Step-wise Rubric Rewards for LLM Reasoning

Weichu Xie[1,2,*] **Haozhe Zhao**[1,*] **Wenpu Liu**[1,2,*]
Yongfu Zhu[2] **Liang Chen**[1] **Minghao Ye**[3] **Zirong Chen**[4]
Yuqi Xu[1,2] **Shuai Dong**[5,2] **Ziyue Wang**[1,2] **Xinbo Xu**[1]
Kean Shi[1] **Ruoyu Wu**[1] **Xiaoying Zhang**[2] **Wenqi Shao**[2]
Baobao Chang[1,†] **Nan Duan**[2] **Jiaqi Wang**[2]

[1]Peking University [2]JD Explore Academy
[3]Shanghai Jiao Tong University [4]Tsinghua University
[5]Shanghai Innovation Institute

[*]Equal contribution. [†]Corresponding author.

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) is widely used to improve the reasoning capabilities of large language models, but its reward is derived only from the correctness of the final answer and provides no supervision over intermediate reasoning steps. Recent rubric-based methods, such as Rubrics as Rewards (RaR), introduce finer-grained supervision by scoring rollouts against a structured set of evaluation criteria. However, the resulting rubric scores are still aggregated into a single scalar that is applied to the entire response, leading to three structural weaknesses, namely loss of the multi-criterion rubric structure, uniform supervision of correct and incorrect reasoning steps, and reward hacking in the trained model through unbounded self-correction. On a sample of 1,000 problems, we find that 18.2% of steps within answer-correct responses are themselves wrong yet positively rewarded, while 49.9% of steps within answer-incorrect responses are in fact correct yet penalized. Therefore, we introduce **Step-wise Rubrics as Rewards (SRaR)**, an RLVR framework that (i) uses an LLM judge to attribute each rubric item to a specific reasoning step, (ii) normalizes the per-step rubric scores across rollouts so that only steps whose quality varies produce a learning signal, and (iii) combines the resulting per-step reward with the standard outcome reward through a decoupled advantage estimator that keeps the outcome-driven baseline stable. To support training, we further build a 16K-problem rubric dataset by contrastively distilling rubric items from correct and flawed reasoning paths sampled from a strong model and verified against the ground-truth answer. Across six mathematical reasoning benchmarks spanning multiple difficulty levels, SRaR improves the average accuracy over RaR by **3.57** points on Qwen3-8B-Non-Thinking and by **2.75** points on Qwen3-32B-Non-Thinking, raises the Faithful Reasoning Rate on AIME 2025 from 34.5% to 46.7% (every correct-answer trajectory of SRaR uses entirely correct reasoning steps), and reduces the rate of self-correction looping from 48.1% to 26.5%. The gains extend to code generation, where SRaR is best on three of four benchmarks.

1 Introduction

Large language models (LLMs) have demonstrated remarkable reasoning capabilities across complex tasks such as mathematics, programming, and scientific problem solving [Wei et al., 2022, OpenAI, 2024, Dong et al., 2025, Chen et al., 2026a, Team, 2026, Si et al., 2026, Jin et al., 2026, Xiao et al.,

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

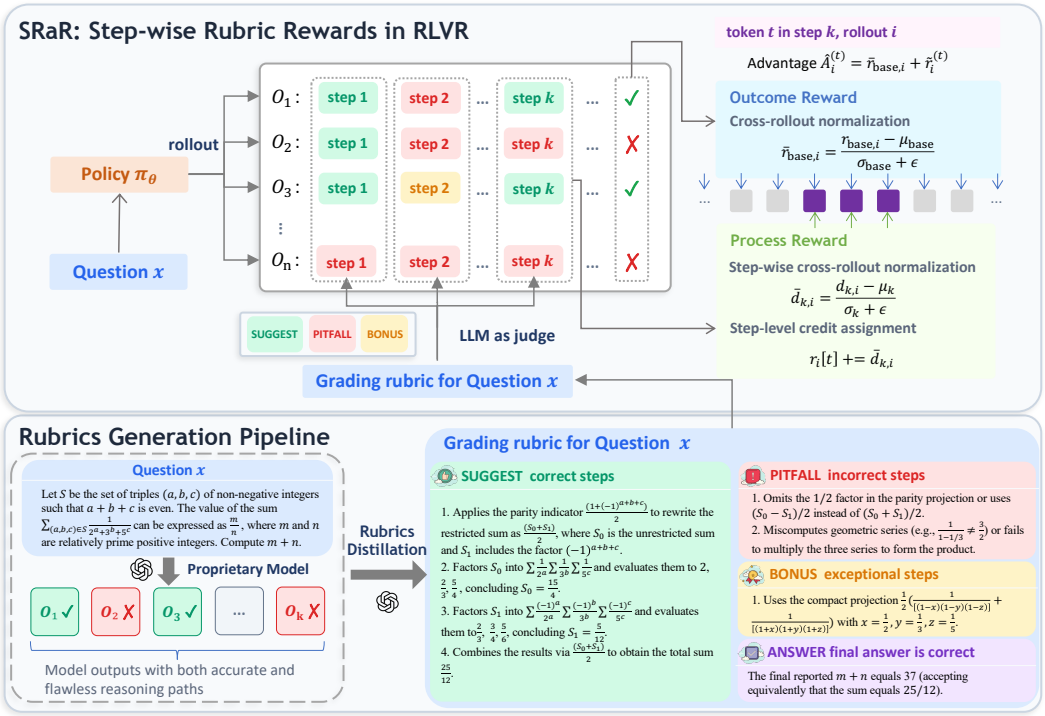


Figure 1: **Overview of SRaR. Top:** The SRaR training framework. An LLM judge attributes each rubric item to a specific reasoning step; per-step rewards are normalized across rollouts and combined with outcome rewards via a decoupled advantage estimator. **Bottom:** The three-stage contrastive rubric distillation pipeline that generates instance-specific rubrics from correct and flawed paths.

2026b, Shen et al., 2026b, Li et al., 2026a]. To further enhance these capabilities, Reinforcement Learning with Verifiable Rewards (RLVR) has emerged as a widely adopted training paradigm in which rule-based verifiers automatically assess the correctness of model-generated answers and produce reward signals for policy optimization [DeepSeek-AI, 2025, Shao et al., 2024, Dong et al., 2026, Liu et al., 2026c, Wang et al., 2026b, Shen et al., 2026a]. While effective, the RLVR reward is derived solely from the correctness of the final answer and provides no supervision over the quality of intermediate reasoning steps. To inject finer-grained supervision, recent work turns to rubrics, structured checklists of evaluation criteria that decompose response quality into multiple verifiable aspects and are scored automatically by an LLM judge [Hashemi et al., 2024, Kim et al., 2024]. Methods such as Rubrics as Rewards (RaR) [Gunjal et al., 2025] and RGR-GRPO [Bi et al., 2025] use these rubric scores as RL reward signals by aggregating the per-criterion scores into a single scalar reward applied to the entire response (Figure 2(a)).

Despite providing richer supervision than pure outcome rewards, current rubric-based methods exhibit three structural weaknesses. **First, existing methods score each rubric item as binary pass/fail and then merge all per-step verdicts into a single response-level scalar, discarding both the type-level and step-level structure of the rubric.** The binary scoring erases the distinction between criteria that reinforce standard reasoning, flag common errors, and credit exceptional insights. The response-level aggregation further discards which step each criterion judges, leaving the policy gradient with no information about the multi-aspect structure that motivated using rubrics in the first place. **Second, this response-level reward is applied uniformly to every token, so correct and incorrect reasoning steps within the same response receive the same reward signal.** As a result, the policy gradient cannot distinguish steps that actually contributed to the response's quality from steps that should have been corrected. To measure how often this matters in practice, we sample 1,000 random solutions from Qwen3-8B-Non-Thinking on the DAPO training set and independently judge each reasoning step. 18.2% of steps within answer-correct responses are themselves wrong and 49.9% of steps within answer-incorrect responses are themselves correct (Figure 2(b)), so any single-scalar response-level reward assigns incorrect supervision to a substantial fraction of intermediate steps. **Third, the rubric score rewards visible reasoning effort such as self-correction and**

re-verification, so the model can inflate it without actually improving the answer. Because existing methods feed this rubric score directly into the policy gradient, the model learns to overproduce such surface behaviors as a form of reward hacking. On AIME 2025, this manifests as severe self-correction looping: 48.1% of RaR rollouts repeatedly rewrite and re-verify intermediate steps to inflate the rubric reward, often failing to terminate (Section 4.3.2). **This raises a natural question: how can a reward signal be both interpretable and fine-grained in supervising a model's reasoning?**

First, step-attributed rubric judging preserves the multi-aspect structure that existing aggregation discards. Each rubric item carries one of three types: SUGGEST for standard reasoning, PITFALL for known errors, and BONUS for exceptional insights. An LLM judge evaluates whether each item is satisfied and ties it to the reasoning step it judges, producing per-step, type-differentiated rewards that are retained at the step level rather than merged into a single response-level scalar. **Second, per-step token assignment with cross-rollout normalization** replaces the uniform token-level reward with step-aligned credit. Each step's reward is broadcast only to its own tokens and normalized across rollouts of the same prompt, so each token receives a learning signal aligned with its own step. **Third, a decoupled advantage estimator** computes the advantage as a sum of two separately normalized parts: one from the outcome reward, one from the rubric. The outcome part is the standard advantage from accuracy alone; the rubric part is added on top as a bounded, per-step adjustment. Because rubric noise stays in the rubric part, no rollout can inflate its overall advantage by gaming the rubric, removing the incentive behind self-correction looping and yielding more stable training.

All three designs assume access to a problem-specific rubric, yet no public corpus provides rubrics paired with reasoning data at the scale needed for RL training. We therefore introduce a **contrastive rubric distillation** pipeline that constructs a 16K-problem rubric corpus in three stages: (i) sampling multiple reasoning rollouts from a strong model (GPT-5) per problem; (ii) verifying them against the ground-truth answer to split correct from flawed reasoning paths; and (iii) feeding both groups back to the same model to distill rubric items, each classified as SUGGEST, PITFALL, or BONUS, that capture what separates successful from failed reasoning. By grounding each rubric item in empirically observed failure modes rather than a single reference answer, this procedure yields criteria that target the errors a policy is actually likely to commit during RL training.

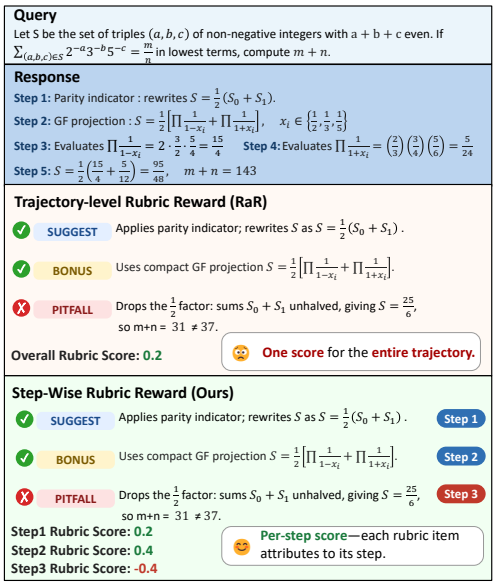
We evaluate *SRaR* on various reasoning benchmarks spanning multiple difficulty levels and model scales. *SRaR* improves average accuracy over the strongest baseline by **3.57** points on Qwen3-8B and **2.75** points on Qwen3-32B, with consistent improvements across all benchmarks and model scales. Three targeted analyses on AIME 2025 attribute these gains to the three designs above. Step-attributed judging with per-step token assignment raises the Faithful Reasoning Rate, the fraction of trajectories with both a correct answer and all-correct intermediate steps, from 34.5% to 46.7%. The resulting step-level credit assignment reduces self-correction looping, a failure mode in which the model repeatedly regenerates or revisits reasoning steps without converging, from 48.1% to 26.5%. Finally, removing cross-rollout step normalization causes training to collapse. The gains also transfer beyond mathematics: on code generation, *SRaR* is best on three of four benchmarks (Appendix P). These results demonstrate that step-level rubric supervision improves both final-answer accuracy and intermediate reasoning quality.

Our contributions are summarized as follows:

1. We identify three structural weaknesses of previous methods that use rubric scores as RL reward signals for finer-grained supervision and quantify their impact via a diagnostic study on 1,000 random model rollouts.

2. We propose *SRaR*, an RLVR framework that delivers step-level rubric supervision via three coordinated designs that provide interpretable and fine-grained reward signals for supervising model reasoning.

3. We release a 16K-problem rubric dataset built via correctness-verified contrastive distillation. *SRaR* outperforms RL baselines by a large margin on six reasoning benchmarks across different model scales, while raising the Faithful Reasoning Rate and substantially reducing the self-correction looping observed in other methods.



(a) Response-level (RaR) vs. step-wise (SRaR) rubric reward

Category	# Samples	# Steps	Reward	Indiscriminate
A✓S✓	194	1012	+	—
A✓S×	194	916	+	350 wrong
A×S✓	18	101	−	101 correct
A×S×	591	3151	−	1521 correct

Indiscriminate reward under response-level signal:

Wrong steps receiving same + as correct (A✓):

$$\frac{350}{1012 + 916} = \frac{350}{1928} = \mathbf{18.2}\%$$

Correct steps receiving same − as wrong (A×):

$$\frac{101 + 1521}{101 + 3151} = \frac{1622}{3252} = \mathbf{49.9}\%$$

(b) Indiscriminate reward under response-level signal

Figure 2: **Motivation for step-wise rubric reward. (a)** RaR applies a single scalar rubric reward to the entire response, so every step receives the same signal; SRaR decomposes the rubric into per-step rewards aligned with each reasoning step. **(b)** Diagnostic study on 1,000 random Qwen3-8B-Non-Thinking solutions from the DAPO training set, each parsed into reasoning steps and independently judged (details in Section 4.3.1). A✓/A× denotes correct/incorrect final answer; S✓/S× denotes all steps correct / at least one step wrong. Under a single-scalar response-level reward, **18.2%** of steps within answer-correct responses are wrong yet receive the same + reward as correct steps, and **49.9%** of steps within answer-incorrect responses are correct yet receive the same − penalty as wrong steps.

2 Related Work

2.1 Process Supervision for Reasoning

RLVR with GRPO and DAPO [Yu et al., 2025] provides only sparse binary outcome signals with no guidance on intermediate steps. Process Reward Models (PRMs) [Lightman et al., 2024, Wang et al., 2024, Luo et al., 2024] score individual steps but require massive human annotations and produce opaque scalars with no explanation for why a step receives its score. GRPO-VPS [Wang et al., 2026a] avoids auxiliary models by measuring the policy's own conditional probability of the correct answer at each step boundary, but the resulting signal is still a single opaque scalar per step that offers no interpretable diagnosis of reasoning quality.

2.2 Rubric-Based Evaluation and Rewards

Rubric-based evaluation decomposes quality into interpretable criteria without auxiliary models. Hashemi et al. [2024] introduce LLM-Rubric for calibrated multi-aspect evaluation; Kim et al. [2024] train open-source rubric-guided evaluators; RaR [Gunjal et al., 2025] and RGR-GRPO [Bi et al., 2025] use rubric scores directly as RL rewards. However, all existing rubric-based RL methods aggregate scores into a single trajectory-level scalar, forfeiting per-step credit: correct steps in failed trajectories are penalized and incorrect steps in successful ones are reinforced. SRaR addresses this overlooked problem by attributing each rubric item to its corresponding reasoning step and normalizing rewards at the step level.

3 Method

3.1 Preliminary

RLVR for LLM Reasoning. Reinforcement Learning with Verifiable Rewards (RLVR) trains a policy π_θ by rewarding responses whose final answer matches the ground truth [DeepSeek-AI, 2025, Shao et al., 2024]. We adopt Group Relative Policy Optimization (GRPO) [Shao et al., 2024]: for each prompt $x \in \mathcal{X}$, where $\mathcal{X}$ denotes the space of problem prompts, GRPO samples n rollouts $\{y_i\}_{i=1}^n \sim \pi_\theta(\cdot \mid x)$, scores each with a scalar reward r_i, and computes the advantage by group normalization

$$A_i = \frac{r_i - \mu}{\sigma + \epsilon}, \quad \mu = \frac{1}{n}\sum_{j=1}^{n} r_j, \quad \sigma^2 = \frac{1}{n}\sum_{j=1}^{n}(r_j - \mu)^2. \tag{1}$$

The same scalar A_i is broadcast as the per-token advantage at every position t of y_i.

Problem Formulation. Each problem $x \in \mathcal{X}$ in our training set $\mathcal{D} = \{(x, a^*, \mathcal{R}_x)\}$ carries a ground-truth answer a^* and a rubric $\mathcal{R}_x = \{(c_j, \tau_j)\}_{j=1}^M$ with criterion text c_j and type $\tau_j \in \{\texttt{SUGGEST}, \texttt{PITFALL}, \texttt{BONUS}\}$. Given a response $y = (y_1, \ldots, y_T) \in \mathcal{V}^T$, a token sequence over the vocabulary $\mathcal{V}$, organized into K steps with token spans $\{(a_k, b_k)\}_{k=1}^K$, our task is to construct a token-level advantage $\hat{A}_i^{(t)}$ for each rollout i and position t, jointly informed by outcome correctness and rubric verdicts, as a drop-in replacement for A_i. A summary of notation is provided in Table 4 (Appendix A).

3.2 Rubrics Design

A rubric $\mathcal{R}_x$ comprises three item types, each contributing a distinct sign to the policy gradient (Section 3.3). **SUGGEST** items reinforce standard problem-solving paradigms (clear step structure, logical coherence, explicit derivations), contributing positive deltas. **PITFALL** items penalize known failure modes (arithmetic errors, logical leaps, hallucinations), contributing negative deltas. **BONUS** items reward valuable optional behaviors (alternative solution paths, cross-validation), contributing additional positive deltas. This three-aspect schema separates baseline correctness, error avoidance, and exceptional quality, each bounded by a per-type budget.

3.3 Step-wise Rubrics Rewards in RLVR (SRaR)

Figure 1 (top) provides an overview. SRaR realizes three coordinated designs: **step-attributed rubric judging**, **per-step token assignment with cross-rollout normalization**, and a **decoupled advantage estimator**.

Step-attributed rubric judging. An LLM judge ties each rubric item to the reasoning step it judges, producing a per-step, per-criterion verdict instead of a single response-level scalar. To enable step attribution, the policy is prompted to emit responses in a structured step format ("`### Step N: ...`") with the final answer in `\boxed{}`, and step token spans $\{(a_k, b_k)\}$ are recovered via the tokenizer's offset mapping. For each response y, the judge $\mathcal{J}$ [Zheng et al., 2023] returns

$$\mathcal{J}(y, \mathcal{R}_x) = \{(s_j, k_j)\}_{j=1}^M, \quad s_j \in \{0, 1\},\ k_j \in \{1, \ldots, K\}, \tag{2}$$

where s_j indicates whether criterion j is satisfied and k_j records the step it judges.

Per-step token assignment with cross-rollout normalization. Each judged rubric item is converted into a per-step reward, normalized across rollouts at the step level, and broadcast only to the tokens of the step it judges. This replaces the uniform token-level reward of standard GRPO with step-aligned credit.

We first map each verdict to a signed delta whose magnitude depends on its rubric type:

$$\delta_j = \begin{cases} +R_{\texttt{SUG}}/N_{\texttt{SUG}} & \tau_j = \texttt{SUGGEST},\ s_j = 1 \\ -|R_{\texttt{PIT}}|/N_{\texttt{PIT}} & \tau_j = \texttt{PITFALL},\ s_j = 1 \\ +R_{\texttt{BON}}/N_{\texttt{BON}} & \tau_j = \texttt{BONUS},\ s_j = 1 \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $R_\tau \in \mathbb{R}$ is the signed reward budget for type $\tau \in \{\texttt{SUGGEST}, \texttt{PITFALL}, \texttt{BONUS}\}$, with $R_{\texttt{SUG}}, R_{\texttt{BON}} > 0$ and $R_{\texttt{PIT}} < 0$, and N_τ counts the items of that type in the current sample, keeping each category's contribution bounded regardless of rubric list length. Deltas attributed to the same step k are summed into a per-step raw rubric reward $d_{k,i} = \sum_{j:k_j=k} \delta_j$ for rollout i, representing the total rubric credit that step k receives before normalization.

To extract a learning signal that reflects relative step quality across rollouts, we normalize $d_{k,i}$ per step across rollouts of the same prompt:

$$\bar{d}_{k,i} = \frac{d_{k,i} - \mu_k}{\sigma_k + \epsilon}, \quad \mu_k = \frac{1}{|\mathcal{G}_k|} \sum_{i \in \mathcal{G}_k} d_{k,i}, \tag{4}$$

where $\mathcal{G}_k$ is the set of rollouts whose judge evaluation contains at least one rubric item attributed to step k, and $\bar{d}_{k,i} = 0$ when $|\mathcal{G}_k| \leq 1$ since a single sample provides no relative signal. This mirrors GRPO's group normalization, applied independently per step: steps that are consistently good or bad across rollouts receive near-zero advantage, while steps whose quality varies receive a strong learning signal. The normalized $\bar{d}_{k,i}$ is then broadcast to every token within the step's span:

$$\tilde{r}_i^{(t)} = \bar{d}_{k,i} \quad \text{for all } t \in [a_k, b_k], \tag{5}$$

so that all tokens of a reasoning step share the same rubric-derived signal.

Decoupled advantage estimator. A naive substitution of the combined base-and-rubric reward into the GRPO advantage would let rubric noise enter the group baseline, allowing a rollout to inflate its overall advantage by gaming the rubric. We instead compute the advantage as a sum of two separately normalized parts:

$$\hat{A}_i^{(t)} = \underbrace{\frac{r_{\text{base},i} - \mu_{\text{base}}}{\sigma_{\text{base}} + \epsilon}}_{\text{outcome advantage}} + \underbrace{\tilde{r}_i^{(t)}}_{\text{rubric offset}}, \tag{6}$$

where $r_{\text{base},i} = (1-\lambda)\, r_{\text{acc},i} + \lambda\, r_{\text{fmt},i}$ combines an accuracy reward $r_{\text{acc},i} \in \{0,1\}$ for whether the extracted answer is correct and a format reward $r_{\text{fmt},i} \in \{0,1\}$ for whether the response contains both step headers and a `\boxed{}` answer, and $\mu_{\text{base}}, \sigma_{\text{base}}$ are computed across rollouts of the same prompt. The outcome term is the standard GRPO advantage from accuracy alone; the rubric term is the bounded, per-step offset constructed above. Because rubric noise stays in the offset and never enters the group baseline, no rollout can inflate its overall advantage by gaming the rubric, removing the incentive behind self-correction looping and yielding more stable training. We provide formal analyses of the estimator's gradient decomposition, variance, and unbiasedness properties in Appendix Q. The full procedure is summarized in Algorithm 1.

3.4 Contrastive Rubric Distillation

No public corpus provides problem-specific rubrics paired with reasoning data at scale. We construct a 16K-problem rubric corpus via contrastive distillation (Figure 1, bottom) in three stages. **(1) Reasoning rollout generation.** We prompt GPT-5 to sample multiple step-by-step trajectories per problem with final answers in a standardized tag. **(2) Trajectory verification.** Each trajectory is matched against the ground-truth answer, partitioning rollouts into correct and flawed paths. **(3) Contrastive rubric distillation.** Both correct and flawed paths are fed back to the model, which contrasts structural differences to distill a fine-grained rubric list. Grounding items in empirically observed failures yields criteria targeting errors a policy is likely to commit during training.

Prompts are in Appendix C.1 and C.2; rubric examples in Appendix J; quality evaluation in Appendix G; judge validation in Appendix H.

4 Experiment

4.1 Experimental Setup

Training Details. We build the training set by applying our pipeline to the DAPO math dataset (statistics in Appendix R) and adopt the same hyperparameters and training setting as DAPO except for the reward signal. Policy models are Qwen3-8B-Non-Thinking and Qwen3-32B-Non-Thinking [Yang

Algorithm 1 SRaR: Step-wise Rubric Rewards in RLVR

Require: Policy π_θ, training set $\mathcal{D}$ with rubrics, LLM judge $\mathcal{J}$, group size n

for each training iteration **do**
  Sample prompts $\{x\}$ from $\mathcal{D}$; for each x, sample n rollouts $\{y_i\}_{i=1}^{n} \sim \pi_\theta(\cdot \mid x)$
  for each prompt x **do**
    for $i = 1, \ldots, n$ **do**
      $r_{\text{base},i} \leftarrow (1-\lambda)\, r_{\text{acc},i} + \lambda\, r_{\text{fmt},i}$
      Detect step spans $\{(a_k^i, b_k^i)\}_{k=1}^{K_i}$ via tokenizer offset mapping
      $\{(s_j, k_j)\}_j \leftarrow \mathcal{J}(y_i, \mathcal{R}_x)$ ▷ Step-attributed rubric judging
      $d_{k,i} \leftarrow \sum_{j:\, k_j = k} \delta_j$ for each step k
    end for
    for each step k present in any rollout **do**
      $\mathcal{G}_k \leftarrow \{i : d_{k,i} \text{ exists}\}$; $\bar{d}_{k,i} \leftarrow 0$ if $|\mathcal{G}_k| \le 1$ else $(d_{k,i} - \mu_k)/(\sigma_k + \epsilon)$
    end for
    for $i = 1, \ldots, n$ **do**
      $\tilde{r}_i^{(t)} \leftarrow \bar{d}_{k,i}$ for all $t \in [a_k^i, b_k^i]$, each step k ▷ Per-step token assignment
      $A_{\text{base},i} \leftarrow (r_{\text{base},i} - \mu_{\text{base}})/(\sigma_{\text{base}} + \epsilon)$
      $\hat{A}_i^{(t)} \leftarrow A_{\text{base},i} + \tilde{r}_i^{(t)}$ for all t ▷ Decoupled advantage
    end for
  end for
  Update θ via clipped PPO with token-level advantages $\{\hat{A}_i^{(t)}\}$
end for

et al., 2025], trained on 8× NVIDIA H200 GPUs with rubric budgets $R_{\texttt{SUG}} = 0.8$, $R_{\texttt{PIT}} = -1.0$, $R_{\texttt{BON}} = 1.0$ and format weight $\lambda = 0.1$. Details are in Appendix D and C.3.

Evaluation Configuration. We evaluate on six benchmarks: AIME 2024/2025, AMC 2023, MATH500 [Hendrycks et al., 2021], Minerva Math [Lewkowycz et al., 2022], and Olympiad Bench [He et al., 2024]. Decoding uses temperature 0.7, top-p 0.8, presence penalty 1.5, repetition penalty 1.0, max length 10,240 tokens. AIME 2024/2025 and AMC 2023 report Avg@32 [Wang et al., 2023]; remaining benchmarks use a single generation.

Baselines. We compare against GRPO [Shao et al., 2024], DAPO [Yu et al., 2025], RaR [Gunjal et al., 2025], GRPO-VPS [Wang et al., 2026a], and RGR-GRPO [Bi et al., 2025]. The latter two are re-implemented under the same training recipe for a fair comparison.

Rubric LLM Judge. We use GPT-OSS-20B as the rubric judge (prompt in Appendix C.4). Robustness to judge choice is validated with Qwen3-30B-A3B-Instruct as an alternative (Appendix I).

4.2 Experimental Results

4.2.1 Main Results

Table 1 reports the main results. SRaR achieves the highest average accuracy at both scales: 72.53 on Qwen3-8B (+3.57 over RaR) and 75.81 on Qwen3-32B (+2.75 over RaR), and improves over GRPO-VPS by 10.85/14.41 points and over RGR-GRPO by 9.29/8.08 points. The advantage over GRPO-VPS stems from rubric items describing specific reasoning behaviors rather than relying on probability increments of a reference answer, enabling credit attribution to step content rather than statistical effects. The advantage over RaR and RGR-GRPO comes from routing each rubric item to the step it evaluates, eliminating indiscriminate supervision where correct steps in incorrect trajectories are penalized. We validate this in Section 4.3.1.

The gains concentrate on longer benchmarks: at 8B, AIME 2025 (+5.83), Olympiad Bench (+5.64), and AMC 2023 (+4.53) improve most, while MATH500 (+1.80) and Minerva move less. Across scales, SRaR raises the base model's average by 19.13 points at 8B and 21.18 points at 32B. Training dynamics are in Appendix E.

Table 1: Main results on six mathematical reasoning benchmarks for Qwen3-8B-Non-Thinking and Qwen3-32B-Non-Thinking. Δ denotes the improvement of SRaR over the strongest existing baseline (RaR). The best results are **bolded**, second best are underlined, and our method is highlighted in light red. For AIME 2024/2025 and AMC 2023, we report Avg@32.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
Qwen3-8B-Non-Thinking							
Baseline	26.88	23.33	72.03	86.8	55.15	56.23	53.40
+ GRPO	36.56	27.71	79.06	89.2	55.51	63.20	58.54
+ GRPO-VPS	44.58	32.40	83.20	88.60	61.03	60.24	61.68
+ RGR-GRPO	45.31	34.69	81.09	90.40	61.03	66.91	63.24
+ DAPO	52.60	41.35	90.23	92.0	59.56	67.80	67.26
+ RaR	57.50	43.23	88.20	92.6	**64.71**	67.51	68.96
+ SRaR (Ours)	**61.15**	**49.06**	**92.73**	**94.4**	**64.71**	**73.15**	**72.53**
Δ	+3.65	+5.83	+4.53	+1.80	+0.00	+5.64	+3.57
Qwen3-32B-Non-Thinking							
Baseline	27.19	21.67	71.48	85.8	62.87	58.75	54.63
+ GRPO	37.92	30.21	80.78	91.0	63.97	66.02	61.65
+ GRPO-VPS	43.02	26.56	80.08	90.20	63.97	64.54	61.40
+ RGR-GRPO	47.71	36.46	87.42	94.40	68.01	72.40	67.73
+ DAPO	61.25	47.81	93.67	**95.4**	63.60	72.11	72.31
+ RaR	63.65	50.94	94.22	94.8	65.44	69.29	73.06
+ SRaR (Ours)	**64.17**	**51.77**	**94.77**	**95.4**	**72.06**	**76.71**	**75.81**
Δ	+0.52	+0.83	+0.55	+0.60	+6.62	+7.42	+2.75

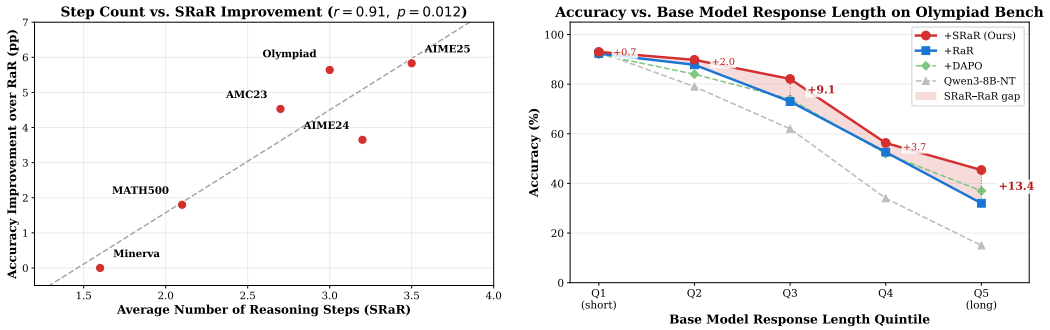


Figure 3: **(Left)** Correlation between the average number of reasoning steps per benchmark and SRaR's accuracy improvement over RaR (Pearson $r = 0.91$, $p = 0.012$). Benchmarks requiring more reasoning steps benefit more from step-wise credit assignment. **(Right)** Accuracy vs. problem difficulty on Olympiad Bench (674 problems), where difficulty quintiles are defined by the base model's response length. SRaR outperforms all baselines across all five quintiles, with the largest margin on the hardest problems (Q5: +13.4 pp over RaR).

4.2.2 Think Longer and Better

Why gains vary across benchmarks. We correlate the average number of reasoning steps per benchmark with SRaR's improvement over RaR (Figure 3, left). The correlation is strong (Pearson $r = 0.91$, $p = 0.012$): benchmarks requiring more steps (AIME 2025: 3.6, Olympiad: 3.0) benefit substantially more than those with shorter solutions (Minerva: 1.6).

Harder problems benefit more. We stratify Olympiad Bench problems into difficulty quintiles by base model response length (Figure 3, right). SRaR outperforms RaR in all quintiles, with the gap widening from +0.7 pp (Q1) to +13.4 pp (Q5). On Q5, SRaR also generates shorter responses (3,645 vs. 5,933 tokens), confirming gains from effective reasoning rather than exhaustive self-revision.

Table 2: Step-level reasoning quality analysis on AIME 2025. **Step Acc.** is the percentage of steps judged correct. The four combination columns show counts per (answer $\times$ step correctness) category. **FRR** (Faithful Reasoning Rate) is A✓S✓ / total (↑), the fraction of trajectories with both a correct answer and all-correct steps. **MTR** (Misaligned Trajectory Rate) is (A✓S× + A×S✓) / total (↓), the fraction of trajectories whose step-level judgement disagrees with the outcome, and serves as the population-level analogue of the misalignment in Figure 2(b).

Model	Ans.	Step	Answer × Step				FRR	MTR
	Acc.	Acc.	A✓S✓	A✓S×	A×S✓	A×S×	(%)↑	(%)↓
Qwen3-8B	13.3	60.8	4	**0**	4	22	13.3	<u>13.3</u>
+ RaR	<u>41.4</u>	<u>80.8</u>	<u>10</u>	2	<u>4</u>	<u>13</u>	<u>34.5</u>	20.7
+ SRaR (Ours)	**46.7**	**85.2**	**14**	**0**	**3**	**13**	**46.7**	**10.0**

Table 3: Repetitive looping analysis on AIME 2025. **Loop Rate** is the percentage of samples exhibiting any form of looping. Accuracy is reported for looping vs. non-looping responses. **Mean S-C** denotes the average count of self-correction phrases per response.

Model	Loop Rate (%)	Acc. (loop)	Acc. (no loop)	Mean S-C
Qwen3-8B	56.8	25.9	16.1	**0.9**
+ RaR	48.1	<u>24.2</u>	**53.4**	25.4
+ SRaR (Ours)	**26.5**	**42.5**	<u>52.8</u>	<u>14.8</u>

4.3 Analysis

4.3.1 Process-Outcome Alignment of Step-wise Rewards

To examine whether step-level reward assignment improves intermediate reasoning quality beyond final-answer accuracy, we conduct a diagnostic on AIME 2025 (single-pass). For each model, we parse solutions into steps and an independent LLM judge evaluates each step's correctness conditioned on preceding steps (prompt in Appendix C.5).

As shown in Table 2, the base model reaches only 60.8% step accuracy, confirming that outcome-only supervision fails to prevent error accumulation. RaR raises step accuracy to 80.8% but still has 2 correct-answer trajectories with erroneous steps (A✓S×=2) and 4 incorrect-answer trajectories with all-correct steps (A×S✓=4). SRaR achieves the highest step accuracy (85.2%) and answer accuracy (46.7%), and is the only model with A✓S×=0: every correct-answer trajectory uses entirely correct reasoning. SRaR also attains the highest Faithful Reasoning Rate (46.7% vs. 34.5%) and the lowest Misaligned Trajectory Rate (10.0% vs. 20.7%), showing that step-level rewards align intermediate quality with outcome correctness. One RaR trajectory is excluded because it does not contain a parseable reasoning trace, giving effective totals of 30, 29, and 30 trajectories for Qwen3-8B, RaR, and SRaR respectively. Case studies are in Appendix K.1.

4.3.2 Reduction of Repetitive Looping

A common failure mode of RLVR-trained models is repetitive looping: unbounded self-correction cycles that revisit computations without converging (case study in Appendix K.2). We quantify this on 960 AIME 2025 samples (30 problems, 32 rollouts each), classifying a response as looping if any of four criteria holds: (i) > 20 self-correction phrases; (ii) `### Step 1` appearing more than once; (iii) repeated step headings; or (iv) $> 10\%$ duplicate paragraphs.

As shown in Table 3, RaR's looping rate is 48.1% with a mean of 25.4 self-correction phrases, and its looping samples achieve only 24.2% accuracy vs. 53.4% for non-looping ones. SRaR reduces the looping rate to 26.5% with mean self-correction count 14.8, and its looping samples reach 42.5% accuracy, suggesting productive reflection rather than uncontrolled repetition.

We attribute this to step-level credit assignment: each step receives an independent reward, so the model can commit to completed steps rather than perpetually revisiting them.

4.4 Ablation Study

Rubric aspect ablation. We test whether PITFALL and BONUS provide value beyond Suggest-only. As shown in Table 10, SRaR outperforms RaR under every single-aspect configuration and under the full rubric set (+2.86 SUGGEST-only, +4.28 PITFALL-only, +2.89 BONUS-only, +3.57 multi-aspect), confirming the primary gain comes from step-wise credit assignment rather than any one rubric aspect. Each aspect also supervises effectively on its own (71.39, 71.97, and 70.89 respectively), and combining all three raises accuracy to 72.53, indicating complementary supervision beyond positive guidance alone.

Method design ablation. We ablate three components on Qwen3-8B (Table 11 in Appendix N). Removing the step-structured format prompt yields 67.85 average accuracy, matching standard DAPO (67.26), since step boundaries are required for attribution. Removing cross-rollout step normalization causes collapse to 56.76: without normalization, group-invariant rubric signals dominate the gradient, and the model degenerates to a single step (Appendix O). Replacing the decoupled advantage with a merged RaR-style scalar yields 72.06, slightly below full SRaR (72.53), because merging injects rubric variance into the GRPO baseline statistics.

5 Conclusion

We present *Step-wise Rubrics as Rewards (SRaR)*, an RLVR framework that delivers fine-grained supervision during RL training via step-level rubrics. Through *step-attributed rubric judging* and *cross-rollout step normalization*, each reasoning step receives a step-wise rubric reward, while a *decoupled advantage estimator* keeps RL training stable against rubric noise. On six mathematical benchmarks, *SRaR* outperforms the state-of-the-art by **3.57** points on Qwen3-8B and **2.75** points on Qwen3-32B, raises the Faithful Reasoning Rate from 34.5% to 46.7%, and reduces self-correction looping from 48.1% to 26.5%. We hope *SRaR* provides a practical paradigm for delivering fine-grained rubric supervision to RL-trained reasoners, showing that the granularity at which supervision is applied matters as much as its content. We discuss limitations and future directions in Appendix B.

Acknowledgments and Disclosure of Funding

This work was supported by JD.com, Peking University, and Unipat AI. Apart from the support acknowledged above and the institutional affiliations disclosed in this paper, the authors declare no other competing interests related to this work.

References

Baolong Bi, Shenghua Liu, Yiwei Wang, Siqian Tong, Lingrui Mei, Yuyao Ge, Yilong Xu, Jiafeng Guo, and Xueqi Cheng. Reward and guidance through rubrics: Promoting exploration to improve multi-domain reasoning. *arXiv preprint arXiv:2511.12344*, 2025.

Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Yiping Bao, et al. Babyvision: Visual reasoning beyond language. *arXiv preprint arXiv:2601.06521*, 2026a.

Zirong Chen, Fuda Ye, Enjun Du, Junfu Pu, Xinlei Wang, Xinyu Zuo, Lisheng Duan, Haijin Liang, Jin Ma, Jiachuan Wang, and Yongqi Zhang. Learning multimodal embeddings with evidence-aligned readout. *arXiv preprint arXiv:2609.33659*, 2026b.

Zirong Chen, Fuda Ye, Kuan Zhang, Enjun Du, Junfu Pu, Xinlei Wang, Xinyu Zuo, Lisheng Duan, Jin Ma, and Yongqi Zhang. Snapbench: Benchmarking snap-and-ask multimodal retrieval for mobile interactions. *arXiv preprint arXiv:2608.29607*, 2026c.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Nature*, 645:633–638, 2025.

Shuai Dong, Siyuan Wang, Xingyu Liu, Chenglin Li, Haowen Hou, and Zhongyu Wei. Interleaved latent visual reasoning with selective perceptual modeling. *arXiv preprint arXiv:2512.05665*, 2025.

Shuai Dong, Yongfu Zhu, Yuqi Xu, Weichu Xie, Ziyue Wang, Kaiwen Tuo, Congcong Wang, Siyuan Wang, Wenqi Shao, Shuai Yang, et al. Rl starts before rl: On policy distillation for better reinforcement learning. *arXiv preprint arXiv:2609.28145*, 2026.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

Helia Hashemi, Jason Eisner, Corby Rosset, Benjamin Van Durme, and Chris Kedzie. LLM-Rubric: A multidimensional, calibrated approach to automated evaluation of natural language texts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13806–13834, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng 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*, 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. *Advances in Neural Information Processing Systems*, 34:7294–7305, 2021.

Jing Jin, Xu Liu, Te Gao, Zhihong Shi, Yixiong Liang, Ruiqing Zheng, Hulin Kuang, Min Zeng, and Shichao Kan. Dynamic residual encoding with slide-level contrastive learning for end-to-end whole slide image representation. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8389–8398, 2025.

Jing Jin, Hao Liu, Yan Bai, Yihang Lou, Zhenke Wang, Tianrun Yuan, Juntong Chen, Yongkang Zhu, Fanhu Zeng, Xuanyu Zhu, et al. Unveiling fine-grained visual traces: Evaluating multimodal interleaved reasoning chains in multimodal stem tasks. *arXiv preprint arXiv:2604.19697*, 2026.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 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.

Boyi Li, Yifan Shen, Yuanzhe Liu, Yifan Xu, Jiateng Liu, Xinzhuo Li, Zhengyuan Li, Jingyuan Zhu, Yunhan Zhong, Fangzhou Lan, et al. Toward cognitive supersensing in multimodal large language model. *arXiv preprint arXiv:2602.01541*, 2026a.

Mingyuan Li, Guangsheng Yu, Xu Wang, and Shaoxiong Ji. Frozen memory is not enough: Rethinking external memory as extraction. *arXiv preprint arXiv:2608.17050*, 2026b.

Mingyuan Li, Guangsheng Yu, Juyuan Zhang, Xu Wang, Zhibo Man, Haonan Zhang, and Shaoxiong Ji. Memoryathena: Adaptive routing over latent and generated memories. *arXiv preprint arXiv:2609.25853*, 2026c.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Mengmeng Liu, Diankun Zhang, Jiuming Liu, Jianfeng Cui, Hongwei Xie, Guang Chen, Hangjun Ye, Francesco Nex, Hao Cheng, and Michael Ying Yang. Universe: Unified video action models for autonomous driving with flexible mask-modulated modality generation. *arXiv preprint arXiv:2607.05133*, 2026a.

Mengmeng Liu, Diankun Zhang, Jiuming Liu, Jianfeng Cui, Hongwei Xie, Guang Chen, Hangjun Ye, Michael Ying Yang, Francesco Nex, and Hao Cheng. Driveva: Video action models are zero-shot drivers. In *European Conference on Computer Vision*, pages 315–335. Springer, 2026b.

Wenpu Liu, Yuqi Xu, Weichu Xie, Yongfu Zhu, Shuai Dong, Ziyue Wang, Wenqi Shao, Xiaoying Zhang, Tong Yang, Nan Duan, et al. Leveraging error diversity in group rollouts for reinforcement learning. *arXiv preprint arXiv:2605.17333*, 2026c.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.

OpenAI. Learning to reason with LLMs. `https://openai.com/index/learning-to-reason-with-llms/`, 2024.

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

Yifan Shen, Yuanzhe Liu, Jingyuan Zhu, Xu Cao, Xiaofeng Zhang, Yixiao He, Wenming Ye, James Rehg, and Ismini Lourentzou. Fine-grained preference optimization improves spatial reasoning in vlms. *Advances in Neural Information Processing Systems*, 38:17929–17960, 2026a.

Yifan Shen, Jian Xu, Boyi Li, Yuner Zhang, Tianjiao Yu, Bingxuan Li, Houze Yang, Rushi Wang, and Xu Cao. Chronovision: Temporal reasoning via latent state reconstruction. *arXiv preprint arXiv:2608.05631*, 2026b.

Shuzheng Si, Haozhe Zhao, Yu Lei, Qingyi Wang, Dingwei Chen, Zhitong Wang, Zhenhailong Wang, Kangyang Luo, Zheng Wang, Gang Chen, Fanchao Qi, Minjia Zhang, and Maosong Sun. From context to skills: Can language models learn from context skillfully?, 2026. URL `https://arxiv.org/abs/2604.27660`.

Meituan LongCat Team. Longcat-next: Lexicalizing modalities as discrete tokens, 2026. URL `https://arxiv.org/abs/2603.27538`.

Jingyi Wang, Lei Zhu, Tengjin Weng, Song-Li Wu, Haochen Tan, Jierun Chen, Chaofan Tao, Haoli Bai, Lu Hou, Lifeng Shang, and Xiao-Ping Zhang. GRPO-VPS: Enhancing group relative policy optimization with verifiable process supervision for effective reasoning. *arXiv preprint arXiv:2604.20659*, 2026a.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Ziyue Wang, Aomufei Yuan, Yongfu Zhu, Shuai Dong, Wenpu Liu, Yiran Yao, Weichu Xie, Yuqi Xu, Caoyuan Ma, Wenqi Shao, et al. Right makes might: Aligning verified hidden states empowers rl reasoning. *arXiv preprint arXiv:2606.03234*, 2026b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, 1992.

Xi Xiao, Xingjian Li, Cheng Han, Tianyang Wang, Lin Zhao, Yunbei Zhang, Guosheng Hu, Runmin Jiang, Xi Li, Xiao Wang, and Min Xu. Adapting vision foundation models with cascaded semantics. *Transactions on Machine Learning Research*, 2026a. ISSN 2835-8856. URL `https://openreview.net/forum?id=SSsobNZJPO`.

Xi Xiao, Tianchen Zhao, Youngeun Kim, Zhuowei Li, Linghan Xu, Jiaye Wu, Zheng Zhang, Xiang Xu, Xuanbai Chen, Farhan Tejani, et al. Rethinking latent visual reasoning: Grounding latent reasoning in visual evidence. *arXiv preprint arXiv:2609.34563*, 2026b.

Yuqi Xu, Rizhen Hu, Zihan Liu, Mou Sun, and Kun Yuan. Grouter: Decoupling routing from representation for accelerated moe training. *arXiv preprint arXiv:2603.06626*, 2026.

An Yang, Anfeng Yang, Baosong Yang, Beichen Bi, Binyuan Hui, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Haozhe Zhao, Zefan Cai, Shuzheng Si, Liang Chen, Jiuxiang Gu, Wen Xiao, Minjia Zhang, and Junjie Hu. Mentor: Efficient multimodal-conditioned tuning for autoregressive vision generation models. *arXiv preprint arXiv:2507.09574*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2023.

A Notation

Table 4 summarizes the notation used throughout the paper.

Table 4: Summary of notation.

Symbol	Description
$x \in \mathcal{X}$	Problem prompt; $\mathcal{X}$ denotes the space of problem prompts
$y = (y_1, \ldots, y_T) \in \mathcal{V}^T$	Response token sequence of length T over the vocabulary $\mathcal{V}$
$\mathcal{D} = \{(x, a^*, \mathcal{R}_x)\}$	Training set of problems with answers and rubrics
a^*	Ground-truth answer of problem x
$\mathcal{R}_x = \{(c_j, \tau_j)\}_{j=1}^{M}$	Rubric of problem x with M items
c_j	Criterion text of rubric item j
τ_j	Type of rubric item j, one of `SUGGEST`, `PITFALL`, `BONUS`
π_θ	Policy model with parameters θ
n	Number of rollouts sampled per prompt (group size)
r_i	Scalar outcome reward of rollout i
A_i	Group-normalized (GRPO) advantage of rollout i
K	Number of reasoning steps in a response
(a_k, b_k)	Token span of step k
$\mathcal{J}$	LLM judge
$s_j \in \{0, 1\}$	Whether rubric item j is satisfied
$k_j \in \{1, \ldots, K\}$	Step to which rubric item j is attributed
δ_j	Signed reward delta of rubric item j
$R_\tau \in \mathbb{R}$	Signed reward budget for type τ; $R_{\texttt{SUG}}, R_{\texttt{BON}} > 0$, $R_{\texttt{PIT}} < 0$
N_τ	Number of rubric items of type τ in the current sample
$d_{k,i}$	Raw rubric reward of step k in rollout i
$\mathcal{G}_k$	Rollouts with at least one rubric item attributed to step k
μ_k, σ_k	Mean and standard deviation of $d_{k,i}$ over $\mathcal{G}_k$
$\bar{d}_{k,i}$	Step-wise cross-rollout normalized rubric reward
$\tilde{r}_i^{(t)}$	Token-level rubric offset at position t of rollout i
$r_{\text{acc},i}, r_{\text{fmt},i} \in \{0, 1\}$	Accuracy and format rewards of rollout i
λ	Mixing weight between accuracy and format rewards
$r_{\text{base},i}$	Base outcome reward $(1 - \lambda)\, r_{\text{acc},i} + \lambda\, r_{\text{fmt},i}$
$\mu_{\text{base}}, \sigma_{\text{base}}$	Mean and standard deviation of $r_{\text{base},i}$ across rollouts of the same prompt
$\hat{A}_i^{(t)}$	Final token-level advantage at position t of rollout i
ϵ	Small constant for numerical stability

B Limitations

We identify the following limitations of our work:

1. **Dependence on the LLM judge.** SRaR relies on an external LLM judge to evaluate rubric satisfaction and attribute each item to a specific reasoning step. Errors in step attribution or satisfaction judgments propagate into the reward signal. Although step-wise cross-rollout normalization mitigates the impact of systematic biases, judge inaccuracies on individual samples may still introduce noise into training.
2. **Additional inference cost.** Each training iteration requires invoking the LLM judge on every rollout, increasing wall-clock time and computational cost compared to standard GRPO or trajectory-level rubric methods. This cost scales linearly with the number of rollouts and rubric items per prompt.
3. **Domain scope.** Our experiments cover mathematical reasoning and code generation (Appendix P). While the framework is domain-agnostic in principle, we have not verified its effectiveness on other multi-step reasoning tasks such as scientific reasoning, tool use, factual multi-hop QA, or open-ended generation, where answer verification and step boundaries are less clean. The rubric generation pipeline may require domain-specific adaptation.
4. **Rubric generation relies on a strong model.** The three-stage data pipeline uses a frontier model (GPT-5) to generate rollouts and distill rubrics. The quality of the resulting rubrics is bounded

by the capabilities of this model, and the pipeline's applicability to settings where only weaker models are available remains unexplored.

5. **Single-model rubric bias.** Because all rubrics are distilled from one strong model, they may be biased toward that model's preferred reasoning patterns and inherit its systematic omissions. We mitigate single-solution bias by sampling multiple independent solutions without a step-format instruction and asking the model to compare and synthesize them, so a rubric is not tied to the proof structure of any single sampled trajectory. Model-level bias may nonetheless remain, particularly when alternative valid proof strategies are underrepresented among the sampled solutions. Multi-model rubric generation and human validation are natural remedies that we leave to future work.

6. **Step boundary detection.** SRaR assumes that the policy model produces responses with explicit step headers (`### Step N`). This requires a format prompt during training and may not generalize to free-form reasoning without structural delimiters, and SRaR remains untested on less structured reasoning where no comparable segmentation interface exists.

7. **Step atomicity is not enforced.** SRaR treats each formatted step as the unit of credit assignment but does not verify that a step corresponds to a single atomic reasoning operation. A policy could in principle pack several atomic operations into one formatted step; this would make the assigned reward coarser, since all tokens in that span share one step-level signal, and would therefore erode the benefit of fine-grained credit assignment rather than inflate the reward. An atomicity-aware judge or an auxiliary penalty on overly composite steps is a promising extension.

C Prompt Details

This section provides the full prompts used in our data pipeline and training procedure.

C.1 Rollout Generation Prompt

Prompt for Rollout Generation

Solve the following math problem step by step. Please strictly follow these formatting instructions:

1. **Step Format:** Organize your reasoning into clear steps. Start each step with the header "### Step N:" (e.g., ### Step 1:, ### Step 2:).
2. **Final Answer:** The last line of your response must be enclosed in tags like this: "<answer>Answer</answer>", where Answer is the calculated result.

Problem: `{Problem}`

C.2 Rubric Distillation Prompt

Rubric Distillation Prompt Template

You are designing a grading rubric for math problem solutions.
The original problem statement is: `{problem_text}`
You are given n model solutions with a correctness label. For each numbered solution: `{solutions_block}`
The true correct answer for this problem is: <ANSWER>`{gold_answer}`</ANSWER>
Based on the samples and the true answer, create a concise, verifiable rubric. Output ONLY structured items using the tags below (one item per line):

- <**SUGGEST**>: Mandatory. One scoring checkpoint per line. Each checkpoint checks whether the candidate CoT covers a key reasoning step from the standard solution path. Make it specific and objectively checkable from the text.
- <**PITFALL**>: Optional. Only include if you are confident. Flags reasoning traps or common errors to avoid.
- <**BONUS**>: Optional. Only include if you are confident. Rewards especially clever, efficient, or elegant reasoning patterns.
- <**ANSWER**>: Mandatory. One line that checks whether the final answer matches the ground truth above (allow equivalent numeric forms).

Guidelines:

- Select only the most essential scoring points; total lines ≤ 8 (SUGGEST/ANSWER mandatory; PITFALL/BONUS only if sure). Do NOT pad to the limit; stop once the main reasoning chain is covered.
- Each <SUGGEST> is exactly one scoring point and must be independently verifiable by an LLM judge from the candidate response.
- Do NOT include formatting or output-structure checks (e.g., headers, tags, markdown). Focus solely on mathematical / logical content steps and correctness of the final answer.
- Do not reference the given samples explicitly; generalize the rubric.
- Keep wording concise and actionable; no extra commentary before or after.

C.3 Training Format Prompt

Format Prompt

Solve the following math problem step by step. Follow these formatting rules:

1. **Steps:** Break your solution into multiple clear steps. Begin each step with "`### Step N:`" (e.g., `### Step 1:`, `### Step 2:`, `### Step 3:`).
2. **Final Answer:** End your response with the answer inside `\boxed{}`.

Problem: `{Problem}`

C.4 LLM Judge Prompt

LLM Judge Prompt

Problem
`{problem}`

Rubric Items
`{rubric_items}`

Student Solution
`{response}`

Student's Final Answer
`{extracted_answer}`

Task
For every rubric item listed above, evaluate the student's solution.
Rules:

- SUGGEST → `satisfied`: true if the student correctly performed that reasoning step.
- PITFALL → `satisfied`: true if the student made that mistake (fell into the pitfall).
- BONUS → `satisfied`: true if the student used that approach.

For `step`: the 1-indexed step number (### Step N) most closely associated with this item. Use 0 if it spans the whole solution; use -1 if there is no relevant step.

Return ONLY a valid JSON array:

```
[
   {"id": <int>, "satisfied": <bool>, "step": <int>},
   ...
]
```

C.5 Step-Level Judge Prompt

Step-Level Correctness Judge Prompt

You are a math reasoning step verifier. Given a math problem and a sequence of reasoning steps, determine whether the CURRENT STEP is logically correct.

Problem
`{question}`

Ground Truth Answer
`{answer}`

Previous Steps (assumed correct for this evaluation)
`{previous_steps}`

Current Step to Evaluate
`{current_step_title}`
`{current_step_content}`

Instructions
Analyze whether the current step contains correct reasoning, calculations, and logic. Consider:

1. Are the mathematical operations correct?
2. Is the logic/deduction valid?
3. Are there any errors in the step?

Respond with EXACTLY one of:

- "CORRECT" if the step is logically and mathematically correct
- "INCORRECT" if the step contains any error

D Training Hyperparameters

Table 5: Training Hyperparameters.

Hyperparameter	Value
Max Prompt Length	2048
Max Response Length	8192
Train Batch Size	128
Learning Rate	1×10^{-6}
Rollout Group Size (n)	8
Training Steps	200
Temperature	1.0

E Training Dynamics

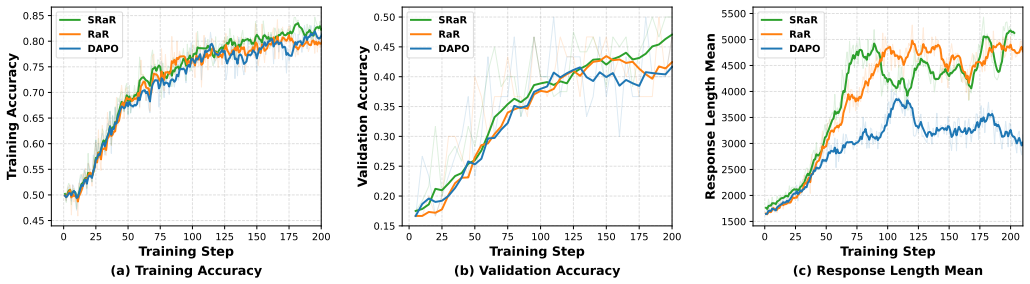


Figure 4: Training dynamics of DAPO, RaR, and SRaR on Qwen3-32B-Non-Thinking. **(a)** Training reward accuracy. **(b)** Validation accuracy. **(c)** Mean response length over training.

Figure 4 illustrates the training dynamics of different methods on Qwen3-32B-Non-Thinking. As shown in Figure 4(a), all three methods converge to similar training reward accuracy ($\sim$0.80–0.83), with SRaR reaching marginally higher values, indicating that step-level credit assignment provides effective gradient signals even at larger scale. Figure 4(b) reports validation accuracy. SRaR consistently achieves the highest validation accuracy throughout training, reaching $\sim$0.47 at convergence compared to RaR ($\sim$0.43) and DAPO ($\sim$0.42), confirming that the training gains translate to held-out performance. Meanwhile, Figure 4(c) shows that both RaR and SRaR produce longer responses than DAPO, with similar length trajectories. Despite comparable response lengths, SRaR achieves substantially higher accuracy (Figure 4(b)), indicating that its gains stem from more effective reasoning per token rather than simply generating more content. The per-aspect rubric score dynamics (Suggest, Bonus, and Pitfall) are provided in Figure 6.

F Format Compliance During Training

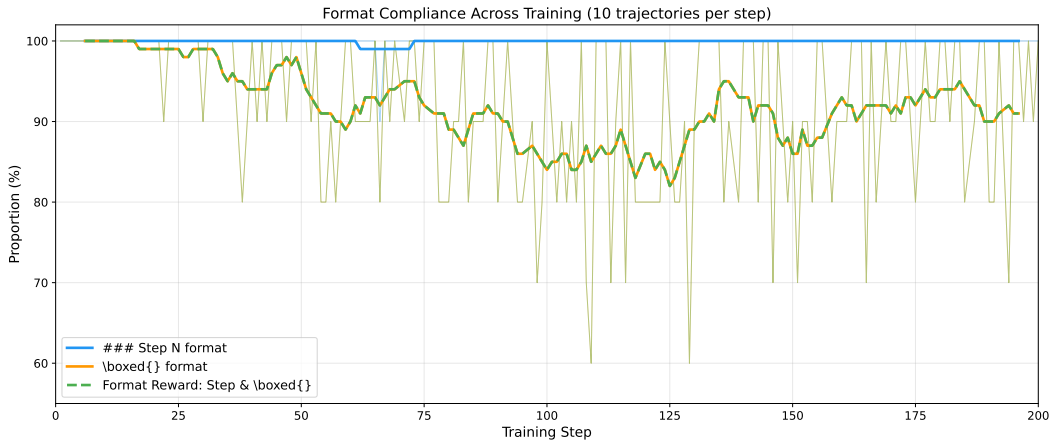


Figure 5: Format compliance across training steps for Qwen3-8B-Non-Thinking. At each training step, 10 rollouts are sampled and checked for (1) `### Step N` headers, (2) `\boxed{}` answer, and (3) the conjunction of both (i.e., the format reward). The step-header format is maintained at near-100% throughout training. The occasional drop in `\boxed{}` compliance is attributable to response truncation at the maximum generation length, which prevents the model from emitting a final boxed answer.

G Rubric Quality Evaluation

To assess the quality of rubrics produced by our pipeline, we conducted an automated validity evaluation on 51,583 rubric items spanning ∼8,000 problems, using Gemini-3-Flash as the LLM judge. Each item was independently judged as valid or invalid with a brief justification. **Overall validity.** After this treatment, **98.0%** of evaluated rubric items are judged valid, with only 1,027 items (2.0%) flagged as invalid. **Validity by rubric type.** Table 6 shows validity rates broken down by rubric aspect. All three types achieve high and consistent validity rates in the range 96.3%–98.5%, confirming that the pipeline generates reliable rubrics across all aspects.

Table 6: Rubric validity rates by type, evaluated over 51,583 items from ∼8,000 problems.

Rubric Type	Total	Valid	Validity Rate
SUGGEST	36,532	35,994	98.5%
PITFALL	9,822	9,458	96.3%
BONUS	5,213	5,089	97.6%
Overall	**51,583**	**50,556**	**98.0%**

Human–AI agreement. Two of the authors independently annotated a 100-item cross-check sample using the same prompt below, achieving 92% agreement with the LLM judge.

Rubric Validity Judge Prompt

You are a mathematics expert evaluating the quality of a rubric item for a math problem.
Math Problem
`{problem}`
Ground Truth Answer
`{ground_truth}`
Rubric Item
Type: `{rubric_type}`
Content: `{rubric_text}`
Rubric Type Definitions

- **SUGGEST**: A recommended reasoning step or approach for solving the problem.
- **PITFALL**: A common mistake or error that students might make.
- **BONUS**: An alternative valid approach or insight.
- **ANSWER**: The expected final answer.

Your Task
Judge whether this rubric item is **valid** for the given math problem:

1. Is the content mathematically correct?
2. Is it relevant to this specific problem?
3. Is the type label (SUGGEST/PITFALL/BONUS/ANSWER) appropriate?

Respond with ONLY a JSON object (no markdown, no extra text):
`{"valid": true or false, "reason": "brief explanation"}`

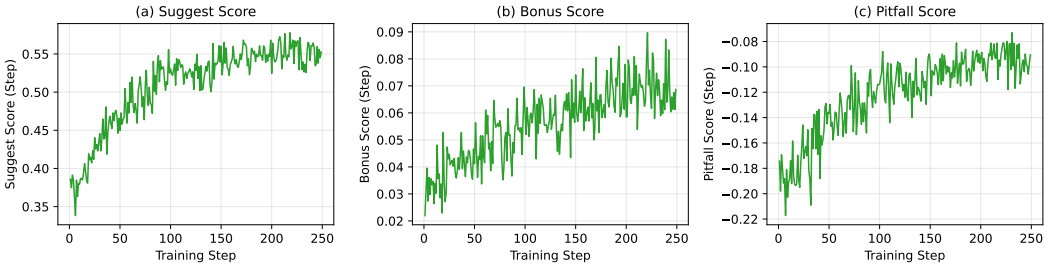


Figure 6: Step-level rubric score dynamics of SRaR during training on Qwen3-8B-Non-Thinking. **(a)** Suggest score steadily increases, indicating the model increasingly follows recommended reasoning steps. **(b)** Bonus score rises as the model discovers alternative solution strategies. **(c)** Pitfall score increases (toward zero), showing the model progressively avoids common errors.

H LLM Judge Agreement Analysis

A core component of SRaR is the LLM judge that evaluates each rubric item against the model's response and attributes it to a specific reasoning step. To validate the reliability of this judge, we conduct an agreement study on 100 training trajectories sampled uniformly across training steps 20–200 (10 trajectories per checkpoint, totaling 664 rubric items).

Setup

We use GPT-OSS-20B as the LLM judge (identical to the judge used during training) and compare its outputs against independent human annotations by two of the authors on the same 664 rubric items. For each item, both the LLM judge and the human annotators produce a binary satisfied judgment and a step attribution (the step number most closely associated with the rubric item).

Agreement on Rubric Satisfaction

Table 7 reports the agreement between the two independent evaluations.

Table 7: LLM judge agreement on 664 rubric items from 100 training trajectories.

Metric	Value
Satisfied agreement rate	89.3%
Satisfied Cohen's κ	0.76
Step attribution agreement	80.7%

The two evaluations agree on 89.3% of rubric items with a Cohen's κ of 0.76, corresponding to substantial agreement on the Landis–Koch scale. For step attribution, when both evaluations assign a specific step number, they agree 80.7% of the time.

Judge Stochasticity

The agreement above measures whether the judge matches human annotators. A separate question is whether the judge is self-consistent, since an unstable judge would inject noise into the reward regardless of its accuracy. We therefore re-evaluate each of 10 fixed trajectories 10 times with the same judge and prompt, and measure how often repeated evaluations of the same trajectory agree. Rubric-satisfaction judgments agree at a rate of 0.983 and step attributions at 0.997. Judge stochasticity is thus small enough that it cannot account for the differences between RaR and SRaR, which use identical rubrics and the same judge.

Rubric Scores and Answer Correctness

We further analyze the relationship between rubric satisfaction rates and final answer correctness across the 100 trajectories (Table 8).

Table 8: Rubric satisfaction rates for correct vs. incorrect trajectories (100 trajectories, 664 rubric items). Point-biserial correlations are between binary correctness and satisfaction rate.

Rubric Type	Correct	Incorrect	Diff.	r_{pb}	p-value
SUGGEST satisfied	74.2%	24.7%	+49.5 pp	0.62	<0.001
PITFALL triggered	2.5%	24.2%	−21.7 pp	−0.31	0.002
BONUS satisfied	12.5%	0.0%	+12.5 pp	0.28	0.005

All three rubric aspects are significantly correlated with answer correctness ($p < 0.01$). Correct trajectories satisfy 74.2% of SUGGEST items (vs. 24.7% for incorrect ones), trigger only 2.5% of PITFALL items (vs. 24.2%), and satisfy 12.5% of BONUS items (vs. 0%). This confirms that the rubric scores produced by the LLM judge are meaningful indicators of reasoning quality and are well-aligned with outcome correctness.

I Judge Model Robustness

To validate that SRaR is not dependent on a specific LLM judge, we replace the default GPT-OSS-20B judge with two alternative open-source models while keeping all other training hyperparameters identical: Qwen3-30B-A3B-Instruct, a judge of comparable capacity, and Qwen3-4B, a substantially weaker judge that probes how much judge capacity SRaR actually requires. Table 9 reports the results on Qwen3-8B-Non-Thinking.

With Qwen3-30B-A3B-Instruct as the judge, SRaR still achieves a $+1.90$ average improvement over RaR (70.86 vs. 68.96), demonstrating that the step-wise reward framework is effective with a different judge model. The gap relative to the GPT-OSS-20B judge (-1.67 avg.) is modest, suggesting that judge choice primarily affects the magnitude of improvement rather than the viability of the approach. This is consistent with the expectation that different judges produce varying step attributions, which are partially mitigated by cross-rollout normalization (noise that is uncorrelated across rollouts averages out during normalization).

Dropping to Qwen3-4B, a judge roughly an order of magnitude smaller than the default, costs only 1.53 points on average (71.00 vs. 72.53) and still improves over RaR by $+2.04$. The degradation is

Table 9: SRaR performance on Qwen3-8B-Non-Thinking with different LLM judges. All three judges produce improvements over the RaR baseline (68.96 avg.), confirming that SRaR's gains are robust to the choice of judge model.

Judge Model	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
RaR (no step-level)	57.50	43.23	88.20	92.6	64.71	67.51	68.96
SRaR w/ GPT-OSS-20B	**61.15**	**49.06**	**92.73**	**94.4**	64.71	**73.15**	**72.53**
SRaR w/ Qwen3-30B-A3B	57.81	46.04	91.25	93.20	**66.54**	70.33	70.86
SRaR w/ Qwen3-4B	**61.15**	47.60	92.11	**95.00**	58.46	71.67	71.00

not uniform across benchmarks: the weaker judge matches or exceeds the default judge on AIME24 (61.15) and MATH500 (95.00) while losing ground mainly on MinervaMath (58.46 vs. 64.71). SRaR therefore does not require a particularly strong judge to be effective, though judge capacity does set the ceiling on how much of the available gain is realized.

J Rubric Examples

We present two representative examples from our curated training set to illustrate the structure and quality of the generated rubrics. Each problem is accompanied by a rubric list consisting of SUGGEST, PITFALL, BONUS, and ANSWER items produced by our three-stage pipeline.

Example 1

Problem. Given that nonzero real numbers x and y satisfy $x + \frac{1}{y} = 3$ and $y + \frac{1}{x} = 4$, find the value of $xy + \frac{1}{xy}$.

Ground Truth. 10

Generated Rubric

<SUGGEST> Multiplies the two equations to obtain $\left(x + \frac{1}{y}\right)\left(y + \frac{1}{x}\right) = 12$.
<SUGGEST> Expands the product correctly to $xy + 2 + \frac{1}{xy}$.
<SUGGEST> From $xy + 2 + \frac{1}{xy} = 12$, isolates and states $xy + \frac{1}{xy} = 10$.
<PITFALL> Misexpanding $\left(x + \frac{1}{y}\right)\left(y + \frac{1}{x}\right)$, especially omitting the $+2$ term from $x \cdot \frac{1}{x}$ and $\frac{1}{y} \cdot y$.
<BONUS> Computes $xy + \frac{1}{xy}$ directly via the product expansion without solving for x or y individually.
<ANSWER> Final value reported for $xy + \frac{1}{xy}$ equals 10.

Example 2

Problem. Let S be the set of triples (a, b, c) of non-negative integers such that $a + b + c$ is even. The value of $\sum_{(a,b,c)\in S} \frac{1}{2^a 3^b 5^c}$ can be expressed as $\frac{m}{n}$ where $\gcd(m, n) = 1$. Compute $m + n$.

Ground Truth. 37

Generated Rubric

<SUGGEST> Applies the parity indicator $\frac{1+(-1)^{a+b+c}}{2}$ to rewrite the restricted sum as $\frac{S_0+S_1}{2}$, where S_0 is the unrestricted sum and S_1 includes the factor $(-1)^{a+b+c}$.
<SUGGEST> Factors S_0 into $\sum \frac{1}{2^a} \cdot \sum \frac{1}{3^b} \cdot \sum \frac{1}{5^c}$ and evaluates to $2 \cdot \frac{3}{2} \cdot \frac{5}{4} = \frac{15}{4}$.
<SUGGEST> Factors S_1 into $\sum \frac{(-1)^a}{2^a} \cdot \sum \frac{(-1)^b}{3^b} \cdot \sum \frac{(-1)^c}{5^c}$ and evaluates to $\frac{2}{3} \cdot \frac{3}{4} \cdot \frac{5}{6} = \frac{5}{12}$.
<SUGGEST> Combines via $\frac{S_0+S_1}{2} = \frac{15/4+5/12}{2} = \frac{25}{12}$, giving $m+n=37$.
<PITFALL> Omits the $\frac{1}{2}$ factor in the parity projection, or uses $\frac{S_0-S_1}{2}$ (which sums over odd $a+b+c$) instead of $\frac{S_0+S_1}{2}$.
<BONUS> Uses the compact form $\frac{1}{2}\left(\frac{1}{(1-x)(1-y)(1-z)} + \frac{1}{(1+x)(1+y)(1+z)}\right)$ with $x=\frac{1}{2}, y=\frac{1}{3}, z=\frac{1}{5}$.
<ANSWER> Final reported $m+n$ equals 37.

K Case Studies

We provide representative case studies from AIME 2025 (repeat_id=0) to illustrate the qualitative differences between RaR and SRaR identified in our analysis.

K.1 Process–Outcome Alignment Case Study

We present two problems where RaR produces incorrect answers with step-level errors, while SRaR produces correct answers with all steps judged correct. **Case 1: AIME 2025 Problem 20.** Circle ω_1 with radius 6 centered at point A is internally tangent at point B to circle ω_2 with radius 15... The area of rectangle $EFGH$ is $\frac{m}{n}$... Find $m+n$. (Ground truth: 293.)

RaR Response (Problem 20) — Incorrect, Looping

`### Step 1: Set up coordinates`
We can place the center of ω_2 at the origin... Since the circles are internally tangent at point B... the center A of ω_1 lies along the line connecting the centers...

[The model enters an extended self-revision loop with 34 revision phrases across 30,922 characters. It repeatedly reconsiders the geometric constraints without converging. The response is cut off mid-computation without producing a `\boxed{}` answer.]

`...So the inequality becomes: 33 - p < p + 15...18`
[Response truncated — no final answer produced.]

SRaR Response (Problem 20) — Correct, All Steps Correct

`### Step 1: Understand the configuration and setup coordinates`
We are given two circles, ω_1 and ω_2... Let's place the center of ω_2 at the origin. Then $A = (9, 0)$, $O = (0, 0)$, $B = (15, 0)$.

`### Step 2: Find key points`
Since BC is a diameter of ω_2, $C = (-15, 0)$. Since $BC \perp AD$, D lies directly above A... $D = (9, 12)$.

`### Step 3: Compute the rectangle and verify equal areas`
... area of the rectangle is $288/5$, so $m = 288$, $n = 5$, $m + n = 293$.

`### Step 4: Final Answer`
`\boxed{293}`

Analysis. RaR's trajectory-level reward fails to prevent the model from entering an extended self-revision loop: it repeatedly reconsiders the geometric constraints without converging, ultimately exhausting the generation budget across 30,922 characters. In contrast, SRaR's step-wise reward allows the model to commit to each step and progress efficiently through the solution in four clean steps with only 6,640 characters. **Case 2: AIME 2025 Problem 24.** Sixteen chairs are arranged in a row. Eight people each select a chair in which to sit so that no person sits next to two other people.

Let N be the number of subsets... Find the remainder when N is divided by 1000. (Ground truth: 907.)

RaR Response (Problem 24) — Incorrect, Unstructured Looping

We are given 16 chairs arranged in a row... Wait, actually the problem says "no person sits next to two other people." Hmm, let me parse that carefully.

[The model never produces any `### Step` headers. Instead, it generates an unstructured stream of reasoning with 63 self-correction phrases across 33,378 characters. It repeatedly restarts the dynamic programming computation without converging. The response ends mid-computation:]

```
...k=6:  dp[7][6] + dp[6][5] + dp[5][4] = 0 + 0 + 1 = 1
k=7:  0
k=8:  0
n=8:
k=0:  1
k=1:  8
```

[Response truncated — no final answer produced.]

SRaR Response (Problem 24) — Correct, All Steps Correct

`### Step 1: Understand the problem`
We have 16 chairs arranged in a row. Eight people each select a chair... The condition means no person is adjacent to two other people, i.e., no three consecutive chairs are occupied.

`### Step 2: Count the valid subsets`
...our calculation of 2907 should be correct. Thus the answer is 907.

`### Step 3: Compute the final answer`
The remainder when N is divided by 1000 is `\boxed{907}`.

Analysis. Both models initially grapple with interpreting the constraint, but RaR degenerates into an unstructured stream of 63 self-corrections without ever organizing its reasoning into steps, ultimately truncating without an answer. SRaR organizes the solution into three clear steps, correctly interprets the constraint, and arrives at the right answer.

K.2 Looping Case Study

We present AIME 2025 Problem 13 as an illustrative example of the looping behavior analyzed in Section 4.3.2.

Let $ABCDE$ be a convex pentagon with $AB = 14$, $BC = 7$, $CD = 24$, $DE = 13$, $EA = 26$, and $\angle B = \angle E = 60°$. For each point X in the plane, define $f(X) = AX + BX + CX + DX + EX$. The least possible value of $f(X)$ can be expressed as $m + n\sqrt{p}$... Find $m + n + p$. (Ground truth: 60.)

RaR Response (Problem 13) — Severe Looping, No Answer

`### Step 1: Understanding the Problem`
The function $f(X)$ represents the sum of distances from a variable point X to each of the five vertices... This is the Fermat-Torricelli problem generalized to five points...

[The model enters a revision loop with 31 self-correction phrases across 23,874 characters. It repeatedly restarts the coordinate geometry computation without converging. Representative excerpts:]

```
...Wait, if DA is (a, b), then perpendicular vectors are (-b, a) and (b,
-a)...
...Wait, no:  Wait, if DA is (a, b)...
```

[The response ends mid-sentence without producing a `\boxed{}` answer.]

SRaR Response (Problem 13) — Structured, Produces Answer

`### Step 1: Understand the problem and identify key elements`
We are given a convex pentagon $ABCDE$ with specific side lengths and angles at B and E each equal to 60°. We need to find the minimum of $f(X) = AX + BX + CX + DX + EX \ldots$

`### Step 2: Apply geometric reasoning`
$\ldots$ the total minimal value might be $7\sqrt{3} + 13\sqrt{3} = 20\sqrt{3}$ plus other terms$\ldots$
[The model works through the problem with self-correction (11 phrases), but stays structured and converges to an answer.]

`\boxed{58}` (incorrect, but a complete structured response)

Analysis. This problem is difficult for both models (neither answers correctly), but the behavioral contrast is stark. RaR produces a response with 31 self-correction phrases, repeatedly restarting its coordinate computation and ultimately failing to produce any answer across 23,874 characters. SRaR, guided by step-wise rewards, maintains a clear two-step structure and converges to a definite (though incorrect) answer in 15,483 characters. This illustrates how step-wise rewards act as a structural prior that prevents degeneration into aimless revision, even on problems the model cannot ultimately solve.

L Data-Size Sensitivity

We analyze how model performance changes with the amount of synthetic data. Instead of varying task types, we vary only the number of synthetic rubrics training samples. Concretely, we train for exactly one epoch on subsets of size 1k, 2k, 4k, 8k, and 16k. All subsets are obtained by random sampling with a fixed seed, so improvements mainly reflect data amount rather than selection artifacts.

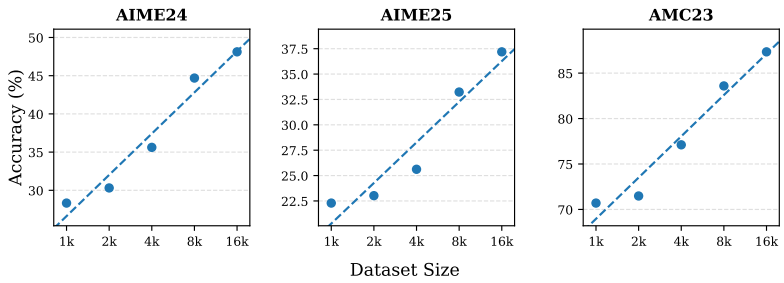


Figure 7: **Data-Size Effects of Synthetic Rubrics.** We report performance trends on AIME 2024, AIME 2025, and AMC 2023 as the amount of synthetic rubrics training data increases. "1k/2k/4k/8k/16k" are random subsets sampled with a fixed seed. The training recipe matches the main experimental setting. Results show a consistent upward trend across all benchmarks as synthetic data increases.

Figure 7 reports accuracy across three benchmarks as data size scales from 1k to 16k. Several observations stand out. First, performance improves monotonically on every benchmark: even the smallest 1k subset already yields meaningful gains, while each doubling of data continues to push accuracy higher, confirming that our synthetic rubrics pipeline produces consistently useful training signal. Second, the steepest improvement occurs between 4k and 8k samples. For instance, AIME25 jumps from 25.62% to 33.23%, and AMC23 rises from 77.11% to 83.59%, suggesting a critical mass of rubric diversity is reached in this regime. Third, at the full 16k scale, the curves have not yet plateaued. For example, AIME24 reaches 48.12%, AIME25 reaches 37.19%, and AMC23 reaches 87.34%, indicating that further scaling the synthetic data is likely to yield additional gains. Finally, the trend is consistent across benchmarks of different difficulty levels, from the easier AMC23 (70.70%→87.34%) to the harder AIME25 (22.29%→37.19%), demonstrating that the benefit of more synthetic rubrics is not confined to a narrow difficulty band.

M Rubric Aspect Ablation

We ablate the rubric aspects to disentangle the contribution of step-wise credit assignment from the rubric design. Table 10 compares RaR and SRaR under four rubric configurations on Qwen3-8B-Non-Thinking: three single-aspect settings that retain only SUGGEST, only PITFALL, or only BONUS items, and the full multi-aspect set. Within each configuration, RaR and SRaR use exactly the same rubric items, judge, training data, and training recipe.

Table 10: Rubric aspect ablation on Qwen3-8B-Non-Thinking. Δ denotes SRaR improvement over RaR within the same rubric configuration. Best results per configuration are **bolded**.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
SUGGEST-only rubrics							
RaR	53.23	41.98	**91.64**	93.2	61.40	69.73	68.53
SRaR	**58.65**	**46.98**	91.56	**93.6**	**66.18**	**71.36**	**71.39**
Δ	+5.42	+5.00	–0.08	+0.40	+4.78	+1.63	+2.86
PITFALL-only rubrics							
RaR	52.08	39.79	87.50	93.8	**66.18**	66.77	67.69
SRaR	**60.63**	**49.06**	**92.34**	**94.2**	62.14	**73.45**	**71.97**
Δ	+8.55	+9.27	+4.84	+0.40	–4.04	+6.68	+4.28
BONUS-only rubrics							
RaR	55.21	40.73	88.05	93.0	**63.97**	67.06	68.00
SRaR	**59.38**	**47.08**	**92.89**	**95.0**	60.67	**70.33**	**70.89**
Δ	+4.17	+6.35	+4.84	+2.00	–3.30	+3.27	+2.89
Full rubrics (SUGGEST + PITFALL + BONUS)							
RaR	57.50	43.23	88.20	92.6	**64.71**	67.51	68.96
SRaR	**61.15**	**49.06**	**92.73**	**94.4**	**64.71**	**73.15**	**72.53**
Δ	+3.65	+5.83	+4.53	+1.80	+0.00	+5.64	+3.57

Two conclusions emerge. First, SRaR outperforms RaR under every rubric configuration (+2.86 with SUGGEST-only, +4.28 with PITFALL-only, +2.89 with BONUS-only, +3.57 with full rubrics), confirming that the primary source of improvement is step-wise credit assignment rather than the specific rubric aspects. Second, each aspect provides meaningful supervision on its own (SRaR reaches 71.39, 71.97, and 70.89 with SUGGEST, PITFALL, and BONUS items alone), and the full combination is best at 72.53, indicating that the three aspects are complementary rather than redundant. The single-aspect variants trade benchmarks against one another: PITFALL-only and BONUS-only both give up ground on MinervaMath relative to their RaR counterparts while gaining substantially on AIME24/AIME25, which is consistent with error avoidance and exceptional-insight credit being sharper but narrower signals than the full multi-aspect schema.

Role of BONUS relative to SUGGEST. SUGGEST and BONUS carry the same positive sign in Eq. (3) because both mark desirable reasoning behaviour, but they cover different content. SUGGEST items are the core reasoning checkpoints expected of any standard valid solution, such as deriving a necessary intermediate result or carrying out an essential calculation. BONUS items are optional and reward reasoning that goes beyond the standard solution path: applying an advanced result, recognizing a compact identity, constructing a substantially shorter derivation, or supplying an alternative solution. BONUS credit is evaluated independently of final-answer correctness by design, since a response may use an elegant intermediate argument correctly and then make an unrelated error later; SRaR rewards the locally correct BONUS step in that case, which is precisely the behaviour trajectory-level aggregation cannot express.

N Method Design Ablation

We ablate three key design choices of SRaR on Qwen3-8B-Non-Thinking to isolate their individual contributions. Table 11 reports results across all six benchmarks.

Table 11: Component ablation on Qwen3-8B-Non-Thinking. Each variant removes one component from the full SRaR system. Best results are **bolded**.

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
Base (Qwen3-8B, w/ step format)	26.77	22.40	71.88	85.0	55.51	56.53	53.02
DAPO (w/o step format)	55.73	41.25	87.66	92.0	61.76	68.69	67.85
SRaR (w/o step norm.)	45.10	28.96	73.52	87.2	49.26	56.53	56.76
SRaR (w/o decoupled adv.)	**62.60**	46.67	**93.28**	**94.6**	**64.71**	70.47	72.06
SRaR (full)	61.15	**49.06**	92.73	94.4	**64.71**	**73.15**	**72.53**

Step-structured format prompt. Comparing DAPO trained without step format constraints (67.85 avg) against the full DAPO baseline in Table 1 (67.26 avg), the two are virtually identical, indicating that requiring the model to organize its chain-of-thought into explicit `### Step` blocks is a benign structural constraint that introduces no measurable performance penalty. This makes it a practical prerequisite for SRaR, which relies on step boundaries to attribute rubric items to individual reasoning steps. **Cross-rollout step normalization.** Removing step normalization causes severe training collapse (56.76 avg), performing even worse than the base model on several benchmarks. Without normalization, rubric items that are uniformly satisfied across all rollouts still produce large positive advantages. The model exploits this by collapsing all reasoning into a single `### Step 1`, maximizing rubric reward without distributing reasoning across multiple steps. We provide a representative example in Appendix O. Cross-rollout normalization prevents this by zeroing out the advantage for rubric items whose scores do not vary across rollouts, ensuring that only genuinely discriminative rubric items drive learning. **Decoupled advantage estimation.** Instead of maintaining separate outcome and rubric advantage streams, this variant computes a single RaR-style scalar reward (combining outcome correctness and rubric satisfaction) and then allocates it to the tokens of steps that received rubric scores. This approach yields competitive performance (72.06 avg) but falls short of the full SRaR (72.53 avg). The gap is most pronounced on harder benchmarks (AIME 2025: 46.67 vs. 49.06), suggesting that merging outcome and rubric signals into one scalar before allocation loses the benefit of keeping the outcome-driven baseline stable.

O Case Study: Training Collapse without Step Normalization

Below is a representative response from the SRaR (w/o step normalization) model after training collapse. The model places its entire chain-of-thought (spanning over 4,000 tokens of deliberation) inside a single `### Step 1`, never producing any subsequent step. This behavior emerges because, without cross-rollout normalization, rubric items that are consistently satisfied across all rollouts (e.g., "identify the modular invariant") still carry large positive advantages. By concentrating all reasoning in one step, the model ensures that this single step collects all rubric credit, removing any incentive to distribute reasoning across multiple steps.

SRaR (w/o step norm.) – Single-Step Collapse

`### Step 1: Understand the problem and analyze the operations`
We are given a set of numbers from 1 to 2004 written on a desk. A move consists of selecting some numbers, calculating the remainder of their sum modulo 11, writing this remainder on the desk, and deleting the selected numbers. After several such moves, only two numbers remain on the desk, one of which is 1000. We are to find the other number.
First, let's understand the invariant... Each move replaces some numbers with their sum modulo 11. So, if we consider the numbers on the desk as elements of the set, the key is that the sum of all numbers on the desk modulo 11 is preserved...
[... 4,000+ tokens of reasoning, all within Step 1, including repeated self-verification, alternative approaches, and redundant restatements...]
Therefore, the answer is 4.

```
### Final Answer
\boxed{4}
```

Analysis. The response contains a single reasoning step that spans the entire solution. While the final answer happens to be correct for this particular problem, the degenerate structure, including no step decomposition and excessive self-verification loops within one monolithic block, is a direct

consequence of the un-normalized rubric reward. In contrast, the full SRaR model with cross-rollout normalization produces well-structured multi-step solutions, because normalization zeros out the advantage for rubric items that do not vary across rollouts, leaving only genuinely discriminative items to shape the policy gradient.

P Generalization to the Code Domain

The main experiments are confined to mathematical reasoning. To test whether step-wise rubric rewards transfer to a domain with different step semantics and a different verification mechanism, we repeat the full pipeline on code generation. We take 12K problems from Code-R1, generate problem-specific rubrics with exactly the same three-stage contrastive distillation pipeline used for math (Section 3.4), and train DAPO, RaR, and SRaR under the same experimental setup, changing only the reward verifier to unit-test execution. Table 12 reports pass rates on four code benchmarks.

Table 12: Code-domain results on Qwen3-8B-Non-Thinking, trained on 12K Code-R1 problems with rubrics generated by the same pipeline as the math corpus. Best results are **bolded**.

Method	LiveCodeBench v6	HumanEval+	MBPP+	TACO
Qwen3-8B	22.93	80.49	68.78	12.59
+ DAPO	40.07	87.80	**77.25**	30.69
+ RaR	40.93	87.80	75.13	29.52
+ SRaR (Ours)	**41.93**	**87.98**	76.60	**30.72**

SRaR is best on LiveCodeBench v6, HumanEval+, and TACO, and remains within 0.65 points of the best method on MBPP+. The margin over trajectory-level RaR is consistent across all four benchmarks ($+1.00$, $+0.18$, $+1.47$, $+1.20$), which is the comparison that isolates step-wise credit assignment since both methods use identical rubrics, judge, data, and recipe. Notably, RaR fails to beat DAPO on MBPP+ and TACO whereas SRaR does on TACO, suggesting that trajectory-level rubric aggregation is more fragile than step-wise attribution when the reward signal is already dense from unit-test execution. Absolute gains are largest on LiveCodeBench v6 and TACO, the two benchmarks with the longest multi-step solutions, consistent with the length-dependent trend observed on math (Section 4.2.2).

Q Theoretical Properties of SRaR

We provide formal analyses of the key properties of SRaR. Throughout this section, we consider a single prompt x with n rollouts $\{y_1, \ldots, y_n\}$ sampled from the policy $\pi_\theta(\cdot|x)$. A small constant $\epsilon > 0$ is added to every normalization denominator for numerical stability; since each numerator is a mean-centred deviation, this does not affect the properties established below.

Q.1 Policy Gradient Decomposition

Proposition 1 (Decomposition of the SRaR Policy Gradient)**.** *Let K_i denote the number of steps in rollout i, and let $\mathcal{T}_{k,i}$ denote the set of token indices belonging to step k in rollout i. Under the SRaR advantage, the policy gradient estimate decomposes as:*

$$\hat{g}_{\textit{SRaR}} = \hat{g}_{\textit{GRPO}} + \hat{g}_{\textit{rubric}} \tag{7}$$

where

$$\hat{g}_{\textit{GRPO}} = \frac{1}{n} \sum_{i=1}^{n} A_{\textit{base},i} \sum_{t=1}^{T_i} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i), \tag{8}$$

$$\hat{g}_{\textit{rubric}} = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K_i} \bar{d}_{k,i} \sum_{t \in \mathcal{T}_{k,i}} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i). \tag{9}$$

Proof. The SRaR policy gradient estimate is:

$$\hat{g}_{\text{SRaR}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{T_i}\hat{A}_i^{(t)}\,\nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i). \tag{10}$$

Substituting $\hat{A}_i^{(t)} = A_{\text{base},i} + \tilde{r}_i^{(t)}$ and using linearity of summation:

$$\hat{g}_{\text{SRaR}} = \underbrace{\frac{1}{n}\sum_{i=1}^{n} A_{\text{base},i}\sum_{t=1}^{T_i}\nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i)}_{\hat{g}_{\text{GRPO}}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n}\sum_{t=1}^{T_i}\tilde{r}_i^{(t)}\,\nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i)}_{\hat{g}_{\text{rubric}}}. \tag{11}$$

For the rubric term, since $\tilde{r}_i^{(t)} = \bar{d}_{k,i}$ for all $t \in \mathcal{T}_{k,i}$ (full-span placement) and $\tilde{r}_i^{(t)} = 0$ for tokens not belonging to any step with rubric evaluations, we can rewrite:

$$\hat{g}_{\text{rubric}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K_i}\bar{d}_{k,i}\sum_{t\in\mathcal{T}_{k,i}}\nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i). \tag{12}$$

□

Remark 1 (Interpretation)**.** *Equation* (8) *is exactly the standard GRPO gradient, which uniformly upweights or downweights the log-probability of the entire response based on outcome correctness. Equation* (9) *provides a step-selective correction: for each step k, the log-probabilities of tokens within that step are scaled by $\bar{d}_{k,i}$, which is positive when the step is relatively better than the same step in other rollouts, and negative otherwise. This allows the gradient to selectively reinforce or suppress individual reasoning steps within an otherwise uniformly scored response.*

Q.2 Variance Analysis

Proposition 2 (Scale Alignment Between Outcome and Rubric Signals)**.** *After normalization, both the outcome advantage and the step-wise rubric signals have controlled second moments:*

$$\frac{1}{n}\sum_{i=1}^{n} A_{base,i}^2 = 1, \tag{13}$$

$$\frac{1}{|\mathcal{G}_k|}\sum_{i\in\mathcal{G}_k}\bar{d}_{k,i}^2 = 1, \quad \forall\, k. \tag{14}$$

Proof. For the outcome advantage (Eq. 13), using population variance $\sigma_{\text{base}}^2 = \frac{1}{n}\sum_i (r_{\text{base},i} - \mu_{\text{base}})^2$:

$$\frac{1}{n}\sum_{i=1}^{n} A_{\text{base},i}^2 = \frac{1}{n\,\sigma_{\text{base}}^2}\sum_{i=1}^{n}(r_{\text{base},i} - \mu_{\text{base}})^2 = \frac{n\,\sigma_{\text{base}}^2}{n\,\sigma_{\text{base}}^2} = 1. \tag{15}$$

For the step-wise rubric signal (Eq. 14), using population variance $\sigma_k^2 = \frac{1}{|\mathcal{G}_k|}\sum_{i\in\mathcal{G}_k}(d_{k,i} - \mu_k)^2$:

$$\frac{1}{|\mathcal{G}_k|}\sum_{i\in\mathcal{G}_k}\bar{d}_{k,i}^2 = \frac{1}{|\mathcal{G}_k|\,\sigma_k^2}\sum_{i\in\mathcal{G}_k}(d_{k,i} - \mu_k)^2 = \frac{|\mathcal{G}_k|\,\sigma_k^2}{|\mathcal{G}_k|\,\sigma_k^2} = 1. \tag{16}$$

Both quantities equal 1, confirming that the two signals operate on exactly the same scale. □

Remark 2 (Variance Reduction via Step-wise Normalization)**.** *Without step-wise normalization (e.g., reward pattern 1 or 2), the raw rubric deltas $d_{k,i}$ may have highly variable magnitudes depending on the number and type of rubric items. In the worst case, the rubric signal can dominate or be negligible relative to the outcome advantage, destabilizing training. Step-wise normalization guarantees unit population variance per step, preventing any single step from contributing disproportionately to the gradient. Since the outcome advantage has unit population variance as well (Proposition 2), the two components of the SRaR advantage remain on the same scale throughout training, regardless of the raw rubric reward magnitudes.*

Q.3 Unbiasedness of the Policy Gradient Estimator

Proposition 3 (SRaR Preserves Unbiasedness of the REINFORCE Estimator)**.** *Define the SRaR reward for rollout i at token t as $R_i^{(t)} = r_{base,i} + \tilde{r}_i^{(t)}$, where $\tilde{r}_i^{(t)}$ is the rubric reward at token t. Then the SRaR policy gradient estimator*

$$\hat{g}_{SRaR} = \frac{1}{n} \sum_{i=1}^{n} \sum_{t=1}^{T_i} \left(R_i^{(t)} - b_i^{(t)} \right) \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \tag{17}$$

is an unbiased estimator of $\nabla_\theta J_{SRaR}(\theta)$ for any baseline $b_i^{(t)}$ that is independent of the action a_t^i, where $J_{SRaR}(\theta) = \mathbb{E}_{y \sim \pi_\theta}[R(y)]$ is the expected SRaR reward.

Proof. This follows from the standard REINFORCE theorem [Williams, 1992]. The rubric reward $\tilde{r}_i^{(t)}$ is a function of the complete response y_i (evaluated by the judge after generation), thus it is a valid return signal. For any baseline $b_i^{(t)}$ that does not depend on a_t^i:

$$\mathbb{E}_{a_t^i \sim \pi_\theta} \left[b_i^{(t)} \nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) \right] = b_i^{(t)} \nabla_\theta \sum_a \pi_\theta(a \mid s_t^i) = b_i^{(t)} \nabla_\theta 1 = 0. \tag{18}$$

Therefore subtracting the baseline does not introduce bias.

In the SRaR decoupled advantage, the group-level statistics μ_{base}, σ_{base}, μ_k, and σ_k serve as components of the baseline. In the leave-one-out interpretation, these statistics computed from the other $n-1$ rollouts are independent of a_t^i for any given rollout i, preserving unbiasedness. When computed including rollout i (as in the standard GRPO implementation), the resulting bias is $O(1/n)$ and vanishes as the group size grows, consistent with the analysis of GRPO □

R Dataset Statistics

Our training set consists of 16,560 math problems, each annotated with a structured rubric composed of four tag types: `SUGGEST` (key solution steps), `PITFALL` (common errors), `BONUS` (alternative approaches or insights), and `ANSWER` (final answer verification). We provide a detailed statistical analysis below.

R.1 Overall Summary

Table 13: Overall dataset statistics.

Statistic	Value
Total samples	16,560
Mean rubric items per sample	7.53
Median rubric items per sample	8
Mean words per rubric item	18.5

R.2 Rubric Item Counts by Type

Table 14: Statistics of rubric item counts by tag type.

Tag Type	Mean	Std	Median	Min	Max	Avg. Ratio
SUGGEST	4.59	0.86	5	2	8	60.9%
PITFALL	1.26	0.51	1	0	3	16.7%
BONUS	0.67	0.47	1	0	2	8.9%
ANSWER	1.00	0.00	1	1	1	13.3%
Total	7.53	0.79	8	4	12	100%

R.3 Distribution of Total Rubric Item Count

Figure 8 shows the distribution of total rubric item count per sample.

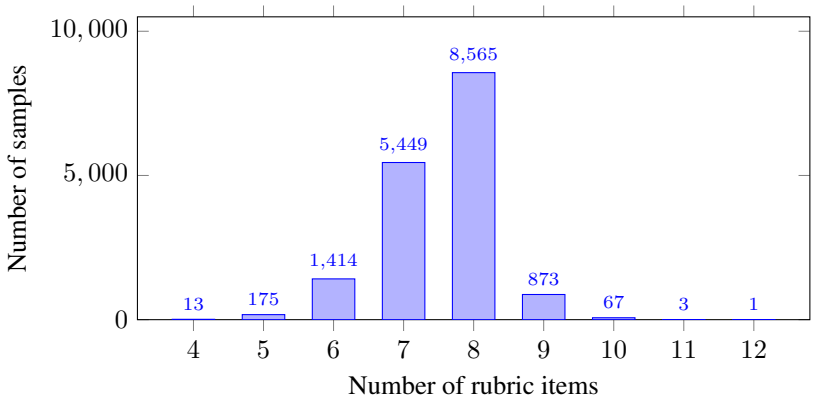


Figure 8: Distribution of total rubric item count per sample. The majority of samples (84.6%) contain 7–8 rubric items.

R.4 Distribution of SUGGEST Count

Figure 9 shows the distribution of SUGGEST items per sample.

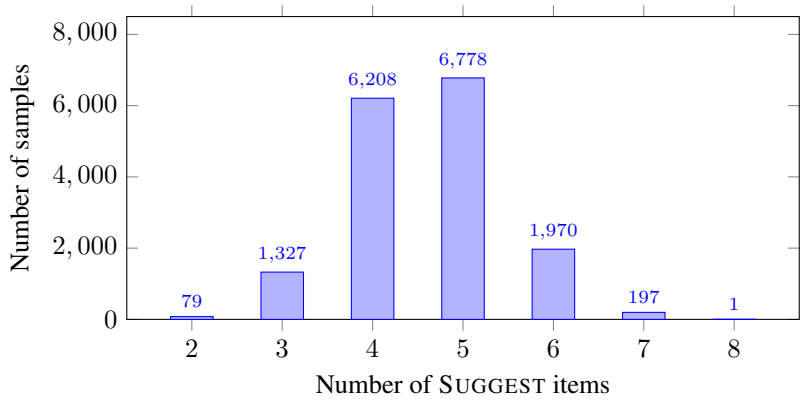


Figure 9: Distribution of SUGGEST items per sample. Most samples contain 4–5 solution steps.

R.5 Distribution of PITFALL Count

Figure 10 shows the distribution of PITFALL items per sample.

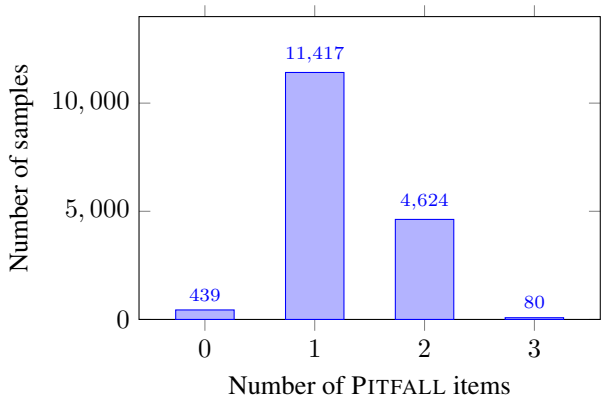


Figure 10: Distribution of PITFALL items per sample. 97.3% of samples have 1–2 pitfall warnings.

R.6 Distribution of BONUS Count

Figure 11 shows the distribution of BONUS items per sample.

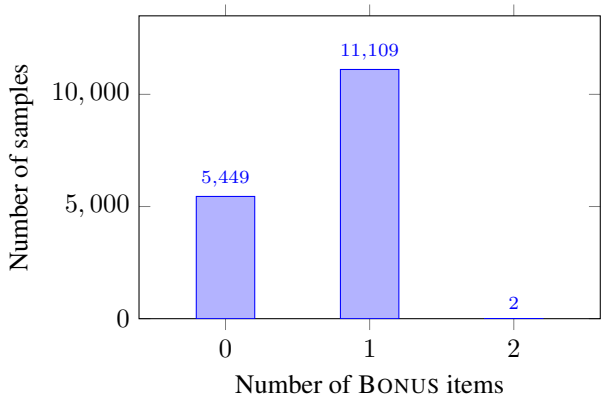


Figure 11: Distribution of BONUS items per sample. 67.1% of samples include one bonus insight.

R.7 Co-occurrence of PITFALL and BONUS

Table 15: Co-occurrence matrix of PITFALL and BONUS item counts (# of samples).

	Bonus=0	Bonus=1	Bonus=2	Total
Pitfall=0	369	70	0	439
Pitfall=1	3,416	8,000	1	11,417
Pitfall=2	1,617	3,006	1	4,624
Pitfall=3	47	33	0	80
Total	5,449	11,109	2	16,560

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Refer to Abstract and Introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Refer to Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Refer to Appendix Q.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Refer to Training Details in Experimental Setup and Appendix D and Appendix C.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will open source our datasets and code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Refer to Appendix D and Appendix C.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The paper does not report error bars or provide statistical significance measures for the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Refer to Training Details in Experimental Setup.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of our work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: There is no such risk in the paper.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We comply with the licenses of existing assets used in the paper and provide necessary references.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Refer to Appendix R.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects. The human annotations reported in Appendix H are performed by two of the authors as internal validation, not by external participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve research with human subjects. No IRB approval is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The proposed algorithm can be used for LLMs post-training. Thus we present some experiments about LLMs post-training. Meanwhile, we use LLMs for editing.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.