TEACH TO LEARN: HINT ANNEALING FOR SELF-IMPROVING LLM REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Group Relative Policy Optimization (GRPO) improves language-model reasoning by comparing verified rewards among multiple solution rollouts for each query. However, difficult training queries can yield only incorrect rollouts, leaving GRPO with no reward contrast or learning signal. Prior hint-based methods construct auxiliary hints from solution evidence and use them to re-solve failed queries, recovering learning signal. Yet the resulting trajectories are typically treated as ordinary solution trajectories despite being generated under an assisted condition unavailable at evaluation. We discover *hinted reward shift*: recovered reward contrast can concentrate policy updates on hinted trajectories, limiting improvement without hints. This also creates a trade-off: increasing hinted trajectories can accelerate early learning but intensify reward shift later. To address this problem, we propose HATCH (*Hint-Annealed Self-Teaching*), an online single-policy framework that learns from both generating and using its own hints to improve reasoning without assistance. To mitigate hinted reward shift, we introduce online weighting to anneal the contribution of hinted trajectories. However, learning to generate hints can conflict with improving query solving. We therefore use gradient projection to remove the opposing component of hint-generation updates. Together, these designs support self-improvement by enabling the policy to create learning opportunities for itself and turn them into stronger reasoning without hints. We evaluate our method on mathematical reasoning benchmarks and outperform state-of-the-art methods by 1.02 pp on Llama-3.2-1B-Instruct, 2.84 pp on Qwen3-1.7B, and 4.32 pp on Qwen3-8B.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) improves reasoning in language models by training a policy model with automatically checked answers (Shao et al., 2024; Yu et al., 2025). For each query, the policy model samples several solution rollouts. An automatic verifier determines whether each final answer is correct and assigns a reward accordingly. Group Relative Policy Optimization (GRPO) then compares these rewards within the rollout group and converts their differences into relative advantages: responses that perform better than their group peers are reinforced, while worse responses are suppressed. Thus, under binary correctness rewards, those groups containing both successful and unsuccessful rollouts provide the reward contrast needed to specify a local direction for improving the policy. However, queries within a batch vary in difficulty. For queries that the current policy rarely solves, a limited set of sampled rollouts can all be incorrect, leaving no reward contrast. As a result, these identical rewards yield zero relative advantages, leaving the policy without learning signal from queries it still struggles to solve.

Existing work addresses this issue along two complementary dimensions: improving rollout budget allocation and recovering learning signal from failed queries. The first line of work improves the use of the rollout budget through adaptive rollout allocation, selective rollout sampling, and dynamic filtering (Li et al., 2026b; Qu et al., 2025; Zheng et al., 2025; Yu et al., 2025). These strategies improve how computation is allocated and rollouts are selected for updates based on reward contrast already exposed by original rollouts. The second line of work constructs hints to recover learning signal from failed queries. These methods alter the condition under which difficult queries are solved by supplying partial reasoning, solution-derived hints, or learned guidance (Li et al., 2026a; Zhang et al., 2026b; Chen et al., 2026; Liao et al., 2026; Xia et al., 2026). By making successful

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

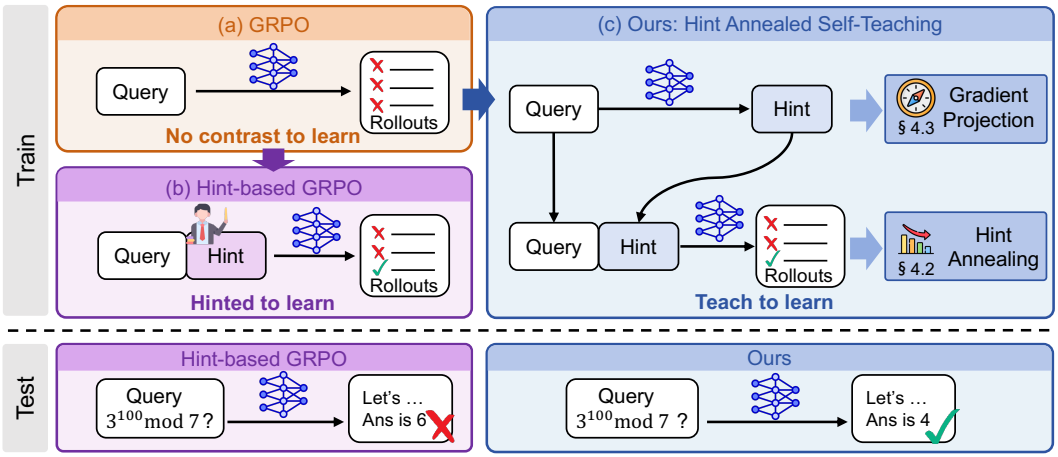


Figure 1: Comparison between GRPO, hint-based GRPO, and ours. (a) GRPO receives no group-relative learning signal for hard queries with all-incorrect rollouts. (b) Hint-based GRPO restores learning signal by re-solving under a hint, yielding strong improvement during training but limited improvement during testing. (c) Ours learns from generating and using its own hints, while gradually shifting training toward the original solving ability without hints.

solution paths easier to discover, this assistance can turn all-incorrect original rollout groups into re-solves with both successes and failures, thereby recovering reward contrast. Yet prior methods focus primarily on constructing hints, while the resulting trajectories are optimized under an assisted condition and can shift learning away from original-query solving.

Our analysis reveals that reward contrast recovered under hints can make the shared update favor hinted trajectories over the original-query trajectories used at evaluation. When original rollouts provide no reward contrast, hinted re-solving can recover learning signal, but the resulting updates directly optimize solving under hints. As examined in Section 3, hinted and original solving can induce substantially different gradient directions, even when both conditions yield reward contrast. Sustained hinted feedback can therefore continue to optimize solving under hints without producing corresponding gains on the original query. We call this phenomenon *hinted reward shift*. Hint-based RL must therefore not only recover learning signal from failed queries, but also prevent the shared update from favoring solving with hints over original-query solving.

To address this challenge, we propose HATCH (*Hint-Annealed Self-Teaching*), which frames self-improvement as a teach-to-learn process: the policy uses self-generated hints to turn its own failures into learnable re-solves, and learns from both the resulting solutions and the effectiveness of its teaching (Figure 1). Meanwhile, hints should serve as an annealing assistance: hinted-solve feedback is most valuable while unassisted rollouts provide little reward contrast and should recede as original solving improves. We realize this annealing through data-driven online weighting. Yet hint-generation serves a distinct reasoning role from solving, so the gradient signals need not point in the same direction. We address this conflict through gradient projection, removing the hint-generation component that opposes solution updates. Together, these designs enable the policy to guide its own learning toward sustained self-improvement in unassisted reasoning. Our contributions are:

- We discover that learning signal recovered under hints can favor hinted solving. We term this *hinted reward shift*, revealing a trade-off between early gains from hint recovery and sustained improvement on original query solving.
- We propose an online self-teaching framework that jointly learns hint generation and query solving with two components: online weighting to regulate the contribution of hinted trajectories and gradient projection to coordinate hint-generation learning with query solving.
- We evaluate our method across nine mathematical reasoning benchmarks and deliver consistent accuracy gains over state-of-the-art methods across three model scales: 1.02 pp on Llama-3.2-1B-Instruct, 2.84 pp on Qwen3-1.7B, and 4.32 pp on Qwen3-8B.

2 Related Work

RL Signal Utilization. GRPO derives learning signal from within-group reward contrast, while difficult queries yielding only incorrect rollouts provide no relative preference signal. To make better use of existing learning signal under the original rollout condition, prior work acts in three complementary ways. Sampling methods assign heterogeneous rollout budgets across prompts or predict prompt difficulty to prioritize those likely to yield informative gradients (Li et al., 2026b; Qu et al., 2025). Filtering and signal-shaping methods retain groups at appropriate difficulty, skip prompts predicted to be uninformative, down-sample redundant rollouts, or construct advantages for zero-variance groups (Bae et al., 2026; Zheng et al., 2025; Yu et al., 2025; Xu et al., 2025; Le et al., 2026). Curriculum methods schedule tasks from easy to hard or adapt the task mixture according to the policy's evolving learning progress (Bengio et al., 2009; Chen et al., 2025; Parashar et al., 2026). Together, these approaches improve the allocation of computation and updates under the original rollout condition. They are complementary to our work, which uses solution-derived hints to recover learning signal from queries that remain uninformative under original rollouts.

Hint-Based Signal Recovery. On the other hand, hint-based RL recovers learning signal from difficult queries by re-solving them under auxiliary contexts. One line of work explores auxiliary contexts that make difficult queries more accessible through partial solutions, tiered scaffolds, step-wise reasoning prefixes, or selected knowledge guidance (Li et al., 2026a; Zhang et al., 2026b;a; Yu et al., 2026). Another form of such auxiliary context guides exploration closer to successful reasoning paths through oracle prefixes or expert-anchored rollouts (Qu et al., 2026; Zhang et al., 2025). A second line studies how useful hints can be generated and maintained during training through self-generated abstractions, privileged solution-derived hints, or learned hinters conditioned on current policy failures (Chen et al., 2026; Liao et al., 2026; Xia et al., 2026). Within this direction, external guidance can further be kept compatible with online policy updates through abstract meta-hints and affinity-aware optimization (Wang et al., 2026). Together, these methods establish how auxiliary contexts can recover reward contrast and how useful hints can be produced. However, effective hints can still induce auxiliary updates that favor hinted solving over solving without hints. We instead study how learning to provide guidance can itself improve a policy's unassisted reasoning.

3 Observation

3.1 Preliminaries

GRPO and Hint-Based Optimization. Given verifiable outcome rewards, GRPO samples G responses $\{y_{i,k}\}_{k=1}^{G}$ for each query $q_i \in \mathcal{B}_t$ at each update step t, where i indexes individual queries and k indexes sampled responses within each query group. The verifier V assigns each response the outcome reward $R_{i,k} = V(q_i, y_{i,k})$. GRPO then computes the group-relative advantage

$$A_{i,k} = \frac{R_{i,k} - \frac{1}{G}\sum_{k'=1}^{G} R_{i,k'}}{\mathrm{Std}\big(\{R_{i,k'}\}_{k'=1}^{G}\big) + \epsilon}. \tag{1}$$

This advantage is assigned to the tokens of $y_{i,k}$. GRPO minimizes the standard clipped objective

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) = -\mathbb{E}_{q_i \sim \mathcal{B}_t}\left[\frac{1}{G}\sum_{k=1}^{G} \ell_{\mathrm{clip}}(y_{i,k}, A_{i,k})\right], \tag{2}$$

where ℓ_{clip} denotes the standard clipped surrogate (Schulman et al., 2017) for each sampled response. We define query groups with zero, some, or all correct rollouts as *solve-none*, *solve-partial*, or *solve-all*, denoted by $\mathcal{B}_t^{\mathrm{none}}$, $\mathcal{B}_t^{\mathrm{partial}}$, and $\mathcal{B}_t^{\mathrm{all}}$, respectively. Hint-based methods re-solve queries in $\mathcal{B}_t^{\mathrm{none}}$ under generated hints to recover reward contrast. Let $\mathcal{Y}$ and $\mathcal{Y}^h$ denote original and hinted trajectories with advantages A^{orig} and A^{hinted}. Applying the GRPO objective in Eq. (2) we have

$$\mathcal{L}_{\text{hint-based}}(\theta) = \underbrace{\mathcal{L}_{\mathrm{GRPO}}\big(\mathcal{Y}, A^{\mathrm{orig}}; \theta\big)}_{\mathcal{L}_{\mathrm{solve}}} + \underbrace{\mathcal{L}_{\mathrm{GRPO}}\big(\mathcal{Y}^h, A^{\mathrm{hinted}}; \theta\big)}_{\mathcal{L}_{\text{hinted-solve}}}. \tag{3}$$

Existing work has primarily focused on how to construct effective hints: by generating compact abstractions from solution, revealing privileged intermediate reasoning, or training a separate hinter to produce guidance for a subsequent attempt (Chen et al., 2026; Liao et al., 2026; Xia et al., 2026).

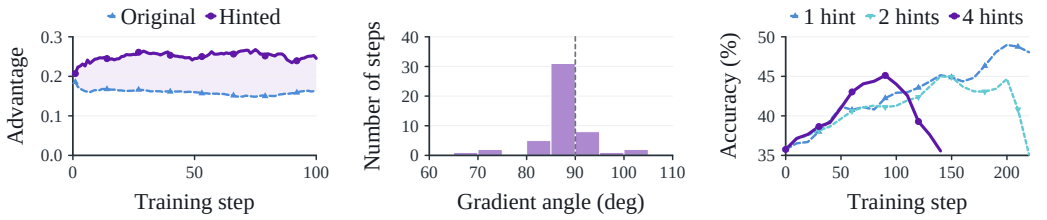


Figure 2: Evidence of hinted reward shift on Qwen3-8B. (a) Left: Mean absolute advantages of original and hinted trajectories. (b) Middle: Gradient-angle distribution between original and hinted solving over steps 1–50. (c) Right: Original-query accuracy with different numbers of hints.

3.2 Hinted Trade-off Problem

Hint-Assisted Contrast Recovery Leads to Reward Shift. Figure 2(a–b) examines the learning signals of the two objectives in Eq. (3) through their mean absolute advantages and gradient directions. Over training, hinted-solve trajectories retain mean absolute advantages comparable to or larger than those of original-solve trajectories, while the two objectives often produce nearly orthogonal gradients. Such advantage comparison reflects an asymmetry: queries in $\mathcal{B}_t^{\text{none}}$ provide no group-relative learning signal through original rollouts, while re-solving them under hints can constantly recover reward contrast. Because this recovered signal directly optimizes solving under hints, its sustained contribution can favor $\mathcal{L}_{\text{hinted-solve}}$ over $\mathcal{L}_{\text{solve}}$ in policy optimization. We call this tendency for learning signal to concentrate on hinted solving *hinted reward shift*. These observations suggest that sustained reward contrast under hints can lead to favoring hinted solving without ensuring corresponding gains on the original query.

Trade-off Between Early Gains and Hinted Reward Shift. Given this shift, we examine how policy improvement and the shift evolve under stronger hinted learning signals. To this end, we increase the number of hints per failed query, adding more hinted-solve groups to $\mathcal{L}_{\text{hinted-solve}}$. Figure 2(c) shows that multiple hints yield faster early accuracy gains but subsequently deteriorate, whereas the single-hint setting continues to improve at a slower pace. This reflects the changing availability of learning signal from original rollouts: when $\mathcal{B}_t^{\text{none}}$ is large, hinted-solve groups turn zero-signal queries into informative comparisons; but their continued influence may limit later improvement as original learning signal emerges. Therefore, hinted trajectories should contribute most when many queries lack learning signal and recede as that signal emerges, allowing the model to retain its early gains. This observation suggests a trade-off between faster early gains and sustained original-query improvement, consistent with *hinted reward shift*. An effective method should therefore fully exploit the learning potential of hinted trajectories while keeping the ability to solve without hints.

4 Method

To address the challenge in Section 3, we propose HATCH (*Hint-Annealed Self-Teaching*), an online single-policy framework (Figure 3). It constructs hint-generation and hinted-solving objectives from initially failed queries in Section 4.1, regulates the contribution of hinted trajectories through online weighting in Section 4.2, and uses gradient projection to address role divergence in Section 4.3.

4.1 Online self-hinting for valuable trajectories

Section 3 shows the potential of additional hinted trajectories to improve policy learning. To continually generate more useful hints, we jointly train hint generation and query solving within one policy, using re-solving outcomes to supervise hint generation. This introduces $\mathcal{L}_{\text{hint-gen}}$ alongside $\mathcal{L}_{\text{solve}}$ and $\mathcal{L}_{\text{hinted-solve}}$ in Eq. (3). Specifically, for each query $q_i \in \mathcal{B}_t^{\text{none}}$, we condition the policy on its reference solution z_i to sample M hints as

$$h_{i,j} \sim \pi_t(\cdot \mid P_{\text{hint}}(q_i, z_i)), \qquad j = 1, \ldots, M. \tag{4}$$

To evaluate each hint through its effect on solving, we append $h_{i,j}$ to q_i and sample G re-solves $\{y_{i,j,k}^h\}_{k=1}^G$ from the same policy. We score these responses and compute $A_{i,j,k}^{\text{hinted}}$ within each hinted-

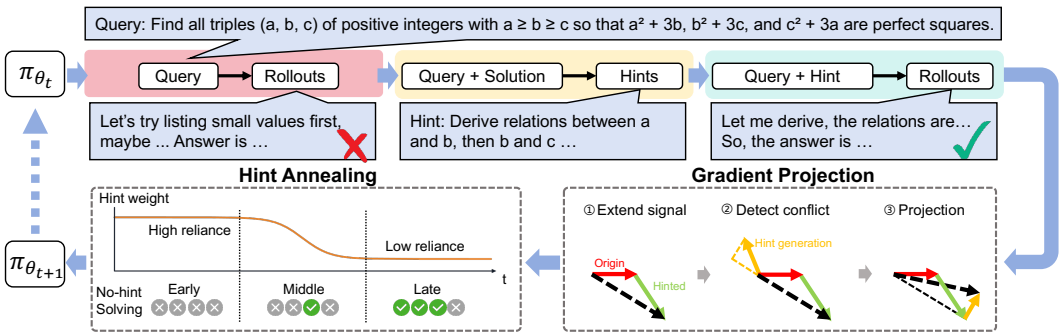


Figure 3: Overview of our method. For solve-none queries, it first self-generates hints from solutions and then re-solves the query with the hints, yielding three complementary training trajectories: native solving, hint generation, and hinted solving. Online weighting adaptively controls the contribution of hinted-solving feedback, while gradient projection removes conflicting components from hint-generation updates. The resulting signals are jointly used to update a single shared policy.

solve group using Eq. (1). The same outcomes also supervise hint generation through the reward

$$R_{i,j}^{\text{hint-gen}} = \mathcal{R}_{\text{hint}}(\Delta\text{Pass}_{i,j}, \Delta \log p_{i,j}) \,. \tag{5}$$

Here, $\Delta\text{Pass}_{i,j}$ measures the rollout accuracy gain, and $\Delta \log p_{i,j}$ measures the log-probability difference of the successful trajectory under the hinted and original conditions (Xia et al., 2026). Applying Eq. (1) across the M hint rewards yields $A_{i,j}^{\text{hint-gen}}$. Using these advantages to optimize the hint-generation trajectories with Eq. (2) defines $\mathcal{L}_{\text{hint-gen}}$. We retain all M hinted-solve groups rather than selecting one. Let T_O, T_S, and T_H denote the numbers of active response tokens in original solving, hinted solving, and hint generation, respectively, with $T_F = T_O + T_S + T_H$. Each loss is averaged over the active response tokens of its own trajectory. The joint objective is

$$\mathcal{L}_{\text{joint}} = \frac{T_O}{T_F}\mathcal{L}_{\text{solve}} + \frac{T_S}{T_F}\mathcal{L}_{\text{hinted-solve}} + \frac{T_H}{T_F}\mathcal{L}_{\text{hint-gen}}. \tag{6}$$

As the policy learns to solve queries, its re-solving outcomes also refine the hints it generates for its remaining failures. The following subsections regulate the contribution of hinted trajectories and the direction of hint-generation updates so that both support learning to solve the original queries.

4.2 Online Weighting of Hinted Trajectories

To mitigate hinted reward shift in the joint training of Section 4.1, we adjust the contribution of $\mathcal{L}_{\text{hinted-solve}}$ according to the learning signal available from original rollouts. The relative proportions of unsolved groups and groups with reward contrast indicate how much training can rely on original rollouts. Let $\rho_{\text{none}}^{(t)}$ and $\rho_{\text{partial}}^{(t)}$ denote $|\mathcal{B}_t^{\text{none}}|/|\mathcal{B}_t|$ and $|\mathcal{B}_t^{\text{partial}}|/|\mathcal{B}_t|$, respectively. We smooth these proportions with Exponential Moving Averages (EMA) to determine the weight β_t as

$$p_t = \frac{\text{EMA}(\rho_{\text{none}}^{(t)})}{\text{EMA}(\rho_{\text{none}}^{(t)}) + \text{EMA}(\rho_{\text{partial}}^{(t)}) + \epsilon}, \qquad \beta_t = \text{clip}(\kappa p_t^{\gamma}, 0, 1) \,. \tag{7}$$

As the balance shifts from $\mathcal{B}_t^{\text{none}}$ toward $\mathcal{B}_t^{\text{partial}}$, β_t decreases, reducing reliance on hinted trajectories as original reward contrast emerges. The scale κ controls the overall strength, while γ controls how sharply the weight decreases. We apply β_t only to hinted-solve advantages as

$$\widetilde{A}_{i,j,k}^{\text{hinted}} = \beta_t A_{i,j,k}^{\text{hinted}}. \tag{8}$$

Online weighting thus mitigates hinted reward shift by letting hinted trajectories supplement learning when original reward contrast is scarce, while reducing their contribution as that contrast emerges. Still, the direction of hint-generation updates requires separate treatment.

4.3 Gradient Projection for Hint Generation

Although hint generation is rewarded by re-solving outcomes, its updates need not support query solving within the shared policy. Figure 4 shows that the angle between hint-generation gradient and the combined gradient from original-solve and weighted hinted-solve trajectories fluctuates around 90° and frequently exceeds it, indicating a learning divergence. We therefore use gradient projection to make hint-generation updates compatible with the update induced by solution trajectories.

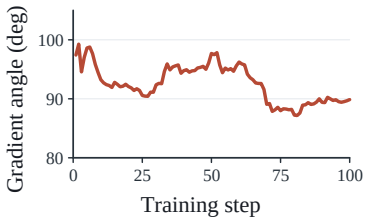


Figure 4: Angle between the hint-generation gradient and the solution-side gradient.

Let $\mathcal{L}_{\text{full}}$ denote the joint loss of all three objectives after applying Eq. (8). We then compute its full gradient and the hint-generation gradient separately as

$$g_F = \nabla_\theta \mathcal{L}_{\text{full}}, \qquad g_H = \nabla_\theta \mathcal{L}_{\text{hint-gen}}. \tag{9}$$

Since g_H is normalized over hint-generation tokens alone, its contribution to the full gradient across all three objectives is scaled by T_H/T_F as

$$g'_H = \frac{T_H}{T_F} g_H, \qquad g_P = g_F - g'_H. \tag{10}$$

Here, g_P combines the gradients from the solution side trajectories as the reference direction. When $\langle g'_H, g_P \rangle$ is nonnegative, hint-generation learning does not oppose the solution-side update and is retained unchanged. Otherwise, we remove only its component opposite to g_P as

$$g_{\text{update}} = g_P + \begin{cases} g'_H - \dfrac{\langle g'_H, g_P \rangle}{\|g_P\|_2^2 + \epsilon} g_P, & \langle g'_H, g_P \rangle < 0, \\ g'_H, & \text{otherwise.} \end{cases} \tag{11}$$

The projection preserves g_P and the non-opposing component of hint-generation learning. This allows the policy to learn useful hints without counteracting the update from solution trajectories.

4.4 Hint Annealed Self Teaching

Having specified online weighting and gradient projection, we now combine the three objectives in a single policy update. Let $\widetilde{\mathcal{L}}_{\text{hinted-solve}}$ denote the hinted-solve loss evaluated with the weighted advantages $\widetilde{A}^{\text{hinted}}$ from Eq. (8). The joint loss before gradient projection is

$$\mathcal{L}_{\text{full}}(\theta) = \frac{T_O}{T_F}\mathcal{L}_{\text{solve}} + \frac{T_S}{T_F}\widetilde{\mathcal{L}}_{\text{hinted-solve}} + \frac{T_H}{T_F}\mathcal{L}_{\text{hint-gen}}. \tag{12}$$

We apply Eq. (11) to the hint-generation component of $\nabla_\theta \mathcal{L}_{\text{full}}$ and use the resulting g_{update} to update the policy. Algorithm 1 summarizes one training update. At inference, the policy receives only the original query, without hints or reference solutions.

Algorithm 1 *Hint-Annealed Self-Teaching*

Input: rollout policy π_t, training batch $\{(q_i, z_i)\}$
Sample G original-solve responses for each q_i; compute $R_{i,k}$, $A^{\text{orig}}_{i,k}$, and $\mathcal{B}^{\text{none}}_t$
for each $q_i \in \mathcal{B}^{\text{none}}_t$ **do**
  Sample M hints $h_{i,j}$ conditioned on q_i, z_i
  For each hint, sample G hinted-solve responses
  Compute $A^{\text{hinted}}_{i,j,k}$ and $A^{\text{hint-gen}}_{i,j}$ from the re-solve outcomes
end for
Compute β_t using Eq. (7) and obtain $\widetilde{A}^{\text{hinted}}_{i,j,k}$ using Eq. (8)
Form $\mathcal{L}_{\text{full}}$, compute g_F, g_H, g'_H, g_P, and obtain g_{update} using Eq. (11)
Update $\theta_{t+1} \leftarrow \text{Optimizer}(\theta_t, g_{\text{update}})$

Table 1: Per-dataset no-hint accuracy (%) for the initial models and each trained method. Bold marks the best result and underlining marks the second-best distinct result with ties retained.

Method	Math500	Minerva	Oly.	AIME24/25/26	AMC23	HMMT25	BRUMO25	Avg.
Llama-3.2-1B-Instruct	24.04	4.41	4.71	1.11/0.00/0.00	5.00	0.00	1.11	4.49
+ GRPO	25.45	5.02	4.71	**2.22**/1.11/1.11	6.67	1.11	1.11	5.39
+ QuESTA	26.92	6.13	5.21	**2.22**/1.11/1.11	7.50	**2.22**	2.22	6.07
+ HiLL	26.12	5.63	4.96	1.11/**2.22**/1.11	8.33	0.00	2.22	5.74
+ HATCH (ours)	**27.38**	**6.62**	**5.65**	**2.22/2.22/3.33**	**10.83**	**2.22**	**3.33**	**7.09**
Qwen3-1.7B	79.76	52.57	48.66	16.67/16.67/10.00	42.50	6.67	16.67	32.24
+ GRPO	82.57	54.41	49.80	17.78/21.11/16.67	50.00	8.89	26.67	36.43
+ QuESTA	83.17	54.29	**51.49**	21.11/18.89/17.78	54.17	10.00	22.22	37.01
+ HiLL	83.23	53.68	50.79	20.00/22.22/17.78	53.33	11.11	22.22	37.15
+ HATCH (ours)	**83.97**	**55.02**	**51.49**	**23.33/25.56/18.89**	**58.33**	**14.44**	**28.89**	**39.99**
Qwen3-8B	83.57	57.72	48.36	26.67/20.00/20.00	62.50	6.67	16.67	38.02
+ GRPO	86.37	59.07	55.26	37.78/31.11/31.11	74.17	16.67	36.67	47.58
+ QuESTA	87.84	61.76	58.38	47.78/32.22/32.22	72.50	10.00	38.89	49.07
+ HiLL	88.58	58.33	58.09	41.11/34.44/31.11	73.33	16.67	43.33	49.44
+ HATCH (ours)	**90.71**	**63.11**	**60.27**	**52.22/38.89/35.56**	**80.83**	**17.78**	**44.44**	**53.76**

5 Experiments

5.1 Experimental Setup

Training settings. We train Llama-3.2-1B-Instruct (Meta, 2024), Qwen3-1.7B (Team et al., 2025), and Qwen3-8B (Team et al., 2025). Within each model scale, all methods are initialized from the same pretrained checkpoint. We sample 10,000 problems from NuminaMath (LI et al., 2024), splitting into 9,800 training and 200 validation set. Training uses VERL framework (Sheng et al., 2024), with Megatron (Shoeybi et al., 2019) backend for distributed policy optimization and vLLM (Kwon et al., 2023) for asynchronous rollout generation. We use GRPO with $G = 8$ rollouts per query. For each solve-none query, our method generates $M = 4$ hints from the current policy and samples $G = 8$ hinted re-solves under each hint. The maximum prompt and generation lengths are 4,096 and 8,192 tokens, respectively. Solving rollouts use temperature 0.85 and top-$p = 1.0$, while hint generation uses temperature 0.3 and top-$p = 0.95$. We use a batch size of 128 and optimize the policy with Adam (Kingma & Ba, 2014) using a learning rate of 1×10^{-6}, a 5-step warmup and a KL penalty of 0.01. Experiments are conducted on 4 nodes with 8 NVIDIA H20 GPUs each.

Baselines. Rows labeled with model names in Table 1 report initial accuracy. *GRPO* (Shao et al., 2024) directly optimizes the original queries using group-relative advantages. *QuESTA* (Li et al., 2026a) is reimplemented following its two-stage curriculum, which gradually reduces the amount of solution guidance. For a matched online comparison, *HiLL* (Xia et al., 2026) learns utility-scored hints and trains on the best hint. *Ours* uses the hint annealed self-teaching described in Section 4.

Evaluation settings. For each run, we select the checkpoint with the highest no-hint validation accuracy and report its no-hint accuracy on nine benchmarks. We report pass@1, averaged over three independent runs with different random seeds in Table 1. The sample standard deviation over the three runs is at most 0.8 pp. We evaluate on Math500 (Lightman et al., 2024), Minerva Math (Lewkowycz et al., 2022), OlympiadBench (He et al., 2024), AIME 2024–2026 (Zhang & Math-AI, 2024; 2025; 2026), AMC 2023 (Art of Problem Solving), HMMT 2025 (Dekoninck et al., 2026), and BRUMO 2025 (Dekoninck et al., 2026). Among these benchmarks, HMMT 2025 and BRUMO 2025 provide additional tests of generalization across competition-specific problem sets.

5.2 Main Results

Table 1 reports original-query pass@1 across three model scales. Our method achieves the highest macro average at every scale, improving over GRPO by 1.70 pp on Llama-3.2-1B-Instruct, 3.56 pp

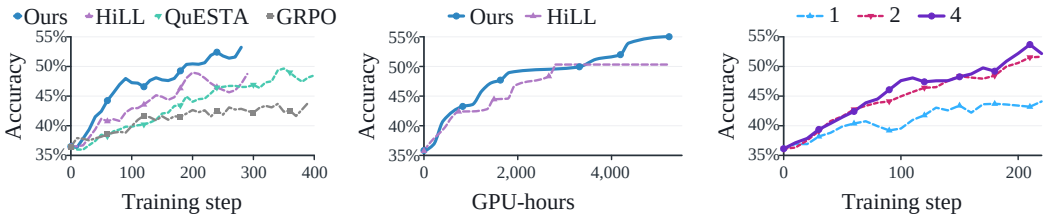


Figure 5: Left: average benchmark accuracy over training updates. Middle: best-so-far no-hint benchmark accuracy against cumulative GPU-hours for the two online methods. Right: accuracy of training by our method with single-, two-, and four retained hinted-solve groups.

on Qwen3-1.7B, and 6.18 pp on Qwen3-8B. It also outperforms both hint-based baselines at all three scales. The improvement is broad rather than concentrated on a single benchmark: on Qwen3-8B, our method obtains the best score on every benchmark in the evaluation suite, including HMMT 2025 and BRUMO 2025, which assess generalization across different competition-specific problem distributions. These results show that the recovered trajectories improve the policy's ability to solve the original queries, rather than merely improving performance under hinted inputs.

Figure 5 traces how the endpoint gains emerge during training and how they depend on the number of retained hinted-solve groups on Qwen3-8B. In the left panel, our method surpasses 50% accuracy by around step 200 and reaches approximately 53% near step 280, whereas HiLL, QuESTA, and GRPO remain below 49%. The middle panel provides the corresponding compute view: up to approximately 49% best-so-far accuracy, our method reaches each level with lower cumulative GPU-hours than HiLL. The right panel reports the trajectory-count ablation: by step 220, the two- and four-group settings reach approximately 51% accuracy, whereas the single-group setting remains near 44%. Together, these dynamics show that our method achieves stronger no-hint learning while making effective use of the additional trajectories recovered under hints.

5.3 Further Analysis

Improvement from single policy joint training. To assess the value of jointly learning hint generation and query solving, we compare our method with three variants in Table 2. The *Offline Hints* and *External Hints* variants use fixed hints generated offline by Qwen3-8B and Qwen3-30B-A3B, respectively. These variants achieve 48.07% and 48.80%, suggesting that a larger external hinter alone does not reproduce the gains from joint learning. To isolate the benefit of learning hint construction, *Online Hints* retains current-policy hints but removes $\mathcal{L}_{\text{hint-gen}}$. It achieves 50.16%, 3.60 pp below ours. Together, these results support learning from hint construction, beyond using hints solely as auxiliary inputs.

Table 2: Ablation results on Qwen3-8B.

Variant	Acc. (%)
HATCH (ours)	**53.76**
Offline Hints	48.07
External Hints	48.80
Online Hints	50.16
Fixed Weighting	44.96
Linear Decay	48.25
Staged Exp. Decay	51.25
No Projection	51.93
Reverse Projection	49.24

Hinted Reward Shift Correction by Online Weighting. We compare data-driven hint annealing with fixed weighting and two predefined decay schedules in Table 2. Fixed weighting reduces accuracy to 44.96%. Linear decay and staged exponential decay achieve 48.25% and 51.25%. These results support the effectiveness of data-driven hint annealing relative to the fixed weighting and predefined decay schedules. The middle panel of Figure 6 shows how online weighting reduces hinted-solve contributions as original reward contrast emerges. The right panel shows that $\gamma = 5$ best balances this trade-off: annealing too quickly weakens useful early feedback, whereas annealing too slowly sustains hinted reward shift. Together, these results support hint annealing as a way to exploit assistance early without allowing it to dominate once original rollouts become informative.

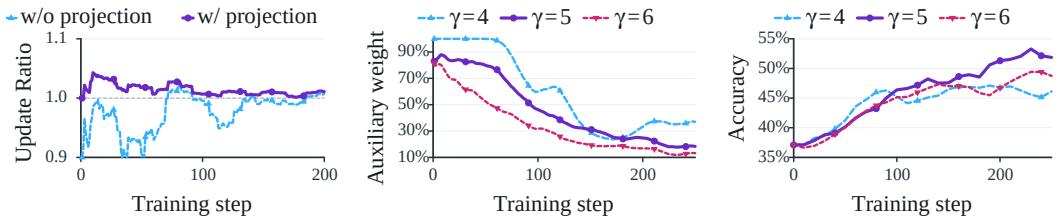


Figure 6: Ablation dynamics on Qwen3-8B. Left: relative strength in the solution direction in the final shared update gradient. Middle: auxiliary-weight schedules induced by the three γ settings. Right: the corresponding nine-benchmark accuracy curves of each selected γ.

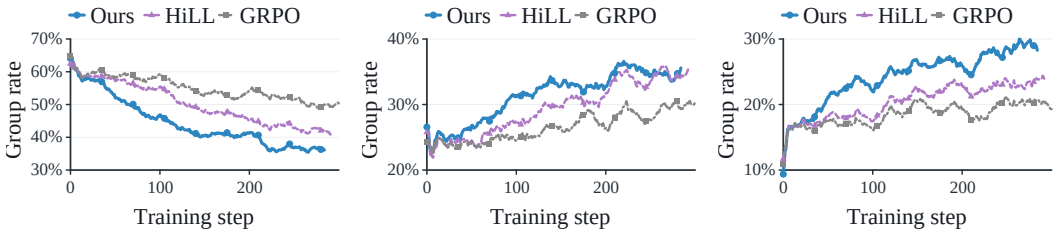


Figure 7: No-hint group-state dynamics across Ours, HiLL, and GRPO on Qwen3-8B. Ours progressively converts solve-none groups into solve-partial and solve-all groups, indicating stronger unassisted solving. From left to right panels are: solve-none, solve-partial and solve-all groups.

Beneficial hint learning with gradient projection. To examine whether gradient projection helps hint-generation learning improve query solving, we analyze its effect on solution-side updates and final accuracy. The left panel of Figure 6 shows that projection preserves the shared update in the solution direction, whereas the unprojected update is repeatedly weakened by the conflicting component. Correspondingly, Table 2 shows that removing projection reduces accuracy to 51.93%. Reverse projection, which retains only the opposing component, performs substantially worse at 49.24%. These results show that the gain comes from preserving directionally compatible hint-generation learning, rather than allowing it to counteract original-query solving.

Improved query solving. Recovered hinted trajectories should improve how the policy solves a query itself, rather than only its behavior after receiving a hint. To examine this, Figure 7 tracks the composition of rollout groups throughout training. Compared with other methods, our method reduces the fraction of solve-none groups more rapidly while increasing solve-partial and solve-all groups. This shows that we better improve solving behavior beyond the hinted condition.

5.4 Limitations and Future Work

First, our evaluation focuses on mathematical reasoning with verifiable rewards and reference. In interactive tasks such as tool use, new observations can change the guidance needed at each step. Extending HATCH to such tasks therefore requires verification and reference suited to the context. Second, our experiments use only textual inputs and hints. Future work could explore multimodal self-teaching by incorporating visual evidence into hint construction and subsequent reasoning.

6 Conclusion

We discover *hinted reward shift*, where recovered signal can favor hinted solving without corresponding gains without hints. To address this, we propose HATCH (*Hint-Annealed Self-Teaching*), which jointly learns hint generation and query solving within one policy, using online weighting and gradient projection to regulate the strength and direction of auxiliary learning. Experiments across three model scales show stronger unassisted mathematical reasoning and suggest that models can improve their reasoning by learning to provide effective guidance for themselves.

AI USE STATEMENT

We used generative AI tools to assist with manuscript writing, improve clarity and logic, and proofread the text. The authors take full responsibility for the final manuscript.

ETHICS STATEMENT

This study uses publicly available mathematical reasoning datasets and pretrained language models. It does not involve human participants or the collection of personal information. Appendix F documents the sources and license terms of the principal datasets, models, and software used in our experiments. Generated hints and responses may inherit errors and biases from the underlying models and data. Although our evaluation focuses on mathematical reasoning, applications beyond this should undergo task-specific validation and include appropriate safeguards and human oversight.

REPRODUCIBILITY STATEMENT

Section 4 and Algorithm 1 specify the training procedure, while the experimental setup describes the backbones, training settings, baselines, and evaluation protocol. Appendix A provides data-processing details, additional hyperparameters, the hint-generation reward, and evaluation settings. Appendix B presents the prompt templates. Detailed ablation results and computational measurements are provided in Appendices C and E, respectively.

REFERENCES

Art of Problem Solving. Amc problems and solutions. https://artofproblemsolving.com/wiki/index.php?title=AMC_Problems_and_Solutions.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, 2026.

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

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. In *The Fourteenth International Conference on Learning Representations*, 2026.

Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*, 2025.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms. *arXiv preprint arXiv:2605.00674*, 2026.

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

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Thanh-Long V Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in llm reinforcement learning via entropy-guided advantage shaping. In *International Conference on Learning Representations*, volume 2026, pp. 121956–121982, 2026.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 3843–3857, 2022.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. https://huggingface.co/datasets/AI-MO/NuminaMath-1.5, 2024.

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*, 2026a.

Ziniu Li, Congliang Chen, Tianyun Yang, Tian Ding, Ruoyu Sun, Ge Zhang, Wenhao Huang, and Zhi-Quan Luo. Knapsack RL: Compute-efficient reinforcement learning via heterogeneous rollout allocation. In *Forty-third International Conference on Machine Learning*, 2026b.

Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning. *arXiv preprint arXiv:2602.03143*, 2026.

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 *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

AIa Meta. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. *Meta AI Blog. Retrieved December*, 20(2024), 2024.

Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Curriculum reinforcement learning from easy to hard tasks improves LLM reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. Optimizing test-time compute via meta reinforcement fine-tuning. *arXiv preprint arXiv:2503.07572*, 2025.

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.

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

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

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. *arXiv preprint arXiv: 2409.19256*, 2024.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Qwen Team et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 6(7):13, 2025.

Xinyi Wang, Jinyi Han, Zishang Jiang, Jiaqing Liang, Sihang Jiang, Zhaoqian Dai, Ma Shuguang, Fei Yu, Yanghua Xiao, et al. Don't tell the answer, truly guide the reasoning during rl rollouts. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 3437–3455, 2026.

Yu Xia, Canwen Xu, Zhewei Yao, Julian McAuley, and Yuxiong He. Learning to hint for reinforcement learning. *arXiv preprint arXiv:2604.00698*, 2026.

Yixuan Even Xu, Yash Savani, Fei Fang, and J Zico Kolter. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025.

Linhao Yu, Tianmeng Yang, Siyu Ding, Renren Jin, Naibin Gu, Xiangzhao Hao, Shuaiyi Nie, Deyi Xiong, Weichong Yin, Yu Sun, et al. Knowrl: Boosting llm reasoning via reinforcement learning with minimal-sufficient knowledge guidance. *arXiv preprint arXiv:2604.12627*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113222–113244, 2025.

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. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37846–37864, 2026a.

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. In *International Conference on Learning Representations*, volume 2026, pp. 131946–131974, 2026b.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. Bread: Branched rollouts from expert anchors bridge sft & rl for reasoning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 96726–96752, 2025.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2026, 2026.

Haizhong Zheng, Yang Zhou, Brian Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for llm reasoning via selective rollouts. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 124321–124346, 2025.

A Implementation and Evaluation Details

A.1 Training Data and Sampling

The NuminaMath subset contains 9,800 training queries and 200 held-out validation queries. Reference solutions are attached by matching the query text to NuminaMath-1.5 (LI et al., 2024), first after stripping surrounding whitespace and then after normalizing internal whitespace. Whether a query requires assistance is determined from its current original rollout group, rather than from a fixed difficulty label. For each query in $\mathcal{B}_t^{\text{none}}$, the policy generates four hints and samples eight re-solves per valid hint. Original and hinted responses use the same final-answer verification interface. The reference solution conditions hint generation but is not supplied directly to the solving prompt.

A.2 Training Configuration

Table 3 supplements the settings in the main paper. Hint annealing changes the contribution of hinted-solve trajectories after advantage normalization; it does not truncate the hint text.

Table 3: Additional training settings. The common settings apply to all three backbones; the final three rows specify the online weighting parameters.

Setting	Value
Original responses / hints / re-solves per hint	$G = 8$, $M = 4$, $G = 8$
Prompt / response / hint generation limit	4,096 / 8,192 / 8,192 tokens
Reference-solution input limit	8,000 characters
Solution sampling temperature / top-p	0.85 / 1.0
Hint sampling temperature / top-p	0.3 / 0.95
Rollout batch / PPO minibatch / PPO epochs	128 / 128 / 1
Per-GPU microbatch size	1
Learning rate / warmup / weight decay	10^{-6} / 5 steps / 0.1
GRPO clipping bounds	$\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.28$
KL coefficient: solution / hint-generation tokens	0.01 / 0
EMA update coefficient / numerical stabilizer	0.1 / 10^{-6}
Hint reward temperature / leakage penalty	0.3 / -0.2
Post-normalization hint-generation advantage scale	1.0
Llama-3.2-1B-Instruct: (κ, γ)	$(1.2, 4)$
Qwen3-1.7B: (κ, γ)	$(65, 10)$
Qwen3-8B: (κ, γ)	$(5, 5)$

Valid trajectories. For Qwen models, the thinking prefix is removed before the hint is appended to the query; a generation without a closing thinking tag is treated as an empty hint. This extraction does not redefine the sampled hint-generation trajectory used for training. The Llama hint-generation configuration does not enable a thinking block. Responses reaching the configured generation limit are assigned zero outcome reward by the overlength mask.

Auxiliary updates. The EMAs in Section 4.2 are initialized from the first observed rollout composition and subsequently updated with coefficient 0.1. The resulting β_t scales hinted-solve advantages after within-group normalization. Hint-generation advantages retain unit scale, and no KL penalty is applied to their tokens. Gradient projection is applied over all trainable parameters at each policy update containing hint-generation rows.

A.3 Hint-Generation Reward

To make the reward construction in Section 4.1 explicit, let $\mathcal{C}_{i,j} = \{k : R_{i,j,k}^{\text{hinted}} > 0.5\}$ be the indices of verified successful re-solves for hint $h_{i,j}$, and let $\hat{p}_{i,j} = |\mathcal{C}_{i,j}|/G$. Since the original group is solve-none, $\hat{p}_{i,j}$ also measures its rollout accuracy gain. We score each successful response under both the hinted and original inputs. Writing $\bar{\ell}_\theta(y \mid x)$ for its mean log-probability over non-padding response tokens, we compute

$$\Delta \log p_{i,j} = \frac{1}{|\mathcal{C}_{i,j}|} \sum_{k \in \mathcal{C}_{i,j}} \left[\bar{\ell}_\theta(y_{i,j,k}^h \mid q_i) - \bar{\ell}_\theta(y_{i,j,k}^h \mid q_i \oplus h_{i,j}) \right]. \tag{A.1}$$

This quantity is evaluated only when successful responses are available. Following HiLL (Xia et al., 2026), the implemented hint reward is

$$R_{i,j}^{\text{hint-gen}} = \begin{cases} -0.2, & \text{answer leakage,} \\ 0, & \text{if } \hat{p}_{i,j} \in \{0, 1\}, \\ \left[1 - \hat{p}_{i,j}^{G} - (1 - \hat{p}_{i,j})^{G}\right] \exp\left(\dfrac{\min(\Delta \log p_{i,j}, 0)}{0.3}\right), & \text{otherwise.} \end{cases} \tag{A.2}$$

The first factor favors groups with reward contrast, while the exponential factor discounts successful responses that are much less likely without the hint. Thus, hint reward is not simply re-solving accuracy: an all-correct group also provides no group-relative contrast. The leakage check detects a boxed reference answer or its occurrence in the hint. For answers of at most two characters, it instead checks answer-revealing patterns such as "answer is" or "equals" to reduce incidental matches.

A.4 Evaluation Protocol

Benchmarks and decoding. For each run, the checkpoint is selected by its highest accuracy on the held-out 200-problem validation set. The nine benchmark scores are reported at this same checkpoint. Evaluation samples one response per original query with a maximum of 8,192 generated tokens, without hints or reference solutions. Accuracy is computed within each of the nine benchmarks in Table 1 and then averaged with equal weight across benchmarks. The main results average three independent training runs. The detailed ablation table and supplementary curves have the reporting conventions specified in Appendix C.

Supplementary curves. Learning curves show individual training runs in the main table. Per-benchmark curves use exponential smoothing with weight 0.3 and show raw evaluations in lighter colors. Training diagnostics use trailing 20-step means with raw observations in lighter colors. Smoothing is used only for visualization, not to compute the reported accuracies.

B Prompt Templates

B.1 Original and Hinted Solving

Original solving uses the message sequence stored with each example, rendered through the backbone's chat template. Any existing system message and answer-format instructions are preserved. Hinted solving uses the same sequence, appending the generated hint to the final user message after a blank line. The following shows the change to that message; the braced field denotes its original content, including any existing instructions. Neither the reference answer nor the solution is supplied directly to the solving policy. Evaluation uses the original message without the hint.

Original solving

```
{original user message}
```

Hinted solving

```
{original user message}

[Hint: {hint}]
```

B.2 Reference-Conditioned Hint Construction

The system and user messages below are passed through the backbone's chat template. The same instruction is used throughout training; the generated hint changes with the current policy and query. Reference solutions exceeding the character limit in Table 3 are truncated.

B.3 Outcome Verification

Hint quality is computed from verified re-solving outcomes using Eq. (A.2), rather than an additional judge prompt. For answer correctness, original and hinted training responses are routed to the same configured reward service. Evaluation uses answer checkers selected by the example's data-source

System message
You are a helpful math tutor. You will be given a problem, its reference answer, AND a complete reference solution. Your job is to write a substantive hint that guides the student through the key ideas and reasoning steps needed to solve the problem. Cover the important approach, key observations, and any crucial intermediate steps — give enough guidance for the student to make real progress, not just a vague nudge. Do NOT directly state the final answer. Do NOT copy the reference solution verbatim. Do NOT merely restate the problem.

User message
Problem:
{question}
Reference answer: {ground_truth}
Reference solution (for YOUR eyes only, do NOT copy it):
{solution}
Now write the hint. Respond with the hint text only, no preamble, no restating of the problem, and do NOT reveal the final answer.

Figure 8: The hint-generation prompt. Braced fields are replaced with the training query and its reference information.

identifier. These compare the extracted answer with the stored ground truth; the hint-generation reward is not an evaluation metric. Thus, answer verification supplies the outcome rewards, while Eq. (A.2) uses those outcomes to assign credit to hint construction.

C Additional Results and Ablations

C.1 Component Ablations

Table 4 expands ablation results. The comparisons examine learning to generate hints, annealing hinted-solve contributions, and gradient projection. The HATCH row shows the three-run average in Table 1; each ablation variant in Tables 2 and 4 is evaluated using a single training run.

Table 4: Qwen3-8B ablation accuracy (%) by benchmark. Avg. gives equal weight to all nine benchmarks. Variant names follow Table 2.

Variant	Math500	Minerva	Oly.	AIME24/25/26	AMC23	HMMT25	BRUMO25	Avg.
HATCH	90.71	63.11	60.27	52.22/38.89/35.56	80.83	17.78	44.44	**53.76**
Offline Hints	86.97	63.60	56.25	43.33/30.00/30.00	72.50	13.33	36.66	48.07
External Hints	88.77	62.50	58.77	40.00/26.66/30.00	72.50	16.66	43.33	48.80
Online Hints	90.38	64.70	58.92	43.33/36.66/33.33	77.50	10.00	36.66	50.16
Fixed Weighting	87.98	61.76	54.91	43.33/30.00/26.67	60.00	10.00	30.00	44.96
Linear Decay	90.38	65.07	58.77	40.00/26.66/33.33	70.00	10.00	40.00	48.25
Staged Exp. Decay	90.58	65.80	60.71	40.00/33.33/36.66	77.50	16.66	40.00	51.25
No Projection	89.38	63.97	60.71	46.67/33.33/30.00	80.00	16.67	46.67	51.93
Reverse Projection	90.38	61.03	61.76	43.33/26.67/26.67	80.00	13.33	40.00	49.24

Learning from hint construction. The *Offline Hints* variant uses one fixed hint per query generated by Qwen3-8B and achieves 48.07%. To evaluate guidance from a larger external model, *External Hints* uses four fixed hints per query from Qwen3-30B-A3B, retaining online weighting. It achieves 48.80%. We further distinguish fixed guidance from online generation through *Online Hints*, which retains four hints generated by the current policy and online weighting but removes the hint-generation loss. Its accuracy reaches 50.16%, still below HATCH's 53.76%. Across these variants, hints assist re-solving without directly training hint construction. Their results support learning hint generation within the solving policy, beyond supplying hints alone.

Annealing the hinted-solve contribution. Fixed Weighting ($\beta_t = 1$), Linear Decay ($1 \to 0$ over steps 1–300), and Staged Exp. Decay (initial weight 1 at step 1, halved every 25 steps) achieve 44.96%, 48.25%, and 51.25%, respectively, compared with 53.76% for HATCH. For both predefined schedules, accuracy begins to decline while $\beta_t > 0.1$, before the weight approaches zero.

Correcting update direction. No Projection and Reverse Projection obtain 51.93% and 49.24%, respectively, at their reported checkpoints. The directional mechanism is examined separately in Section 4.3 and Figure 6; the present table provides the benchmark breakdown of the ablations.

C.2 Annealing Parameters

The online coefficient in Eq. (7) depends on the current rollout composition rather than a prescribed end step. At a fixed composition, increasing γ lowers the unclipped weight, while κ sets its scale. Figure 6 compares $\gamma \in \{4, 5, 6\}$ with $\kappa = 5$ on Qwen3-8B. We select these hyperparameters using the held-out validation set. Table 3 reports the configurations used for the other backbones; the Qwen3-8B comparison does not establish a universally optimal exponent across model scales.

C.3 Per-Benchmark Training Curves

Figure 9 shows learning trajectories on the nine benchmarks. Improvements on Math500, Minerva Math, and OlympiadBench are visible across repeated evaluations, while the contest benchmarks exhibit larger checkpoint-to-checkpoint changes. This variation motivates reporting the full benchmark breakdown alongside the average. The figure uses one recorded run per method to show training behavior; it does not replace the multi-run comparison in the main results.

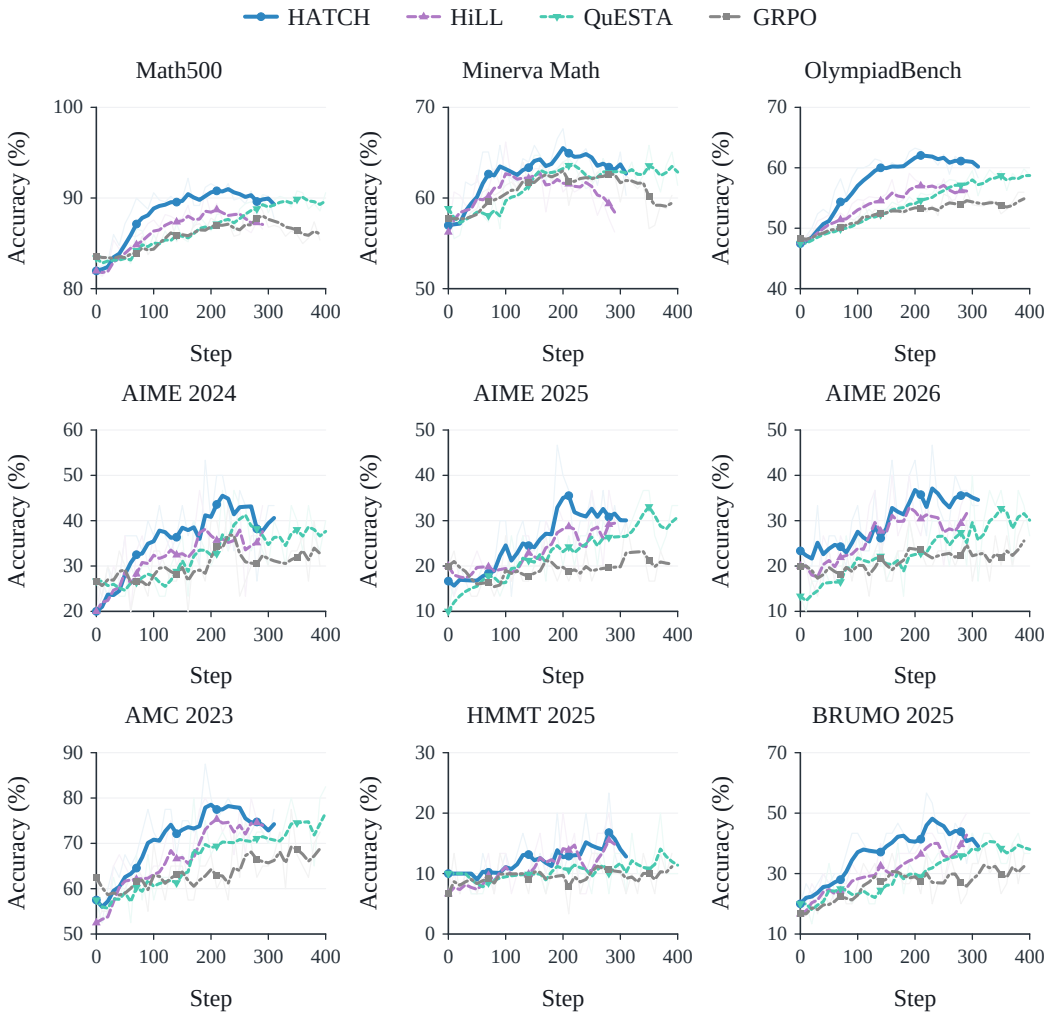


Figure 9: Per-benchmark no-hint accuracy through step 400 for individual Qwen3-8B training runs. Dark curves use exponential smoothing with weight 0.3; light curves show raw evaluations.

D Learning Dynamics

D.1 Reward Contrast on Failed Queries

The original-query group composition in the main paper measures changes in the policy's solving ability. Here we examine the groups created by hints for queries whose original responses are all incorrect. Figure 10(a) reports the fraction of hinted-solve groups that recover group-relative learning signal. A group is counted as recovered when its re-solves contain both correct and incorrect responses, restoring the reward contrast absent from its original rollouts. Averaging over the four hint indices and then over updates, theproportion is 25.1% during steps 1–50 and 14.9% during steps 251–300. The latter groups still provide learning signal despite being drawn from queries that remain solve-none under the original condition. Thus, assistance continues to recover informative comparisons on the policy's current failures, even as their frequency changes over training.

D.2 Learning from Re-Solving Outcomes

To inspect the hint-generation objective itself, Figure 10(c–d) tracks its mean reward and the fraction of hints flagged by the answer-leakage check. Between the same two training windows, mean reward increases from 0.065 to 0.082, while the flagged fraction falls from 17.7% to 0.7%. Meanwhile, Figure 10(b) shows that hinted re-solving accuracy falls from 13.6% to 4.5%. Higher reward and fewer flagged hints are consistent with learning the hint-construction objective.

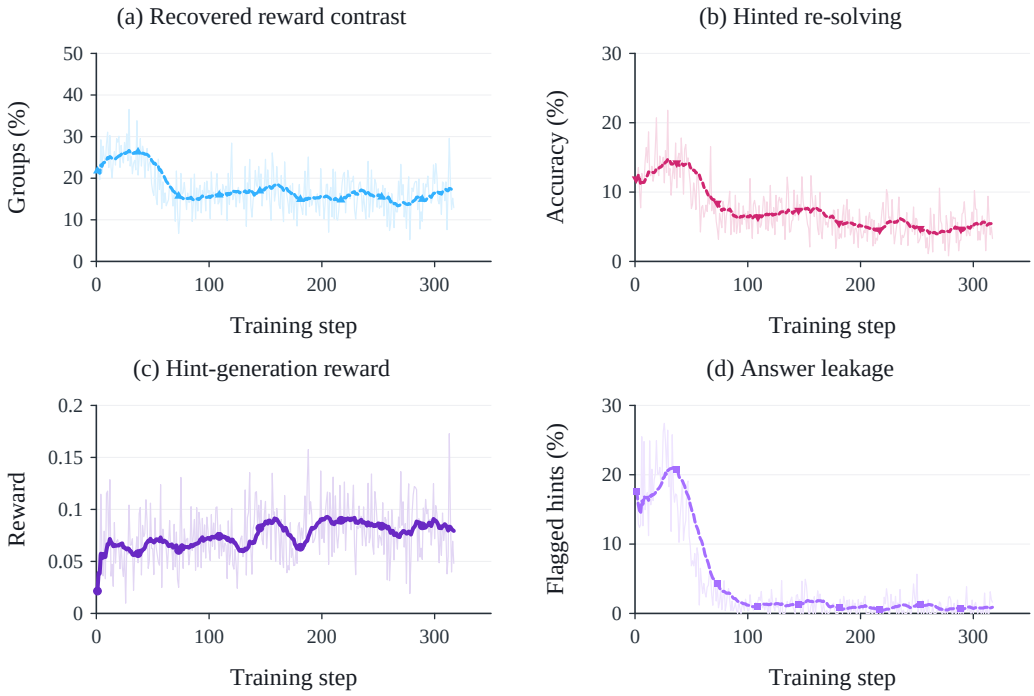


Figure 10: Hint-related training dynamics in the Qwen3-8B run. (a) Hinted groups with both correct and incorrect responses. (b) Hinted re-solving accuracy. (c) Mean hint-generation reward. (d) Hints flagged by the answer-leakage check. Panels (a–b) average the logged group statistics across four hint indices. Dark curves are trailing 20-step means; light curves are raw values.

E Computational Cost

All runs use four nodes with eight NVIDIA H20 GPUs per node. Figure 11(a) reports the time spent on original generation, the hint-assisted pipeline, and policy updates in a Qwen3-8B run. As training progresses, fewer queries require hint generation, as shown in Figure 11(b). The hint-assisted pipeline consequently takes less time, while original-generation time remains approximately

stable. These trends show that the auxiliary computational cost decreases as the policy becomes more capable of solving queries without hints.

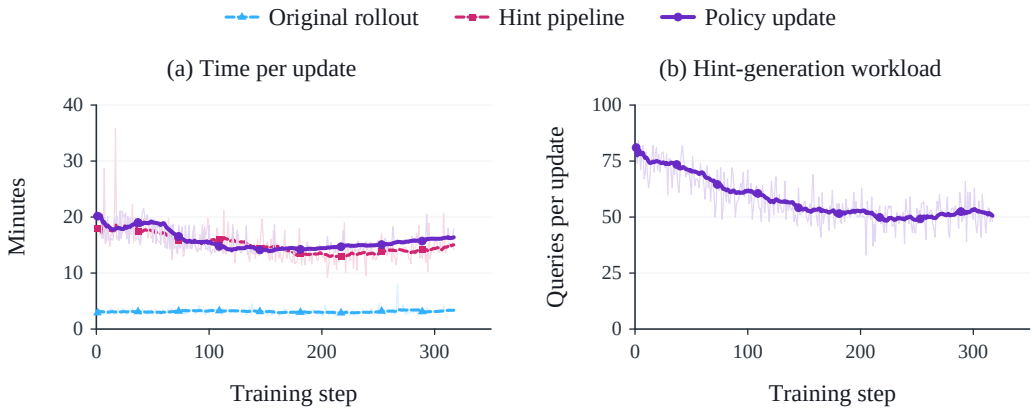


Figure 11: Time and workload in the Qwen3-8B run. Left: separately logged original-generation, hint-pipeline, and policy-update times. Right: queries entering hint generation per update. Dark curves are trailing 20-step means; light curves show raw observations.

F Assets and Licenses

Table 5 summarizes the licenses of models, training data, and software used in our experiments.

Table 5: Principal external assets and their licenses.

Asset	License
Llama-3.2-1B-Instruct	Llama 3.2 Community License
Qwen3-1.7B	Apache-2.0
Qwen3-8B	Apache-2.0
Qwen3-30B-A3B-Base	Apache-2.0
NuminaMath-1.5	Apache-2.0
verl	Apache-2.0
Megatron-LM	BSD-3-Clause; third-party licenses
vLLM	Apache-2.0