SELF-HINTING LANGUAGE MODELS ENHANCE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Group Relative Policy Optimization (GRPO) often stalls on hard prompts under sparse terminal rewards: identical rewards within a rollout group collapse relative advantages and eliminate updates. We propose *self-hint aligned GRPO with privileged supervision* (SAGE), an on-policy method that uses training-only hints compressed from reference solutions to reshape the rollout distribution. For each prompt x, the model samples a compact hint h and generates a solution τ conditioned on (x, h), while retaining the same verifier reward $R(x, \tau)$. At each training step, the scheduler starts without a hint and increases hint strength only if a rollout group has no correct response. This makes informative groups more likely under finite sampling and can restore learning signals on hard prompts. At test time, we deploy the no-hint policy with $h = \varnothing$. By refreshing self-hints as the learner changes, SAGE provides a better-calibrated curriculum than fixed hints from an initial policy or an external model. Across six benchmarks and three backbones, SAGE consistently outperforms GRPO, gaining up to 4.0 percentage points in average accuracy when trained on hard prompts.

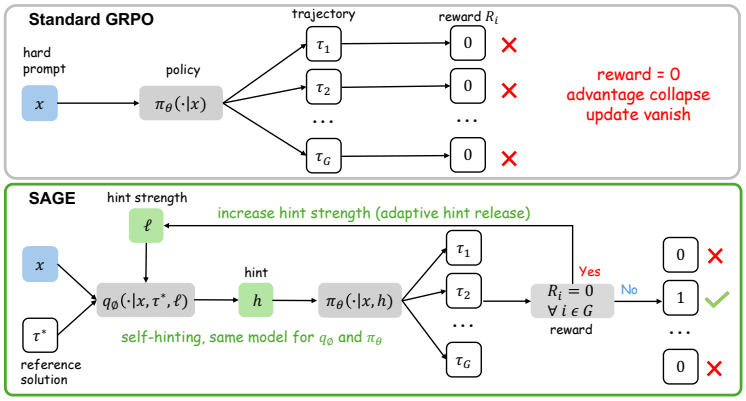

Figure 1: SAGE overview. For prompts where the LLM samples no correct trajectories, it **self-generates** a hint from the reference solution and conditions rollouts on the prompt and hint, mitigating advantage collapse and enabling policy updates. Hint strength starts at $\ell = 0$, where $h = \varnothing$; thus, GRPO is a special case of SAGE. For non-difficult prompts, SAGE falls back to GRPO.

1 INTRODUCTION

Reinforcement learning (RL) trains and aligns LLMs using verifiable objectives such as exact-match answers, unit tests, and automated checkers (Ouyang et al., 2022; Schulman et al., 2017; DeepSeek-AI et al., 2025). The goal is to maximize expected reward over prompts, yet finite-sample policy-gradient estimates can have high variance or vanish on hard prompts.

Group Relative Policy Optimization (GRPO) (Shao et al., 2024) centers, and often standardizes, rewards within rollout groups, requiring mixed outcomes for nonzero advantages under binary rewards. Under sparse $0/1$ rewards, difficult prompts often yield all-zero groups, collapsing the advantages and making the minibatch policy-gradient estimate identically zero. This **finite-sample**

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.

pathology leaves many prompts without a learning signal even when the expected objective is not flat.

Existing remedies modify data collection: skipping degenerate groups and resampling prompts biases training toward easier cases (Yu et al., 2025; Xiong et al., 2025a); adaptive sampling or curricula allocate more rollouts to difficult prompts (Yao et al., 2025; Xiong et al., 2025b; Li et al., 2025; Zhang et al., 2025c); and offline data or candidates from stronger models bootstrap learning (Zhang et al., 2025a; Yan et al., 2025; Zhang et al., 2025b). These approaches may improve training but skew its distribution or introduce context mismatch.

We propose SAGE (**S**elf-hint **A**ligned **G**RPO with Privil**E**ged Supervision), a complementary approach based on *privileged hinting*. During training, we provide an additional hint h, defined as a lossy compression of a reference solution $\tau^\star$, and sample rollouts from the hint-conditioned policy $\pi_\theta(\cdot \mid x, h)$. The hint does not alter the task reward; instead, it reshapes the rollout distribution so that finite rollout groups are more likely to contain mixed outcomes. At test time, hints are removed and the no-hint policy is deployed. We refer to hints generated by the policy itself as *self-hints*, and to the corresponding generation procedure as *self-hinting*.

The benefit of hinting follows from a simple Bernoulli model. Let $p_\theta(x)$ be the no-hint success probability for prompt x and G the group size. The probability that a rollout group contains both successful and unsuccessful samples is $1-(1-p_\theta(x))^G-p_\theta(x)^G \approx Gp_\theta(x)$, when $p_\theta(x)$ is small. Thus, GRPO updates vanish when $Gp_\theta(x) \ll 1$. Privileged hints help by increasing the effective success probability, making mixed-outcome groups more likely at the same group size.

Contributions. (i) We introduce SAGE, an on-policy RL framework that conditions training rollouts on privileged hints while keeping the task reward unchanged and removing hints at test time. (ii) We develop a policy-dependent hint-strength scheduler that activates hints when within-group rewards collapse, yielding an automatic curriculum. (iii) We propose an online self-hinting scheme that periodically refreshes the hint distribution to remain calibrated to the learner. (iv) We analyze GRPO collapse through the mixed-outcome probability under Bernoulli rewards, and empirically show that SAGE improves sample efficiency and final accuracy on challenging reasoning benchmarks, gaining up to 4.0 percentage points over GRPO on the hard training set and achieving the highest in-distribution average across three LLMs.

2 RL WITH PRIVILEGED HINTING

Vanilla GRPO works well when a prompt x yields occasional positive rollouts. In hard regimes, groups often receive identical rewards, collapsing within-group advantages and stalling learning. We address this by injecting *privileged hints* during training, keeping the reward unchanged while reshaping the rollout distribution to surface informative trajectories under finite sampling.

2.1 SETUP AND THE GRPO STALL

Let $x \sim \mathcal{D}$ be a prompt. A policy π_θ generates a trajectory $\tau = (y_1, \ldots, y_T)$, written as $\tau \sim \pi_\theta(\cdot \mid x)$. We use a binary reward $R(x,\tau) \in \{0,1\}$. Define the success probability $p_\theta(x) = \Pr_{\tau\sim\pi_\theta(\cdot|x)}[R(x,\tau)=1]$. GRPO implementations often standardize groupwise advantages by the within-group standard deviation. For a group of G rollouts $\{\tau_i\}_{i=1}^G$, let

$$R_i = R(x,\tau_i), \quad \bar{R} = \tfrac{1}{G}\sum_{i=1}^{G} R_i, \quad s^2 = \tfrac{1}{G}\sum_{i=1}^{G}(R_i-\bar{R})^2,$$

and define standardized advantages

$$A_i = (R_i - \bar{R})/(s+\epsilon), \tag{1}$$

where $\epsilon \geq 0$ is a numerical stabilizer and $R_i \in \{0,1\}$. When $p_\theta(x)$ is tiny, a group is often all-zero and $A_i = 0$ for all i. The chance of a non-degenerate group is

$$1-(1-p_\theta(x))^G - p_\theta(x)^G \approx 1-(1-p_\theta(x))^G \approx Gp_\theta(x),$$

[1]Here we train on 64k prompts, and sample 8 traces per prompt per step.

so training stalls whenever $Gp_\theta(x) \ll 1$ for most prompts. Figure 2 shows this in practice: for many hard prompts, correct trajectories are never sampled for long stretches, yielding no learning signal.

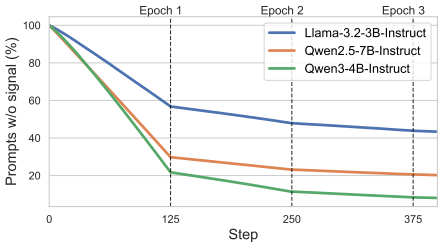

Figure 2: The percentage of prompts whose correct trajectories have NEVER been sampled w.r.t. the training step.[1] A large portion of prompts is wasted during RL due to all 0 rewards, especially for a weaker LLM.

2.2 Privileged hinting as sampling

When a reference trajectory $\tau^\star$ is available, we generate a hint h as a lossy compression of $\tau^\star$ and append it to the prompt.

Hint strength and the no-hint case. We control hint informativeness with a discrete strength level $\ell \in \{0, 1, \ldots, L\}$, where larger ℓ reveals more information about $\tau^\star$. We sample

$$\ell \sim p(\ell), \qquad h \sim q(h \mid x, \tau^\star, \ell), \tag{2}$$

where $\ell = 0$ is the no-hint setting and $q(h \mid x, \tau^\star, 0) = \delta_\varnothing(h)$, i.e., $h = \varnothing$.

Policy-dependent scheduling of ℓ. Hints are used only when a prompt provides no learning signal. We let ℓ depend on the policy through a simple statistic, such as a collapse indicator $c(x) = \mathbb{I}\big[\mathrm{Var}\big(\{R(x,\tau_i)\}_{i=1}^G\big) = 0\big]$, computed from a small probe group under π_θ. A minimal scheduler is as following, so $\ell > 0$ is activated only when the group collapses.

$$p(\ell \mid x) = \begin{cases} \delta_0, & c(x) = 0, \\ p(\ell), & c(x) = 1, \end{cases} \tag{3}$$

With a hint, we sample from $\pi_\theta(\cdot \mid x, h)$. The success rate increases: $p_\theta^{(\ell)}(x) = \Pr_{\tau\sim\pi_\theta(\cdot|x,h)}[R(x,\tau) = 1], h \sim q(\cdot \mid x, \tau^\star, \ell)$. Hinting is useful when it raises $p_\theta^{(\ell)}(x)$ enough that $Gp_\theta^{(\ell)}(x)$ is no longer tiny, making non-degenerate groups common and enabling RL updates.

2.3 GRPO with hints and the final loss

Given x, sample ℓ and h, then draw a group of rollouts

$$\tau_i \sim \pi_\theta(\cdot \mid x, h), \qquad R_i = R(x, \tau_i), \quad i = 1, \ldots, G.$$

Compute A_i with Eq. (1). The hint-conditioned loss is

$$\mathcal{L}(\theta) = -\mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} A_i \sum_{t=1}^{T_i} \log \pi_\theta(y_{i,t} \mid x, h, y_{i,<t})\right] + \beta\, \mathbb{E}\big[\mathrm{KL}\big(\pi_\theta(\cdot \mid x, h) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x, h)\big)\big], \tag{4}$$

where the expectation is over $x \sim \mathcal{D}$, hint sampling from Eq. (2), and rollouts from $\pi_\theta(\cdot \mid x, h)$. At test time, we set $\ell = 0$, $h = \varnothing$, and run $\pi_\theta(\cdot \mid x, \varnothing) \equiv \pi_\theta(\cdot \mid x)$.

Summary. Sparse rewards can stall GRPO because finite groups often contain no positive samples, causing advantage collapse. Privileged hinting avoids this by changing the rollout distribution while keeping the reward unchanged. A policy-dependent scheduler activates hints only on collapsed groups, yielding an automatic curriculum. Training remains on-policy since rollouts are drawn from $\pi_\theta(\cdot \mid x, h)$. Deployment uses $\ell = 0$ and requires no hints or privileged information.

3 Analysis: standardized GRPO as a gated update objective

Fix a context (x, h) and draw G rollouts with rewards $R_i \in \{0, 1\}$. Let $\bar{R} = \frac{1}{G}\sum_i R_i$ and $s^2 = \frac{1}{G}\sum_i (R_i - \bar{R})^2$. Standardized GRPO uses $A_i = \frac{R_i - \bar{R}}{s + \epsilon}$.

Corollary 3.1 (Signal energy equals a gate probability)**.** *Define the advantage energy* $E := \frac{1}{G}\sum_{i=1}^{G} A_i^2$. *If* $\epsilon > 0$,

$$E = \frac{s^2}{(s+\epsilon)^2} \in [0, 1], \tag{5}$$

which is monotone in s *and still collapses to* 0 *when* $s = 0$.

For GRPO, the prompt-level update magnitude is dominated by whether the group is *non-degenerate* ($s > 0$). In other words, training behaves like a gated procedure that updates only when the rollout group contains mixed outcomes.

Proposition 3.2 (Gate opening probability under Bernoulli rewards)**.** *Let* $p_\theta(x, h) = \Pr[R(x, \tau) = 1 \mid x, h]$. *Then*

$$\Pr[s > 0 \mid x, h] = 1 - (1 - p_\theta(x, h))^G - p_\theta(x, h)^G, \tag{6}$$

$\Pr[s > 0 \mid x, h]$ *is maximized at* $p_\theta = \frac{1}{2}$. *In the sparse regime* $p_\theta(x, h) \ll 1$, $\Pr[s > 0 \mid x, h] \approx G\, p_\theta(x, h)$.

Thus, SAGE should choose the hint strength to move hard prompts out of the regime where $Gp_\theta \ll 1$, and avoid overly strong hints that push $p_\theta \approx 1$ to close the gate again.

Proposition 3.3 (Optimal hint distribution is policy-dependent)**.** *Fix a prompt* x *and group size* $G \geq 2$. *Let* $p_\theta(x, h) = \Pr_{\tau \sim \pi_\theta(\cdot|x,h)}[R(x, \tau) = 1]$ *and define*

$$u(p) \coloneqq 1 - (1 - p)^G - p^G. \tag{7}$$

Under Bernoulli rewards, $u(p_\theta(x, h)) = \Pr[s > 0 \mid x, h]$.

For any distribution $q(\cdot \mid x)$, *define the expected probability*

$$J_x(\theta, q) \coloneqq \mathbb{E}_{h \sim q(\cdot|x)}[u(p_\theta(x, h))]\,. \tag{8}$$

Then u *is symmetric and strictly concave on* $[0, 1]$, *and is maximized at* $p = \frac{1}{2}$. *Consequently, any maximizer*

$$q^\star_\theta(\cdot \mid x) \in \arg\max_q J_x(\theta, q) \tag{9}$$

must place its mass on calibrating hints *that make* $p_\theta \approx \frac{1}{2}$.

In general, the set of calibrating hints depends on θ. *Unless* $p_\theta(x, h)$ *is invariant in* θ *for* q*-almost all* h, *a fixed* q *cannot remain (near) optimal for* $J_x(\theta, q)$ *throughout training. Updating* q *online therefore reduces this gate-mismatch and increases the frequency of non-degenerate GRPO updates.*

Remark 3.4 (Why not sample many hints per prompt.). Let $u(p) = 1 - (1 - p)^G - p^G$ denote the gate probability in Eq. (6). For $G \geq 2$, u is concave on $[0, 1]$ since $u''(p) = -G(G - 1)\big(p^{G-2} + (1 - p)^{G-2}\big) \leq 0$. Therefore, for any hint distribution $h \sim q$,

$$\mathbb{E}_{h \sim q}\big[u(p_\theta(x, h))\big] \leq u(\mathbb{E}_{h \sim q}[p_\theta(x, h)])\,. \tag{10}$$

At a fixed mean success rate, additional randomness across hints can only reduce the expected frequency of non-degenerate groups. Motivated by Eq. (10), we sample a single hint realization per prompt per epoch and spend compute on G or better strength scheduling.

Overall, standardized GRPO turns sparse-reward learning into maximizing a gate probability. The proposed method should operationalize this by (i) ensuring on-policy conditioning on h, (ii) scheduling ℓ when the gate is closed, (iii) updating the self-hint generator online to keep p_θ calibrated.

4 Design of SAGE

4.1 On-policy training

Why hints must be in the conditioning context? SAGE appends a hint h to the prompt and samples rollouts from the hint-conditioned policy $\pi_\theta(\cdot \mid x, h)$. This keeps training *on-policy* for the augmented context. The loss is $\sum_t \log \pi_\theta(y_t \mid x, h, y_{<t})$. If one instead samples $\tau \sim \pi_\theta(\cdot \mid x, h)$ but evaluates $\log \pi_\theta(\tau \mid x)$, i.e., drops h inside the log-probability, the update no longer corresponds to the gradient of an on-policy objective under the sampling process. In practice, this mismatch behaves like an off-policy update and is less stable. We include a study in Sec. 5.2 for this setting.

[2]The 4.5k training prompts are extremely hard; correct trajectories have never been sampled during training (Figure 2). The number of rollouts per prompt per step here is set to 32 to encourage exploration.

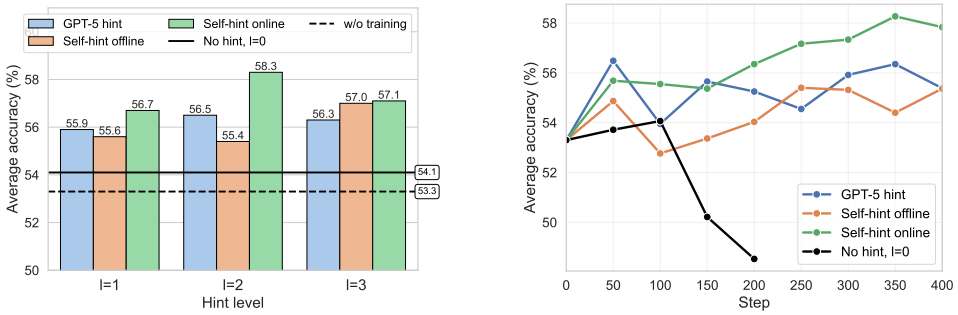

Figure 3: Average accuracy over 6 benchmarks of Qwen3-4B-Instruct trained on hard prompts.[2] **Left**: Performance on various hints. Hinting improves performance over no-hint training, with online self-hinting performing the best. **Right**: Average accuracy w.r.t. the training steps for hint level $\ell = 2$. No-hint training degrades over time under sparse rewards, while online self-hinting improves steadily.

4.2 Online Self-Hinting

We produce privileged hints in two ways: (1) **Offline hints.** A fixed hint generator, e.g., extracted once from $\tau^\star$, is simple but may become miscalibrated as the learner evolves. (2) **Online hints.** We periodically refresh the hint generator using a copy of the current policy π_θ. In SAGE, hint generation is *online*: we implement $q_\phi(h \mid x, \tau^\star, \ell)$ by prompting π_θ to produce a *procedure-only* plan aligned with the reference trajectory.

We evaluate three variants: (1) **Fixed privileged hints:** q_ϕ is derived from π_{θ_0} and frozen after initialization. (2) **Online privileged hints (SAGE):** q_ϕ is derived from π_θ and refreshed during training. (3) **External teacher hints:** q_ϕ is produced by a stronger frozen model, when available.

Figure 3 shows that hints improve performance across levels compared with no hint, consistent with increasing the chance of informative trajectories under sparse rewards. Online self-hinting performs best, suggesting that refreshed self-hints are better calibrated to the learner than fixed hints.

4.3 Policy-Dependent Scheduling

We control hint informativeness with a discrete strength variable $\ell \in \{0, 1, \ldots, L\}$, where $\ell = 0$ is the deployable no-hint setting. The scheduler is *policy-dependent*: it adapts ℓ using recent rollout statistics under π_θ with stop-gradient. Hint strength rises when success is too rare and can fall as the policy improves.

Scheme 1 (SAGE-LIGHT): epoch-level accuracy threshold. Let $\hat{p}_{t-1}(x)$ be the empirical success rate of prompt x in the previous epoch under its then-current hint level. Given a threshold $\alpha \in (0, \frac{1}{2})$, we increase hint strength when success is rare and decrease it when success exceeds one half:

$$\ell_t(x) = \begin{cases} \min\{\ell_{t-1}(x) + 1, L\}, & \hat{p}_{t-1}(x) < \alpha, \\ \max\{\ell_{t-1}(x) - 1, 0\}, & \hat{p}_{t-1}(x) > \frac{1}{2}, \\ \ell_{t-1}(x), & \text{otherwise.} \end{cases} \tag{11}$$

Thus, the level can move in either direction as the policy improves or the prompt remains difficult.

Scheme 2 (SAGE): group-degeneracy trigger. GRPO requires within-group outcome differences to produce nonzero updates. At each training step t, we restart at $\ell = 0$ and probe successively stronger hints until a group contains a positive sample or $\ell = L$. At each level ℓ, sample a hint $h_t^{(\ell)}$ and a group $\{\tau_{t,i}^{(\ell)}\}_{i=1}^G$ from $\pi_{\theta_t}(\cdot \mid x, h_t^{(\ell)})$, and define $z_t^{(\ell)}(x) = \mathbb{I}\left[\sum_{i=1}^G R(x, \tau_{t,i}^{(\ell)}) = 0\right]$. The level used for the update is the first probed level with a positive rollout, or L if none succeeds:

$$\ell_t(x) = \min\Big(\big\{\ell \in \{0, \ldots, L\} : z_t^{(\ell)}(x) = 0\big\} \cup \{L\}\Big). \tag{12}$$

No hint level is carried over to the next training step; as the policy improves, the same prompt can be updated at a lower level. This targets sparse-reward collapse: when a group has no positives, standardized GRPO advantages collapse and the policy-gradient estimator vanishes.

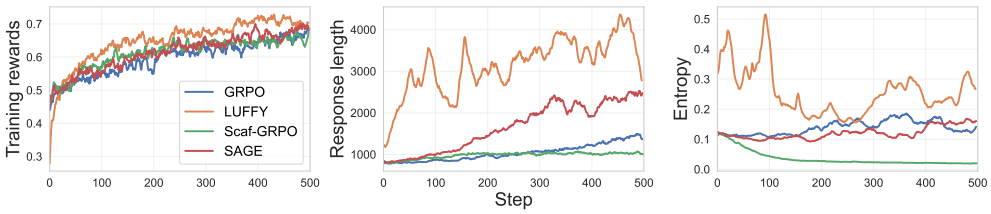

Figure 4: Training dynamics on Qwen2.5-7B-Instruct.[4]For rewards, we focus on trends rather than values, since SAGE, Scaf-GRPO, and LUFFY modify prompt difficulty. LUFFY is unstable, with low initial reward and oscillating length/entropy, likely due to off-policy imitation. Scaf-GRPO has the lowest entropy, suggesting less exploration. SAGE preserves GRPO's on-policy property, maintains moderate entropy, and shows stable response-length growth, indicating improved reasoning.

Discussion. SAGE-LIGHT is compute-efficient because it updates hint strength only at the epoch level without extra rollouts, but it may react slowly to reward collapse. SAGE is more reactive and directly targets GRPO's failure mode via a no-positives trigger, at the cost of additional probe rollouts. We report both schemes and find that the no-positives trigger usually recovers faster from stalled training on hard prompts and performs better. Algorithm 1 in Appendix F summarizes the SAGE implementation. At each probed level, all G rollouts for a prompt share *one* sampled hint. This design reduces unnecessary variance from hint sampling.

5 Empirical Results

Models. We evaluate SAGE on three LLMs with varying math specialization: Llama-3.2-3B-Instruct (Meta, 2024), Qwen2.5-7B-Instruct (Yang et al., 2024), and Qwen3-4B-Instruct-2507 (Yang et al., 2025).

Baselines. We compare SAGE with: (1) Supervised Fine-Tuning (SFT), which finetunes on DeepSeek-R1 reasoning traces; (2) GRPO (Shao et al., 2024), which uses no hints; (3) LUFFY (Yan et al., 2025), which replaces one on-policy trajectory with the corresponding correct DeepSeek-R1 trajectory; and (4) Scaf-GRPO (Zhang et al., 2025b), which uses hints generated by GPT-5.2. SFT, LUFFY, and Scaf-GRPO rely on a stronger external LLM, whereas SAGE learns only from self-generated hints. For fairness, we reproduce LUFFY and Scaf-GRPO with their open-source implementations on the same training set.[3]

Training sets. We sample training prompts from OpenR1-Math-220k (Hugging Face, 2025), which combines NuminaMath 1.5 prompts (Li et al., 2024) with DeepSeek-R1 reasoning traces (DeepSeek-AI et al., 2025). After filtering with *Math-Verify* (Kydlíček), we obtain 64k verifiable prompts. For resource-efficient RL training, we further select 15k prompts with reasoning traces shorter than 8,192 tokens, as required by LUFFY. We call this the **easy set**: it is not pass-rate filtered and spans diverse difficulty levels. To construct a **hard set** for each LLM, we remove prompts from the easy set with $pass@10 \geq 0.5$ under that model. The easy set includes easier prompts that provide GRPO with learning signals; the hard set tests how GRPO, LUFFY, Scaf-GRPO, and SAGE learn from more challenging prompts. Unless otherwise specified, all LLMs are trained on the easy set.

Evaluation sets. We primarily evaluate on six mathematical benchmarks: AIME24 (MAA Committees, 2024), AIME25 (MAA Committees, 2025), AMC23 (Li et al., 2024), MATH-500 (Hendrycks et al., 2021), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). We also include two non-mathematical benchmarks, GPQA-diamond (Rein et al., 2024) and MMLU-Pro (Wang et al., 2024), to assess generalization. **Hints are used only during training; evaluation uses the prompt alone.**

Implementation details. We set the max hint strength $L = 3$, and $\alpha = 0.35$ for SAGE-LIGHT. The prompts used for hint generation and injection are in Appendix E. The complete training and evaluation settings are shown in Appendix G.

[3]Only align the batch size and training steps.

[4]Please refer to Figure 8 for the training dynamics of all 3 LLMs.

Table 1: Accuracy on in-distribution and out-of-distribution tasks across three LLMs trained on the **easy training set**. The **best** and second-best results are in bold and underlined, respectively. SAGE and SAGE-LIGHT consistently outperform baselines on average across various LLMs.

Method	In-distribution							Out-of-distribution			
	AIME24 / 25	AMC23	MATH-500	Minerva	Olympiad	Avg.	Δ	GPQA	MMLU-Pro	Avg.	Δ
Llama-3.2-3B-Instruct	6.5 / 0.6	22.8	44.7	17.8	14.2	17.8	0	17.9	27.0	22.5	0
SFT	0.4 / 0.6	9.5	26.9	5.1	6.5	8.2	−9.6	11.6	18.8	15.2	−7.3
GRPO	6.7 / 0.8	29.5	52.1	20.5	21.8	21.9	+4.1	26.3	39.8	33.1	+10.6
LUFFY	4.4 / 0.4	18.6	38.9	14.3	11.9	14.7	−3.1	16.0	26.7	21.4	−1.1
Scaf-GRPO	7.7 / **2.3**	28.8	51.7	19.4	19.5	21.5	+3.7	24.1	38.0	31.0	+8.5
SAGE-LIGHT	8.8 / 1.9	32.2	54.1	**20.8**	20.1	23.0	+5.2	26.8	39.6	33.2	+10.7
SAGE	**9.2** / 0.8	**34.7**	**56.3**	20.1	**22.0**	**23.9**	**+6.1**	**27.3**	**40.7**	**34.0**	**+11.5**
Qwen2.5-7B-Instruct	13.8 / 6.7	53.4	75.7	38.1	39.2	37.8	0	37.1	56.4	46.7	0
SFT	3.5 / 7.1	30.9	56.2	20.0	21.7	23.2	−14.6	9.5	35.6	22.5	−24.2
GRPO	15.0 / **13.5**	55.5	79.2	39.1	44.5	41.1	+3.3	37.2	57.6	47.4	+0.7
LUFFY	**17.1 / 13.5**	55.2	**81.3**	39.0	44.2	41.7	+3.9	**38.1**	59.1	**48.6**	**+1.9**
Scaf-GRPO	14.6 / 12.7	58.8	78.0	**39.8**	42.0	41.0	+3.2	36.6	58.4	47.5	+0.8
SAGE-LIGHT	**17.1** / 11.7	58.1	79.9	38.6	**46.1**	41.9	+4.1	36.6	58.8	47.7	+1.0
SAGE	16.0 / 12.5	**60.3**	80.0	39.3	45.9	**42.3**	**+4.5**	38.0	**59.3**	**48.6**	**+1.9**
Qwen3-4B-Instruct	52.1 / 43.1	92.2	93.6	46.1	67.7	65.8	0	57.6	70.9	64.3	0
SFT	14.4 / 22.1	55.2	78.9	38.1	40.2	41.5	−24.3	29.4	52.6	41.0	−23.3
GRPO	55.8 / 45.0	**95.0**	96.0	**50.1**	70.4	68.7	+2.9	57.0	72.0	64.5	+0.2
LUFFY	42.3 / 36.0	86.4	91.1	48.2	59.4	60.6	−5.2	31.9	46.6	39.3	−25.0
Scaf-GRPO	**59.8** / 45.2	92.2	95.1	48.9	69.8	68.5	+2.7	54.3	72.1	63.2	−1.1
SAGE-LIGHT	59.2 / 47.1	92.2	95.1	49.5	70.5	68.9	+3.1	57.1	72.0	64.5	+0.2
SAGE	58.1 / **52.1**	94.2	**95.4**	49.2	**71.2**	**70.0**	**+4.2**	**57.8**	**72.5**	**65.2**	**+0.9**

Table 2: Accuracy of LLMs trained on the **hard training set**.

Method	In-distribution							Out-of-distribution			
	AIME24 / 25	AMC23	MATH-500	Minerva	Olympiad	Avg.	Δ	GPQA	MMLU-Pro	Avg.	Δ
Llama-3.2-3B-Instruct	6.5 / 0.6	22.8	44.7	17.8	14.2	17.8	0	17.9	27.0	22.5	0
GRPO	7.5 / 1.7	23.8	51.5	18.8	19.0	20.4	+2.6	21.8	36.5	29.2	+6.7
LUFFY	1.7 / 0.0	13.1	32.2	11.2	8.3	11.1	−6.7	11.4	23.5	17.4	−5.1
Scaf-GRPO	6.7 / 1.7	33.1	50.2	18.8	18.8	21.5	+3.7	22.7	37.9	30.3	+7.8
SAGE	**10.8 / 3.3**	**33.8**	**54.1**	**20.2**	**21.5**	**23.9**	**+6.1**	**27.9**	**38.9**	**33.4**	**+10.9**
Qwen2.5-7B-Instruct	13.8 / 6.7	53.4	75.7	38.1	39.2	37.8	0	37.1	56.4	46.7	0
GRPO	10.0 / 11.7	51.9	77.0	37.2	42.6	38.4	+0.6	35.2	57.8	46.5	-0.2
LUFFY	15.0 / 14.2	51.3	77.7	38.6	40.0	39.5	+1.7	36.6	57.1	46.9	+0.2
Scaf-GRPO	11.7 / 10.8	50.6	78.0	39.4	41.6	38.7	+0.9	37.9	58.7	48.3	+1.6
SAGE	**15.8 / 16.7**	**56.9**	**80.4**	**40.3**	**44.4**	**42.4**	**+4.6**	**39.8**	**59.2**	**49.5**	**+2.8**
Qwen3-4B-Instruct	52.1 / 43.1	**92.2**	93.6	46.1	67.7	65.8	0	57.6	70.9	64.3	0
GRPO	51.7 / 41.7	88.1	92.5	**50.4**	67.4	65.3	-0.5	57.8	71.9	64.9	+0.6
LUFFY	50.8 / 45.0	88.8	93.6	49.1	69.9	66.2	+0.4	57.5	**73.2**	65.3	+1.0
Scaf-GRPO	52.5 / **46.5**	90.0	93.3	49.4	69.4	66.8	+1.0	57.1	72.1	64.6	+0.3
SAGE	**57.5** / 45.8	91.3	**95.2**	49.5	**70.8**	**68.3**	**+2.5**	**60.7**	**73.2**	**66.9**	**+2.6**

5.1 MAIN RESULTS

Easy training set. We report the results of all methods and LLMs on eight benchmarks in Table 1, with the corresponding training dynamics shown in Figure 4. Overall, SAGE consistently achieves the highest average performance among all baselines, yielding average improvements of +6.1, +4.5, and +4.2 over the base LLMs for Llama-3.2, Qwen2.5, and Qwen3, respectively.

Compared with GRPO, SAGE achieves an average improvement of +1.2–2.0. LUFFY exhibits substantially different behavior across LLMs, likely due to its off-policy training setup. Specifically, LUFFY replaces one on-policy trajectory with a trajectory generated by a teacher model. When there is a large distribution gap between the learner and the teacher model, this design can destabilize training, as shown in Figure 4. Scaf-GRPO uses a teacher model, GPT-5.2, to generate hints. Although this strategy is effective, it still underperforms SAGE. We hypothesize that the online self-hinting mechanism in SAGE produces hints that are better calibrated to the learner, whereas hints generated by a stronger teacher may provide overly explicit guidance and thereby limit the model's exploration. Consequently, Scaf-GRPO exhibits lower entropy, as shown in Figure 4.

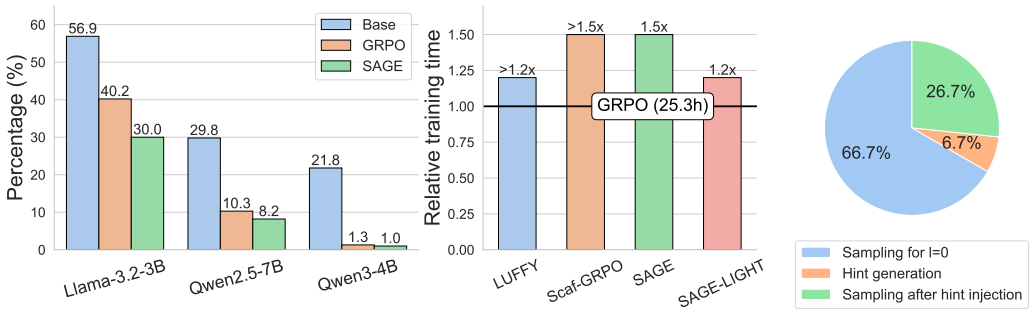

Figure 5: **Left**: Prompts without any learning signal. I.e., rewards are always 0 during training. **Middle**: Relative training time on Qwen2.5-7B-Instruct. Here, we don't take the extra time into account (the time used for trajectory generation from DeepSeek-R1 for LUFFY and hint generation from GPT-5.2 for Scaf-GRPO). **Right**: Training time decomposition for SAGE.

SAGE-LIGHT achieves slightly lower accuracy than SAGE but improves efficiency, requiring 80% of SAGE's training time.

Notably, we use the same fixed easy training set for all LLMs, despite their different levels of optimization for mathematical reasoning. This training set is favorable to GRPO because it contains fewer hard prompts. However, as shown in Figure 5 (Left), incorporating hard prompts improves RL performance. For example, on Qwen3, SAGE uses only 0.3% more prompts than GRPO, yet achieves an average accuracy improvement of +1.3 points. This suggests that hard prompts encourage the model to perform more extensive reasoning.

Hard training set. Table 2 compares SAGE with GRPO, LUFFY, and Scaf-GRPO across all three backbones on the hard prompt set. SAGE achieves the highest in-distribution and out-of-distribution average for each backbone. Relative to GRPO, its six-benchmark in-distribution average is higher by 3.5, 4.0, and 3.0 percentage points on Llama-3.2, Qwen2.5, and Qwen3, respectively. Its out-of-distribution average is also higher on all three backbones. Many hard prompts yield all-zero rollout rewards under GRPO, providing no within-group advantage for those updates. SAGE generates hints for such prompts to increase the chance of sampling mixed outcomes. These results support using SAGE when training emphasizes hard prompts.

OOD tasks. In Tables 1 and 2, SAGE attains the highest or joint-highest out-of-distribution average across all three backbones on both training sets, indicating that the gains extend beyond the mathematical training distribution.

Table 3: Performance on open-ended tasks.

Method	WB	AE2	CWv3
Qwen2.5-7B-Instruct	22.2	37.1	49.8
GRPO (Bhaskar et al., 2025)	36.8	39.3	74.6
SAGE	**38.2**	**41.5**	**75.3**

Open-ended tasks. We also evaluate WildBench (WB) (Lin et al., 2024), AlpacaEval2 (AE2) (Dubois et al., 2024), and CreativeWritingV3 (CWv3) (Paech, 2025). Following the RLMT protocol (Bhaskar et al., 2025), SAGE outperforms GRPO on all three (Table 3). Appendix G.3 gives the training and evaluation settings.

5.2 Discussion

Latency. Figure 5 (Middle) shows that SAGE has training time comparable to baselines, with SAGE-LIGHT faster than SAGE. The extra $0.5\times$ cost over GRPO mainly comes from second-turn sampling after hint injection. Figure 5 (Right) decomposes SAGE into three stages: (1) sampling with $\ell = 0$, equivalent to GRPO; (2) hint generation for prompts with no positive rewards, which is efficient since hints are short (<512 tokens); and (3) sampling with hints, which dominates the extra cost. SAGE-LIGHT uses the same stages but is more efficient in the third stage, since hint levels are determined from the previous epoch and do not require worst-case sampling over three hint levels. The two variants offer flexible efficiency trade-offs and both consistently outperform baselines.

Offline with more hints. In Figure 3, online self-hinting refreshes hints during training, whereas offline self-hinting uses fixed pre-generated hints. To test whether online gains come mainly from hint diversity, we add an offline variant with multiple hints in Figure 6. Before training, the base

LLM generates 10 hints with `temperature=1.0`. During training, each prompt uses different hints across steps. This improves over standard offline self-hinting by +0.9 (56.3 vs. 55.4), but still trails online self-hinting by 2.0 points (58.3 vs. 56.3). This suggests that online hinting helps beyond diversity by better calibrating hints to the learner during training.

Off-policy. We ablate whether the policy-gradient matches the sampling context: SAGE optimizes $\log \pi_\theta(\cdot \mid x, h)$, while an off-policy variant rolls out with h but optimizes $\log \pi_\theta(\cdot \mid x)$. Figure 6 shows a clear drop for the off-policy variant (56.5) compared with on-policy SAGE (59.2) and even the single level hint level baseline (58.3).

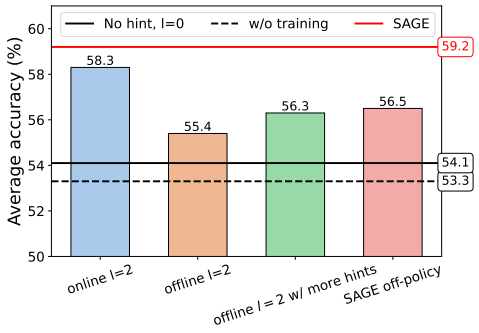

Figure 6: Ablation studies on Qwen3-4B trained on the same prompts as Figure 3.

Same level hint vs. SAGE. SAGE does not rely on a fixed hint level. Instead, it adaptively increases the hint strength only when the current level fails to yield a correct response. In Figure 6, this strategy leads to a clear performance gain over using a constant hint level (online $l = 2$), achieving an improvement of +0.9 (59.2 vs 58.3). This design enables more effective utilization of the hard prompt, allowing the LLM to progressively learn from weaker hints, down to $l = 0$.

Less hint w.r.t. step. In Figure 7, we can observe that the LLM uses fewer and fewer hints during the training. It indicates that self-hinting indeed enhances RL. The LLM becomes increasingly powerful and can gradually solve difficult problems without the help of a hint.

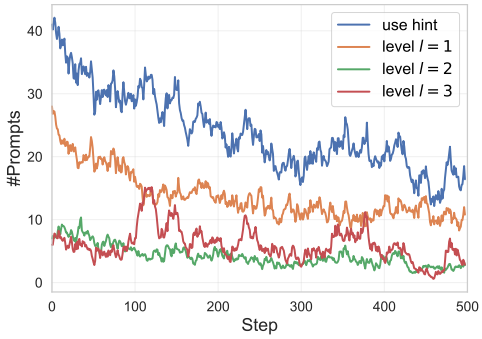

Figure 7: Number of prompts using hint during training on Llama-3.2-3B-Instruct. The model uses less hint w.r.t. the training step, indicating that the model becomes more powerful.

More results. Appendix H.3 illustrates how hints help. Higher maximum hint strength L improves results at greater training cost (Appendix H.4), and SAGE outperforms HINT, GHPO, and SFT+GRPO (Appendix H.5). DeepSeek-R1 reference trajectories also work; we assess suboptimal trajectories in Appendix H.6. Answer-only hints work for coding (Appendix H.7), and results are stable across training seeds (Appendix H.1).

6 Related Work

Prior work addresses sparse-reward collapse through prompt resampling or adaptive sampling (Yu et al., 2025; Yao et al., 2025; Li et al., 2025), and through teacher or offline trajectories (Zhang et al., 2025a; Yan et al., 2025). These approaches can bias the training distribution or introduce context mismatch. Hint-based methods use partial solutions or external scaffolds (Liu et al., 2025b; Wang et al., 2025; Zhang et al., 2025b); HiPO (Qiyuan et al., 2026) uses self-hints from rare successful trajectories. SAGE addresses prompts with no sampled success by generating self-hints online from reference solutions, adapting hint strength to group collapse, and updating on the same hinted context used for rollouts. See Appendix A for details and Appendix B for limitations.

7 Conclusion

We identify a finite-sample degeneracy in GRPO under sparse 0/1 rewards. When group rewards are identical, advantage standardization collapse, and the minibatch gradient vanishes on hard prompts. We propose SAGE, a privileged procedural hinting method that injects reference-solution-derived hints during training to shift the rollout distribution while preserving the original reward definition. A policy-dependent schedule gates hint strength based on detected group collapse, and inference uses the no-hint policy. Extensive experiments validate the improvements across tasks.

AI USE STATEMENT

Generative AI tools assisted with editing the manuscript and converting its LaTeX layout to the ICLR format. The authors remain responsible for the paper's claims, proofs, references, and reported results.

REFERENCES

Adithya Bhaskar, Xi Ye, and Danqi Chen. Language models that think, chat better. *arXiv preprint arXiv:2509.20357*, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, KaShun SHUM, and Tong Zhang. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=m7p5O7zblY.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*, 2024.

Yunzhen Feng, Ariel Kwiatkowski, Kunhao Zheng, Julia Kempe, and Yaqi Duan. Pilaf: Optimal human preference sampling for reward modeling. *arXiv preprint arXiv:2502.04270*, 2025.

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

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

Jian Hu, Xibin Wu, Zilin Zhu, Xianyu, Weixun Wang, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Hsiu-Yuan Huang, Chenming Tang, Weijie Liu, Clive Bai, Saiyong Yang, and Yunfang Wu. Think outside the policy: In-context steered policy optimization. *arXiv preprint arXiv:2510.26519*, 2025.

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

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

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.

Hynek Kydlíček. Math-Verify: Math Verification Library. URL `https://github.com/huggingface/math-verify`.

Junghyun Lee, Branislav Kveton, Anup Rao, Subhojyoti Mukherjee, Ryan A Rossi, Sunav Choudhary, and Alexa Siu. Learning to reason in llms by expectation maximization. *arXiv preprint arXiv:2512.20169*, 2025.

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

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

Ziniu Li, Congliang Chen, Tianyun Yang, Tian Ding, Ruoyu Sun, Ge Zhang, Wenhao Huang, and Zhi-Quan Luo. Knapsack rl: Unlocking exploration of llms via optimizing budget allocation. *arXiv preprint arXiv:2509.25849*, 2025.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking llms with challenging tasks from real users in the wild. *arXiv preprint arXiv:2406.04770*, 2024.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*, 2024.

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

Ziru Liu, Cheng Gong, Xinyu Fu, Yaofang Liu, Ran Chen, Shoubo Hu, Suiyun Zhang, Rui Liu, Qingfu Zhang, and Dandan Tu. Ghpo: Adaptive guidance for stable and efficient llm reinforcement learning. *arXiv preprint arXiv:2507.10628*, 2025b.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, et al. Deepcoder: A fully open-source 14b coder at o3-mini level. *Notion Blog*, 1, 2025.

MAA Committees. AIME Problems and Solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions, 2024.

MAA Committees. AIME Problems and Solutions. https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Meta. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI Blog*, 2024. https://ai.meta.com/blog/meta-llama-3/.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Samuel J. Paech. EQ-Bench Creative Writing Benchmark v3. https://github.com/EQ-bench/creative-writing-bench, 2025. Accessed: 2026-05-05.

Deng Qiyuan, Kehai Chen, Min Zhang, and Zhongwen Xu. Hipo: Self-hint policy optimization for rlvr. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

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, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. 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. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Ruizhe Shi, Runlong Zhou, and Simon S Du. The crucial role of samplers in online direct preference optimization. *arXiv preprint arXiv:2409.19605*, 2024.

Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.

Xinyi Wang, Jinyi Han, Zishang Jiang, Tingyun Li, Jiaqing Liang, Sihang Jiang, Zhaoqian Dai, Shuguang Ma, Fei Yu, and Yanghua Xiao. Hint: Helping ineffective rollouts navigate towards effectiveness. *arXiv preprint arXiv:2510.09388*, 2025.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. 2023.

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, et al. A minimalist approach to llm reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025a.

Wei Xiong, Chenlu Ye, Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, Jiang Bian, Nan Jiang, and Tong Zhang. Reinforce-ada: An adaptive sampling framework under non-linear rl objectives. *arXiv preprint arXiv:2510.04996*, 2025b.

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

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jiarui Yao, Yifan Hao, Hanning Zhang, Hanze Dong, Wei Xiong, Nan Jiang, and Tong Zhang. Optimizing chain-of-thought reasoners via gradient variance minimization in rejection sampling and rl. *arXiv preprint arXiv:2505.02391*, 2025.

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

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

Chuheng Zhang, Wei Shen, Li Zhao, Xuyun Zhang, Lianyong Qi, Wanchun Dou, and Jiang Bian. Policy filtration in rlhf to fine-tune llm for code generation.

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

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

Yuheng Zhang, Wenlin Yao, Changlong Yu, Yao Liu, Qingyu Yin, Bing Yin, Hyokun Yun, and Lihong Li. Improving sampling efficiency in rlvr through adaptive rollout and response reuse. *arXiv preprint arXiv:2509.25808*, 2025c.

A Detailed related work

Data resampling and external guidance. Data selection and filtering are widely used in online RL for LLMs (Zhang et al.; Dong et al., 2023; Xiong et al., 2023; Dong et al., 2024; Shi et al., 2024; Liao et al., 2025; Feng et al., 2025), and become particularly important for GRPO-style methods where groupwise advantages can collapse under sparse rewards. Prior work mitigates this issue mainly by reshaping the training distribution or injecting external guidance. A common workaround is to skip degenerate groups and resample or upweight prompts (Yu et al., 2025; Xiong et al., 2025a; Yao et al., 2025; Li et al., 2025), which improves efficiency but biases training toward prompts with non-trivial success probability. Another direction bootstraps learning by adding positive trajectories from stronger teachers, reference models, or offline buffers (Zhang et al., 2025a; Yan et al., 2025; Huang et al., 2025), but this can introduce context or distribution mismatch when mixed with on-policy rollouts. In contrast, SAGE preserves a clean on-policy objective by changing the rollout distribution through privileged hinting, without discarding hard prompts or relying on static external buffers.

Privileged hinting and SAGE. While leveraging intermediate guidance, such as plans or gold solutions, has a rich history in RL (Ng et al., 1999; Szepesvári, 2022; Ouyang et al., 2022), recent LLM-specific adaptations often implement hinting via heuristic "batch surgery." For instance, Zhang et al. (2025b) mitigates collapse by augmenting rollout batches with hinted trajectories upon detecting failure. This approach mixes contexts (e.g., x and x, h) within a single group, which blurs the interpretation of groupwise baselines and advantage normalization. Furthermore, external hint generators may not be calibrated to the learner's current capabilities. SAGE distinguishes itself through three key design choices: (1) it formalizes hinting as an explicitly augmented on-policy sampling process, ensuring the GRPO loss remains well-defined; (2) it employs a policy-dependent strength scheduler that activates hints only when the "learning gate" is closed; and (3) it utilizes online self-hinting via a lagged policy, ensuring the hint distribution tracks the learner's evolving support rather than relying on a potentially misaligned external teacher.

Self-generated rationales and hints. Prior work has explored using intermediate reasoning artifacts to improve LLM reasoning (Lee et al., 2025). STaR (Zelikman et al., 2022) bootstraps reasoning by generating rationales, filtering correct ones, and finetuning on them. Quiet-STaR (Zelikman et al., 2024) learns latent rationales for language modeling, while Self-Refine (Madaan et al., 2023) improves outputs through self-feedback and revision mainly at inference time. More recent RLVR methods use hints or external guidance to recover learning signal on hard prompts. GHPO (Liu et al., 2025b) appends truncated ground-truth solutions as hints, HINT (Wang et al., 2025) uses teacher-generated core insights, and Scaf-GRPO (Zhang et al., 2025b) relies on stronger-model scaffolds. LUFFY (Yan et al., 2025) is related but differs from hinting methods, since it injects an off-policy correct trajectory rather than a compact hint. HiPO (Qiyuan et al., 2026) is closest in spirit to SAGE, as it also uses self-hints for RLVR by repurposing rare successful trajectories as guidance. In contrast, SAGE targets collapsed hard prompts where successful trajectories may not be sampled at all. It generates online self-hints as privileged training-time context, adaptively schedules multiple hint-strength levels according to rollout collapse, and conditions both rollouts and log-probabilities on the hinted context. Hints are not supervised targets and are removed at test time. Thus, SAGE treats hinting as policy-calibrated exploration: weak hints are used when sufficient, stronger hints are activated only when the learning signal collapses, and overly fixed scaffolds are avoided.

B Limitation

SAGE is designed for settings where training-time supervision is available, such as reference solutions, reasoning trajectories, or final answers. This matches many standard RLVR, math, and coding benchmarks, and our experiments show that SAGE remains effective across different supervision formats. In addition, we also show SAGE can be applied to open-ended tasks, showing its robustness across verifiable and non-verifiable tasks.

SAGE adds some training-time computation for hint generation and hinted sampling. This overhead is configurable through the maximum hint level, rollout budget, and SAGE-LIGHT, which provides a simple efficiency-oriented variant. This extra training time is comparable to time used by other baselines, while SAGE outperforms them.

SAGE uses hints only during training and removes them at inference. In our experiments, the deployable no-hint policy consistently improves across models and tasks, suggesting that hints act as useful temporary scaffolds rather than a test-time requirement.

C BROADER IMPACT

This work can reduce training cost and improve stability of RL for LLMs on verifiable tasks. Risks include misuse to optimize harmful verifiable objectives.

D PROOFS

This appendix provides detailed proofs for the results in Section 3 (and additional analysis).

D.1 PRELIMINARIES: BERNOULLI GROUPS AND SAMPLE VARIANCE

Fix a context (x, h) and draw $G \geq 2$ i.i.d. rollouts with terminal rewards $R_i \in \{0, 1\}$. Let

$$\bar{R} \coloneqq \frac{1}{G}\sum_{i=1}^{G} R_i, \qquad s^2 \coloneqq \frac{1}{G}\sum_{i=1}^{G}(R_i - \bar{R})^2, \qquad s \coloneqq \sqrt{s^2}.$$

When standardized GRPO is used, advantages are

$$A_i \coloneqq \frac{R_i - \bar{R}}{s + \epsilon},$$

where $\epsilon > 0$ is a numerical stabilizer (as used in the main text).

We will repeatedly use the fact that since $R_i \in \{0, 1\}$,

$$\sum_{i=1}^{G}(R_i - \bar{R})^2 = \sum_{i=1}^{G} R_i^2 - G\bar{R}^2 = \sum_{i=1}^{G} R_i - G\bar{R}^2 = G\bar{R} - G\bar{R}^2 = G\bar{R}(1 - \bar{R}), \tag{13}$$

hence

$$s^2 = \bar{R}(1 - \bar{R}). \tag{14}$$

In particular, $s^2 = 0$ iff $\bar{R} \in \{0, 1\}$, i.e., iff all R_i are identical.

D.2 PROOF OF COROLLARY 3.1

Proof. Recall the advantage energy

$$E \coloneqq \frac{1}{G}\sum_{i=1}^{G} A_i^2, \qquad A_i = \frac{R_i - \bar{R}}{s + \epsilon}.$$

For $\epsilon > 0$, we compute

$$E = \frac{1}{G}\sum_{i=1}^{G}\frac{(R_i - \bar{R})^2}{(s+\epsilon)^2} = \frac{1}{(s+\epsilon)^2}\cdot\frac{1}{G}\sum_{i=1}^{G}(R_i - \bar{R})^2 = \frac{s^2}{(s+\epsilon)^2},$$

which is exactly Eq. (5).

Since $s \geq 0$ and $\epsilon > 0$, we have $0 \leq \frac{s}{s+\epsilon} < 1$, hence

$$0 \leq E = \left(\frac{s}{s+\epsilon}\right)^2 < 1,$$

so $E \in [0, 1)$ (and in the limit $\epsilon \to 0^+$, $E \to \mathbb{I}[s > 0]$).

Monotonicity in s follows by differentiation: define $f(s) \coloneqq \frac{s^2}{(s+\epsilon)^2}$ for $s \geq 0$. Then

$$f'(s) = \frac{2s(s+\epsilon)^2 - s^2 \cdot 2(s+\epsilon)}{(s+\epsilon)^4} = \frac{2s\epsilon}{(s+\epsilon)^3} \geq 0,$$

so E is non-decreasing in s.

Finally, if the group is degenerate, then $s = 0$ and therefore $E = 0$. This shows the standardized signal energy collapses to 0 exactly when within-group variance collapses. □

D.3 Proof of Proposition 3.2

Proof. Write $p \coloneqq p_\theta(x,h) = \Pr[R(x,\tau)=1 \mid x,h]$. Then $R_1,\dots,R_G$ are i.i.d. Bernoulli(p).

As noted in Section D.1, $s=0$ iff all rewards are identical. There are exactly two degenerate cases: (i) all-zero: $R_1=\cdots=R_G=0$; (ii) all-one: $R_1=\cdots=R_G=1$. Thus

$$\Pr[s>0 \mid x,h] = 1 - \Pr[\text{all-zero}] - \Pr[\text{all-one}] = 1-(1-p)^G-p^G,$$

which proves Eq. (6).

To locate the maximizer, define $u(p) \coloneqq 1-(1-p)^G-p^G$ on $[0,1]$. We have symmetry $u(p)=u(1-p)$. Moreover, for $G\geq 2$,

$$u''(p) = -\,G(G-1)\Big(p^{G-2}+(1-p)^{G-2}\Big) < 0,$$

so u is strictly concave, hence has a unique maximizer. By symmetry, the unique maximizer must be at $p=\frac{1}{2}$.

Finally, in the sparse regime $p \ll 1$,

$$u(p) = 1-(1-p)^G-p^G = 1-\Big(1-Gp+O(p^2)\Big)-O(p^G) = Gp+O(p^2),$$

so $\Pr[s>0 \mid x,h] \approx Gp$. □

D.4 Proof of Proposition 3.3

Proof. Fix x and $G\geq 2$. Recall $u(p)=1-(1-p)^G-p^G$ and

$$J_x(\theta,q) = \mathbb{E}_{h\sim q(\cdot|x)}\Big[u\big(p_\theta(x,h)\big)\Big].$$

We first verify the claims about u. Symmetry: $u(1-p)=1-p^G-(1-p)^G=u(p)$. For strict concavity, compute for $G\geq 2$:

$$u''(p) = -\,G(G-1)\Big(p^{G-2}+(1-p)^{G-2}\Big) < 0,$$

so u is strictly concave on $[0,1]$. By symmetry and strict concavity, the unique maximizer is $p=\frac{1}{2}$.

For a fixed θ, define the measurable function

$$v_\theta(h) \coloneqq u(p_\theta(x,h)) \in [0,1].$$

Then $J_x(\theta,q)=\mathbb{E}_{h\sim q}[v_\theta(h)]$. Over all probability distributions $q(\cdot \mid x)$ supported on the admissible hint space, the maximizers must concentrate probability mass on (essential) maximizers of v_θ: indeed, since the objective is linear in q, any optimizer can be chosen to put all mass on

$$\arg\max_h v_\theta(h) = \arg\max_h u(p_\theta(x,h))\,.$$

Because u is uniquely maximized at $p=\frac{1}{2}$ and is strictly decreasing as p moves away from $\frac{1}{2}$ (due to strict concavity and symmetry), the maximizers of $v_\theta(h)$ are exactly the *calibrating hints* that make $p_\theta(x,h)$ as close as possible to $\frac{1}{2}$ (and equal to $\frac{1}{2}$ when achievable). This proves the statement that an optimal $q^\star_\theta(\cdot \mid x)$ places its mass on calibrating hints.

In general, $p_\theta(x,h)$ changes with θ because the rollout distribution under $\pi_\theta(\cdot \mid x,h)$ changes with θ. Therefore the set of calibrating hints

$$\mathcal{H}^\star_\theta(x) \coloneqq \arg\max_h u(p_\theta(x,h))$$

typically varies with θ. Unless $p_\theta(x,h)$ (hence $v_\theta(h)$) is invariant in θ for q-almost all h, a fixed distribution q that was optimal (or near-optimal) early in training will drift away from being optimal later, reducing $J_x(\theta,q)$ relative to a θ-adapted choice. This formalizes why updating the hint generator online using the policy can reduce gate-mismatch. □

D.5 Proof of the Jensen inequality in Remark 1

Proof. Fix $G \geq 2$ and define $u(p) = 1 - (1-p)^G - p^G$. We showed above that u is concave on $[0,1]$ because $u''(p) \leq 0$. Let $Z := p_\theta(x,h) \in [0,1]$ be the random success probability induced by sampling $h \sim q$. Then Jensen's inequality for concave u gives

$$\mathbb{E}_h\big[u(Z)\big] \leq u(\mathbb{E}_h[Z])\,,$$

which is exactly Eq. (10). Equality holds only when Z is almost surely constant (or when u is affine on the support, which does not happen for $G \geq 2$ except in degenerate cases). This shows that, at a fixed mean success probability, additional variability across hints can only decrease the expected gate-opening frequency. □

D.6 A sharper small-p expansion of the gate probability

For completeness, we also record an exact decomposition that makes the Gp scaling explicit. Let $p = \Pr[R = 1 \mid x, h]$. Then

$$\begin{aligned}\Pr[s > 0 \mid x, h] &= 1 - (1-p)^G - p^G \\ &= \sum_{k=1}^{G-1} \binom{G}{k} p^k (1-p)^{G-k}. \end{aligned} \tag{15}$$

When $p \ll 1$, the dominant term is $k = 1$:

$$\Pr[s > 0 \mid x, h] = Gp(1-p)^{G-1} + O(p^2) \approx Gp,$$

and the neglected p^G term is exponentially smaller in G.

D.7 Non-standardized GRPO signal energy

Non-standardized advantage is also widely used in GRPO-like algorithms (Liu et al., 2025a), which also provide insights about the behavior.

Setup. Define non-standardized (mean-centered) advantages

$$\tilde{A}_i := R_i - \bar{R}, \qquad \bar{R} = \frac{1}{G}\sum_{i=1}^{G} R_i,$$

and define the (non-standardized) energy

$$\tilde{E} := \frac{1}{G}\sum_{i=1}^{G} \tilde{A}_i^2.$$

Note that $\tilde{E} = s^2$ by definition.

Proposition D.1 (Expected non-standardized energy under Bernoulli rewards)**.** *Conditioned on (x,h) with success probability $p = p_\theta(x,h)$,*

$$\mathbb{E}\big[\tilde{E} \mid x, h\big] = \frac{G-1}{G}\, p(1-p). \tag{16}$$

Proof. Let $S = \sum_{i=1}^{G} R_i$ so that $\bar{R} = S/G$. Using identity (13),

$$\tilde{E} = \frac{1}{G}\sum_{i=1}^{G}(R_i - \bar{R})^2 = \bar{R}(1-\bar{R}).$$

Taking expectation:

$$\mathbb{E}[\tilde{E}] = \mathbb{E}[\bar{R}] - \mathbb{E}[\bar{R}^2].$$

We have $\mathbb{E}[\bar{R}] = p$. Also,

$$\mathbb{E}[\bar{R}^2] = \mathrm{Var}(\bar{R}) + (\mathbb{E}[\bar{R}])^2 = \frac{1}{G^2}\mathrm{Var}(S) + p^2 = \frac{1}{G^2}\cdot Gp(1-p) + p^2 = \frac{p(1-p)}{G} + p^2.$$

Therefore

$$\mathbb{E}[\tilde{E}] = p - \left(\frac{p(1-p)}{G} + p^2\right) = \frac{G-1}{G}\,p(1-p).$$

□

Proposition D.2 (Optimal calibrated difficulty for mean-centered updates)**.** *For fixed $G \geq 2$, the right-hand side of Eq.* (16) *is uniquely maximized at $p = \frac{1}{2}$.*

Proof. Let $f(p) = p(1-p) = p - p^2$. Then $f'(p) = 1 - 2p$ and $f''(p) = -2 < 0$, so f is strictly concave and uniquely maximized at $p = \frac{1}{2}$. The prefactor $(G-1)/G$ does not affect the maximizer. □

Remark D.3 (Hint randomness incurs a variance penalty at fixed mean). Let $Z = p_\theta(x, h)$ be random due to $h \sim q$ and $\bar{p} = \mathbb{E}[Z]$. Then

$$\mathbb{E}\big[Z(1-Z)\big] = \bar{p}(1-\bar{p}) - \mathrm{Var}(Z).$$

Thus, at fixed mean success rate, additional variability in $p_\theta(x, h)$ across hints reduces the expected energy.

Proof. Compute

$$\mathbb{E}[Z(1-Z)] = \mathbb{E}[Z] - \mathbb{E}[Z^2] = \bar{p} - \big(\mathrm{Var}(Z) + \bar{p}^2\big) = \bar{p}(1-\bar{p}) - \mathrm{Var}(Z).$$

□

E PROMPT FOR HINT GENERATION AND THE USAGE OF HINT

System prompt for hint generation

You are a tutoring assistant that generates progressive hints to help students solve difficult problems without revealing the solution directly.

TASK:
Given a question and its solution, generate 3 levels of hints that progressively guide the student toward solving the problem independently.

HINT LEVELS:
- Level 1: Minimal hint - Points to the key concept or approach without specifics
- Level 2: Medium hint - Provides more direction on the method or intermediate steps
- Level 3: Detailed hint - Gives substantial guidance while still requiring the student to complete the solution

GUIDELINES:
- Never reveal the final answer
- Each level should be built on the previous one
- Hints should inspire problem-solving, not just provide steps to copy
- Tailor hint difficulty to bridge the gap between the student's level and the solution

OUTPUT FORMAT:

```
```json
{
  "level_1": "minimal hint text",
  "level_2": "medium hint text",
  "level_3": "detailed hint text"
}
```
```

User prompt for hint generation

Question:
{problem}

Solution:
{solution}

System prompt for RL

Please reason step by step, and put your final answer within \\boxed{}.

User prompt for RL

{problem}

Here is a hint to help you:
{hint}

F ALGORITHM

In algorithm 1, we summarize the implementation of SAGE and SAGE-LIGHT.

Algorithm 1 SAGE / SAGE-LIGHT: Self-hint Aligned GRPO with Privileged Supervision

Require: Training set $\mathcal{D} = \{(x, \tau^\star)\}$, policy model π_θ, group size G, KL weight β, stabilizer $\epsilon > 0$, max hint level L, reference policy π_{ref}, hint generator $q_\phi(h \mid x, \tau^\star, \ell)$ based on π_θ, threshold α (SAGE-LIGHT only).

Initialize policy parameters θ; for SAGE-LIGHT, initialize the per-prompt level map $\ell(x) \leftarrow 0$ for all $x \in \mathcal{D}$.
Repeat for epochs $t = 1, 2, \ldots$:
  for each minibatch $\{(x_b, \tau_b^\star)\}_{b=1}^B$:
    for $b = 1, \ldots, B$ **do**
      if SAGE-LIGHT **then**
        if $t > 1$ **and** $\hat{p}_{t-1}(x_b) < \alpha$ **then** $\ell(x_b) \leftarrow \min\{\ell(x_b) + 1, L\}$
        else if $t > 1$ **and** $\hat{p}_{t-1}(x_b) > \frac{1}{2}$ **then** $\ell(x_b) \leftarrow \max\{\ell(x_b) - 1, 0\}$
        Sample $h_b \sim q_\phi(h \mid x_b, \tau_b^\star, \ell(x_b))$
      else (SAGE)
        for $\ell = 0, \ldots, L$ **do**
          Sample $\tilde{h}_b \sim q_\phi(h \mid x_b, \tau_b^\star, \ell)$
          Sample $\tilde{\tau}_{b,i} \sim \pi_\theta(\cdot \mid x_b, \tilde{h}_b)$ for $i = 1, \ldots, G$ and compute $\tilde{R}_{b,i} \leftarrow R(x_b, \tilde{\tau}_{b,i})$
          if $\sum_{i=1}^G \tilde{R}_{b,i} > 0$ **or** $\ell = L$ **then**
            $h_b \leftarrow \tilde{h}_b,\ \tau_{b,i} \leftarrow \tilde{\tau}_{b,i},\ R_{b,i} \leftarrow \tilde{R}_{b,i}$; **break**
      (If SAGE-LIGHT) Sample $\tau_{b,i} \sim \pi_\theta(\cdot \mid x_b, h_b)$ for $i = 1, \ldots, G$ and compute $R_{b,i} \leftarrow R(x_b, \tau_{b,i})$.
      Compute $\bar{R}_b \leftarrow \frac{1}{G}\sum_{i=1}^G R_{b,i}$, $s_b \leftarrow \sqrt{\frac{1}{G}\sum_{i=1}^G (R_{b,i} - \bar{R}_b)^2}$, $A_{b,i} \leftarrow \frac{R_{b,i} - \bar{R}_b}{s_b + \epsilon}$.
      if SAGE-LIGHT **then** store $\hat{p}_t(x_b) \leftarrow \bar{R}_b$ for the next epoch.
    $\mathcal{L}_{\text{pg}} \leftarrow -\frac{1}{BG}\sum_{b=1}^B \sum_{i=1}^G A_{b,i} \sum_{j=1}^{T_{b,i}} \log \pi_\theta(y_{b,i,j} \mid x_b, h_b, y_{b,i,<j})$
    $\mathcal{L}_{\text{kl}} \leftarrow \frac{1}{B}\sum_{b=1}^B \text{KL}(\pi_\theta(\cdot \mid x_b, h_b) \,\|\, \pi_{\text{ref}}(\cdot \mid x_b, h_b))$
    Update $\theta \leftarrow \theta - \eta \nabla_\theta \big(\mathcal{L}_{\text{pg}} + \beta \mathcal{L}_{\text{kl}}\big)$
Deployment: set $\ell = 0$ so $h = \varnothing$, and run $\pi_\theta(\cdot \mid x)$.

G Detailed implementation settings

G.1 Training settings

Training set. Our training data are drawn from OpenR1-Math-220k (Hugging Face, 2025), using prompts from NuminaMath 1.5 (Li et al., 2024) and reasoning traces generated by DeepSeek-R1 (DeepSeek-AI et al., 2025). The initial dataset contains 94k prompts. To ensure answer verifiability, we apply the *Math-Verify* tool (Kydlíček) to remove prompts whose DeepSeek-R1 reasoning traces are incorrectly verified, resulting in 64k prompts. These prompts are used in Figure 2. Due to limited resources, we further subsample 15k prompts from this set, restricting the corresponding DeepSeek-R1 reasoning traces to fewer than 8,192 tokens. This constraint is necessary because one of our baselines, LUFFY (Yan et al., 2025), relies on these reasoning traces, and excessively long traces would significantly increase RL training time. As we do not filter prompts based on pass rate, the resulting 15k prompts span a wide range of difficulty levels, resembling a practical RL training dataset.

Implementation details. We run all experiments on 8 A100 GPUs, and use verl (Sheng et al., 2025) for training and vLLM (Kwon et al., 2023) for sampling. Following DAPO (Yu et al., 2025), we disable the KL term by setting $\beta = 0$, and apply asymmetric clipping with $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$. Unless otherwise specified, the maximum response length is set to 8096 for both training and evaluation,[5] with a batch size of 128, 8 trajectories per prompt,[6] and 500 training steps in total. We evaluate every 50 steps, and report the best average accuracy over all checkpoints. We set max hint strength L as 3, and $\alpha = 0.35$ for SAGE-LIGHT. On the hard training set, we only sample 4 trajectories and train for 400 steps due to the resource constraints.

SFT. We use OpenRLHF Hu et al. (2024) for SFT, and set the learning rate as 5e-5, the batch size as 64, warmup ratio as 10%, and number of epochs as 3. We evaluate on the final checkpoints.

GRPO. GRPO shares the same training settings as SAGE, as stated in §5.

LUFFY. We use the open-source implementation[7] to reproduce LUFFY, and set the batch size as 128, and *ppo_mini_batch_size* as 64. These two hyper-parameters stay the same for all RL methods.

Scaf-GRPO. We use the open-source implementation[8] to reproduce Scaf-GRPO.

G.2 Evaluation settings

For the main results in Table 1, we evaluate all models with a max response length of 8192, *temperature=0.6* and *top_p=0.95*. For the rest, we set a max response length of 2048. For all tasks during evaluation, we take 8 random runs and report the average accuracy.

G.3 Training on open-ended tasks

Following Bhaskar et al. (2025), we apply SAGE to three open-ended reasoning tasks. We starting from the SFT warm-start Qwen2.5-7B-Instruct (no_think) released by Bhaskar et al. (2025), and train it with SAGE on the same training set. Unlike math reasoning task that has a ground truth, open-ended task requires a reward model to judge the generation. Here we utilize Skywork-v1-Llama-3.1-8B-v0.2 (Liu et al., 2024) as the reward model, the same as Bhaskar et al. (2025). The reward score ranges in $(-\infty, +\infty)$. Without any search, we set a threshold 10 for SAGE. I.e., if the rewards of all trajectories are < 10, we deem the prompts are hard, and generate hint from the solution. The training and evaluation settings stay the same as Bhaskar et al. (2025).

[5] We use 8096 for the main results (required by LUFFY), and 2048 for the remaining experiments due to resource constraints.

[6] We use 4 trajectories for Qwen3-4B-Instruct due to slower training caused by its long response length.

[7] https://github.com/ElliottYan/LUFFY

[8] https://github.com/JIA-Lab-research/Scaf-GRPO

H MORE RESULTS

H.1 SENSITIVITY TO TRAINING RANDOM SEEDS

We retrain SAGE with three random seeds on Qwen2.5-7B-Instruct and Qwen3-4B-Instruct, using both the easy and hard training sets. Table 4 reports the six-benchmark in-distribution average at the best and final checkpoints. For each seed, the $\pm$ value is the standard deviation over eight evaluation runs. Seed 0 corresponds to the run reported in Tables 1 and 2; the aggregate rows summarize the three training seeds. Across these four settings, best-checkpoint averages vary by at most 0.8 points across seeds, and final-checkpoint averages by at most 1.0 point.

Table 4: Six-benchmark in-distribution average accuracy of SAGE across training random seeds.

Training set	Backbone	Seed	Best checkpoint	Final checkpoint
Easy	Qwen2.5-7B	0 (reported)	42.3 ± 0.4	42.3 ± 0.4
		1	42.3 ± 0.2	42.3 ± 0.2
		2	42.6 ± 0.4	42.6 ± 0.4
		Mean of 3 seeds	42.4 ± 0.3	42.4 ± 0.3
Easy	Qwen3-4B	0 (reported)	70.0 ± 0.4	69.3 ± 0.3
		1	70.2 ± 0.4	70.2 ± 0.4
		2	69.7 ± 0.3	69.7 ± 0.3
		Mean of 3 seeds	70.0 ± 0.4	69.7 ± 0.3
Hard	Qwen2.5-7B	0 (reported)	42.4 ± 0.5	42.4 ± 0.5
		1	42.6 ± 0.5	42.6 ± 0.5
		2	42.9 ± 0.2	42.7 ± 0.3
		Mean of 3 seeds	42.6 ± 0.4	42.6 ± 0.4
Hard	Qwen3-4B	0 (reported)	68.3 ± 0.2	68.3 ± 0.2
		1	68.1 ± 0.4	67.9 ± 0.3
		2	68.9 ± 0.2	68.9 ± 0.2
		Mean of 3 seeds	68.4 ± 0.3	68.4 ± 0.2

H.2 TRAINING DYNAMICS

In Figure 8, we show the training dynamics for all 3 LLMs.

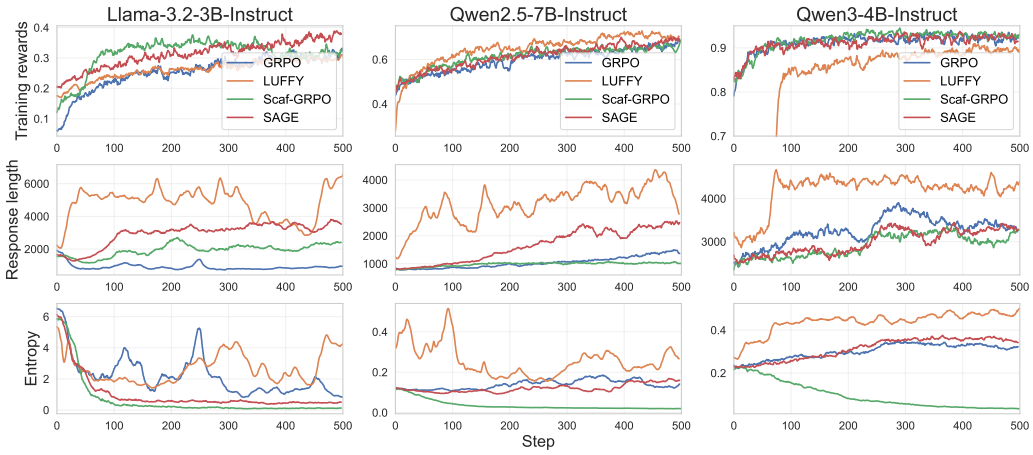

Figure 8: The training dynamics of different methods. For the training rewards, one should focus on the trend instead of the value, since adding hint (SAGE and Scaf-GRPO) modifies the prompt difficulty, and using a correct off-policy trajectory (LUFFY) increases the reward. (1) LUFFY shows the most instability, with a very high entropy for Llama and a very low reward at the beginning of training for Qwen3, because it imitates the off-policy trajectory whose distribution might not be aligned with the policy model. (2) Scaf-GRPO shows the lowest entropy, implying less exploration. (3) SAGE retains the on-policy characteristic, has a mild entropy and shows a stable growth in response length, which normally implies a better reasoning pattern.

H.3 Illustration of Privileged Hinting

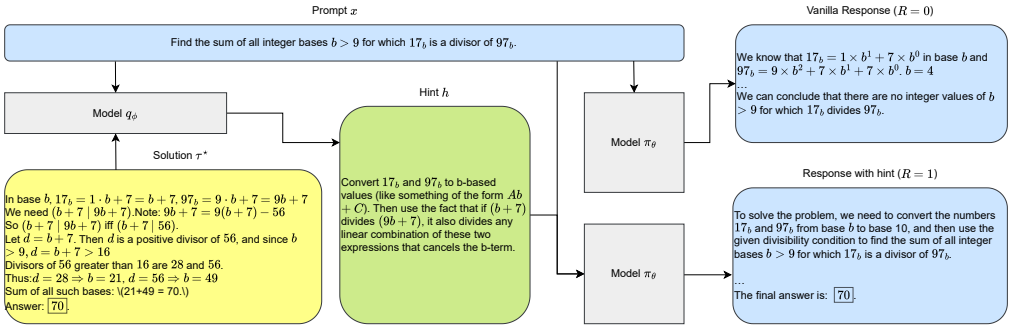

Figure 9: Example of privileged hinting for a single prompt. Given a math prompt x, a hint generator q_ϕ uses the reference solution $\tau^\star$ during training to produce a procedural hint (h) that summarizes intermediate reasoning without revealing the final answer. Rolling out the policy π_θ on the original prompt can yield an incorrect solution with zero reward, while conditioning on (x, h) shifts the rollout distribution and enables a correct solution with positive reward. The task return is unchanged, and at deployment the hint is removed so the model runs on the original prompt only.

Case Study: Progressive privileged hints for $17_b \mid 97_b$

Prompt. Find the sum of all integer bases $b > 9$ for which 17_b is a divisor of 97_b.

Reference solution (available only during training). In base b,

$$17_b = 1 \cdot b + 7 = b + 7, \qquad 97_b = 9 \cdot b + 7 = 9b + 7.$$

We need $b + 7 \mid 9b + 7$. Note that

$$9b + 7 = 9(b + 7) - 56,$$

hence $b + 7 \mid 9b + 7$ iff $b + 7 \mid 56$. Let $d = b + 7$. Then d is a positive divisor of 56, and since $b > 9$ we have $d = b + 7 > 16$. The divisors of 56 greater than 16 are 28 and 56, so $b \in \{21, 49\}$ and the sum is 70.

Privileged hints (used only during training; never shown at test time).

- **Level 1 (minimal).** Rewrite the base-b numerals as ordinary integers in terms of b, then turn the divisibility condition into a statement about a simple linear expression.
- **Level 2 (medium).** Convert 17_b and 97_b into the form $Ab + C$. If $(b + 7)$ divides $(9b + 7)$, it also divides any linear combination of these expressions that cancels the b-term.
- **Level 3 (detailed).** Compute $17_b = b + 7$ and $97_b = 9b + 7$. Subtract a multiple of $(b + 7)$ from $(9b + 7)$ to eliminate b, e.g., $(9b + 7) - 9(b + 7)$. The condition becomes "$(b + 7)$ divides a constant". Enumerate divisors and keep only $b > 9$.

Representative model behaviors (illustrative).

- **No hint ($h = \varnothing$):** the model may mis-expand 97_b (e.g., treating it as a multi-digit polynomial), and incorrectly conclude there is no valid $b > 9$, yielding a wrong terminal decision and thus reward 0.
- **With stronger hints (Level 2/3):** the model is steered toward the key cancellation step $9b + 7 - 9(b + 7) = -56$, after which it can enumerate divisors and recover the correct bases and final sum (70).

Case Study: Privileged Hinting on a Simple Verifiable Math Task. To make the idea of *privileged hinting* concrete, we include a small case study on a verifiable divisibility question. The key point is that hints are *training-time privileged context*: they do not modify the verifier or the terminal reward. Instead, they reshape the rollout distribution so that, under finite sampling, the policy is more likely to generate informative trajectories (e.g., those that perform the correct algebraic reduction). In practice, without hints the model may repeatedly take an incorrect representation path (e.g., mis-expanding base-b numerals) and receive identical zero rewards across a rollout group, causing GRPO advantages to collapse. With progressive hints, the model is guided toward the correct reduction, increasing the chance that a group contains mixed outcomes and thus yields a non-degenerate update.

This example matches the operational role of SAGE: hints do not change the verifier reward, but they increase the probability that at least one rollout in a finite group follows a *useful* trajectory (here, the cancellation-and-divisor-enumeration path). This increases within-group outcome diversity, reduces the frequency of degenerate all-zero groups, and therefore prevents standardized GRPO advantages from collapsing on hard prompts under finite sampling.

H.4 NUMBER OF HINT LEVELS L

Table 5: Performance with different number of hint levels L for SAGE. Due to the resource and time constraint, here we set the number of training steps as 200 instead of 500. The rest settings stay the same as the main experiments.

Hint Level	AIME24	AIME25	AMC23	MATH-500	Minerva	Olympiad	Avg.	Δ
Llama-3.2-3B-Instruct	6.5	0.6	22.8	44.7	17.8	14.2	17.8	0.0
$L = 2$	6.9	1.0	28.8	51.5	19.7	18.9	21.1	+3.3
$L = 3$	**9.2**	0.8	**34.7**	**56.3**	**20.1**	**22.0**	**23.9**	**+6.1**
$L = 4$	**9.2**	**2.5**	30.0	53.2	19.1	19.2	22.2	+4.4
$L = 5$	**9.2**	0.0	25.6	51.0	20.0	19.0	20.8	+3.0
Qwen2.5-7B-Instruct	13.8	6.7	53.4	75.7	38.1	39.2	37.8	0.0
$L = 2$	14.0	8.5	56.6	78.9	36.5	43.0	39.6	+1.8
$L = 3$	15.0	10.6	**57.0**	79.0	37.0	**44.6**	40.5	+2.7
$L = 4$	14.8	11.5	56.6	78.2	**39.7**	43.0	40.6	+2.8
$L = 5$	**15.6**	**12.9**	56.1	**80.0**	39.2	44.1	**41.3**	**+3.5**

In Table 5, we investigate the effectiveness of the max hint level L. Overall, a larger number of hint levels leads to a better average accuracy: (1) Qwen2.5 shows an increasing average accuracy w.r.t. to a larger L; (2) Llama-3.2 achieves a peak accuracy with $L = 3$, but degrades with a larger L. After investigation, we notice that this degradation comes from the poor instruction following ability of Llama-3.2. As shown in Appendix E, model needs to follow the hint instruction for the generation of different levels of hints. With a larger L, the instruction becomes complex. We often observes that Llama-3.2 could not generate the hints in a required output format. So the hints aren't used, since they can't be parsed correctly. We suggest setting a higher L if the model has a good instruction-following capability.

However, with a larger L, the time used for sampling after hint injection also becomes longer. It is a trade-off between latency and performance, leaving it as a flexible budget control factor to the user.

H.5 MORE BASELINES

We further compare SAGE to another three baselines: (1) GHPO (Liu et al., 2025b): GHPO directly uses the reference solution as hint, and append different portion of the reference solution to the prompt to control the hint strength. (2) HINT (Wang et al., 2025): The hints are generated offline by a stronger model, using each problem's question and final answer. This setting is similar to our GPT-5 hint variant. (3) SFT+GRPO: We first apply SFT on the samples with pass@10=0, and then apply GRPO on the rest samples.

Table 6: Performance from training on the same training set used by GHPO (Liu et al., 2025b). The baseline results are directly borrowed from Liu et al. (2025b). Here we use the exactly same training and evaluation settings as Liu et al. (2025b), like *epoch*, *batch_size*, *max_gen_tokens*, *temperature* and so on.

Method	AIME24	MATH-500	Olympiad	AMC23	Minerva	GPQA-Diamond	Avg	Δ
Qwen2.5-Base-7B	9.8	69.4	34.0	40.0	19.5	21.7	32.4	0.0
GRPO	13.1	75.2	40.8	47.5	31.2	30.8	39.8	+7.4
GHPO	13.3	78.6	**41.5**	57.5	34.6	39.4	44.2	+11.8
SAGE	**16.7**	**79.1**	41.0	**58.8**	**36.8**	**41.6**	**45.7**	**+13.3**

Table 7: Performance from training on the same training set used by HINT (Wang et al., 2025). The baseline results are directly borrowed from Wang et al. (2025). Here we use the exactly same training and evaluation settings as Wang et al. (2025), like *epoch*, *batch_size*, *max_gen_tokens*, *temperature* and so on.

Method	AIME24	MATH-500	Olympiad	Minerva	Avg	Δ
Qwen2.5-Base-7B	9.8	50.2	34.0	19.5	28.4	0.0
GRPO	12.8	75.2	40.8	31.2	40.0	+11.6
GHPO	13.1	75.6	42.2	30.0	40.2	+11.8
HINT	13.3	**79.6**	43.6	31.0	41.9	+13.5
SAGE	**14.2**	79.0	**44.5**	**35.2**	**43.2**	**+14.8**

In Table 6 and 7, we can observe that SAGE outperforms both methods. We argue: (1) The setting of GHPO and HINT is similar to our offline hint setting, which is worse than online self-hinting; (2) SAGE self-generates hints per step. What matters more is not the hint quality but the calibration to the learner's current capability, which is the key contribution of SAGE.

Table 8: Compare to SFT+GRPO: we first apply SFT on the samples with pass@10=0, and then apply GRPO on the rest samples. Pass@10 is calculated with the trajectories sampled by the base model. Here we use the same training settings as the main experiments, like *training_steps*, *max_gen_tokens* and so on.

Method	#Training Samples	AIME24	AIME25	AMC23	MATH-500	Minerva	Olympiad	Avg.	Δ
Qwen2.5-7B-Instruct	-	13.8	6.7	53.4	75.7	38.1	39.2	37.8	0.0
(1) SFT+GRPO	15K								
First SFT	3.4K	2.5	2.9	12.5	33.0	15.2	15.4	13.6	-24.2
Then GRPO	11.6K	13.3	**15.8**	57.5	78.3	37.5	44.4	41.1	+3.3
GRPO	15.0K	15.0	13.5	55.5	79.2	39.1	44.5	41.1	+3.3
SAGE	15.0K	**16.0**	12.5	**60.3**	**80.0**	39.3	**45.9**	**42.3**	**+4.5**

In Table 8, we can observe that the performance of SFT+GRPO is similar to GRPO's, still underperforming SAGE.

H.6 Using reference trajectory to generate hint

By default, we use the reference solution to generate the hint. And the reference solution is commonly available. For some tasks or questions, we might not be able to have a gold solution. Here we explore whether we can use a reasoning trajectory from a teacher model to generate a valid hint. As we know, the reasoning trajectory normally contain some sub-optimal steps, like the steps used for reflection, back-tracking and so on. We also explore whether sub-optimal steps affect the hint quality.

To do so, we

- first select 200 prompts with pass@10=0 (sampling with Qwen3-4B-Instruct). These prompts are deemed hard prompts that SAGE mainly focuses on. OpenR1-Math-220k already offers the reasoning trajectories for these prompts from DeepSeek-R1.
- then apply GPT-5.4 (xhigh reasoning effort) to extract the partial errors from these trajectories given reference solution.
- next prompt Qwen3-4B-Instruct to generate hint for these 200 prompts from question and trajectories.
- finally apply GPT-5.4 to classify whether these hints contain the partial errors with three levels: clean, mild and severe. "Clean" means the hint doesn't contain any partial error.
- also show the sampling accuracy (n=10 for each prompt) with different levels of hint to show the effectiveness of hint.

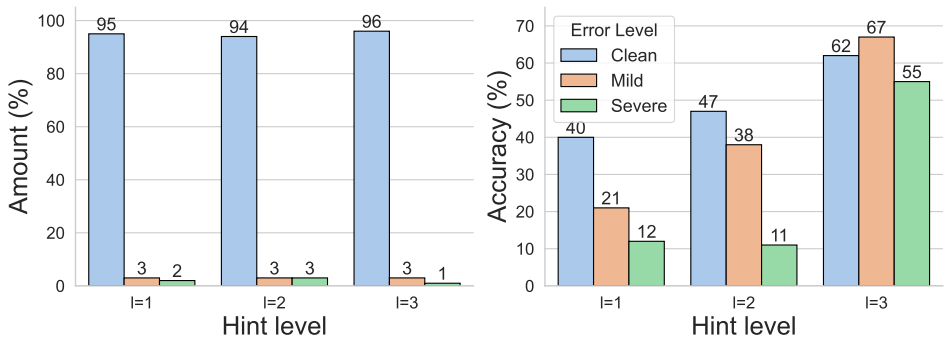

Figure 10: Hint generation from DeepSeek-R1 trajectory instead of reference solution. **Left**: The percentage of hints that contain sub-optimal error from trajectory. **Right**: The sampling accuracy with the hints generted from trajectory.

As shown in Figure 10 (Left), The hint rarely contains the error from the sub-optimal steps. From Figure 10 (Right), all hints in different error levels help the sampling of correct trajectories. Notably, the accuracy of these prompts are 0 without any hint.

H.7 Using answer to generate the hint

For some tasks, the solution (reasoning+answer) is not available, like coding, where we only have the ground truth or answer. Can you generate a valid hint from the ground truth? Following the training and evaluation settings of Luo et al. (2025), we apply SAGE to the coding task, and use the ground truth to generate hint.

Table 9: Using answer to generate hint for coding task.

Method	**LiveCodeBench v5 (Jain et al., 2024)**
Qwen3-4B-Instruct	27.5
GRPO	31.0
SAGE	33.4

As shown in Table 9, SAGE still outperforms GRPO by using the answer to generate hint.