow \hat{A}^{\text{cue}}(\tau) + \hat{A}^{\text{nocue}}(\tau)$

  // -- SGPO policy update (Eq. equation 4) --

  for *each* $\tau_{t,e} \in \bigcup_{t=1}^{T} \mathcal{T}_t$ **do**

    for $i = 1$ **to** $|\tau_{t,e}|$ **do**

      $$\rho_{t,e,i}(\theta) \leftarrow \frac{\pi_\theta(\tau_{t,e,i} \mid \tau_{t,e,<i}, q)}{\pi_{\text{old}}(\tau_{t,e,i} \mid \tau_{t,e,<i}, q)}$$

  Update θ by maximizing $J_{\text{ReSum}}(\theta)$, where each token in $\tau_{t,e}$ uses rollout-level $\hat{A}(\tau_{t,e})$