From Reasoning Chains to Verifiable Subproblems: Curriculum Reinforcement Learning Enables Credit Assignment for LLM Reasoning

Xitai Jiang[1,2,3,4,*], **Zihan Tang**[4,*], **Wenze Lin**[1,2,3,4,*]
Yang Yue[1,2,3], Shenzhi Wang[1,2,3], Gao Huang[1,2,3,†]

[1]Department of Automation, Tsinghua University
[2]Beijing Key Laboratory of Embodied Intelligence Systems
[3]Institute for Embodied Intelligence and Robotics, Tsinghua University
[4]Qiuzhen College, Tsinghua University

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

Abstract

Reinforcement learning from verifiable rewards (RLVR) has shown strong promise for LLM reasoning, but typical outcome-based RLVR methods remain inefficient on hard problems. Correct final-answer rollouts are rare, and standard sample-level credit assignment fails to leverage partial reasoning progress embedded in unsuccessful attempts. To address this problem, we introduce **SCRL** (**S**ubproblem **C**urriculum **R**einforcement **L**earning), a curriculum reinforcement learning framework built on verifiable subproblems derived from reasoning chains. Given a reference solution, SCRL derives a series of verifiable subproblems and constructs a subproblem curriculum, with the final subproblem fixed as the original problem. This converts partial progress on hard problems into verifiable learning signals. Algorithmically, we propose *subproblem-level normalization*, a training technique based on RLVR that normalizes rewards independently at each subproblem position within the rollout group. By assigning the resulting advantages to the corresponding answer spans, we enable finer-grained credit assignment without external rubrics or reward models. Our theoretical analysis shows that this subproblem curriculum makes hard problems more learnable by lifting them out of gradient dead zones, with larger relative gains as the original problem becomes harder. Across seven mathematical reasoning benchmarks, SCRL outperforms strong curriculum-learning baselines, yielding +4.1 and +1.9 average-point gains compared to GRPO on Qwen3-4B-Base and Qwen3-14B-Base respectively. On three hard benchmarks (AIME24, AIME25, and IMO-Bench), SCRL further yields point gains of +3.7 in pass@1 and +4.6 in pass@64 on Qwen3-4B-Base, suggesting improved exploration on hard reasoning problems. Further ablations show that the proposed credit assignment is effective and that the gains do not require highly curated subproblems or strong external generators, highlighting SCRL as a practical curriculum framework that enables fine-grained credit assignment for LLM reasoning. A no-reference-solution variant also improves average accuracy over GRPO, showing that reference solutions are helpful but not required for these gains.

1 Introduction

Reinforcement learning from verifiable rewards (RLVR) has emerged as a dominant paradigm for training large language models on mathematical reasoning, delivering strong empirical gains across

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

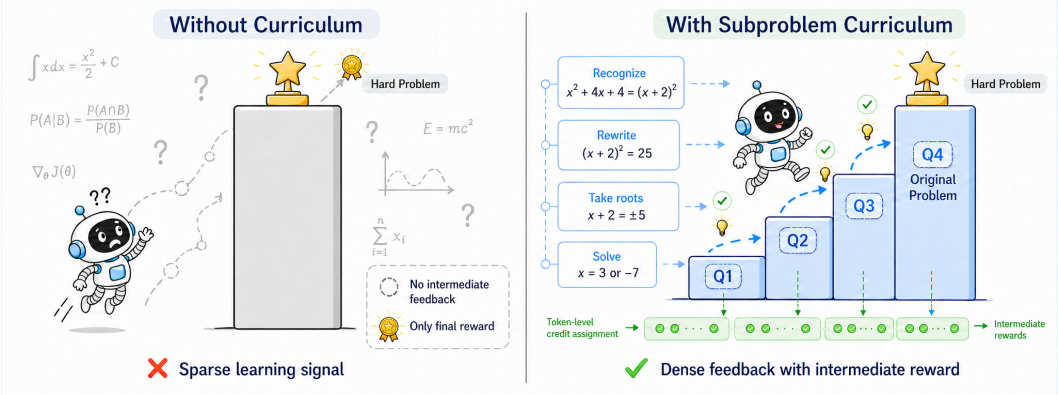

Figure 1: Main idea of SCRL. Standard outcome-based RLVR provides only sparse final-answer rewards on hard problems. SCRL instead decomposes a hard problem into verifiable subproblems, turning intermediate progress into dense learning signals and enabling finer-grained credit assignment.

benchmarks spanning grade-school arithmetic to olympiad-level competition [12, 44, 28, 13, 37]. The key to its success is that a correct final answer provides an unambiguous and automatically checkable reward signal. This removes the need for costly human annotation and avoids the reward hacking risks of learned reward models [33].

A central goal of RLVR is to help models solve previously unsolved problems and improve their reasoning ability. However, prior work suggests that direct RLVR often improves sampling efficiency more than it substantially expands the model's capability boundary [45, 32, 1]. Further studies indicate that training at the edge of the model's current capability with challenging problems is key for better reasoning ability [23, 15, 9, 21]. This makes hard problems particularly valuable for RL training. Yet typical RLVR methods like GRPO [12] struggle precisely on these problems. First, rewards are normalized within a group of rollouts sampled from the same prompt, so a group in which all rollouts fail provides no learning signal. Second, outcome-based RLVR assigns one *sample-level* advantage to the entire rollout. Thus, a near-miss attempt receives the same credit as an immediate failure. It is therefore crucial to extract learning signals from such hard-but-informative problems.

A natural way to learn from hard problems is to make better use of expert trajectories. Existing methods mainly follow two routes. One route is compensating for sparse rewards by training the model to imitate expert-generated trajectories, such as supervised fine-tuning and some off-policy RL methods [14, 40, 10, 48, 20]. However, they replace the model's own on-policy exploration with supervised imitation, and the resulting distribution shift between the expert and student policies can hurt training stability and out-of-distribution generalization [29, 8]. The other route uses on-policy curriculum RL. These methods provide an expert reasoning prefix or other hints and train the model to complete the remaining solution. [2, 49, 38, 24, 26, 40, 31]. However, these hints are treated as fixed conclusions rather than targets the model must derive, so the model does not need to discover the critical reasoning steps on its own, and the supplied context still shifts the model away from its own generation distribution. In fact, solving hard problems requires the model to explore and master the intermediate conclusions behind these hints by itself. This raises a central question: **how can we build a curriculum for hard problems that keeps the model exploring on its own, while also properly giving credit to the intermediate progress it solves along the way?**

We propose **SCRL** (**Subproblem Curriculum Reinforcement Learning**), drawing inspiration from a familiar structure in mathematical competitions: the multi-part problem. In a competition exam, a hard problem is broken into a sequence of subproblems of increasing difficulty, all visible at once; solving an earlier part yields a result that serves as a natural basis for the next. Given the expert solution to a hard problem, we offline construct a sequence of K verifiable subproblems using an external LLM. The subproblems are ordered from easier to harder, with each later subproblem building on the previous ones, and each subproblem has a verifiable answer. We fix the final subproblem as the original problem itself and ask the model to answer all K subproblems in a *single on-policy rollout*. This organically realizes a curriculum learning structure: when the model correctly solves an earlier subproblem, its answer becomes a natural basis for the next, guiding the model toward increasingly

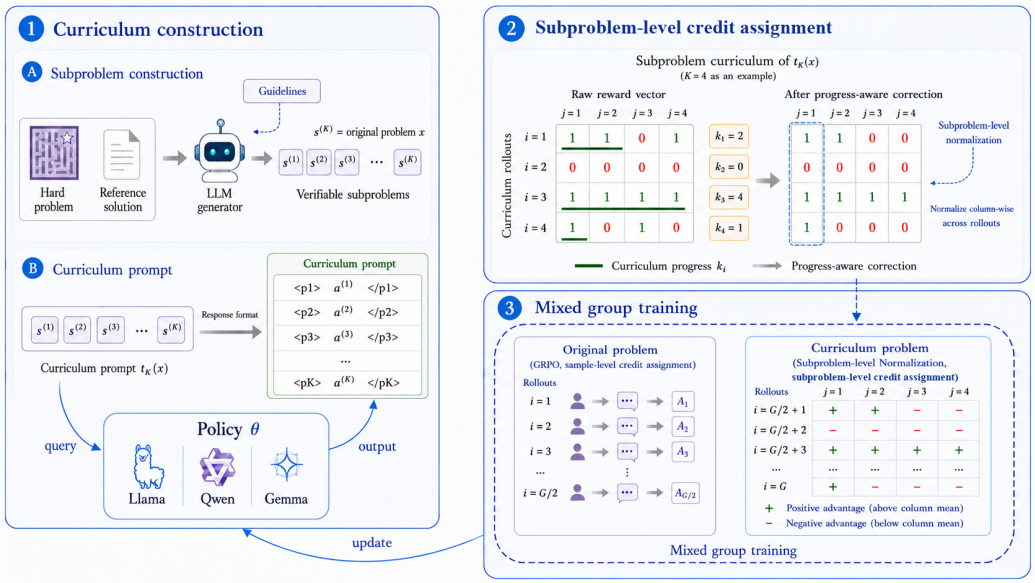

Figure 2: Overview of SCRL. SCRL constructs verifiable subproblems from a reference solution, uses structured responses to assign subproblem-level rewards back to answer-span tokens, and jointly trains curriculum rollouts with original-problem rollouts through mixed group training.

difficult reasoning. Critically, the reasoning steps that bridge consecutive subproblems are self-produced, earned through the model's own on-policy rollout. These intermediate results provide verifiable process-level supervision, naturally enabling finer-grained credit assignment within the rollout. We realize this through *subproblem-level normalization*, a novel RLVR training technique that normalizes rewards independently at each subproblem position and assigns the resulting advantages to the corresponding answer spans. In particular, to prevent the model from rewarding later subproblems without solving earlier ones, we align credit with curriculum progress by counting only the longest consecutively solved subproblem sequence. For example, the subproblem reward $[1, 1, 0, 1]$ is treated as $[1, 1, 0, 0]$, because progress after the first failed subproblem is not credited.

We validate SCRL with both theory and experiments. Theoretically, we show that subproblem decomposition lifts hard problems out of gradient dead zones by recovering non-degenerate learning signals from earlier subproblems. We formalize this as a metric recovery result, where optimization is lifted from the original policy manifold to a subproblem product manifold and the recovery ratio grows with problem difficulty. The empirical results are consistent with this prediction: SCRL improves over strong curriculum-learning baselines across mathematical reasoning benchmarks. Ablations further confirm the effectiveness of subproblem-level credit assignment and show that SCRL does not rely on highly curated subproblems or strong subproblem generators. A variant constructed without reference solutions also improves average accuracy over GRPO.

Our main contributions are:

- **SCRL framework for curriculum learning.** We propose a curriculum RL framework that turns each hard problem into a sequence of verifiable subproblems, enabling process-level supervision within a single on-policy rollout. This keeps the model exploring near the boundary of its current capability, making hard problems more effective for training.
- **Subproblem-level normalization for fine-grained credit assignment.** We introduce *subproblem-level normalization*, which normalizes rewards independently at each subproblem position and assigns the resulting advantages to the corresponding answer spans, enabling fine-grained credit assignment without external rubrics or additional reward models.
- **Theoretical and empirical validation.** We provide a metric recovery analysis showing that subproblem decomposition lifts hard problems out of gradient dead zones, with larger relative gains as the original problem becomes harder. Experiments across seven mathematical reasoning benchmarks verify these predictions and show consistent gains over

strong baselines (+4.1/+1.9 average-point gains on Qwen3-4B/14B; +3.7 pass@1 and +4.6 pass@64 points on three hard benchmarks).

2 Related Work

Reinforcement Learning with Verifiable Rewards (RLVR) Recent advances in Large Language Models (LLMs) have highlighted the effectiveness of Reinforcement Learning (RL) in domains with deterministic verifiers such as mathematics and programming [28, 13, 34, 41, 25, 36]. Unlike open-ended generation, these tasks provide unambiguous feedback, allowing for the optimization of policy models through algorithms like Proximal Policy Optimization (PPO) [27] or the more memory-efficient Group Relative Policy Optimization (GRPO) [12]. However, RLVR faces a significant challenge: for difficult problems, the reward signal becomes extremely sparse, leading to a failure in obtaining meaningful policy gradients [35]. This challenge is often framed as a credit assignment problem: outcome-based rewards provide a global signal but fail to pinpoint which specific reasoning steps contributed to the final success or failure [19]. While iterative self-improvement methods like STaR [46] and ReST [11, 47] attempt to bridge this gap through rejection sampling on easier instances, they still struggle when the task's difficulty exceeds the model's current exploration horizon. Consequently, curriculum learning [4, 42, 16, 22, 39] has become a common way to densify learning signals for hard problems by breaking hard tasks into manageable stages.

Curriculum Learning for Reasoning Existing curriculum learning methods for mathematical reasoning can be broadly categorized into two paradigms. The first category focuses on providing external hints or guidance when the model fails to solve a challenging problem. Notable works such as StepHint [48], Scaf-GRPO [49]and other hint-driven RL frameworks [38, 24, 26, 40, 31], utilize teacher model or self-generated rationales as auxiliary prefixes to lower the exploration threshold. The second category involves rewriting the original problem into simpler versions or augmenting the prompt with supplementary information to facilitate reasoning [6, 39, 17, 9, 15, 18]. As seen in MQR [9] and QuestA [15], these methods effectively create a difficulty gradient by manipulating the problem context. However, a fundamental limitation shared by these methods is their reliance on additional context. By providing the hint or reformulated problem as a static prefix, these approaches primarily optimize the model's continuation capability. As a result, the model fails to internalize the underlying scaffolding logic, as it is never required to generate the hints or auxiliary structures itself. In contrast, SCRL requires the model to generate the entire scaffolded multi-part sequence within a structured response, ensuring that the policy learns to both construct the intermediate reasoning steps and solve the final target problem.

3 Method

We propose **SCRL** (**Subproblem Curriculum Reinforcement Learning**), a curriculum RL framework that turns hard problems into verifiable subproblem curricula for finer-grained credit assignment. SCRL has three steps. First, given a reference solution, an external LLM derives K verifiable subproblems from the reasoning chain and constructs the subproblem curriculum. Second, the policy answers all K subproblems in one on-policy rollout. We then verify each subproblem answer and apply progress-aware correction to obtain progress-aware subproblem rewards. Subproblem-level normalization computes an advantage for each subproblem position, which is then used for subproblem-level credit assignment. Finally, to reduce prompt mismatch, SCRL uses mixed-group training, jointly optimizing curriculum rollouts and original-problem rollouts in the same update.

3.1 Preliminaries: GRPO

Given a prompt q, GRPO samples G rollouts $\{o_i\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid q)$ and assigns each rollout a scalar verifiable reward r_i. It then optimizes the clipped objective

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) = -\frac{1}{\sum_i L_i} \sum_{i=1}^{G} \sum_{t=1}^{L_i} \min\Big(\rho_{i,t} A_i,\ \mathrm{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon) A_i\Big) - \beta D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}). \quad (1)$$

Here $A_i = \frac{r_i - \text{mean}(\{r_i\}_{i=1}^G)}{\text{std}(\{r_i\}_{i=1}^G)}$ is the group-normalized advantage, and $\rho_{i,t} = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t}|q,o_{i,<t})}$ is the importance sampling ratio at token t. Since the same A_i is assigned to every token in o_i, GRPO performs sample-level credit assignment.

3.2 SCRL Framework

Build subproblems. For each hard problem x, we start from an existing chain-of-thought reference solution. An external LLM rewrites its intermediate progress nodes into K verifiable subproblems, rather than solving the problem from scratch. The exact generation prompt is provided in Appendix I, and the main guidelines are summarized below.

Guidelines for Subproblem Generation

1. Increasing difficulty. Subproblems are ordered from easier to harder, with the final subproblem fixed as the original problem:

$$s^{(1)} \prec s^{(2)} \prec \cdots \prec s^{(K)} = x.$$

2. Linked but self-contained. Earlier subproblems provide useful intermediate results for later ones, while each subproblem remains a complete standalone question.

3. Verifiable answers. Each subproblem has an objectively checkable answer for independent verification.

Verification and quality checks. During offline construction, we check the generated subproblems for the required format and completeness, but do not independently re-solve each subproblem to certify its semantic correctness or difficulty ordering. During RL, each subproblem answer is compared with its reference answer using Math-Verify's mathematical-equivalence checks. Integer answers require exact equality; numerical tolerance is used only for floating-point answers. Malformed curriculum responses receive zero reward, as specified below.

Curriculum prompt. Let x denote the original problem. We define the curriculum prompt $t_K(x)$ as the prompt that presents all K subproblems $s^{(1)}, \ldots, s^{(K)}$ simultaneously and asks the model to solve them in order. Thus, x corresponds to the original-problem rollout, while $t_K(x)$ corresponds to the curriculum rollout. The detailed prompt template is provided in Appendix J.1.

Response format. During curriculum rollouts, the model is asked to answer the K subproblems using explicit tags `<pj>` and `</pj>`:

$$\texttt{<p1>}\ a^{(1)}\ \texttt{</p1>}\ \cdots\ \texttt{<pK>}\ a^{(K)}\ \texttt{</pK>},$$

where $a^{(j)}$ is the response to subproblem j. These tags not only specify the response format, but also mark the token span of each subproblem answer. This allows us to verify each answer separately and later assign the corresponding subproblem-level advantage back to the tokens inside that span.

3.2.1 Progress-Aware Subproblem Rewards

Curriculum progress. For a curriculum rollout $o_i \sim \pi_\theta(\cdot \mid t_K(x))$, verifying the K extracted subproblem answers gives a raw reward vector $\mathbf{r}_i = (r_i^{(1)}, \ldots, r_i^{(K)}) \in \{0,1\}^K$. If the response does not follow the required format, we set $\mathbf{r}_i = \mathbf{0}$. We define the curriculum progress $k_i \in \{0, 1, \ldots, K\}$ as the maximum number of consecutively solved subproblems from the beginning:

$$k_i := \max\big\{j \in \{0,1,\ldots,K\} \;\big|\; r_i^{(1)} = \cdots = r_i^{(j)} = 1\big\}. \tag{2}$$

Thus, $k_i = 0$ means the first subproblem is incorrect, while $k_i = K$ means all subproblems are solved. The curriculum progress k_i tracks the current policy's capability boundary on the hard problem, and also identifies the intermediate progress actually achieved by the rollout.

Progress-aware correction. Directly rewarding each subproblem independently may credit later subproblems despite earlier failures, creating a potential reward-hacking shortcut. We therefore align rewards with curriculum progress by keeping only the consecutively solved prefix:

$$\tilde{r}_i^{(j)} := \begin{cases} r_i^{(j)}, & j \le k_i, \\ 0, & j > k_i, \end{cases} \qquad \tilde{\mathbf{r}}_i = \big(\tilde{r}_i^{(1)}, \tilde{r}_i^{(2)}, \ldots, \tilde{r}_i^{(K)}\big). \tag{3}$$

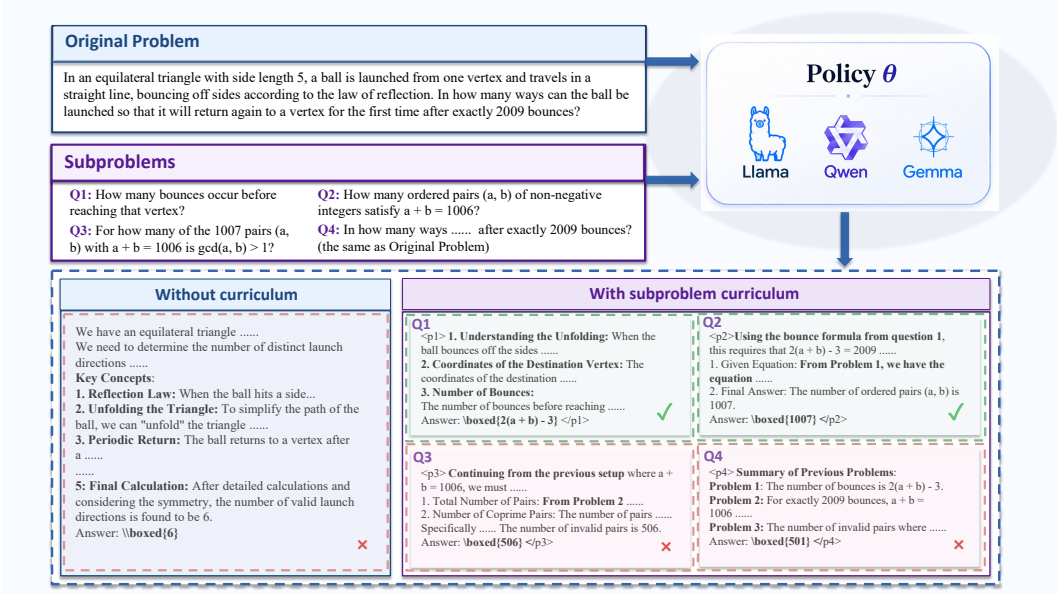

Figure 3: Illustration of mixed training rollouts. The policy generates both original-problem rollouts and curriculum rollouts. The tagged response format <pj>...</pj> in curriculum rollouts identifies the answer span of each subproblem, enabling finer-grained credit assignment.

For example, when $K = 4$, $(1, 1, 0, 1)$ is corrected to $(1, 1, 0, 0)$. For notational convenience, we use $R_i^{(j)} := \tilde{r}_i^{(j)}$ as the final subproblem reward for training.

3.2.2 SCRL Training Algorithm

In this section, we describe the training details of SCRL, including subproblem-level normalization for advantage computation, subproblem-level credit assignment, and mixed-group training. The full training procedure is summarized in Appendix C.

Subproblem-level normalization. Given G curriculum rollouts o_i for $i = 1, \ldots, G$, we normalize the final subproblem rewards at each subproblem position j across the rollout group:

$$A_i^{(j)} = \frac{R_i^{(j)} - \text{mean}\big(\{R_i^{(j)}\}_{i=1}^{G}\big)}{\text{std}\big(\{R_i^{(j)}\}_{i=1}^{G}\big)}. \tag{4}$$

Thus, the subproblem-level advantage $A_i^{(j)}$ measures the relative success of rollout i at subproblem position j within the rollout group, independent of rewards at other subproblem positions.

Subproblem-level credit assignment. After computing the subproblem-level advantages, we assign them back to the tokens of the corresponding subproblem answers. Using the structured response format, we define $\text{sub}_i(t) = j$ if token $o_{i,t}$ lies between <pj> and </pj>; then $A_{i,t} = A_i^{(\text{sub}_i(t))}$ assigns the same subproblem-level advantage to each token in that answer span. Tokens outside all answer spans receive zero advantage, and if the response does not follow the required format, all tokens in that response receive zero advantage. This converts subproblem-level progress into learning signals for the corresponding subproblem answer spans.

Mixed-group training. Training only on the curriculum prompt $t_K(x)$ can cause prompt mismatch, because evaluation uses the original prompt x. We therefore use mixed-group training: for each problem x, $G/2$ rollouts are sampled from $t_K(x)$ and optimized with subproblem-level advantages assigned to answer spans, while the other $G/2$ rollouts are sampled from x and optimized with

standard outcome-based GRPO. The final SCRL objective is

$$\mathcal{L}_{\mathrm{SCRL}}(\theta) = -\frac{1}{\sum_{i=1}^{G} L_i}\left[\sum_{i=1}^{G/2}\sum_{t=1}^{L_i}\min\Big(\rho_{i,t}A_{i,t}, \mathrm{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)A_{i,t}\Big) + \sum_{i=G/2+1}^{G}\sum_{t=1}^{L_i}\min\Big(\rho_{i,t}A_i, \mathrm{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)A_i\Big)\right] - \beta D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}}). \quad (5)$$

The two bracketed terms correspond to curriculum rollouts and original-problem rollouts respectively. The complete training procedure is summarized in Algorithm 1.

4 Theoretical Analysis

Using the information geometry of the policy manifold $\mathcal{M} = \{\pi_\theta : \theta \in \Theta\}$ equipped with the Fisher–Rao metric [3], we show that hard problems can place outcome-based GRPO in a *gradient dead zone*, while subproblem decomposition lifts optimization to a product manifold that recovers useful gradient information. Full discussions and proofs are provided in Appendix B.

Definition 4.1 (Effective and Lifted Gradient Information Matrices)**.** *Under GRPO, let* $o_1, \ldots, o_G \overset{\text{i.i.d.}}{\sim} \pi_\theta(\cdot \mid x)$ *be* G *sampled rollouts. The* ***effective gradient information matrix (EGIM)*** *of* x *and the* ***lifted EGIM*** *of its subproblem transformation* $\mathcal{T}(x) = t_K(x)$ *are*

$$\boldsymbol{F}_x(\theta) = \frac{1}{G}\sum_{i=1}^{G}\mathbb{E}\big[g_i(x)g_i(x)^\top\big], \qquad \text{where } g_i(x) = \hat{A}_i(x)\nabla_\theta \log \pi_\theta(o_i \mid x), \quad (6)$$

$$\boldsymbol{F}_{\mathcal{T}(x)}(\theta) = \frac{1}{K}\sum_{j=1}^{K}\frac{1}{G}\sum_{i=1}^{G}\mathbb{E}\Big[\hat{A}_i^{(j)2}\nabla_\theta \log \pi_\theta(o_i^{(j)} \mid t_K(x))\nabla_\theta \log \pi_\theta(o_i^{(j)} \mid t_K(x))^\top\Big], \quad (7)$$

Here $\hat{A}_i(x)$ *and* $\hat{A}_i^{(j)}$ *denote the original-problem and subproblem-position advantages respectively, and the smallest eigenvalue* $\lambda_{\min}(\boldsymbol{F}_{\mathcal{T}(x)}(\theta))$ *measures the weakest useful gradient signal.*

Theorem 4.2 (Gradient Dead Zone)**.** *Let* $p(x;\theta) = \Pr_{\pi_\theta}[r(x,o) = 1]$ *be the probability that the current policy solves* x*. If* $p(x;\theta) < \delta$*, then*

$$\lambda_{\min}(\boldsymbol{F}_x(\theta)) \le G\delta \cdot C_{\hat{A}}^2 \cdot B_s^2 = O(\delta), \quad (8)$$

where $C_{\hat{A}} \le \sqrt{G-1}$ bounds the normalized advantage magnitude and B_s bounds the score norm (both derived in Appendix B.3).

Theorem 4.2 shows that direct RLVR training becomes ineffective on hard problems: when correct rollouts are rare, reward groups collapse and the worst-case effective gradient signal vanishes.

Theorem 4.3 (Metric Recovery via Subproblem Decomposition)**.** *Let* x *be in the gradient dead zone with* $p(x;\theta) < \delta$*. Suppose the subproblem construction satisfies*

$$p_j(x;\theta) := \Pr[R^{(j)} = 1 \mid t_K(x)] \in [p^\star, 1-p^\star] \quad \textit{for all } j < K,$$

where $p^\star \in (\delta, 1/2]$*. Under the conditional identifiability assumption (*$\mathbb{E}[(v^\top\nabla_\theta \log \pi_\theta)^2 \mid r{=}r_0] \ge \sigma_{\min}^2 > 0$ *for all unit* v*,* $r_0 \in \{0,1\}$*),*

$$\lambda_{\min}\big(\boldsymbol{F}_{\mathcal{T}(x)}(\theta)\big) \ge \frac{1}{K}\,c(p^\star, G, \sigma_{\min}) > 0, \qquad \frac{\lambda_{\min}(\boldsymbol{F}_{\mathcal{T}(x)}(\theta))}{\lambda_{\min}(\boldsymbol{F}_x(\theta))} = \Omega\left(\frac{1}{\delta}\right), \quad (9)$$

where $c(p^\star, G, \sigma_{\min}) = \big(1 - (p^\star)^G - (1-p^\star)^G\big)\sigma_{\min}^2$ *is a positive constant independent of* δ*.*

Theorem 4.3 shows that subproblem curriculum helps hard problems by recovering a non-degenerate learning geometry, even when the original problem provides almost no useful gradient signal. Moreover, the recovery ratio grows as $p(x;\theta) \to 0$, predicting larger relative gains on harder problems.

5 Experiment

5.1 Experimental Setup

Models. To investigate the scalability and effectiveness of our proposed method across different model capacities, we conduct experiments on the Qwen and Llama series. Specifically, we utilize `Qwen3-4B-Base`, `Qwen3-14B-Base` and `Llama3.2-3B-Instruct` as our base policies.

Training Setup. We use the training set `hard_1024`, a subset of 1,024 problems randomly selected from the high-difficulty competition mathematics dataset provided by [43]. For SCRL, subproblems are generated with the DeepSeek-V3.2 API. All models are trained using the **Verl** framework [30] for a total of 300 steps. Detailed hyperparameter configurations are provided in Appendix F.1.

Benchmark. We evaluate the models on seven widely used mathematical reasoning benchmarks: `OlympiadBench`, `Minerva`, `MATH-500`, `AIME 2024`, `AIME 2025`, `AMC`, and `IMO-Bench`.

Baseline Settings. We compare our method against the following competitive baselines: `SFT`, `GRPO` [12], `DAPO` [44], `QuestA` [15], `NuRL` [5], `Scaf-GRPO` [49], and `StepHint` [48]. Scaf-GRPO and StepHint are evaluated on Qwen3-4B-Base. Implementation details are provided in Appendix F.3.

5.2 Main Results and Further Analysis

Table 1: **Main Results** on mathematical reasoning benchmarks. The best results are highlighted in **bold**, and the second-best results are underlined.

Base Model	Method	Olym.B	Minerva	MATH	AIME'24	AIME'25	AMC	IMO-B	AVG
Llama3.2-3B-Instruct	Initial Model	13.1	13.7	44.0	6.4	0.7	20.6	4.5	14.7
	SFT	13.1	11.0	44.1	2.5	0.7	18.1	2.6	13.2
	GRPO	13.7	14.9	44.5	**10.3**	0.5	20.9	5.4	15.7
	DAPO	14.6	15.1	**45.9**	9.8	0.3	**22.2**	5.8	16.2
	QuestA	14.6	14.8	**45.9**	8.5	0.4	21.4	6.1	16.0
	NuRL	15.1	14.9	45.2	10.2	0.3	21.7	5.9	16.2
	SCRL	**15.2**	**15.2**	45.2	**10.3**	**0.8**	21.4	**6.8**	**16.4**
Qwen3-4B-Base	Initial Model	29.0	15.6	64.4	10.9	6.4	38.1	6.1	24.4
	SFT	31.1	25.4	71.0	10.6	10.0	41.0	7.1	28.0
	GRPO	35.1	29.7	76.2	14.5	7.2	46.1	7.8	30.9
	DAPO	34.9	30.1	76.0	11.8	9.3	46.8	8.3	31.0
	QuestA	35.0	32.0	75.9	14.1	11.7	46.8	8.3	32.0
	NuRL	35.6	30.3	77.6	13.2	8.4	45.4	7.1	31.1
	Scaf-GRPO	39.1	29.4	78.2	16.0	11.0	46.1	**8.9**	32.7
	StepHint	38.8	24.3	79.0	**16.5**	**17.0**	49.4	6.9	33.1
	SCRL	**39.2**	**33.1**	**79.8**	**16.5**	15.3	**52.4**	8.7	**35.0**
Qwen3-14B-Base	Initial Model	35.3	23.7	75.7	14.4	10.7	50.2	7.4	31.1
	SFT	**41.2**	24.3	72.0	20.0	**20.0**	49.3	11.0	34.0
	GRPO	38.4	33.4	81.3	21.6	11.9	58.1	9.9	36.4
	DAPO	37.5	32.6	79.9	22.7	12.3	59.2	9.9	36.3
	QuestA	40.4	34.0	82.6	18.5	11.6	**60.2**	10.4	36.8
	NuRL	38.3	32.6	82.0	23.6	11.2	58.1	9.4	36.5
	SCRL	40.9	**34.7**	**83.2**	**24.4**	13.1	60.0	**11.9**	**38.3**

The main results across seven mathematical reasoning benchmarks are summarized in Table 1, with full experimental results provided in Appendix E.

Superior Performance Across All Benchmarks. As shown in Table 1, **SCRL** consistently outperforms vanilla GRPO and competitive baselines including DAPO, QuestA, and NuRL across three model scales: Llama3.2-3B, Qwen3-4B, and Qwen3-14B. In terms of average accuracy (Avg), SCRL achieves the best performance in all settings. The gain is especially clear on Qwen3-4B, where SCRL reaches an average score of **35.0%**, improving over the second-best baseline StepHint (**33.1%**) by **1.9** points and over vanilla GRPO (**30.9%**) by **4.1** points. On challenging benchmarks such as AIME'25, SCRL also shows strong gains, achieving **15.3%** compared with QuestA's **11.7%**.

Curriculum progress transfers to hard-problem solving. Figure 4 shows pass@k curves on AIME24, AIME25, and IMO-Bench. SCRL generally improves pass@k over GRPO and the plotted curriculum baselines, indicating stronger hard-problem solving ability at many sampling budgets.

Table 2: Effect of subproblem generator quality on Qwen3-4B-Base.

Method	Subproblem Generator	Olym.B	Minerva	MATH	AIME'24	AMC	AVG
GRPO	–	35.1	29.7	76.2	14.5	46.1	40.3
SCRL	Qwen3-4B-Instruct	36.3	34.2	80.0	16.7	47.9	43.0
SCRL	DeepSeek-V3.2	39.2	33.1	79.8	16.5	52.4	44.2

Table 3: Effect of removing the reference solution during offline subproblem construction on Qwen3-4B-Base.

Method	Olym.B	Minerva	MATH	AIME'24	AIME'25	AMC	IMO-B	AVG
GRPO	35.10	29.70	76.20	14.50	7.20	46.10	7.80	30.90
SCRL w/o Reference Solution	36.40	**33.46**	75.60	**17.08**	13.23	50.60	**10.20**	33.80
SCRL	**39.20**	33.10	**79.80**	16.50	**15.30**	**52.40**	8.70	**35.00**

However, it does not lead at every k: on AIME24, DAPO achieves 56.7% pass@64 compared with SCRL's 46.7% (Appendix E).

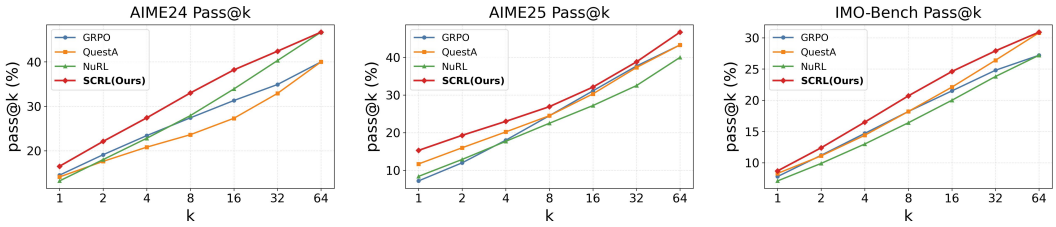

Figure 4: Pass@k curves on AIME24, AIME25, and IMO-Bench on Qwen3-4B-Base.

Figure 5 further tracks the ratio of solvable problems during training, where a problem is counted as solvable once it is fully solved at least once. The *full group* statistic counts success in either the original-problem or curriculum format, while the *half group* statistic uses only half-budget original-problem rollouts, matching SCRL's mixed-group setting. SCRL achieves a higher solvable ratio than GRPO under both protocols, showing that curriculum progress transfers back to direct hard-problem solving rather than only improving curriculum-format rollouts.

SCRL does not rely on highly curated subproblems. We further examine whether SCRL depends on high-quality subproblem construction. Table 2 compares subproblems generated by DeepSeek-V3.2 and a weaker Qwen3-4B-Instruct generator, using the same generation prompt and downstream training pipeline. In both cases, the generator is given the dataset reference solution, so it only decomposes an already solved problem rather than solving it from scratch. SCRL remains effective with the weaker generator, improving over GRPO by +2.7 points on average, while DeepSeek-V3.2 further increases the gain to +3.9 points.

Dependence on reference solutions. The default SCRL construction uses the dataset reference solution to generate subproblems offline. To test this dependence, we generate subproblems without the reference solution while keeping the original problem, final answer, generator, and downstream training configuration unchanged. As shown in Table 3, the no-reference variant achieves 33.80% average accuracy, 2.90 points above GRPO and 1.20 points below the default SCRL setting. Reference solutions improve curriculum quality but are not required for SCRL to outperform GRPO on average. The reference solution is used only during offline subproblem construction and is not directly provided to the policy during RL training.

Figure 6 shows that even with DeepSeek-V3.2, the ratio of curriculum instances fully solved at $k_i = 4$ remains lower than the GRPO solvable ratio (the GRPO bar at $k_i = 4$ reports the ratio of original problems solved by GRPO under the same half-group counting protocol). Nevertheless, SCRL still achieves a substantial performance gain. This indicates that SCRL does not require subproblems to be easy or perfectly curated. At the same time, DeepSeek-V3.2 produces a larger $k_i = 4$ ratio than Qwen3-4B-Instruct, suggesting that better subproblem quality can further increase SCRL's gains.

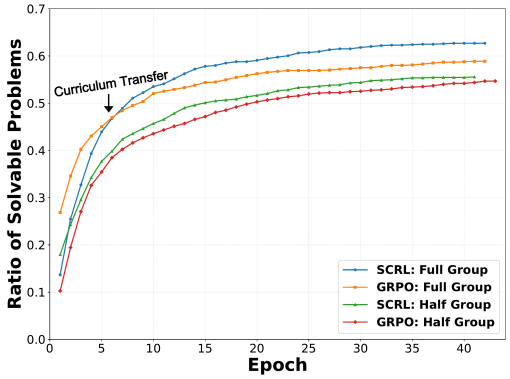

Figure 5: Ratio of solvable problems during training of Qwen3-4B-Base.

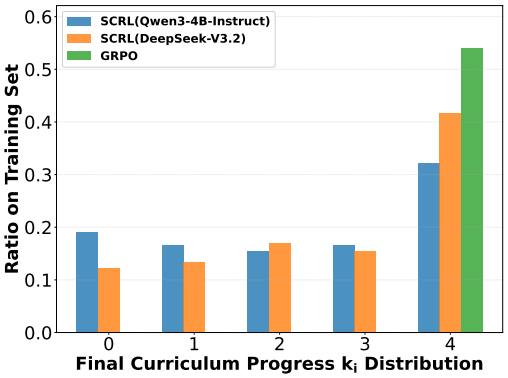

Figure 6: Final curriculum-progress distribution on the training set.

Table 4: Ablation on credit assignment. Here "correction" denotes progress-aware correction.

Credit Assignment Method	Olym.B	Minerva	MATH	AIME'24	AMC	AVG
Both-GRPO	38.2	33.1	79.4	14.0	54.8	43.9
Subproblem-Level Norm. w/o corr.	35.7	33.1	78.2	13.8	48.5	41.9
Subproblem-Level Norm. w/ corr.	39.2	33.1	79.8	16.5	52.4	44.2

Subproblem-level normalization enables better credit assignment. We further ablate on Qwen3-4B-Base how credit is assigned within curriculum rollouts. Table 4 compares the full method with two alternatives: removing progress-aware correction, and Both-GRPO, which keeps mixed training but verifies only the final subproblem and applies sample-level GRPO to curriculum rollouts.

Subproblem-level normalization with progress-aware correction performs best. Without correction, dense subproblem signals may reward later steps after earlier failures, while Both-GRPO uses subproblems only as hints and cannot credit valid intermediate progress. Thus, effective curriculum training needs both subproblem-specific signals and progress-aware correction.

6 Conclusion

We propose SCRL, a subproblem curriculum RL framework for hard LLM reasoning with verifiable rewards. SCRL derives verifiable subproblems from reasoning chains and uses subproblem-level normalization to convert partial rollout progress into subproblem-level learning signals, enabling fine-grained credit assignment without external reward models or process annotations. Our theory shows that subproblem decomposition can lift hard problems out of gradient dead zones, and experiments show consistent gains over strong RLVR and curriculum-learning baselines.

Acknowledgments and Disclosure of Funding

This work is supported in part by the National Key R&D Program of China under Grant 2024YFB4708200, the National Natural Science Foundation of China under Grants U24B20173, U2541227 and 62625604, and the Beijing Major Science and Technology Project under Contract No. Z251100008125064. The authors declare no competing interests.

References

[1] Md Tanvirul Alam and Nidhi Rastogi. Limits of generalization in rlvr: Two case studies in mathematical reasoning. *arXiv preprint arXiv:2510.27044*, 2025.

[2] Mohammad Hossein Amani, Aryo Lotfi, Nicolas Mario Baldwin, Samy Bengio, Mehrdad Farajtabar, Emmanuel Abbe, and Robert West. Rl for reasoning by adaptively revealing

rationales. *ArXiv*, abs/2506.18110, 2025. URL https://api.semanticscholar.org/CorpusID:280000657.

[3] Shun-ichi Amari. *Information Geometry and Its Applications*, volume 194 of *Applied Mathematical Sciences*. Springer Japan, Tokyo, 2016. ISBN 978-4-431-55977-1. doi: 10.1007/978-4-431-55978-8.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[5] Justin Chih-Yao Chen, Becky Xiangyu Peng, Prafulla Kumar Choubey, Kung-Hsiang Huang, Jiaxin Zhang, Mohit Bansal, and Chien-Sheng Wu. Nudging the boundaries of llm reasoning. *arXiv preprint arXiv:2509.25666*, 2025.

[6] Justin Chih-Yao Chen, Archiki Prasad, Zaid Khan, Joykirat Singh, Runchu Tian, Elias Stengel-Eskin, and Mohit Bansal. Cog-drift: Exploration on adaptively reformulated instances enables learning from hard reasoning problems. *arXiv preprint arXiv:2604.04767*, 2026.

[7] Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[8] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*, 2025.

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

[10] Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. Srft: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*, 2025.

[11] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

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

[13] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[14] Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G. Patil, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *ArXiv*, abs/2502.07374, 2025. URL https://api.semanticscholar.org/CorpusID:276258697.

[15] Jiazheng Li, Hongzhou Lin, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Yi Wu, and Jingzhao Zhang. Questa: Expanding reasoning capacity in LLMs via question augmentation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3MifB0f7qR.

[16] Renda Li, Hailang Huang, Fei Wei, Feng Xiong, Yong Wang, and Xiangxiang Chu. Adacurl: Adaptive curriculum reinforcement learning with invalid sample mitigation and historical revisiting. *ArXiv*, abs/2511.09478, 2025. URL https://api.semanticscholar.org/CorpusID:282939669.

[17] Xuchen Li, Jing Chen, Xuzhao Li, Hao Liang, Xiaohuan Zhou, Taifeng Wang, and Wentao Zhang. Mathmixup: Boosting llm mathematical reasoning with difficulty-controllable data synthesis and curriculum learning. *arXiv preprint arXiv:2601.17006*, 2026.

[18] Xiao Liang, Zhong zhi Li, Yeyun Gong, Yelong Shen, Yingchun Wu, Zhijiang Guo, and Weizhu Chen. Beyond pass@1: Self-play with variational problem synthesis sustains rlvr. *ArXiv*, abs/2508.14029, 2025. URL https://api.semanticscholar.org/CorpusID:280686520.

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

[20] Xingtai Lv, Yuxin Zuo, Youbang Sun, Hongyi Liu, Yuntian Wei, Zhekai Chen, Xuekai Zhu, Kaiyan Zhang, Bingning Wang, Ning Ding, et al. Towards a unified view of large language model post-training. *arXiv preprint arXiv:2509.04419*, 2025.

[21] Lu Ma, Hao Liang, Meiyi Qiang, Lexiang Tang, Xiaochen Ma, Zhen Hao Wong, Junbo Niu, Chengyu Shen, Runming He, Bin Cui, and Wentao Zhang. Learning what reinforcement learning can't: Interleaved online fine-tuning for hardest questions. *ArXiv*, abs/2506.07527, 2025. URL https://api.semanticscholar.org/CorpusID:279251208.

[22] Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep M. Kalathil, and Shuiwang Ji. Curriculum reinforcement learning from easy to hard tasks improves llm reasoning. *ArXiv*, abs/2506.06632, 2025. URL https://api.semanticscholar.org/CorpusID:279251658.

[23] Benjamin Pikus, Pratyush Ranjan Tiwari, and Burton Ye. Hard examples are all you need: Maximizing grpo post-training under annotation budgets. *ArXiv*, abs/2508.14094, 2025. URL https://api.semanticscholar.org/CorpusID:280692329.

[24] Deng Qiyuan, Kehai Chen, Min Zhang, and Zhongwen Xu. HiPO: Self-hint policy optimization for RLVR. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rcb20pHmT1.

[25] Yuxiao Qu, Anikait Singh, Yoonho Lee, Amrith Rajagopal Setlur, Ruslan Salakhutdinov, Chelsea Finn, and Aviral Kumar. Rlad: Training llms to discover abstractions for solving reasoning problems. *ArXiv*, abs/2510.02263, 2025. URL https://api.semanticscholar.org/CorpusID:281724383.

[26] Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. Pope: Learning to reason on hard problems via privileged on-policy exploration. *arXiv preprint arXiv:2601.18779*, 2026.

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

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

[29] Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. *ArXiv*, abs/2509.04259, 2025. URL https://api.semanticscholar.org/CorpusID:281103647.

[30] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *EuroSys*, pages 1279–1297, 2025. URL https://doi.org/10.1145/3689031.3696075.

[31] Weijie Shi, Yanxi Chen, Zexi Li, Xuchen Pan, Yuchang Sun, Jiajie Xu, Xiaofang Zhou, and Yaliang Li. R3l: Reflect-then-retry reinforcement learning with language-guided exploration, pivotal credit, and positive amplification. *ArXiv*, abs/2601.03715, 2026. URL https://api.semanticscholar.org/CorpusID:284532205.

[32] Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*, 2025.

[33] Joar Max Viktor Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking. *ArXiv*, abs/2209.13085, 2022. URL https://api.semanticscholar.org/CorpusID:252545256.

[34] Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[35] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

[36] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*, 2025.

[37] Xumeng Wen, Zihan Liu, Shun Zheng, Zhijian Xu, Shengyu Ye, Zhirong Wu, Xiao Liang, Yang Wang, Junjie Li, Ziming Miao, Jiang Bian, and Mao Yang. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *ArXiv*, abs/2506.14245, 2025. URL https://api.semanticscholar.org/CorpusID:279410727.

[38] Jinyang Wu, Chonghua Liao, Mingkuan Feng, Shuai Zhang, Zhengqi Wen, Pengpeng Shao, Huazhe Xu, and Jianhua Tao. Thought-augmented policy optimization: Bridging external guidance and internal capabilities. *arXiv preprint arXiv:2505.15692*, 1(8):10, 2025.

[39] Muling Wu, Qi Qian, Wenhao Liu, Xiaohua Wang, Zisu Huang, Di Liang, LI Miao, Shihan Dou, Changze Lv, Zhenghua Wang, Zhibo Xu, Lina Chen, Tianlong Li, Xiaoqing Zheng, and Xuanjing Huang. Progressive mastery: Customized curriculum learning with guided prompting for mathematical reasoning. *ArXiv*, abs/2506.04065, 2025. URL https://api.semanticscholar.org/CorpusID:279154725.

[40] Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=vO8LLoNWWk.

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

[42] Changpeng Yang, Jinyang Wu, Yuchen Liu, Shuai Zhang, Yang Li, Qiliang Liang, Hongzhen Wang, Shuai Nie, Jiaming Xu, Runyu Shi, Ying Huang, and Guoquan Zhang. From imitation to discrimination: Toward a generalized curriculum advantage mechanism enhancing cross-domain reasoning tasks. In *AAAI Conference on Artificial Intelligence*, 2025. URL https://api.semanticscholar.org/CorpusID:283458223.

[43] Matthew YR Yang, Hao Bai, Ian Wu, Gene Yang, Amrith Setlur, and Aviral Kumar. Int: Self-proposed interventions enable credit assignment in llm reasoning. *arXiv preprint arXiv:2601.14209*, 2026.

[44] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Honglin Yu, Weinan Dai, Yuxuan Song, Xiang Wei, Haodong Zhou, Jingjing Liu, Wei Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yong-Xu Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *ArXiv*, abs/2503.14476, 2025. URL https://api.semanticscholar.org/CorpusID:277104124.

[45] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

[46] Eric Zelikman, Yuhuai Wu, and Noah D Goodman. Star: Self-taught reasoner. *arXiv preprint arXiv:2203.14465*, 2022.

[47] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. Rest-mcts*: Llm self-training via process reward guided tree search. *Advances in Neural Information Processing Systems*, 37:64735–64772, 2024.

[48] Kaiyi Zhang, Ang Lv, Jinpeng Li, Yongbo Wang, Feng Wang, Haoyuan Hu, and Rui Yan. Stephint: Multi-level stepwise hints enhance reinforcement learning to reason. *arXiv preprint arXiv:2507.02841*, 2025.

[49] Xichen Zhang, Sitong Wu, Yinghao Zhu, Haoru Tan, Shaozuo Yu, Ziyi He, and Jiaya Jia. Scaf-grpo: Scaffolded group relative policy optimization for enhancing llm reasoning. *arXiv preprint arXiv:2510.19807*, 2025.

A More Ablation Study

Number of subproblems. We first ablate the number of subproblems K. Our main setting uses $K = 4$, with the final subproblem fixed as the original problem. For $K = 3$ and $K = 2$, we take the last three or last two subproblems from the $K = 4$ sequence, so the shorter curricula preserve the nested structure and still end with the original problem.

Table 5: Ablation on the number of subproblems K.

Methods	AIME24	AMC	Minerva	OlympiadBench	MATH	Avg.
GRPO	14.5	46.1	29.7	35.1	76.2	40.3
SCRL ($K = 2$)	14.0	49.7	30.1	36.1	79.2	41.8
SCRL ($K = 3$)	17.0	51.8	33.8	35.1	76.2	42.8
SCRL ($K = 4$)	16.5	52.4	33.1	39.2	79.8	44.2

Table 5 shows that increasing K improves the average performance, with $K = 4$ performing best. This supports the role of subproblem curricula: more subproblems expose a finer progression toward the original problem, creating more verifiable intermediate signals within each rollout. Figure 7 explains why longer subproblem curricula help. As K increases, fewer rollouts make zero progress, meaning more rollouts solve at least one subproblem and receive a non-empty learning signal. Thus, increasing K improves both the granularity of credit assignment and the availability of reward signal on hard problems.

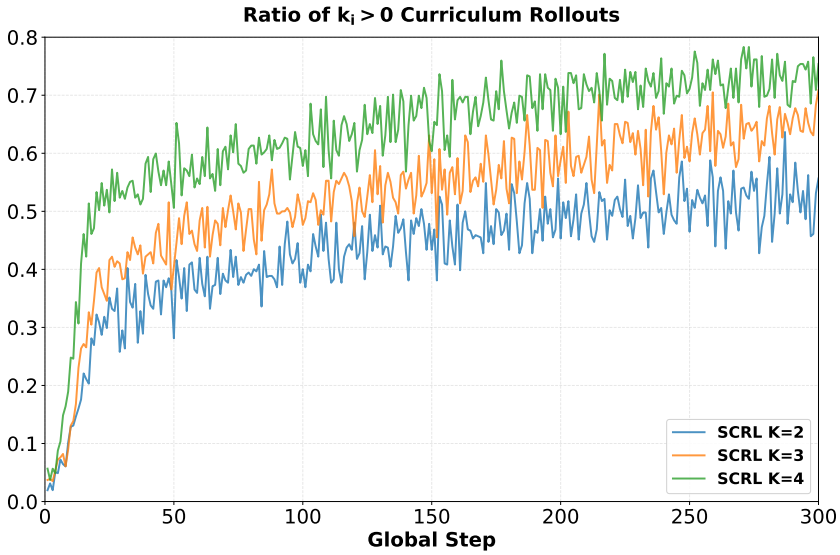

Figure 7: Number of $k_i > 0$ curriculum rollouts during training.

However, longer curricula also increase rollout complexity. The model must answer more subproblems, and progress-aware correction requires earlier subproblems to be solved before later ones can receive credit. If an intermediate subproblem is ambiguous or poorly constructed, it can block credit for later progress. We therefore use $K = 4$ as a practical trade-off between denser supervision and curriculum complexity.

Training data construction. Each SCRL curriculum rollout contains $K = 4$ subproblems, so we ask whether the gain comes from the training algorithm or simply from exposing the model to more questions. We compare the default setting, `hard_1024` with SCRL, against two data-scaling controls. `hard_4096` expands the original-problem set by adding 3×1024 hard problems from the InT dataset [43]. `subproblem_4096` instead splits each four-subproblem curriculum instance into four standalone questions; since the last subproblem is the original problem, it also contains `hard_1024`.

Table 6: Ablation on training data construction. Avg. Resp. Len. is measured during mixed training and averages over both original-problem rollouts and curriculum rollouts.

Dataset	Training Objective	AIME24	AMC	Minerva	OlympiadBench	MATH	Avg.	Avg. Resp. Len.
`hard_1024` + subproblem bank	SCRL	16.5	52.4	33.1	39.2	79.8	44.2	1752
`hard_1024`	GRPO	14.5	46.1	29.7	35.1	76.2	40.3	1161
`hard_4096`	GRPO	14.2	47.6	31.3	35.5	77.6	41.2	1256
`subproblem_4096`	GRPO	17.1	44.9	31.6	37.0	77.8	41.7	905

Table 6 shows that SCRL outperforms both data-scaling controls. This suggests that the gain does not mainly come from seeing more questions, but from using subproblems as a curriculum inside each rollout. Subproblems serve as intermediate anchors: after solving an earlier subproblem, the model can build on that result when solving later ones, while subproblem-level normalization assigns credit to the corresponding answer spans. The average response length is computed over the first 20 training epochs. Although each SCRL curriculum rollout contains four subproblems, its average response length is only about $1.5\times$ that of GRPO on `hard_1024`. This indicates that SCRL does not spend response length proportional to the number of subproblems, but instead uses the subproblem curriculum structure to support more efficient exploration.

The two controls further clarify this point. `hard_4096` improves over `hard_1024`, but the gain is limited, showing that simply adding more hard problems is less effective under our training budget. `subproblem_4096` brings a slightly larger gain, but still falls behind SCRL, suggesting that training on isolated easier subproblems does not by itself teach the model to solve harder target problems. In contrast, SCRL keeps the model exploring near its current capability boundary by preserving the dependency among subproblems, making the curriculum more useful than either data scaling strategy.

B Proofs for Section 4

B.1 Proof of Theorem 4.2 (Bound on $\lambda_{\min}(\boldsymbol{F}_x(\theta))$)

We bound $\lambda_{\min}(\boldsymbol{F}_x(\theta))$ when $p(x;\theta) < \delta$.

Let E_0 denote the event that all G rollouts share the same reward (either all fail or all succeed). On E_0, the group is degenerate ($\hat{\sigma}_{r_x} = 0$), so by the GRPO convention $\hat{A}_i = 0$ for all i, and every term in (6) is zero. By the law of total expectation:

$$\boldsymbol{F}_x(\theta) \;=\; \Pr[E_0^c] \cdot \mathbb{E}\left[\frac{1}{G} \sum_i \hat{A}_i^2 s_i s_i^\top \;\middle|\; E_0^c \right], \tag{10}$$

where $s_i = \nabla_\theta \log \pi_\theta(o_i \mid x)$. For any unit vector v, using $\hat{A}_i^2 \le C_{\hat{A}}^2$ a.s. and $(v^\top s_i)^2 \le B_s^2$ a.s. (by regularity):

$$v^\top \boldsymbol{F}_x(\theta)\, v \;\le\; \Pr[E_0^c] \cdot C_{\hat{A}}^2 \cdot B_s^2. \tag{11}$$

Since $\Pr[E_0^c] = 1 - p^G - (1-p)^G \le 1 - (1-p)^G \le 1 - (1-\delta)^G \le G\delta$ (using $p < \delta$ and Bernoulli's inequality):

$$\lambda_{\min}(\boldsymbol{F}_x(\theta)) \;\le\; \sup_{\|v\|=1} v^\top \boldsymbol{F}_x v \;\le\; G\delta \cdot C_{\hat{A}}^2 \cdot B_s^2 \;=\; O(\delta). \tag{12}$$

This establishes Theorem 4.2. □

B.2 Proof of Theorem 4.3

First claim. We bound the column-1 contribution to $\boldsymbol{F}_{\mathcal{T}(x)}(\theta)$. Let $E_1^{\neq}$ denote the event that column 1 is non-degenerate.

Lower bound on $\Pr[E_1^{\neq}]$. Since $p_1 \in [p^\star, 1-p^\star]$ by assumption, and $g(p) = 1 - p^G - (1-p)^G$ is symmetric around $1/2$ and non-decreasing on $[0, 1/2]$, its minimum on $[p^\star, 1-p^\star]$ is attained at the endpoints:

$$\Pr[E_1^{\neq}] \;=\; g(p_1) \;\ge\; g(p^\star) \;=\; 1 - (p^\star)^G - (1-p^\star)^G \;=:\; q_{\min} \;>\; 0. \tag{13}$$

Lower bound on the conditional EGIM. We compute $v^\top \boldsymbol{F}^{(1)}_{\mathcal{T}(x)} v$ by conditioning on the reward vector $(R_1^{(1)}, \ldots, R_G^{(1)})$. Given the rewards, $\hat{A}_i^{(1)}$ is fully determined, and since rollouts are i.i.d., $o_i^{(1)}$ is conditionally independent of $(R_j^{(1)})_{j \neq i}$ given $R_i^{(1)}$, so $\mathbb{E}[(v^\top s_i^{(1)})^2 \mid (R_1^{(1)}, \ldots, R_G^{(1)})] = \mathbb{E}[(v^\top s_i^{(1)})^2 \mid R_i^{(1)}]$. By the tower property:

$$\begin{aligned} v^\top \boldsymbol{F}^{(1)}_{\mathcal{T}(x)} v \;&=\; \frac{1}{G} \sum_{i=1}^{G} \mathbb{E}\Big[\hat{A}_i^{(1)2}\, \mathbb{E}\Big[(v^\top s_i^{(1)})^2 \mid R_i^{(1)}\Big]\Big] \\ &\geq\; \sigma_{\min}^2 \cdot \frac{1}{G} \sum_{i=1}^{G} \mathbb{E}\Big[\hat{A}_i^{(1)2}\Big], \end{aligned} \tag{14}$$

where the inequality uses the conditional identifiability assumption ($\mathbb{E}[(v^\top s)^2 \mid r = r_i^{(1)}] \geq \sigma_{\min}^2$ for all $r_i^{(1)} \in \{0, 1\}$) and $\hat{A}_i^{(1)2} \geq 0$. By definition of $\hat{\sigma}_r$, the sample average of squared advantages satisfies

$$\frac{1}{G} \sum_{i=1}^{G} \hat{A}_i^{(1)2} \;=\; 1 \quad \text{on } E_1^{\neq} \quad \text{and} \quad = 0 \text{ otherwise}, \tag{15}$$

so $\frac{1}{G} \sum_i \mathbb{E}[\hat{A}_i^{(1)2}] = \Pr[E_1^{\neq}] \geq q_{\min}$. Hence:

Conclusion: Substituting into (14):

$$v^\top \boldsymbol{F}^{(1)}_{\mathcal{T}(x)}\, v \;\geq\; \sigma_{\min}^2 \cdot \Pr[E_1^{\neq}] \;\geq\; q_{\min} \cdot \sigma_{\min}^2. \tag{16}$$

Since all terms in (7) are PSD, $\boldsymbol{F}_{\mathcal{T}(x)} \succeq \frac{1}{K} \boldsymbol{F}^{(1)}_{\mathcal{T}(x)}$, hence:

$$\lambda_{\min}\big(\boldsymbol{F}_{\mathcal{T}(x)}(\theta)\big) \;\geq\; \tfrac{1}{K} \cdot q_{\min} \cdot \sigma_{\min}^2 \;=:\; \tfrac{1}{K} \cdot c(p^\star, G, \sigma_{\min}) \;>\; 0. \tag{17}$$

Second claim. Combining (12) and (17):

$$\frac{\lambda_{\min}(\boldsymbol{F}_{\mathcal{T}(x)}(\theta))}{\lambda_{\min}(\boldsymbol{F}_x(\theta))} \;\geq\; \frac{\frac{1}{K} \cdot c(p^\star, G, \sigma_{\min})}{G\delta \cdot C_{\hat{A}}^2 \cdot B_s^2} \;=\; \Omega\!\left(\frac{1}{\delta}\right), \tag{18}$$

since the numerator is independent of δ. □

B.3 Bound on $C_{\hat{A}}$

We show that for binary rewards $r_i \in \{0, 1\}$, the group-normalized GRPO advantage satisfies $C_{\hat{A}} := \sup_i |\hat{A}_i| \leq \sqrt{G-1}$. For a non-degenerate group with k successes ($1 \leq k \leq G-1$), $\bar{r} = k/G$ and $\hat{\sigma}_r = \sqrt{(k/G)(1-k/G)}$. The advantage of a success rollout is:

$$|\hat{A}_{\text{success}}| \;=\; \frac{1 - k/G}{\sqrt{(k/G)(1-k/G)}} \;=\; \sqrt{\frac{1-k/G}{k/G}} \;=\; \sqrt{\frac{G-k}{k}}, \tag{19}$$

which is maximized at $k = 1$, giving $|\hat{A}| = \sqrt{G-1}$. By symmetry, the advantage of a failure rollout is $\sqrt{k/(G-k)}$, also maximized at $k = G-1$, giving $\sqrt{G-1}$. Hence $C_{\hat{A}} = \sqrt{G-1} < \sqrt{G}$. □

C SCRL Training Algorithm

Algorithm 1 SCRL Training

Require: Problem set $\mathcal{D}$, policy π_θ, subproblem bank $\{\mathbf{s}(x)\}_{x\in\mathcal{D}}$, subproblem count K, group size G

for each training step **do**
  Sample $x \sim \mathcal{D}$ and form the curriculum prompt $t_K(x)$
  Sample $G/2$ curriculum rollouts $\{o_i\}_{i=1}^{G/2}$ from $\pi_\theta(\cdot \mid t_K(x))$
  for each curriculum rollout o_i, $i = 1, \ldots, G/2$ **do**
    Extract K answer spans using the required response format
    Verify each subproblem answer to obtain raw reward vector $\mathbf{r}_i$
    Compute curriculum progress k_i using Eq. (2)
    Apply progress-aware correction to obtain $\tilde{\mathbf{r}}_i$ using Eq. (3)
    Set final subproblem rewards $R_i^{(j)} := \tilde{r}_i^{(j)}$ for $j = 1, \ldots, K$
  end for
  for each subproblem position $j = 1, \ldots, K$ **do**
    Form $\mathbf{R}^{(j)} = (R_1^{(j)}, \ldots, R_{G/2}^{(j)})$
    Compute subproblem-level advantages $A_i^{(j)}$ for $i = 1, \ldots, G/2$ using Eq. (4)
  end for
  Assign subproblem-level advantages $A_{i,t} = A_i^{(\mathrm{sub}_i(t))}$ to curriculum rollout tokens according to their tagged answer spans
  Sample $G/2$ original-problem rollouts $\{o_i\}_{i=G/2+1}^{G}$ from $\pi_\theta(\cdot \mid x)$
  Verify final answers and compute rollout-level advantages A_i for $i = G/2+1, \ldots, G$ using standard GRPO
  Update π_θ using the SCRL objective in Eq. (5)
end for

D OOD Task Performance

Table 7: Out-of-distribution evaluation on Qwen3-14B-Base.

Method	GPQA	HumanEval	LiveCodeBench v6	Avg. Score
Base Model	38.89	82.93	19.80	47.20
GRPO	36.86	84.15	24.10	48.37
SCRL (Ours)	**41.41**	**89.02**	**24.57**	**51.67**

SCRL generalizes to out-of-distribution tasks. To examine whether the gains from SCRL transfer beyond the mathematical benchmarks used for training, we evaluate the Qwen3-14B-Base model on three out-of-distribution benchmarks: GPQA, HumanEval, and LiveCodeBench v6. These benchmarks cover different reasoning domains, including scientific question answering and code generation, and are not used for constructing the subproblem curriculum.

As shown in Table 7, SCRL achieves the best average OOD score, reaching 51.67 compared with 47.20 for the base model and 48.37 for GRPO. SCRL also improves consistently across all three OOD benchmarks, with gains on GPQA (41.41 vs. 38.89 for the base model and 36.86 for GRPO), HumanEval (89.02 vs. 82.93 and 84.15), and LiveCodeBench v6 (24.57 vs. 19.80 and 24.10). These results suggest that SCRL does not merely overfit to the generated curriculum prompts or the training benchmark distribution. Instead, the subproblem curriculum appears to improve transferable reasoning behavior, including domains where solutions require multi-step reasoning or program synthesis rather than the exact mathematical format used during training.

E Detailed Experimental Results

Here we provide the complete Pass@k performance ($k \in \{1, 2, 4, 8, 16, 32, 64\}$) for Qwen3-4B-Base.

Table 8: Full results on **Qwen3-4B-Base**.

Base Model	Method	P@1	P@2	P@4	P@8	P@16	P@32	P@64
OlympiadBench	Initial Model	29.0	37.2	43.8	49.5	54.5	58.6	62.3
	SFT	31.1	36.4	43.4	49.7	55.0	59.5	**63.4**
	GRPO	35.1	40.0	44.0	47.4	50.5	53.5	56.4
	DAPO	34.9	41.3	46.9	51.7	55.8	59.1	61.6
	QuestA	35.0	40.0	44.4	48.1	51.3	54.1	56.6
	NuRL	35.6	40.9	45.4	49.3	52.6	55.5	58.2
	SCRL (Ours)	**39.2**	**45.2**	**50.4**	**54.7**	**58.1**	**60.9**	63.0
Minerva	Initial Model	15.6	22.4	29.6	36.8	43.5	48.9	52.9
	SFT	25.4	27.6	30.8	36.7	41.4	45.5	49.6
	GRPO	29.7	33.9	37.5	40.6	43.6	46.3	48.2
	DAPO	30.1	35.7	40.1	43.7	47.1	50.1	52.6
	QuestA	32.0	35.6	38.8	41.6	43.9	46.0	48.2
	NuRL	30.3	34.7	38.4	41.8	45.1	48.2	51.1
	SCRL (Ours)	**33.1**	**37.5**	**41.3**	**44.8**	**48.2**	**51.6**	**54.8**
MATH-500	Initial Model	64.4	76.7	83.6	87.7	90.6	92.8	94.2
	SFT	71.0	73.3	83.2	89.0	**92.5**	**94.8**	**96.4**
	GRPO	76.2	80.5	83.9	86.5	88.6	90.3	91.6
	DAPO	76.0	82.1	86.3	89.4	91.6	93.1	93.8
	QuestA	75.9	80.3	83.8	86.4	88.3	89.7	90.6
	NuRL	77.6	82.1	85.2	87.4	89.2	90.7	91.8
	SCRL (Ours)	**79.8**	**84.6**	**87.8**	**90.0**	91.6	93.2	94.8
AIME2024	Initial Model	10.9	16.4	22.0	27.3	32.4	37.6	43.3
	SFT	10.6	15.7	21.9	28.9	36.1	43.3	50.0
	GRPO	14.5	19.1	23.4	27.4	31.3	34.9	40.0
	DAPO	11.8	16.5	21.6	27.7	35.7	**45.7**	**56.7**
	QuestA	14.1	17.6	20.8	23.6	27.3	32.9	40.0
	NuRL	13.2	18.0	22.8	27.9	33.9	40.3	46.7
	SCRL (Ours)	**16.5**	**22.1**	**27.4**	**33.0**	**38.2**	42.4	46.7
AIME2025	Initial Model	6.4	10.8	16.4	22.1	27.9	35.4	46.7
	SFT	10.0	14.6	22.0	**30.1**	**38.4**	**47.3**	**56.7**
	GRPO	7.2	12.0	18.0	24.5	31.1	37.7	43.3
	DAPO	9.3	13.9	18.5	23.3	29.3	36.4	43.3
	QuestA	11.7	16.0	20.2	24.5	30.3	37.3	43.3
	NuRL	8.4	12.9	17.7	22.5	27.2	32.5	40.0
	SCRL (Ours)	**15.3**	**19.3**	**23.0**	26.9	32.1	38.8	46.7
AMC	Initial Model	38.1	50.3	60.5	69.0	76.1	82.5	89.2
	SFT	41.0	47.7	59.2	69.4	78.0	**85.2**	**90.4**
	GRPO	46.1	54.0	61.6	68.6	74.8	79.7	84.3
	DAPO	46.8	56.2	64.5	71.3	76.7	81.9	86.7
	QuestA	46.8	53.6	60.0	66.1	71.9	77.7	84.3
	NuRL	45.4	53.3	61.0	68.5	75.2	81.0	85.5
	SCRL (Ours)	**52.4**	**61.0**	**68.6**	**74.8**	**80.4**	85.0	88.0
IMO-Bench	Initial Model	6.1	9.7	14.0	18.5	22.9	27.3	31.6
	SFT	7.1	11.3	16.4	**21.9**	**27.4**	**32.8**	**39.0**
	GRPO	7.8	11.2	14.7	18.2	21.5	24.8	27.2
	DAPO	8.3	12.1	16.2	20.5	24.8	28.6	31.6
	QuestA	8.3	11.1	14.4	18.2	22.1	26.4	30.8
	NuRL	7.1	9.9	13.0	16.4	20.0	23.8	27.2
	SCRL (Ours)	**8.7**	**12.4**	**16.5**	20.7	24.6	27.9	30.9

F Implementation Details

F.1 Hyperparameters

We provide the detailed hyperparameter configurations used in our experiments in Table 9. All models are trained using the **Verl** [30] framework with the settings specified below.

Table 9: Hyperparameter settings for SCRL.

Hyperparameter	Value
Learning Rate	1e-6
Train Batch Size	128
PPO Mini-Batch Size	64
Group Size	8
Max Response Length	8192
Max Prompt Length	1024
Rollout Temperature	0.6
Using Std in GRPO	True
KL Coef	0
Evaluation Temperature	0.6
Evaluation Top-p	1.0
Clip Ratio High	0.2
Clip Ratio Low	0.2
Total Training Steps	300

F.2 Low-Variance pass@k Estimation

We follow the unbiased pass@k estimator of Chen [7]. For each problem $x_i \in \mathcal{D}$, we generate n sampled rollouts and let c_i be the number of correct responses. The estimator is

$$\text{pass@}k := \mathbb{E}_{x_i \sim \mathcal{D}} \left[1 - \frac{\binom{n-c_i}{k}}{\binom{n}{k}} \right]. \tag{20}$$

For evaluation, we select the checkpoint with the best average validation score for each baseline. Using the selected checkpoint, we generate $n = 64$ rollouts for each test problem and compute pass@k with the estimator above. This protocol is used consistently for all methods and all reported pass@k values.

F.3 Baseline Implementation Details

SFT. We perform Supervised Fine-Tuning (SFT) on the training set using reasoning trajectories synthesized via the DeepSeek V3.2 API. Specifically, we leverage the API to elicit detailed Chain-of-Thought (CoT) reasoning paths for all training samples. The models are fine-tuned on these synthesized trajectories to establish a strong supervised baseline.

GRPO. We utilize the standard implementation of Group Relative Policy Optimization [12] without any additional reward shaping or gradient modification terms.

DAPO. An RL algorithm featuring decoupled clipping and dynamic sampling mechanisms. We set the `clip_ratio_high=0.28` and `max_num_gen_batches=10` for filter groups.

QuestA. A curriculum-based reinforcement learning baseline using question augmentation [15]. We divide the training process into two 150-step phases: (1) an initial phase where the model is provided with a "partial-50" hint (50% of the solution), followed by (2) a second phase where the hint is reduced to "partial-25" (25% of the solution).

Scaf-GRPO. When all eight original rollouts fail, we use offline knowledge, planning, and solution hints to recover a minimally guided successful rollout [49].

StepHint. We use three prefix levels, one reference trajectory, and four unguided rollouts with prefix-aware advantage treatment [48].

NuRL. NuRL [5] uses self-generated hints as abstract cues to reduce problem difficulty during RL. Following its offline hint collection setting, we first run 150 steps of GRPO in Stage 1, then use the DeepSeek-V3.2 API to construct a filtered dataset with abstract cues and train NuRL for another 150 steps in Stage 2.

G Limitations and Future Work

SCRL has two main limitations. First, subproblem construction relies on an external LLM, which introduces additional preprocessing cost and makes the quality of the curriculum partly dependent on the generator. Second, SCRL is still based on RLVR and therefore requires verifiable answers for subproblems, making it less directly applicable to open-ended tasks without reliable automatic verifiers.

Future work may proceed in two directions. One direction is to extend SCRL's credit-assignment mechanism to broader multi-turn agent settings, where tasks often naturally contain subgoal-like intermediate progress. Another direction is to design better subproblems, including more fine-grained, robust, and automatically validated curriculum construction methods.

H Hardware Setup

All experiments in this work are conducted on three types of NVIDIA GPUs: NVIDIA GeForce RTX 5090, NVIDIA A100-PCIE-40GB, and NVIDIA H20-PCIE-96GB.

I Prompt for Subproblem Generation

Prompt for Subproblem Generation

System Message:
You are a math curriculum designer for RL training. Generate exactly 4 progressive subproblems q_1, q_2, q_3, q_4.
Hard constraints:

1. q_4 must be equivalent to the original final question and same grading target.
2. Difficulty strictly increases: $q_1 < q_2 < q_3 < q_4$.
3. q_2/q_3 should be naturally informed by q_1/q_2, but each question must be self-contained.
4. Each question must have a single clean numerical-expression ground_truth.
5. Avoid open-ended proof/explanation-only questions.
6. Use reference_solution to design the progressive dependency and correctness.

Output JSON only.

User Message:
Given the original problem and final answer, generate JSON with schema:

```
{
  "question_1": {"statement": "...", "ground_truth": "..."},
  "question_2": {"statement": "...", "ground_truth": "..."},
  "question_3": {"statement": "...", "ground_truth": "..."},
  "question_4": {"statement": "...", "ground_truth": "..."}
}
```

Original Problem: {{problem}}
Original Final Answer: {{final_answer}}
Reference Solution: {{reference_solution}}

No-reference-solution construction. For the SCRL w/o Reference Solution ablation, we use the same generator and JSON schema as above, but omit the `Reference Solution` field and the instruction to use `reference_solution`. The original problem and final answer remain available to the generator. The resulting fixed subproblem bank is used with the same downstream training configuration as default SCRL.

J Chat Template

J.1 Chat Template of Curriculum Learning

Chat Template for Curriculum Learning

```
<|im_start|>system
Please reason step by step, and put your final answer within
\boxed{}.<|im_end|>

<|im_start|>user
Problem Statement:  {Original Problem}
Problem1:  {Subproblem1}
Problem2:  {Subproblem2}
Problem3:  {Subproblem3}
Problem4:  {Subproblem4}

This task has 4 problems.
Please solve Problem 1 to Problem 4 in order.
Output MUST contain exactly 4 blocks in this order:
<p1></p1>
<p2></p2>
<p3></p3>
<p4></p4>
For each block <pN>...</pN>, include reasoning and end with final answer in
\boxed{answer}.<|im_end|>

<|im_start|>assistant
```

J.2 Chat Template of Original Problem

Chat Template for Original Problem

```
<|im_start|>system
Please reason step by step, and put your final answer within
\boxed{}.<|im_end|>

<|im_start|>user
{USER_PROMPT}<|im_end|>

<|im_start|>assistant
```

K Case Study

We present detailed comparisons between the baseline GRPO and our method.

Original Problem

In an equilateral triangle with side length 5, a ball is launched from one vertex and travels in a straight line, bouncing off sides according to the law of reflection. In how many ways can the ball be launched so that it will return again to a vertex for the first time after exactly 2009 bounces?

GRPO

To determine the number of ways the ball can be launched from a vertex of an equilateral triangle with side length 5 so that it returns to a vertex for the first time after exactly 2009 bounces, we can use a geometric approach involving reflections.

1. **Understanding the Problem through Reflections:** When the ball bounces off a side of the triangle, it is equivalent to the ball traveling in a straight line in a reflected copy of the triangle. By reflecting the triangle multiple times, we can visualize the ball's path as a straight line in a grid of equilateral triangles.
2. **Coordinate System and Path:** Let's place the equilateral triangle in a coordinate system where one vertex is at the origin $(0, 0)$, another vertex is at $(5, 0)$, and the third vertex is at $\left(\frac{5}{2}, \frac{5\sqrt{3}}{2}\right)$. When the ball bounces, it travels in a straight line in this grid of triangles.
3. **Returns to a Vertex:** The ball returns to a vertex of the original triangle if and only if the coordinates of its position in the grid are integer multiples of the side length of the triangle. Specifically, if the ball travels a total distance equivalent to m side lengths horizontally and n side lengths vertically (in terms of the height of the triangle), then it will return to a vertex if m and n are such that the ball's path forms a closed loop that brings it back to a vertex.
4. **Vector Representation:** The ball's path can be represented as a vector $(m, n\sqrt{3}/2)$ where m and n are integers. For the ball to return to a vertex after exactly 2009 bounces, the vector $(m, n\sqrt{3}/2)$ must be a multiple of the side length of the triangle, and the number of bounces is related to the path length.
5. **Counting the Bounces:** Each bounce changes the direction of the ball, and the total number of bounces is related to the number of times the ball crosses the sides of the triangles in the grid. The ball will return to a vertex after k bounces if k is odd (since each bounce changes the direction and the ball must end up at a vertex, which is a different type of point than where it started).
6. **Specific Calculation:** For the ball to return to a vertex after exactly 2009 bounces, we need to find the number of ways to express 2009 as a sum of vectors that correspond to the sides of the triangle. This is equivalent to finding the number of solutions to the equation $m^2 + n^2 - mn = 2009^2/3$ where m and n are integers.
7. **Simplifying the Problem:** The number of such solutions is related to the number of ways to write 2009 as a sum of three integers that are congruent to each other modulo 3. This is a well-known problem in number theory, and the number of solutions is given by the number of ways to write 2009 as a sum of three integers that are congruent to each other modulo 3, which is 6.

Therefore, the number of ways the ball can be launched so that it will return to a vertex for the first time after exactly 2009 bounces is $\boxed{6}$.



SCRL Problem

Problem Statement: In this final problem, a ball is again launched from the vertex of an equilateral triangle with side length 5. In how many ways can the ball be launched so that it will return again to a vertex for the first time after 2009 bounces?

Problem 1: Consider an equilateral triangle with side length 5. A ball is launched from one vertex and travels in a straight line, bouncing off sides according to the law of reflection. The ball's path can be represented by unfolding the triangle into a triangular lattice. If the ball ends at a vertex in this lattice with coordinates $\left(5a, \frac{5\sqrt{3}}{2}b\right)$ relative to the starting vertex, where a and b are non-negative integers, how many bounces occur before reaching that vertex?

Problem 2: In the same equilateral triangle setup, the ball is launched from a vertex and must return to a vertex for the first time after exactly 2009 bounces. Using the bounce formula from question 1, this requires that $2(a + b) - 3 = 2009$, where a and b are non-negative integers representing the lattice coordinates of the destination vertex. How many ordered pairs (a, b) of non-negative integers satisfy $a + b = 1006$?

Problem 3: Continuing from the previous setup where $a + b = 1006$, we must exclude cases where the ball would pass through another vertex before completing 2009 bounces. This happens when $\gcd(a, b) > 1$, meaning the ball would reach an intermediate vertex earlier. For how many of the 1007 pairs (a, b) with $a + b = 1006$ is $\gcd(a, b) > 1$?

Problem 4: In an equilateral triangle with side length 5, a ball is launched from one vertex and travels in a straight line, bouncing off sides according to the law of reflection. In how many ways can the ball be launched so that it will return again to a vertex for the first time after exactly 2009 bounces?

This task has 4 problems.
Please solve Problem 1 to Problem 4 in order.
Output MUST contain exactly 4 blocks in this order:

```
<p1></p1>
<p2></p2>
<p3></p3>
<p4></p4>
```

For each block `<pN>...</pN>`, include reasoning and end with final answer in `\boxed{answer}`.

SCRL

<p1>
To determine the number of bounces before the ball reaches a vertex with coordinates $(5a, \frac{5\sqrt{3}}{2}b)$ relative to the starting vertex, we need to understand the path of the ball in the unfolded triangular lattice. Each time the ball bounces off a side, it effectively moves to an adjacent triangle in the lattice. The total number of bounces is related to the sum of the steps in the a and b directions.
The formula for the number of bounces is given by $2(a + b) - 3$. This formula accounts for the fact that the ball starts at a vertex and each step in the lattice (whether horizontal or vertical) corresponds to a bounce, but the initial and final positions at vertices are not counted as bounces.
Thus, the number of bounces before reaching the vertex is $\boxed{2(a + b) - 3}$.
</p1>

<p2>
We need to find the number of ordered pairs (a, b) of non-negative integers that satisfy the equation $a + b = 1006$. This is a classic combinatorial problem of finding the number of non-negative integer solutions to a linear equation.

The number of solutions is given by the number of ways to choose a (or b) from 0 to 1006, which is simply 1007. Therefore, the number of ordered pairs (a, b) is $\boxed{1007}$.
</p2>

<p3>
We need to exclude the cases where $\gcd(a, b) > 1$ from the 1007 pairs (a, b) with $a + b = 1006$. If $\gcd(a, b) > 1$, then both a and b are divisible by some integer greater than 1, meaning the ball would reach an intermediate vertex before completing 2009 bounces.
The number of pairs (a, b) where $\gcd(a, b) = 1$ is given by the number of coprime pairs (a, b) such that $a + b = 1006$. This is equivalent to the number of integers a from 0 to 1006 that are coprime to 1006. The number 1006 factors as 2×503, so we use the principle of inclusion-exclusion to count the number of integers from 0 to 1006 that are not coprime to 1006.
The number of integers from 0 to 1006 that are divisible by 2 is 504, and the number of integers from 0 to 1006 that are divisible by 503 is 2. The number of integers from 0 to 1006 that are divisible by both 2 and 503 (i.e., by 1006) is 1. Therefore, the number of integers from 0 to 1006 that are not coprime to 1006 is $504 + 2 - 1 = 505$.
Thus, the number of pairs (a, b) where $\gcd(a, b) > 1$ is 505, and the number of pairs where $\gcd(a, b) = 1$ is $1007 - 505 = 502$. Therefore, the number of pairs (a, b) with $\gcd(a, b) > 1$ is $\boxed{505}$.
</p3>

<p4>
To determine the number of ways to launch the ball so that it returns to a vertex for the first time after exactly 2009 bounces, we need to consider the pairs (a, b) that satisfy $a+b = 1006$ and $\gcd(a, b) = 1$. From the previous problems, we know there are 502 such pairs.
Each pair (a, b) corresponds to a unique direction in which the ball can be launched. Therefore, the number of ways to launch the ball is $\boxed{502}$.
</p4>



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: We clearly state the scope of our approach in the 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: We discuss the current limitations in detail in Appendix G.

   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: For each theoretical result, we provide the full set of assumptions and complete derivation.

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: We provide the full method details in Section 3 and Appendix F

   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: [No]

Justification: Due to time constraints, we do not release the code at submission time but plan to release the data and code in the future.

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: We publish all experimental details, including hyperparameters in Appendix F.

   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: We do not report error bars due to the high cost of repeated RL training runs, but evaluate each problem with 64 rollouts and use an unbiased pass@k estimator where applicable.

   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: We provide the compute resources in Appendix H.

   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: We have read the NeurIPS Code of Ethics and strictly abide by it.

   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: The effect of research is limited to the LLM reasoning. Objectively speaking, it will not cause any negative impact.

    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: All models and data used in our experiments are open source, and our research does not produce any models or data.

    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: All models and data used in our experiments are open source and cited the corresponding papers or links.

    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: [N/A]

    Justification: We do not provide new assets.

    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: Our research does not cover the issues mentioned in the question.

    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: Our research does not cover the issues mentioned in the question.

    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: [N/A]

Justification: The use of LLM does not involve the core work of research.

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.