StepHint: Multi-level Stepwise Hints Enhance Reinforcement Learning to Reason

Kaiyi Zhang[1,2,3], **Ang Lv**[1,2,3], **Jinpeng Li**[4]
Yongbo Wang[5], **Feng Wang**[5], **Haoyuan Hu**[5], **Rui Yan**[6*]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Beijing Key Laboratory of Research on Large Models and Intelligent Governance
[3]Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE
[4]Peking University, [5]Ant Group
[6]School of Artificial Intelligence, Wuhan University
kyzhang@ruc.edu.cn, rui.yan@whu.edu.cn

Abstract

Reinforcement learning with verifiable rewards (RLVR) is a promising approach for improving the complex reasoning abilities of large language models (LLMs). However, current RLVR methods face two significant challenges: the near-miss reward problem, where a small mistake can invalidate an otherwise correct reasoning process, greatly hindering training efficiency; and exploration stagnation, where models tend to focus on solutions within their "comfort zone," lacking the motivation to explore potentially more effective alternatives. To address these challenges, we propose StepHint, a novel RLVR algorithm that utilizes multi-level stepwise hints to help models explore the solution space more effectively. StepHint partitions valid reasoning chains into reasoning steps using our proposed adaptive partitioning method. The initial few steps are used as hints, and simultaneously, multiple-level hints (each comprising a different number of steps) are provided to the model. This approach directs the model's exploration toward a promising solution subspace while preserving its flexibility for independent exploration. By providing hints, StepHint mitigates the near-miss reward problem, thereby improving training efficiency. Additionally, the external reasoning pathways help the model develop better reasoning abilities, enabling it to move beyond its "comfort zone" and mitigate exploration stagnation. StepHint outperforms competitive RLVR enhancement methods across six mathematical benchmarks and two out-of-domain benchmarks.[1]

1 Introduction

Eliciting the reasoning capabilities of large language models (LLMs) through Reinforcement Learning with Verifiable Rewards (RLVR) has emerged as a powerful paradigm (Jaech et al., 2024; Guo et al., 2025). In RLVR frameworks, a policy model explores the solution space by generating reasoning chains. The model is then optimized using algorithms like PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024), based on the advantages of final outcomes of these chains.

However, free exploration within the vast and complex solution space introduces significant challenges. A key issue is the **near-miss reward problem**, where a single incorrect step voids an otherwise reward-worthy reasoning chain. This leads to training inefficiency, as models expend resources on repeatedly almost-correct solutions. Moreover, as shown by Yue et al. (2025), existing RLVR methods often refine the model's ability to sample known reasoning chains rather than discover novel or higher-quality ones. Consequently, when a task exceeds the model's current capabilities, it tends to remain confined to its "comfort zone," unable to independently advance beyond familiar solutions—an issue we term **exploration stagnation**.

We propose StepHint, a novel augmented RLVR algorithm that integrates multi-level stepwise hints to address these challenges. StepHint leverages reasoning chains from advanced models, partitioning them into discrete reasoning steps.[2] It then provides only the initial few steps as hints for the model to complete the reasoning process. This approach effectively simplifies the solution space while preserving sufficient exploratory flexibility. Specifically, during RLVR—regardless of the policy optimization algorithm used—StepHint's pipeline comprises two key steps:

Step 1: Adaptive stepwise partitioning of reasoning chains. We introduce a probabilistic partitioning strategy that adaptively splits reasoning chains into meaningful steps, moving beyond su-

[*] Corresponding author.

[1]We have released our code at https://github.com/Cardinalere/StepHint.

[2]A *reasoning step* refers to a distinct logical stage within the reasoning chain and typically comprises multiple tokens. It should not be confused with a *token-prediction step* during generation or a *training step*.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 37846–37864

perficial markers like "First" or "Second." Our method estimates, at each token, the model's probability of generating an end-of-reasoning token (e.g., </think>). A position is identified as a candidate endpoint if the estimated probability at this position exceeds that of the next position. From these candidates, we randomly sample several endpoints, subject to a minimum length constraint, resulting in a fixed number of segments. The initial few steps are later provided as hints to guide the model's rollouts during RL training.

Step 2: Multi-level hints for problem solving. We define a hint's "level" as the number of initial reasoning steps it provides. A high-level hint contains many steps, potentially making the task trivial for the model, diminishing training efficacy. Conversely, a low-level hint with too few steps may be insufficient to guide the model, leaving it vulnerable to the "near-miss reward problem." Determining the optimal hint level for a given model-problem pair is inherently difficult, as the model abilities keep improving continuously. Our simple yet effective solution is to generate multi-level hints for each problem. With sufficiently fine-grained step partitioning, at least one hint level is likely to be suitable for the model's current ability.

By adaptively providing multi-level hints, StepHint effectively addresses both the near-miss reward problem and exploration stagnation. First, the model receives appropriate guidance to complete reasoning chains correctly, significantly reducing near-miss rewards and improving training efficiency—leading to faster convergence. Second, exposure to high-quality hints steers the policy toward more sophisticated reasoning patterns, preventing stagnation during independent exploration. This not only enhances the model's ability to break through its "comfort zone" but also avoids the poor generalization typical of SFT-based methods.

We evaluate StepHint by training a series of LLMs on mathematical tasks and comparing their performance against strong RLVR-enhanced baselines. Results demonstrate StepHint's effectiveness on both in-domain (math) and out-of-domain tasks.

• **In-domain performance**: Across six math benchmarks, StepHint surpasses existing RLVR methods. Notably, it achieves significant improvements in pass@k performance on two challenging benchmarks, AIME24 and AIME25.

• **Out-of-domain generalization**: StepHint also achieves the highest results on out-of-domain, non-mathematical benchmarks, highlighting its robust generalization beyond its training domain.

2 Background: Reinforcement Learning with Verifiable Rewards

Reinforcement Learning (RL) has been instrumental in advancing the reasoning capabilities of Large Language Models (LLMs) by enabling them to learn optimal reasoning chains through reward-based feedback (Hu et al., 2025; Guo et al., 2025). A popular paradigm in this domain is Reinforcement Learning with Verifiable Rewards (RLVR), which is a specialized RL paradigm for training LLMs on tasks where the correctness of an outcome can be objectively verified, such as mathematical problem-solving or code generation. In the RLVR framework, the learning process is typically driven by automated, often binary (correct/incorrect), reward signals, which facilitates scalable self-improvement (Gao et al., 2023).

Proximal Policy Optimization (PPO) PPO (Schulman et al., 2017) is a widely-used algorithm that optimizes the LLM's generation policy (π_θ) by maximizing expected rewards while preventing excessively large updates that could destabilize training. Given a problem q, the policy model π_θ samples N rollouts, denoted as $\{y_1, y_2, \cdots, y_N\}$, PPO optimizes the following objective:

$$\begin{aligned}\mathcal{L}_\theta^{PPO} = &\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big[r_{i,t}\hat{A}_{i,t},\\ &\text{clip}\left(r_{i,t}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\Big]\\ &-\beta D_{KL}(\pi_\theta||\pi_{\text{ref}}),\end{aligned} \tag{1}$$

where:

- $r_{i,t} = \frac{\pi_\theta(y_{i,t}|q,y_{i,<t})}{\pi_{\text{old}}(y_{i,t}|q,y_{i,<t})}$ is the probability ratio between the current policy π_θ and the policy before the update, π_{old}. The clip function bounds this ratio.
- $\hat{A}_{i,t}$ is the advantage of taking token $y_{i,t}$ as the t token of rollout i. It quantifies how much better that action is compared to the average action value at that state.
- $\beta D_{KL}(\pi_\theta||\pi_{\text{ref}})$ is a penalty term that discourages the current policy from deviating too far from a reference policy π_{ref}.

A key component of standard PPO is the critic model, which is required to calculate the token-level advantage $\hat{A}_{i,t}$. However, training this critic is computationally expensive and complex.

Group Relative Policy Optimization (GRPO) To address the challenges of PPO, GRPO (Shao et al., 2024) is introduced as a simpler yet effective alternative. It has demonstrated strong performance and efficiency, particularly in complex mathematical reasoning tasks (Liu et al., 2025; Yan et al., 2025; Zeng et al., 2025; Hu et al., 2025).

GRPO computes a single, uniform advantage value for all tokens within a rollout, based on the final outcome of that entire rollout. Specifically, for a given problem q, N rollouts $\{y_1, y_2, \cdots, y_N\}$ are sampled. Each complete rollout y_i is assigned a final reward $R(y_i)$. GRPO then calculates a rollout-level advantage by normalizing this reward across the group. This advantage value is assigned to every token within that rollout:

$$\hat{A}_{i,t}^{GRPO} = \frac{R(y_i) - \text{mean}\left(\{R(y_1), \cdots, R(y_N)\}\right)}{\text{std}\left(\{R(y_1), \cdots, R(y_N)\}\right)}. \tag{2}$$

By replacing the token-level advantage $\hat{A}_{i,t}$ with this rollout-level advantage $\hat{A}_{i,t}^{GRPO}$, GRPO retains the core PPO objective while eliminating the need for a critic model.

3 Method

We frame the reasoning process as a stepwise reduction of a solution space. This perspective inspires our core idea: guiding the model's exploration with stepwise hints. To build this foundation, we first formalize the generation of reasoning chains as a solution space exploration (Section 3.1). This formalization helps identify critical issues in existing methods, such as near-miss rewards and exploration stagnation, and lays the groundwork for our proposed method, StepHint (Section 3.2), which is designed to address these challenges.

3.1 Motivation: a solution space reduction view of reasoning

We model the autoregressive generation of a reasoning chain as a sequential reduction of a solution space $\mathcal{R}$, which comprises all possible reasoning chains for a given prompt $\mathcal{C}$ (Guo et al., 2025). This process is represented by a sequence of states, where each state $S_k = (\mathcal{C}, t_1, \ldots, t_k)$ corresponds to the partial reasoning chain after k tokens. The complexity or uncertainty of the remaining solution space at each state is quantified by the conditional entropy $H(\mathcal{R}|S_k)$.

While this entropy is generally intractable to compute directly, as it requires summing over all possible chains in $\mathcal{R}$, it serves as a powerful conceptual tool for analysis. A higher entropy indicates a complex and unconstrained solution space. The following proposition formalizes the intuition that each generation step, in expectation, reduces the solution space's complexity.

Proposition 1. *Let $\mathcal{R}$ be the solution space and S_{k-1} be the state after $k-1$ tokens have been generated. Upon generating the next token t_k to form state $S_k = (S_{k-1}, t_k)$, the expected entropy of the solution space is bounded by the current entropy:*

$$\mathbb{E}_{t_k \sim P(\cdot|S_{k-1})}[H(\mathcal{R}|S_k)] \leq H(\mathcal{R}|S_{k-1}).$$

We leave the detailed proof in Appendix A. Proposition 1 formally establishes that autoregressive generation is a structured process of uncertainty reduction. However, the reduction in entropy quantifies the convergence of certainty, not necessarily the correctness or logical validity of the reasoning chain. A model can become increasingly certain about a flawed conclusion, which still manifests as a decrease in entropy.

This distinction reveals a critical failure mode. An early error can irrevocably prune the subspace of correct solutions, $R^* \subset R$. Formally, this occurs when the model reaches a state S_k where the probability of the correct solution set collapses to zero: $P(R^*|S_k) \approx 0$. Subsequently, the model may continue to confidently reduce entropy, but it does so within the incorrect subspace $R \backslash R^*$, inevitably arriving at a "confident but wrong" conclusion.

3.2 StepHint: multi-level stepwise hints enhance RLVR

Based on the views above, an initial error can lead to an incorrect final answer, even if the subsequent reasoning is logically sound, as it originates from a flawed premise. This "near-miss" reward problem can be mitigated by providing early guidance. Furthermore, a model's exploration of the solution space is intrinsically constrained by the model's ability. Without external guidance, the exploration will be limited to a narrow subspace (Yue et al., 2025; Lv et al., 2025b). Our proposed method,

StepHint, addresses these two challenges by leveraging part of the strong reasoning chains from a more capable model as hints during training.

Given a problem, StepHint first obtains a valid reasoning chain from a stronger model, and then performs two key stages to enhance the target model being trained: (1) adaptive stepwise partitioning of on-hand reasoning chains and (2) multi-level hints for problem solving.

In the following, we focus on detailing the latter two stages. We will frequently use the term "reasoning step" to refer to an intact unit of thought, which may consist of several sentences. Please note that this is distinct from a "training step," which refers to updating the model after processing a batch of data, or a "next-token prediction step," which generates a single token at a time.

3.2.1 Adaptive stepwise partitioning of on-hand reasoning chains

Definitions Let the reasoning chain be denoted as $G = t_1 \circ t_2 \circ \cdots \circ t_n$, where each t_i represents a single token, and $\circ$ denotes concatenation. A *reasoning step* corresponds to a sequence of tokens $t_i \circ \cdots \circ t_j (1 \leq i \leq j \leq n)$ that forms an intact unit of thought. A *hint* is composed of one or more reasoning steps and serves as a conditioning prompt that guides the target model's reasoning toward a promising direction, helping it explore otherwise intractable solution spaces. The *level of a hint* is determined by the number of reasoning steps it contains—the more reasoning steps included, the richer the guidance it offers to the target model, and thus the higher its level.

Method details We need a flexible method to adaptively partition a complete reasoning chain G into m reasoning steps, then combine appropriate number of reasoning steps as a hint in appropriate level to the target model. Conventional approach relies on syntactic cues, such as keywords like "first," "next," or "Step 1." However, such heuristics are prone to misidentifying the boundaries of reasoning steps and lack the flexibility (Moens, 2017).

In contrast, we leverage the model's output probability distribution to identify the boundaries of reasoning steps. We hypothesize that when a reasoning step concludes, the model's probability of completing the entire chain should be relatively high. Conversely, at the beginning of a new step, this probability should decrease as the model expects additional reasoning to follow. This perceived likelihood of reasoning completion can be captured by the probability assigned to a special "end-of-thinking" token, `</think>`, which is explicitly introduced during pretraining to mark the conclusion of reasoning (Team, 2024; Guo et al., 2025). Formally, the model's tendency to conclude reasoning at token t_i can be quantified by the probability $p(\texttt{</think>} \mid G_i)$, where G_i denotes the token sequence up to t_i.

This hypothesis leads us to our core partitioning method: a token t_i is considered a candidate reasoning step boundary if and only if the probability of concluding the reasoning chain after t_i is greater than the probability of concluding it after the subsequent token, t_{i+1}: $p(\texttt{</think>}|G_i) > p(\texttt{</think>}|G_{i+1})$. By iterating through the entire reasoning chain, we collect all tokens satisfying this condition as candidate boundaries. To maintain high-quality reasoning steps, we enforce two constraints during partitioning: (1) adjacent boundaries must be at least l tokens apart to avoid overly short steps, and (2) the number of steps must equal the predetermined value, m. In practice, multiple valid partitionings may satisfy these constraints, we randomly sample one to proceed with. An alternative selection strategy and its effect is discussed in Appendix F.1. Figure 1 illustrates this token-distribution-based partitioning method.

3.2.2 Multi-level hints for problem solving

Building on the adaptive stepwise partitioning method described above, we divide the reasoning chain into m reasoning steps. A key question is how many of these steps should be included as a complete hint for the target model.

An ideal-level hint matches the model's current capabilities—it is neither too weak nor too strong. Moreover, this optimal level shifts continuously as the model's reasoning ability evolves. Considering these challenges, rather than determining the ideal level at each training step, we provide hints at multiple levels. Our core hypothesis is that if the partitioning is fine-grained enough, there is likely to be a hint that fits the model's needs well. Specifically, we construct a set of $m-1$ multi-level hints, $\mathcal{H}$. Each hint is a prefix of the full chain G, created by concatenating the first j steps:

$$\mathcal{H} = \{h_j | h_j = S_1 \circ \cdots \circ S_j, \quad \text{for } j = 1, \cdots, m-1\},$$

where S_i represents the i-th reasoning step. Low-level hints preserve considerable problem-solving

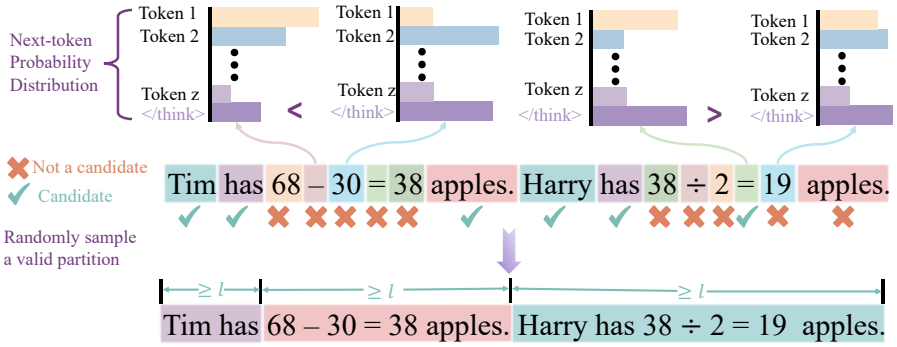

Figure 1: Adaptive stepwise partitioning: step boundaries are identified where the probability of concluding the reasoning chain after the current token, $p(\texttt{</think>}|G_i)$, is greater than concluding after the subsequent token, $p(\texttt{</think>}|G_{i+1})$. A final partition is chosen to meet constraints on step count m and minimum length l.

difficulty. In contrast, high-level hints simplify the solution space, making the completion easier.

For each problem, we construct three types of prompts for the model to complete:

1. **Hinted problems:** For each of the $m-1$ hint levels, the model is asked to complete the reasoning from that hint using k_{hint} rollout attempts per hint.

2. **Unhinted problems:** To preserve the model's independent exploration, it also solves the problem from scratch without any hints. It is allowed k_{unhint} rollouts in this case.

3. **Reference trajectory:** The original ground-truth reasoning chain G is also provided to the target model and used to assign rewards. This ensures that the model is consistently exposed to the correct solution path during training.

Figure 2 illustrates this multi-level hinting and completion process. In total, the model generates $k_{\text{hint}}(m-1)+k_{\text{unhint}}+1$ completions per training problem. StepHint strikes a balance between guiding the model with reliable hints and allowing it to learn from its own exploration mistakes, leading to more effective learning. We further discuss the effectiveness of the multi-level design in Appendix F.11.

The above method applies to most RLVR algorithms but poses issues for GRPO (He et al., 2025), prompting further adaptations and discussion. In GRPO, an incorrect completion assigns negative advantages to all tokens in the rollout—including those in the correct hint prefix. This steers the model away from the correct reasoning chains. To address this, we modify GRPO by clipping negative advantages to zero for tokens in the hint prefix when the completion is incorrect; that is, we set $\hat{A}_{i,t}^{\text{GRPO}} = \max(0, \hat{A}_{i,t}^{\text{GRPO}})$ (see Eq. 2 for the definition of $\hat{A}_{i,t}^{\text{GRPO}}$). This adaptation prevents the model from being penalized for correct prefixes, aligning the training process with our intended mechanism. We empirically validate the effectiveness of this design in Appendix F.5 and prove the convergence of StepHint in Appendix G.

4 Experiments

4.1 Experimental settings

Training data To construct our training data, we generated complete reasoning chains for challenging problems. The specific details of our data construction and processing methods are described in Appendix C.

Hyperparameters We train two backbone models using GRPO (Shao et al., 2024) for 5 epochs each: Qwen-2.5-7B-Instruct (Yang et al., 2024a; Team, 2024) and Qwen-2.5-Math-7B (Yang et al., 2024b). The training prompt template is shown in Appendix B. Other training settings are detailed in Appendix E. The effect of applying an alternative optimization algorithm, Dr.GRPO, is further examined in Appendix F.2.

Baselines We compare our method against five categories of baselines: (1) *Vanilla GRPO:* We train backbone models using vanilla GRPO. (2) *Dr.GRPO:* We train backbone models using Dr.GRPO. (3) *SFT:* We fine-tune the backbone models with our dataset. (4) *Other RLVR Methods:* We evaluate several other RLVR enhancement techniques, including SimpleRL-Zero-7B (Zeng et al., 2025), Qwen-2.5-Math-7B-SimpleRL-Zoo, OpenReasonerZero-7B (Hu et al., 2025), and Oat-

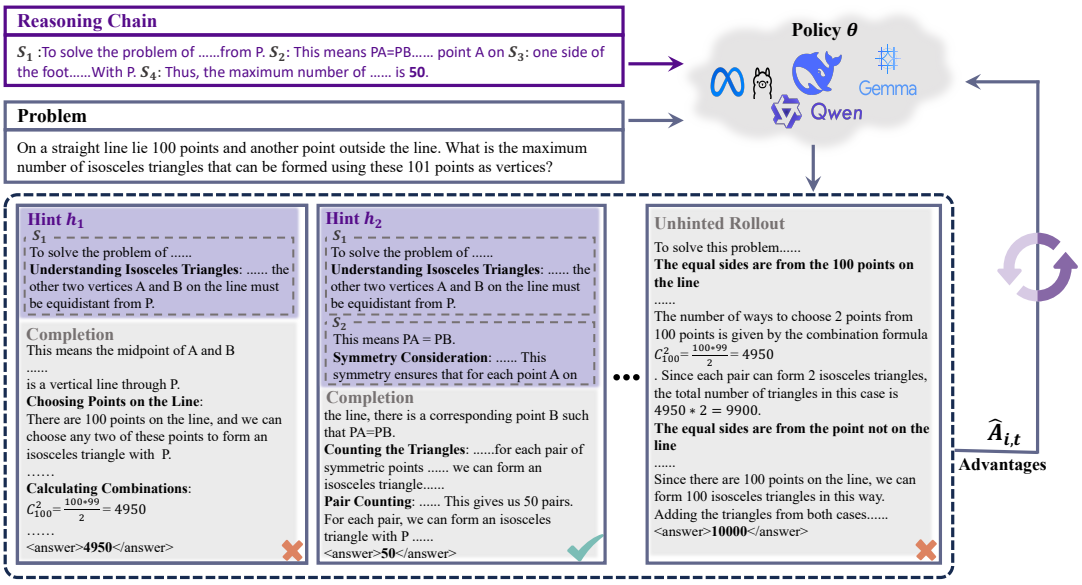

Figure 2: An overview of the multi-level hinting process. The process begins with a ground-truth reasoning chain, which is partitioned into m steps (Section 3.2.1). From these steps, we construct $m-1$ prefix-based hints$(h_1, h_2, \cdots, h_{m-1})$. The model is trained to generate completions from each hint level, as well as from scratch (Unhinted), and a reference trajectory.

7B (Liu et al., 2025). (5) *RL with Reference Trajectory:* We include Luffy-Qwen-2.5-7B-Instruct (Yan et al., 2025) and Luffy-Qwen-Math-7B-Zero. For baselines in categories (4) and (5), we use their publicly released model weights and prompt templates for evaluation.

Evaluation We follow prior work (Yan et al., 2025) and evaluate on six math datasets: AIME 2024, AIME 2025, AMC (Li et al., 2024), Minerva (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), and MATH500 (Hendrycks et al., 2021). For the AIME 24/25 and AMC datasets, given their limited data points, we conduct each evaluation five times and report the weighted average results. To evaluate generalization, we also incorporate two non-math benchmarks, ARC-C (Clark et al., 2018) and GPQA-D (Rein et al., 2024), as out-of-domain tests. The generation length is also set to 4,110. Results were evaluated using Math-Verify[3] and OAT-Grade (Liu et al., 2024).

4.2 Main results

Table 1 shows the overall performance of StepHint and baseline methods. When applied to the general-purpose model, Qwen-2.5-7B-Instruct, StepHint achieves the highest performance on in-domain math tasks among all compared methods. Furthermore, StepHint consistently surpasses the SFT baseline, indicating that StepHint effectively learns beyond simple token imitation, leading to improved reasoning outcomes.

For the specialized math model, the math model performs lower on the out-of-domain benchmarks compared to the general-purpose Qwen-2.5-7B-Instruct. However, StepHint not only leads the board in in-domain math tasks compared with baselines but also enables the Math model to achieve the highest out-of-domain test performance among all baselines. This suggests that the improvements are not solely due to domain-specific knowledge but also reflect an enhancement of the model's general reasoning capabilities.

We now turn our analysis to Luffy, an RL method that employs an *entire* reasoning chain from a stronger model as a reference trajectory. A key limitation of this paradigm is its potential inability to fully address the near-miss reward issue, as the model's exploration is not directly guided. Although using expert trajectories can mitigate exploration stagnation, it may come at the expense of the intrinsic exploration mechanism vital to reinforcement learning. In contrast, StepHint integrates external hints with the model's own exploration,

[3] https://github.com/huggingface/Math-Verify

Model	In-Domain							Out-of-Domain		
	AIME24 (avg@5)	MATH500 (pass@1)	AMC (avg@5)	Olympiad (pass@1)	Minerva (pass@1)	AIME25 (avg@5)	Avg. -	ARC-C (pass@1)	GPQA-D (pass@1)	Avg. -
SFT*	20.00	78.80	53.73	36.89	36.40	10.67	50.05	90.96	23.23	81.17
SFT‡	26.00	82.20	59.52	45.19	34.19	15.33	54.77	67.66	23.74	61.31
On-policy RLVR replication										
Vanilla-GRPO*	24.67	76.60	51.33	43.41	39.34	10.67	52.59	91.30	37.37	83.51
Vanilla-GRPO‡	24.67	78.60	60.72	40.00	36.40	15.33	51.85	79.78	36.87	73.58
Dr.GRPO*	24.00	78.20	51.57	42.81	39.34	12.00	52.87	91.64	35.86	83.58
Dr.GRPO‡	24.67	78.00	57.59	38.22	35.29	15.33	50.55	76.71	33.33	70.44
Other RLVR methods										
ORZ-7B†	24.67	81.00	46.90	45.60	33.46	15.30	53.76	90.53	40.40	83.28
SimpleRL†	22.00	76.00	47.90	39.30	36.40	5.30	49.83	74.74	32.32	68.61
SimpleRL‡	28.00	76.20	57.59	37.93	34.93	12.00	49.80	63.91	27.27	58.61
Oat‡	**36.00**	78.40	59.75	42.52	36.40	10.00	52.92	59.89	33.84	56.13
RL with reference trajectory										
LUFFY*	21.30	77.80	44.82	40.00	36.40	14.67	50.69	81.83	32.32	74.67
LUFFY‡	27.33	83.20	60.24	48.00	38.97	17.33	57.19	81.83	35.86	75.19
StepHint*	29.33	82.80	61.69	47.41	**43.38**	17.30	57.69	**91.89**	**42.42**	**84.74**
StepHint‡	**36.00**	**87.00**	**62.65**	**52.15**	38.24	**18.87**	**60.35**	84.73	38.89	78.10

Table 1: Performance comparison of StepHint with baseline methods on in-domain and out-of-domain benchmarks. The top score in each column is in **bold**, and the second-highest is underlined. Backbone models are denoted by: *Qwen-2.5-7B-Instruct, †Qwen-2.5-7B, ‡Qwen-2.5-Math-7B.

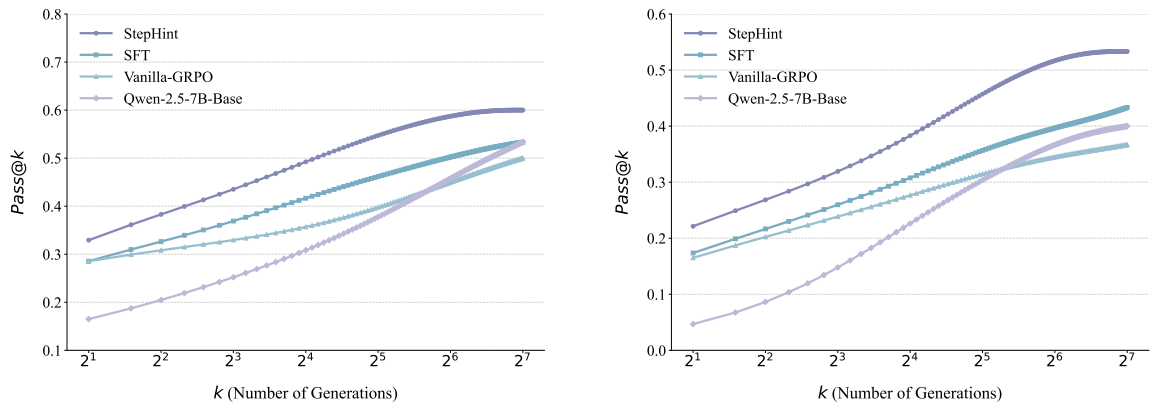

Figure 3: Comparison of pass@k results on the AIME24 and AIME25 datasets. **Left:** AIME24; **Right:** AIME25.

fostering a more robust learning process. Our experimental results substantiate this claim.

To further assess the generalizability of our approach, we conducted experiments on Llama-3.1-8B (Dubey et al., 2024) in Appendix F.3.

4.3 Pass@k evaluation

Many studies (Chen et al., 2021; Wang et al., 2022) show that with a limited number of rollouts, models may perform poorly on certain tasks. However, with a sufficiently large number of rollouts, they are more likely to solve these problems. Therefore, to fully assess the model's potential performance, pass@k accuracy is a more suitable metric (Yue et al., 2025). In this context, a problem is considered solved if any of the k sampled reasoning chains yields a correct answer.

Figure 3 presents the pass@k results on the AIME24 and AIME25 datasets. The results demonstrate that StepHint improves the model's pass@k performance. In contrast, Vanilla-GRPO shows a slower rate of improvement at higher values of k, which aligns with findings from previous work (Yue et al., 2025). The superior performance under pass@k evaluation further validates the effectiveness of StepHint. Additionally, the performance difference can be attributed to the exploration strategies the models employ. While Vanilla-GRPO suffers from "exploration stagnation," StepHint guides the model's exploration,

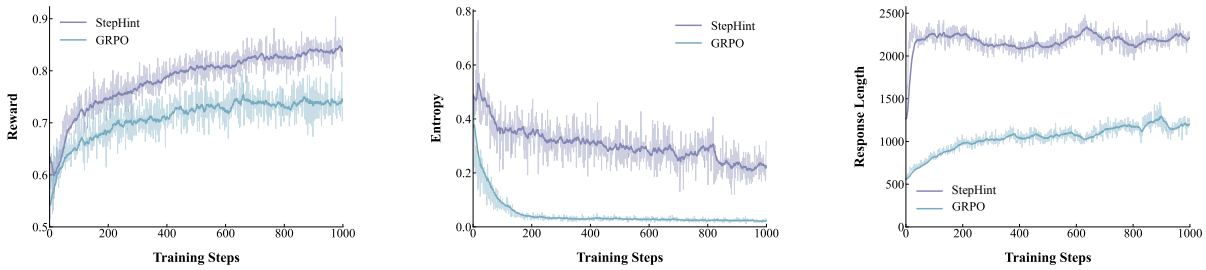

Figure 4: Training dynamics of StepHint compared with GRPO. **Left:** Reward; **Middle:** Entropy; **Right:** Response Length.

helping it break free from its "comfort zone."

4.4 Method analysis from training dynamics

Figure 4 illustrates the training dynamics of StepHint and Vanilla-GRPO on Qwen-2.5-7B-Instruct, comparing them across three key metrics: entropy, response length, and training rewards.

Reward Due to the multi-level stepwise hints provided by StepHint, the problem-solving difficulty for the model is lower compared to Vanilla-GRPO. As a result, the reward score for StepHint is consistently higher, reflecting the mitigation of the near-miss reward issue. A closer examination of the trends in both curves offers further insights. Vanilla-GRPO shows a steady, monotonic increase in reward as it refines its existing policy. In contrast, StepHint experiences a brief initial dip in reward before a rapid and sustained increase. This initial dip likely reflects an adaptation period where the model transitions from simple exploitation to a more complex, hint-guided exploration. Once adapted, the model efficiently discovers higher-reward solutions, leading to faster and effective learning to reason.

Entropy Both methods exhibit an overall decrease in entropy, though their trajectories diverge as training progresses. The entropy of StepHint remains higher than that of Vanilla-GRPO. This suggests that the hints provided by StepHint encourage a more diverse policy, preventing premature convergence to a narrow solution subspace and promoting a higher level of exploration (Cheng et al., 2025). This trend reflects, to some extent, the mitigation of exploration stagnation.

Response length StepHint shows an initial sharp increase in length, which we attribute to the model learning to mimic the structured, stepwise reasoning chains provided by the multi-level hints. These hints are often more detailed than the model's initial, more direct responses.

4.5 Ablation on Partitioning Algorithms

To rigorously evaluate the efficacy of our proposed partitioning strategy, we conducted ablation studies comparing our method, StepHint-Base (denoted as *StepHint-*`</think>` in the results), against two alternative segmentation baselines. The first baseline, **StepHint-Random**, adopts a naive length-based approach that splits reasoning chains into four segments based on token count with uniform distribution and random jitter, thereby disregarding semantic boundaries. The second baseline, **StepHint-`\n`**, employs a syntactic heuristic that utilizes the probability of the newline token ('\n') as a proxy for identifying reasoning steps, effectively simulating sentence or paragraph-level splitting rather than relying on the specific `</think>` token.

The comparative results across four benchmarks are summarized in Table 2. These empirical findings underscore the superiority of our semantic-aware approach. Specifically, **StepHint-Base** significantly outperforms the length-based baseline, confirming that maintaining **Semantic Integrity** is paramount for effective reasoning. While random splits frequently sever logical deductions mid-thought—creating "broken hints" that disrupt coherence and confuse the policy model—our method preserves the logical flow. Furthermore, our approach surpasses the syntactic heuristic (*StepHint-*`\n`). This advantage stems from the fact that reasoning steps do not necessarily align with structural formatting; complex mathematical derivations often span multiple lines, rendering newline characters an unreliable proxy for logical boundaries. In contrast, the probability of the `</think>` token effectively captures the **model's internal confidence** regarding the completion of a thought unit, providing a far more robust signal for exploration than external formatting cues.

Method	AIME24	MATH500	AMC	Olympiad	Minerva	AIME25	Avg.
StepHint-Random	32	85.8	59.04	48.3	34.92	15.3	57.43
StepHint-\n	34	86.40	60.72	50.07	37.13	16.67	58.90
StepHint-</think>	**36.00**	**87.00**	**62.65**	**52.15**	**38.24**	**18.87**	**60.35**

Table 2: Ablation study of different partitioning strategies. **StepHint-Base** denotes our proposed method.

5 Related works

RL-based post-training has demonstrated remarkable success in mathematical reasoning tasks (Shao et al., 2024; Yang et al., 2024b). Research in this area has primarily advanced in three directions: (1) optimizing the models and their training data, (2) refining inference-time strategies, and (3) improving policy optimization methods.

The first direction involves constructing high-quality, large-scale mathematical reasoning datasets (Wang et al., 2023; Ye et al., 2025; Zhao et al., 2025) and designing specialized training or fine-tuning methods (Jaech et al., 2024; Mitra et al., 2024; Zhang et al., 2025a). The second direction focuses on guiding the model's step-by-step thought processes without altering its underlying weights, typically through sophisticated prompting techniques such as Chain-of-Thought (Wei et al., 2022) and innovations in in-context learning (Wu et al., 2024). The third direction aims at developing advanced policy optimization algorithms. GRPO has recently gained popularity due to its simplicity and strong performance (Hu et al., 2025; Zeng et al., 2025). Additionally, several improvements have been proposed for GRPO; for example, Liu et al. (2025) identified inherent length and difficulty biases in vanilla GRPO and addressed these issues. Zhang et al. (2025b) utilizes expert traces but is limited by computationally expensive iterative search and rigid heuristic partitioning, which disrupt semantic flow and training efficiency.

6 Conclusion

In this paper, we introduced StepHint, a novel RLVR algorithm that incorporates multi-level stepwise hints. This mechanism is designed to provide the model with assistance tailored to its evolving capabilities, thereby facilitating the learning process by addressing challenges such as near-miss rewards and exploration stagnation. StepHint not only outperforms strong baselines on mathematical benchmarks but also demonstrates robust generalization to out-of-domain tasks, highlighting the promising potential of the stepwise hinting paradigm for RLVR enhancement.

Limitations

StepHint currently relies on reasoning chains extracted from stronger teacher models to serve as hint sources. While this ensures high-quality guidance, it also means our method's potential is capped by the teacher model's capability. Future work will focus on removing this bottleneck by developing self-supervised mechanisms, allowing the model to generate its own high-confidence hints without external dependency.

Acknowledgements

This work was supported by fund for building world-class universities (disciplines) of Renmin University of China and Ant Group Research Intern Program.

Ethical considerations

Regarding potential risks, although mathematical reasoning is a domain with low inherent social bias, our results indicate that StepHint enhances general reasoning capabilities beyond the training domain; while currently beneficial, improved general reasoning carries theoretical dual-use risks if applied to malicious automated planning, though such capabilities are outside the scope of this work.

In this work, we constructed a derived dataset for training by aggregating mathematical problems from the existing DAPO and DeepMath datasets and generating corresponding multi-step reasoning chains using advanced models. We strictly adhere to the licenses of these source artifacts and cite their original creators. The content of this constructed dataset is strictly limited to mathematical and logical reasoning tasks; therefore, it contains no personally identifiable information or offensive content, and no specific anonymization measures were required. We also utilize standard open-source models and benchmarks for evaluation in accordance

with their intended research use. Documentation on the data construction pipeline is provided in Appendix C. Detailed statistics regarding the training setup, hyperparameters (e.g., learning rate, batch size), and compute resources (8 $\times$ A100 GPUs) are reported in Section 4.1 and Appendix E. Breakdown of evaluation metrics and split details for the benchmarks follow standard community practices as cited in the main text.

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, and 1 others. 2025. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*.

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

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *arXiv preprint arXiv:2503.24290*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and 1 others. 2024. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9.

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

Zichen Liu, Changyu Chen, Xinyi Wan, Chao Du, Wee Sun Lee, and Min Lin. 2024. Oat: A research-friendly framework for llm online alignment. `https://github.com/sail-sg/oat`.

Ang Lv, Yuhan Chen, Kaiyi Zhang, Yulong Wang, Lifeng Liu, Ji-Rong Wen, Jian Xie, and Rui Yan. 2024. Interpreting key mechanisms of factual recall in transformer-based language models. *Preprint*, arXiv:2403.19521.

Ang Lv, Jinpeng Li, Yuhan Chen, Gao Xing, Ji Zhang, and Rui Yan. 2023a. DialoGPS: Dialogue path sampling in continuous semantic space for data augmentation in multi-turn conversations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1267–1280, Toronto, Canada. Association for Computational Linguistics.

Ang Lv, Jinpeng Li, Shufang Xie, and Rui Yan. 2023b. Envisioning future from the past: Hierarchical duality learning for multi-turn dialogue generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7382–7394, Toronto, Canada. Association for Computational Linguistics.

Ang Lv, Jin Ma, Yiyuan Ma, and Siyuan Qiao. 2026. Coupling experts and routers in mixture-of-experts via an auxiliary loss. In *The Fourteenth International Conference on Learning Representations*.

Ang Lv, Ruobing Xie, Yining Qian, Songhao Wu, Xingwu Sun, Zhanhui Kang, Di Wang, and Rui Yan. 2025a. Autonomy-of-experts models. In *Forty-second International Conference on Machine Learning*.

Ang Lv, Ruobing Xie, Xingwu Sun, Zhanhui Kang, and Rui Yan. 2025b. The climb carves wisdom deeper than the summit: On the noisy rewards in learning to reason. *Preprint*, arXiv:2505.22653.

David JC MacKay. 2003. *Information theory, inference and learning algorithms*. Cambridge university press.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. 2024. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*.

Marie-Francine Moens. 2017. Argumentation mining: How can a machine acquire common sense and world knowledge? *Argument & Computation*, 9:1 – 14.

Yury Polyanskiy and Yihong Wu. 2025. *Information theory: From coding to learning*. Cambridge university press.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zengzhi Wang, Rui Xia, and Pengfei Liu. 2023. Generative ai for math: Part i–mathpile: A billion-token-scale pretraining corpus for math. *arXiv preprint arXiv:2312.17120*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jinyang Wu, Mingkuan Feng, Shuai Zhang, Feihu Che, Zengqi Wen, Chonghua Liao, and Jianhua Tao. 2024. Beyond examples: High-level automated reasoning paradigm in in-context learning via mcts. *arXiv preprint arXiv:2411.18478*.

Songhao Wu, Ang Lv, Ruobing Xie, Xingwu Sun, Di Wang, Rui Yan, and Yankai Lin. 2025. Union-of-experts: Experts in mixture-of-experts are secretly routers.

Shufang Xie, Ang Lv, Yingce Xia, Lijun Wu, Tao Qin, Tie-Yan Liu, and Rui Yan. 2022. Target-side input augmentation for sequence to sequence generation. In *International Conference on Learning Representations*.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. 2025. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

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

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*.

Kaiyi Zhang, Ang Lv, Yuhan Chen, Hansen Ha, Tao Xu, and Rui Yan. 2024. Batch-ICL: Effective, efficient, and order-agnostic in-context learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10728–10739, Bangkok, Thailand. Association for Computational Linguistics.

Xiaoqing Zhang, Huabin Zheng, Ang Lv, Yuhan Liu, Zirui Song, Xiuying Chen, Rui Yan, and Flood Sung. 2025a. Divide-fuse-conquer: Eliciting "aha moments" in multi-scenario games. *Preprint*, arXiv:2505.16401.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. 2025b. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. *arXiv preprint arXiv:2506.17211*.

Xueliang Zhao, Wei Wu, Jian Guan, and Lingpeng Kong. 2025. Promptcot: Synthesizing olympiad-level problems for mathematical reasoning in large language models. *arXiv preprint arXiv:2503.02324*.

A Proof of proposition 1

Proposition 1:

$$\mathbb{E}_{t_k \sim p(\cdot|S_{k-1})}[H(\mathcal{R}|S_k)] \leq H(\mathcal{R}|S_{k-1})$$

Proof. We want to prove the following inequality:

$$\mathbb{E}_{t_k \sim p(\cdot|S_{k-1})}[H(\mathcal{R}|S_k)] \leq H(\mathcal{R}|S_{k-1})$$

This inequality states that the expected entropy of solution space $\mathcal{R}$ conditioned on the state S_k is less than or equal to the entropy of $\mathcal{R}$ conditioned on the prior state S_{k-1}. The state S_k is reached from S_{k-1} after an observation or transition t_k. Let's denote the random variable for this transition as T_k.

First, let's clarify the notation. The expression on the left-hand side, $\mathbb{E}_{t_k \sim p(\cdot|S_{k-1})}[H(\mathcal{R}|S_k)]$, represents the conditional entropy $H(\mathcal{R}|S_k)$. The conditional entropy $H(X|Y)$ is defined as the expectation of the entropy of X over the values of Y. The subscript simply makes the underlying probability model explicit: the distribution of the state S_k is induced by the distribution of the prior state S_{k-1} and the transition T_k.

The proof of this relationship relies on the non-negativity of conditional mutual information. The conditional mutual information between two random variables, $\mathcal{R}$ and T_k, given a third variable S_{k-1}, is defined as:

$$I(\mathcal{R}; T_k|S_{k-1}) = H(\mathcal{R}|S_{k-1}) - H(\mathcal{R}|S_{k-1}, T_k)$$

A fundamental property of mutual information is that it is always non-negative (MacKay, 2003; Polyanskiy and Wu, 2025):

$$I(\mathcal{R}; T_k|S_{k-1}) \geq 0$$

From this, it directly follows that:

$$H(\mathcal{R}|S_{k-1}) \geq H(\mathcal{R}|S_{k-1}, T_k)$$

This equation shows that conditioning on an additional variable, T_k, can only decrease (or leave unchanged) the entropy of $\mathcal{R}$.

Now, we must relate the term $H(\mathcal{R}|S_{k-1}, T_k)$ to the term $H(\mathcal{R}|S_k)$. The state S_k is the result of a process that starts in state S_{k-1} and undergoes the transition T_k. This means that the information contained in the pair of variables (S_{k-1}, T_k) fully determines the state S_k. In many typical models, knowing S_k is equivalent to knowing the pair (S_{k-1}, T_k) that produced it. If we assume this equivalence, then conditioning on S_k is the same as conditioning on the pair (S_{k-1}, T_k). Therefore, we have:

$$H(\mathcal{R}|S_k) = H(\mathcal{R}|S_{k-1}, T_k)$$

Substituting this equality back into our previous inequality, we arrive at the desired result:

$$H(\mathcal{R}|S_k) \leq H(\mathcal{R}|S_{k-1})$$

This completes the proof. □

B Template of Qwen-2.5

Template of Qwen-2.5

<|im_start|>system
You are a helpful assistant. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think><answer> answer here </answer>. Now the user asks you to solve a mathematical reasoning problem. After thinking, when you finally reach a solution, clearly state the answer marked with \boxed{} and within <answer> </answer> tags, i.e., <answer>\boxed{answer}</answer>
<|im_end|>
<|im_start|> user
{question}
<|im_end|>
<|im_start|>assistant
<think>

C Training data construction

We gathered all problems from the DAPO dataset (Yu et al., 2025) and selected problems of difficulty level 7 or higher from the DEEPMATH dataset (He et al., 2025). The DEEPMATH dataset provides solution reasoning chains, while for each question in the DAPO dataset (Yu et al., 2025), we sample a total of 12 reasoning chains using DAPO-Qwen-32B (Yu et al., 2025), QWQ-32B (Team, 2025), and DeepSeek-R1-Distill-Qwen-32B (Guo et al., 2025), with 4 samples per model. The sampling is conducted under a 0-shot setting, with a temperature of 1 and a maximum length of 8,192. We filter these to retain all reasoning chains that are both correct and have a length of no more than 4,110. In cases where multiple chains satisfy these conditions, we randomly select one.

All reasoning chains are partitioned into $m = 4$ steps with the method we proposed in Section 3.2.1, each longer than $l = \frac{L}{8}$ tokens, where L is the total length of a reasoning chain G, with QWQ-32B (Team, 2025). This threshold ensures that even the shortest step contains a substantive amount of information, while allowing for the natural length variation between different steps in a complex reasoning process.

D Training settings for Llama

Template of Llama-3.1

system
You are a helpful assistant. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think><answer> answer here </answer>. Now the user asks you to solve a mathematical reasoning problem. After thinking, when you finally reach a solution, clearly state the answer marked with \boxed{} and within <answer> </answer> tags, i.e., <answer>\boxed{answer}</answer>
user
{question}
assistant
<think>

Due to temporal and computational constraints, we train the Llama-3.1-8B model for 4 epochs on our dataset using two methods: StepHint and GRPO. We increased the maximum generation length to 4,500 during training to ensure it could accommodate the longest external reasoning chains within the dataset. To align with the original Llama pre-training paradigm, we introduced slight modifications to the prompt template. All other hyperparameters remained consistent with those of the Qwen-2.5 models.

E Training setup

The training is based on the VeRL framework (Sheng et al., 2024). We set a global batch size of 128 and a fixed learning rate of $1e-6$. Following (Yan et al., 2025), we set the KL loss coefficient $\beta = 0$, indicating no reference model is used for regularization. We configure $k_{\text{hint}} = 2$ and $k_{\text{unhint}} = 5$. During training, the temperature for rollout generation is set to 1.0. Our training is completed on 8×A100s. We adopt a community-released variant of Qwen-2.5-Math-7B model (Yang et al., 2024b) that extends the context

length to 32k tokens.[4]

F Supplementary experiments

F.1 Stepwise partitioning strategy

In this section, we conduct an ablation study on the stepwise partitioning strategy introduced in Section 3.2.1. We compare two approaches: *Base*, which selects k tokens uniformly at random from candidate tokens that satisfy the condition $(p(\texttt{</think>}|G_i) > p(\texttt{</think>}|G_{i+1}))$ and the interval constraints; and *Salient*, which selects the top-k candidate tokens exhibiting the largest probability drop $(p(\texttt{</think>}|G_i) - p(\texttt{</think>}|G_{i+1}))$ while satisfying the interval constraints.

As shown in Table 3, both strategies achieve comparable overall performance.

F.2 Optimization algorithm

In this section, we evaluate the effect of different optimization algorithms on our proposed StepHint method. Specifically, we employ Dr.GRPO as the optimization algorithm and compare its performance against Vanilla-GRPO. The results are summarized in Table 4.

F.3 Results on Llama

We trained both StepHint and the baseline GRPO on Llama-3.1-8B, and the results are presented in Table 5. The results demonstrate that StepHint successfully trains Llama-3.1-8B. Training settings are detailed in Appendix D.

F.4 Computational overhead and cost trade-off

It is worth noting that StepHint does not introduce significant computational overhead during either the preprocessing or training phases. The generation of multiple hint levels is a one-time offline procedure requiring only a single forward pass over the collected reasoning chains. Consequently, this preprocessing cost is negligible compared to the overall reinforcement learning pipeline. Furthermore, during the training phase, the computational budget is strictly controlled. As outlined in Section 3.2.2, if standard GRPO utilizes a total of n rollouts per problem, StepHint maintains the exact same budget of n rollouts. The only distinction is that StepHint flexibly distributes these n rollouts between hint-guided completions and standard unhinted rollouts, ensuring that the enhanced learning process does not incur additional rollout costs.

F.5 Non-negative advantage for hints

We perform an ablation study with the Qwen2.5-7B-Instruct model to assess the effect of enforcing a non-negative constraint on the advantages of hints. As shown in Table 6, *w/ Constraint* denotes the configuration where the advantage values of hints are constrained to be non-negative, whereas *baseline* corresponds to the baseline without this constraint. The results demonstrate that introducing the non-negative constraint improves the model's mathematical reasoning performance.

F.6 Ablation on Hint Levels and Exploration Trade-off

The parameter m is crucial in controlling the balance between guided hint exploration and free exploration. Specifically, a larger m introduces more hint levels, which inherently requires $2(m-1)$ guided hint completions. Given a fixed total rollout budget n per problem, the number of rollouts remaining for free (unhinted) exploration is strictly constrained to $n - 2m + 1$.

To empirically validate our design choice, we conduct an ablation study over different values of $m \in \{3, 4, 6\}$ using Qwen2.5-7B-Instruct. To facilitate efficient validation, these ablation experiments are trained for only 3 epochs, rather than the 5 epochs utilized in our main experiments. In this setup, the total rollout budget remains fixed at $n = 12$. The rollout distribution and corresponding performance across multiple benchmarks are detailed in Table 7.

As demonstrated, $m = 4$ strikes an optimal balance between hint-guided learning and free exploration, achieving the best overall performance. Conversely, when $m = 6$, the increased number of hint completions leaves only a single rollout for free exploration. This severely restricts the model's exploratory capacity, leading to a significant degradation in performance.

Furthermore, regarding the minimum hint length constraint, setting the threshold to $L/8$ acts as a structural safeguard. It ensures that step boundaries are not placed too closely together, thereby preserving the semantic diversity and utility of the generated hints.

[4] https://huggingface.co/open-r1/Qwen2.5-Math-7B-RoPE-300k

Strategy	AIME24	MATH500	AMC	Olympiad	Minerva	AIME25	Avg.
Qwen2.5-7B-Instruct-StepHint							
Base	29.33	82.80	61.69	47.41	43.38	17.30	57.69
Salient	29.33	83.80	59.28	47.41	43.01	18.67	57.84
Qwen2.5-Math-7B-StepHint							
Base	36.00	87.00	62.65	52.15	38.24	18.87	60.35
Salient	36.00	86.60	63.86	53.33	38.24	18.67	60.78

Table 3: Performance comparison of different stepwise partitioning strategies.

	AIME24	MATH500	AMC	Olympiad	Minerva	AIME25	Avg.
Qwen2.5-7B-Instruct-StepHint							
GRPO	29.33	82.80	61.69	47.41	43.38	17.30	57.69
Dr.GRPO	32.00	82.40	61.00	49.04	42.65	18.0	58.15
Qwen2.5-Math-7B-StepHint							
GRPO	36.00	87.00	62.65	52.15	38.24	18.87	60.35
Dr.GRPO	40.67	88.60	65.30	51.70	40.44	22.67	61.33

Table 4: Performance comparison of StepHint under different optimization algorithms.

F.7 Impact of Error Position on Solution Trajectory

To empirically validate that early errors disproportionately prune the correct solution subspace (Xie et al., 2022; Lv et al., 2023a,b), we quantify how the position of a logical error affects the final mathematical reasoning accuracy.

We designed an error-injection experiment using a randomly selected subset of 10 problems from the MATH500 dataset. For each problem, we manually constructed a standardized 4-step reasoning chain. We then artificially introduced a logical error at a specific reasoning step $K \in \{1, 2, 3\}$ and prompted the StepHint-Instruct-7B model to generate the remainder of the solution. For each injection point K, we sampled 10 completions per problem, yielding a total of 100 completions per condition.

The results, detailed in Table 8, reveal a clear monotonic trend: the earlier an error occurs in the reasoning chain, the more severely the final accuracy degrades.

This provides direct empirical evidence that early errors irreversibly derail the reasoning trajectory, effectively trapping the model in a flawed solution subspace from which recovery is highly improbable. Studying the mechanisms underlying the existence of such a subspace (Lv et al., 2024) or developing lightweight methods to steer away from it (Zhang et al., 2024) represents a valuable direction for future research.

F.8 Robustness to Teacher Model Quality

A notable advantage of the StepHint framework is its robustness regarding the choice of the teacher model. The algorithm does not strictly necessitate highly advanced or computationally expensive models. Instead, the core prerequisite is simply access to a logically correct reasoning chain prefix, which functions as a valid anchor to safely guide the student model's exploration.

Consequently, smaller and more accessible models can effectively serve as substitute teachers. While a weaker model may require a higher sampling budget to successfully generate a correct reasoning chain, this trade-off is practically viable. Because hint generation is a one-time, offline preprocessing procedure, scaling the sampling attempts for a smaller model remains a highly feasible and cost-effective alternative. In our primary data construction, stronger teacher models were employed primarily to maximize preprocessing efficiency and initial yield, rather than out of a structural dependency on their unique reasoning capabilities.

To empirically validate this flexibility, we evaluate a "Self-Teaching" configuration, where the stu-

	AIME24	MATH500	AMC	Olympiad	Minerva	AIME25	Avg.
GRPO	4.00	9.00	6.02	3.56	12.13	0	6.81
StepHint	6.00	44.00	18.56	13.48	20.22	1.30	24.13

Table 5: Results on Llama-3.1-8B.

	AIME24	MATH500	AMC	Olympiad	Minerva	AIME25	Avg.
Qwen2.5-7B-Instruct-StepHint							
baseline	28.00	81.40	57.30	45.30	39.70	14.70	55.43
w/ Constraint	29.33	82.80	61.69	47.41	43.38	17.30	57.69

Table 6: Ablation results on the effect of applying non-negative advantage constraint for hints.

m	AIME24	Math500	AIME25	AMC
3	27.3	79.6	13.3	53.3
4	**28.0**	**80.8**	**16.0**	**57.6**
6	16.7	34.0	10.0	28.2

Table 7: Ablation study on the number of hint levels (m) with a fixed rollout budget ($n = 12$).

Injection Point (K)	$K = 1$	$K = 2$	$K = 3$
Accuracy (%)	13.0	17.0	**44.0**

Table 8: Impact of the error injection point (K) on final accuracy across a 4-step reasoning chain.

dent model itself (Qwen2.5-7B-Instruct) is utilized as the teacher to generate the requisite reasoning chains. To facilitate an efficient and fair comparison, both the standard-teacher and self-teacher configurations are trained on a representative subset of the dataset and evaluated at an equivalent early training stage. The comparative results are detailed in Table 9.

As demonstrated in Table 9, the performance of the Self-Teaching configuration closely matches that of the Standard Teacher setup across all evaluated benchmarks. This confirms that StepHint is highly flexible and can be effectively deployed even when advanced teacher models are unavailable, provided a sufficient sampling budget is allocated for correct chain generation.

F.9 Generalization to General-Domain Tasks

To assess the generalization capability of our method beyond mathematical and reasoning-intensive domains, we extend our evaluation to general knowledge tasks using the MMLU and MMLU-Pro benchmarks. For this analysis, we evaluate the models on a randomly sampled subset of 2,000 problems from each dataset.

The comparative results are presented in Table 10. StepHint demonstrates substantial improvements over standard GRPO, achieving an accuracy of 78.4% on MMLU and 57.6% on MMLU-Pro. These significant gains indicate that the structural reasoning enhancements fostered by StepHint are not strictly localized to math or logic problems, but successfully transfer to broader, general-domain cognitive tasks.

Here, we note another limitation regarding generalizability: MoE models are known to be unstable during RLVR post-training. Moving forward, we will focus our efforts on testing the stability of StepHint on MoE models. Additionally, a related research direction involves exploring stable MoE structures (Lv et al., 2025a, 2026; Wu et al., 2025) that consistently make capability-aligned expert selection decisions.

F.10 Categorizing Problem Difficulty via Reasoning Step Density

Traditional metrics, such as prompt length, often fail to accurately reflect the intrinsic difficulty of mathematical reasoning problems. To provide a more informative and granular analysis, we leverage our adaptive segmentation method to categorize problems based on the complexity of their required solution paths.

Specifically, for each problem in the AMC and MATH500 datasets, we sample five distinct reasoning chains using DeepSeek-R1-Distill-Qwen-32B. We then apply our segmentation technique to calculate the number of discrete reasoning steps within each chain. To normalize against varying response

Teacher Configuration	AIME24	Math500	AIME25	AMC
Self-Teaching	20.0	77.0	9.3	46.8
Standard	**21.3**	**77.4**	**10.0**	**47.5**

Table 9: Performance comparison between Standard Teacher and Self-Teaching configurations. Both models are evaluated at an equivalent early training stage on a data subset.

Method	MMLU	MMLU-Pro
GRPO	74.1	43.7
StepHint	**78.4**	**57.6**

Table 10: Performance comparison on general-domain tasks.

lengths, we compute a step density ratio, defined as the number of reasoning steps divided by the total chain length. Problems are subsequently partitioned based on the median of this ratio: the bottom 50%, which exhibit a lower density of logical transitions, are classified as "Easy," while the top 50%, characterized by a higher density of logical transitions, are classified as "Hard."

We evaluate our StepHint-Instruct-7B model across these categorized splits. The comparative performance is detailed in Table 11.

As demonstrated in the results, the model exhibits a clear performance degradation on the "Hard" subsets across both benchmarks. For instance, accuracy drops from 63.33% to 58.50% on the AMC dataset, and from 86.00% to 81.20% on the MATH500 dataset as the density of logical transitions increases. These empirical findings confirm that problems necessitating denser logical transitions are inherently more challenging. Furthermore, it validates that our step segmentation method can serve as an effective and robust proxy for quantifying intrinsic problem complexity.

F.11 Impact of Multi-Level Hint Design

To validate the necessity of our multi-level strategy, we introduced a baseline variant termed **StepHint-Single**. Unlike our proposed method, which utilizes hierarchical split points, the *StepHint-Single* baseline randomly samples only one split point from the candidate set for each problem. To ensure a fair comparison, both models were trained with an identical total number of rollouts. As shown in Table 12, the multi-level design consistently outperforms the single-level baseline across all benchmarks. Most notably, on AIME24, our method achieves a score of 36.0, surpassing the baseline by a substantial margin of 6.7 points.

We attribute this performance gain to an implicit **curriculum learning** effect facilitated by the multi-level design. By training on a diverse mixture of hint granularities—encompassing short, medium, and long contexts simultaneously—the model learns to navigate varying degrees of reasoning uncertainty. Conversely, training on a single random hint level introduces high variance in task difficulty within batches, leading to optimization instability. Although the multi-level strategy incurs a marginal computational overhead during the data generation phase, the significant improvement in performance justifies this cost, particularly for high-stakes reasoning tasks where accuracy is the primary objective.

G Proof of convergence

We first recall two standard assumptions from stochastic non-convex optimization.

Assumption 1 (Smoothness)**.** *The objective function $J(\theta)$ is continuously differentiable, and its gradient is L-Lipschitz:*

$$\|\nabla J(\theta_1) - \nabla J(\theta_2)\| \le L\|\theta_1 - \theta_2\|, \quad \forall \theta_1, \theta_2.$$

Consequently, for any update θ_{t+1}:

$$J(\theta_{t+1}) \ge J(\theta_t) + \langle \nabla J(\theta_t), \theta_{t+1} - \theta_t \rangle - \tfrac{L}{2}\|\theta_{t+1} - \theta_t\|^2.$$

Assumption 2 (Bounded Variance)**.** *The variance of the stochastic gradient estimator $\hat{g}_t$ is bounded:*

$$\mathbb{E}\big[\|\hat{g}_t - \mathbb{E}[\hat{g}_t|\theta_t]\|^2 \,\big|\, \theta_t\big] \le \sigma^2.$$

Proposition 2. *Let $\hat{g}_t$ denote the token-averaged PPO/GRPO stochastic gradient estimator used by StepHint at iteration t. Let $p_{hint} \in [0, 1]$ be the expected fraction of tokens in a rollout that are* hint *tokens, and let $\alpha \in [0, 1]$ be the probability that a rollout produced under the current policy yields a correct final outcome. Assume the following modeling approximations at iteration t:*

Model	AMC-Easy	AMC-Hard	Math500-Easy	Math500-Hard
StepHint-Instruct-7B	63.33	58.50	86.00	81.20

Table 11: Performance comparison of StepHint-Instruct-7B on problem splits categorized by reasoning step density.

Method	**AIME24**	**MATH500**	**AMC**	**Olympiad**	**Minerva**	**AIME25**	**Avg.**
StepHint-Single	29.3	84.8	58.55	48.8	34.56	13.3	57.16
StepHint-Base	**36.00**	**87.00**	**62.65**	**52.15**	**38.24**	**18.87**	**60.35**

Table 12: Ablation study on Multi-Level design. **StepHint-Base** denotes our proposed method.

(A1) ***Token homogeneity:*** *the expected per-token policy-gradient contribution within a rollout is approximately the same across token positions, i.e.*

$$\mathbb{E}\big[\nabla_\theta \log \pi_\theta(\tau_{i,j})\hat{A}_{i,j}\big] \approx c_t \nabla J(\theta_t)$$

for some scalar $c_t > 0$, so that token-level expectations align with the full policy gradient direction.

(A2) ***Clipping/ratio approximation for hint tokens:*** *for hint tokens in rollouts that produce a correct outcome the PPO probability ratio $r_{i,j} = \pi_\theta(\tau_{i,j})/\pi_{old}(\tau_{i,j})$ is typically above the upper clip $1+\epsilon$, hence the clipped surrogate multiplies the advantage by approximately $1+\epsilon$; for non-hint tokens the ratio is approximated as $r_{i,j} \approx 1$ so clipping is inactive.*

Under (A1)–(A2) define

$$\beta_t := (1 - p_{hint}) + \alpha\, p_{hint}(1+\epsilon).$$

Then the conditional expectation of the StepHint gradient estimator can be approximated by

$$\mathbb{E}[\hat{g}_t \mid \theta_t] \approx \beta_t \nabla J(\theta_t).$$

Proof. Write $\hat{g}_t$ for the token-averaged stochastic gradient estimator at iteration t. Let a rollout contain multiple tokens and let g_j denote the gradient contribution (surrogate objective / policy-gradient contribution) associated with a generic token position j in a rollout. We drop the rollout index for notational simplicity and condition all expectations on θ_t.

The token-averaged estimator can be written as an expectation over token positions:

$$\begin{aligned}\mathbb{E}[\hat{g}_t \mid \theta_t] =& \mathbb{E}[g_j \mid \theta_t] \\ =& (1 - p_{hint})\, \mathbb{E}[g_j \mid j \text{ is not a hint}, \theta_t] \\ &+ p_{hint}\, \mathbb{E}[g_j \mid j \text{ is a hint}, \theta_t],\end{aligned} \tag{3}$$

since a token is either a hint token (fraction p_{hint}) or a non-hint token (fraction $1 - p_{hint}$).

We treat the two terms separately.

Non-hint tokens. For non-hint tokens the surrogate is the standard PPO/GRPO surrogate built from the student policy's own probabilities and advantages. Under Assumption (A1) we approximate the per-token expected contribution by a common direction proportional to the full policy gradient:

$$\mathbb{E}[g_j \mid \text{non-hint}, \theta_t] \approx \mathbb{E}[\nabla_\theta \log \pi_\theta(\tau)\, \hat{A}],$$

Hint tokens. For tokens that are provided as hints we further condition on whether the *entire* rollout ends in a correct final outcome. Let α denote the probability (under the current policy and the environment) that a rollout is correct. Then, conditioning on hint token and rollout correctness,

$$\begin{aligned}\mathbb{E}[g_j \mid \text{hint}, \theta_t] =& (1 - \alpha)\, \mathbb{E}[g_j \mid \text{hint}, \text{incorrect}, \theta_t] \\ &+ \alpha\, \mathbb{E}[g_j \mid \text{hint}, \text{correct}, \theta_t].\end{aligned} \tag{4}$$

By the StepHint design (and the modification to GRPO described in the paper), when a rollout is incorrect the negative advantages assigned to hint tokens are clipped to zero (i.e. the algorithm prevents penalizing the model for hint tokens in incorrect rollouts). Therefore the contribution from hint tokens in incorrect rollouts is (approximately) zero:

$$\mathbb{E}[g_j \mid \text{hint}, \text{rollout incorrect}, \theta_t] \approx 0.$$

For hint tokens in *correct* rollouts, Assumption (A2) posits that the PPO probability ratio $r_j = \frac{\pi_\theta(\tau_j)}{\pi_{old}(\tau_j)}$ is typically above the upper clipping threshold $1+\epsilon$, so that the clipped surrogate evaluates approximately to $(1+\epsilon)\hat{A}$. Together with the token-homogeneity approximation (A1) that aligns per-token expectation with G_t, we obtain

$$\mathbb{E}[g_j \mid \text{hint}, \text{rollout correct}, \theta_t] \approx (1+\epsilon)\, G_t.$$

Combining the two subcases for hint tokens gives

$$\begin{aligned}\mathbb{E}[g_j \mid \text{hint}, \theta_t] &\approx (1-\alpha)\cdot 0 \;+\; \alpha\cdot(1+\epsilon)\, G_t \\ &= \alpha(1+\epsilon)\, G_t.\end{aligned} \tag{5}$$

Combine hint and non-hint contributions. Substitute the approximations for the two conditional expectations back into the decomposition at the top:

$$\begin{aligned}\mathbb{E}[\hat{g}_t \mid \theta_t] &\approx (1-p_{hint})\, G_t \;+\; p_{hint}\left(\alpha(1+\epsilon)\, G_t\right) \\ &= \left((1-p_{hint}) + \alpha p_{hint}(1+\epsilon)\right) G_t.\end{aligned} \tag{6}$$

Recalling $G_t \equiv \mathbb{E}[\nabla_\theta \log \pi_\theta(\tau)\, \hat{A}] = \nabla J(\theta_t)$ (the standard policy-gradient direction under our estimator), we obtain

$$\begin{aligned}&\mathbb{E}[\hat{g}_t \mid \theta_t] \approx \beta_t\, \nabla J(\theta_t), \\ &\qquad \text{where} \quad \beta_t := (1-p_{hint}) + \alpha p_{hint}(1+\epsilon).\end{aligned} \tag{7}$$

This completes the derivation. □

Theorem G.1 (Convergence to stationarity)**.** *Suppose Assumptions 1–2 hold and that Proposition 2's approximation is valid with $\beta_t \in [\beta_{\min}, \beta_{\max}]$ for constants $0 < \beta_{\min} \le \beta_{\max} < \infty$. Consider stochastic ascent updates*

$$\theta_{t+1} = \theta_t + \eta \hat{g}_t$$

with fixed step size satisfying

$$\eta \le \frac{1}{L\, \beta_{\max}}.$$

Let $J^* = \sup_\theta J(\theta) < \infty$. Then for every integer $T \ge 1$,

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\big\|\nabla J(\theta_t)\big\|^2 \le \frac{2\,(J^* - J(\theta_0))}{\eta T \beta_{\min}} + \frac{L\,\eta\,\sigma^2}{\beta_{\min}}.$$

Hence, choosing $\eta = \Theta(1/\sqrt{T})$ yields the standard stochastic non-convex rate

$$\min_{0\le t<T} \mathbb{E}\big\|\nabla J(\theta_t)\big\|^2 = O\big(T^{-1/2}\big),$$

i.e., the iterates converge to a stationary point in expectation at the usual $O(1/\sqrt{T})$ speed.

Proof. Starting from Assumption 1 and substituting $\Delta = \theta_{t+1} - \theta_t = \eta \hat{g}_t$ gives

$$J(\theta_{t+1}) \ge J(\theta_t) + \eta \langle \nabla J(\theta_t), \hat{g}_t \rangle - \tfrac{L\eta^2}{2}\|\hat{g}_t\|^2.$$

Take the conditional expectation $\mathbb{E}_t[\cdot] = \mathbb{E}[\cdot \mid \theta_t]$ and apply Proposition 2 to replace $\mathbb{E}_t[\hat{g}_t]$ by $\beta_t \nabla J(\theta_t)$ (approximately):

$$\mathbb{E}_t[J(\theta_{t+1})] \ge J(\theta_t) + \eta\beta_t \|\nabla J(\theta_t)\|^2 - \tfrac{L\eta^2}{2}\mathbb{E}_t\|\hat{g}_t\|^2.$$

Use the variance decomposition $\mathbb{E}_t\|\hat{g}_t\|^2 = \|\mathbb{E}_t[\hat{g}_t]\|^2 + \mathbb{E}_t\|\hat{g}_t - \mathbb{E}_t[\hat{g}_t]\|^2$ together with Assumption 2 to obtain

$$\mathbb{E}_t\|\hat{g}_t\|^2 \le \beta_t^2 \|\nabla J(\theta_t)\|^2 + \sigma^2.$$

Substituting back,

$$\begin{aligned}\mathbb{E}_t[J(\theta_{t+1})] - J(\theta_t) \ge \Big(\eta\beta_t - \tfrac{L\eta^2\beta_t^2}{2}\Big)\|\nabla J(\theta_t)\|^2 \\ - \tfrac{L\eta^2\sigma^2}{2}.\end{aligned} \tag{8}$$

With the choice $\eta \le 1/(L\beta_{\max})$ we have $L\eta\beta_t \le 1$, hence

$$\eta\beta_t - \tfrac{L\eta^2\beta_t^2}{2} = \eta\beta_t\Big(1 - \tfrac{L\eta\beta_t}{2}\Big) \ge \tfrac{\eta\beta_t}{2}.$$

Thus

$$\tfrac{\eta\beta_t}{2}\|\nabla J(\theta_t)\|^2 \le \mathbb{E}_t[J(\theta_{t+1}) - J(\theta_t)] + \tfrac{L\eta^2\sigma^2}{2}.$$

Taking total expectation, summing $t = 0, \ldots, T-1$, and using the bound $\beta_t \ge \beta_{\min} > 0$ yields

$$\frac{\eta\beta_{\min}}{2}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla J(\theta_t)\|^2 \le \mathbb{E}[J(\theta_T)] - J(\theta_0) + \tfrac{TL\eta^2\sigma^2}{2}.$$

Since $\mathbb{E}[J(\theta_T)] \le J^*$ we obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla J(\theta_t)\|^2 \le \frac{2(J^* - J(\theta_0))}{\eta T \beta_{\min}} + \frac{L\eta\sigma^2}{\beta_{\min}},$$

which proves the theorem. Choosing $\eta \propto T^{-1/2}$ gives the stated $O(1/\sqrt{T})$ rate. □