WHEN ALL ROLLOUTS FAIL, CRITIQUE PROVIDES DIRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Group-relative reinforcement learning is widely used to improve reasoning in large language models. However, it cannot learn from a problem when all sampled answers are wrong, because their relative advantages collapse to zero. Critique-guided revision can continue exploration beyond these failed answers. Yet when all revisions are also wrong, the learning signal disappears again because outcome rewards cannot distinguish useful critiques from unhelpful ones. We propose Signpost, a training framework based on Critique as Direction. A critique can be useful even if it does not immediately produce a correct answer, as long as it points the model toward one. Signpost trains the same language model to generate both answers and critiques. Its directional reward measures how much a critique increases the likelihood of a target solution under a frozen reference model. This provides a learning signal when all revisions fail, without rewarding incorrect answers. Once a critique helps discover a verified solution, Signpost trains the model to produce that solution directly from the original problem using a bounded-likelihood objective. This transfers critique-guided discoveries into question-only solving and closes the gap between training and inference. Experiments on mathematical reasoning show that Signpost outperforms GRPO, DAPO, Scaf-GRPO, and Critique-GRPO. On Qwen3-8B, it improves pass@8 over GRPO by 6.6 to 10.0 percentage points across AIME 2024–2026. Critiques are used only during training, while inference requires no critiques, revisions, or additional computation. Code is available at `https://anonymous.4open.science/r/signpost-404F/`.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a powerful way to improve the reasoning of large language models (Shao et al., 2024; Guo et al., 2025). Group Relative Policy Optimization (GRPO), a widely used RLVR method, samples a group of answers for each problem and learns from how their rewards differ (Shao et al., 2024). This design has a blind spot: if every answer in the group is wrong, all advantages are zero and the problem gives no learning signal. Such all-wrong groups are common on hard problems, which are often the ones most worth learning from. DAPO simply filters them out (Yu et al., 2025), while other methods make correct answers easier to reach with demonstrations, hints, or other privileged information (Yan et al., 2025; Zhang et al., 2026c; Ding, 2026; Agrawal et al., 2026). Hard problems need guided exploration, not just more samples.

Natural-language critiques are a flexible form of such guidance: by pointing out an error and suggesting another way forward, a critique lets the policy explore solutions that direct sampling rarely reaches. Prior work uses critiques to refine answers at inference time (Madaan et al., 2023) and, more recently, to guide exploration during RL training (Zhang et al., 2026b; Lin et al., 2026; Zhang et al., 2026a). Critiques, however, do not remove the sparse-reward problem; they move it one level deeper. When all direct answers fail, the model writes several critiques and revises its answer according to each. If every revision is still wrong, binary rewards again cannot tell a promising critique from a useless one, and the advantages vanish a second time. Yet a critique may identify the key error and still fail to produce a complete solution in one attempt. When critique utility is inferred from revision outcomes, this partial progress remains indistinguishable from an ineffective critique if all revisions receive the same binary reward (Zhang et al., 2026b; Lin et al., 2026).

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.

Our key observation is that a critique does not need to finish the correction to be useful; it only needs to point the model in the right direction. We therefore evaluate a critique by its direction rather than its immediate success. Following work that uses reference-answer likelihood as graded supervision (Wang et al., 2026; Kwiatkowski et al., 2026; Xiao et al., 2026), we measure how much adding a critique to the same problem and failed answer raises a frozen reference model's likelihood of a reference target available during training. Under the default scoring template, this target may be only the final answer, without solution steps; the score therefore does not verify intermediate reasoning. A positive change means the critique points toward the target, even if its sampled revision is still wrong.

Based on this idea, we propose **Signpost**, a training framework built on the principle of *Critique as Direction*. A single language model serves as both solver and critic through role-specific prompts. Signpost learns useful critiques with a directional reward: for each critique, a frozen reference model measures how much it raises the likelihood of a reference target relative to the same failed answer without the critique. This provides a learning signal even when every sampled revision is wrong, and updates the critique rather than reinforcing failed revisions. When a critique-guided revision yields a verified solution, Signpost removes the failed answer and critique and trains the model to produce the solution directly from the original problem (Lin et al., 2026; Agrawal et al., 2026), using a bounded-likelihood objective alongside standard GRPO updates on direct answers. Critiques are used only during training; at inference, the model answers the problem directly.

We evaluate Signpost on mathematical reasoning benchmarks and compare it with GRPO, DAPO, Scaf-GRPO, and Critique-GRPO. On Qwen3-8B, Signpost improves pass@8 over GRPO by 6.6 to 10.0 percentage points across AIME 2024, 2025, and 2026. Our contributions are:

- Critique-guided learning encounters a second signal failure when all revisions remain wrong: outcome rewards cannot distinguish useful critiques from ineffective ones.
- A directional reward measures how much a critique raises a frozen reference model's likelihood of a reference target, providing supervision before a revision succeeds without reinforcing incorrect revisions.
- Verified critique-guided solutions are transferred to direct solving by removing the failed answer and critique and training under the original problem with a bounded-likelihood objective; direct inference requires no critique or revision.

2 Related Work

Learning from hard problems in RLVR. GRPO provides no learning signal when all answers in a group receive the same binary reward (Shao et al., 2024). Dr. GRPO addresses estimator bias (Liu et al., 2025), while DAPO filters zero-variance groups (Yu et al., 2025); other methods use demonstrations, hints, or privileged information to obtain successful trajectories (Yan et al., 2025; Zhang et al., 2026c; Ding, 2026; Agrawal et al., 2026). Methods including HDPO, Off-Context GRPO, and CAPF also study learning from guided solutions under the original prompt (Ding, 2026; Agrawal et al., 2026; Chen et al., 2026). Signpost instead obtains guidance from critiques generated by the solver itself. The target solution is not shown to the policy; a frozen reference model uses it only to score each critique. Verified revisions are then trained under the original prompt with a bounded-likelihood objective alongside GRPO on direct responses.

Critique-guided learning. Critiques have been used for inference-time self-correction (Madaan et al., 2023; Gou et al., 2024), supervised critique learning (Xi et al., 2024; Wang et al., 2025), and RL-based joint training of solvers and critics (Kumar et al., 2025; Zhang et al., 2026b; Lin et al., 2026; Jiao et al., 2026; Ren et al., 2026; Vu Duc Anh et al., 2026). Critique-GRPO trains on critique-guided revisions, ICRL rewards critiques through their downstream outcomes, and STRIDE uses a separate verifier to provide stepwise feedback (Zhang et al., 2026b; Lin et al., 2026; Zhang et al., 2026a). Outcome-based critique evaluation cannot distinguish useful from ineffective feedback when all revisions fail. Signpost addresses this case by scoring the change a critique induces in a frozen reference model's target-solution likelihood and applying the resulting signal to critique tokens only. Unlike prior likelihood-based rewards for reasoning trajectories (Cui et al., 2025; Wang et al., 2026; Kwiatkowski et al., 2026; Xiao et al., 2026), it evaluates the critique's marginal contribution; the critique is used for training, not inference.

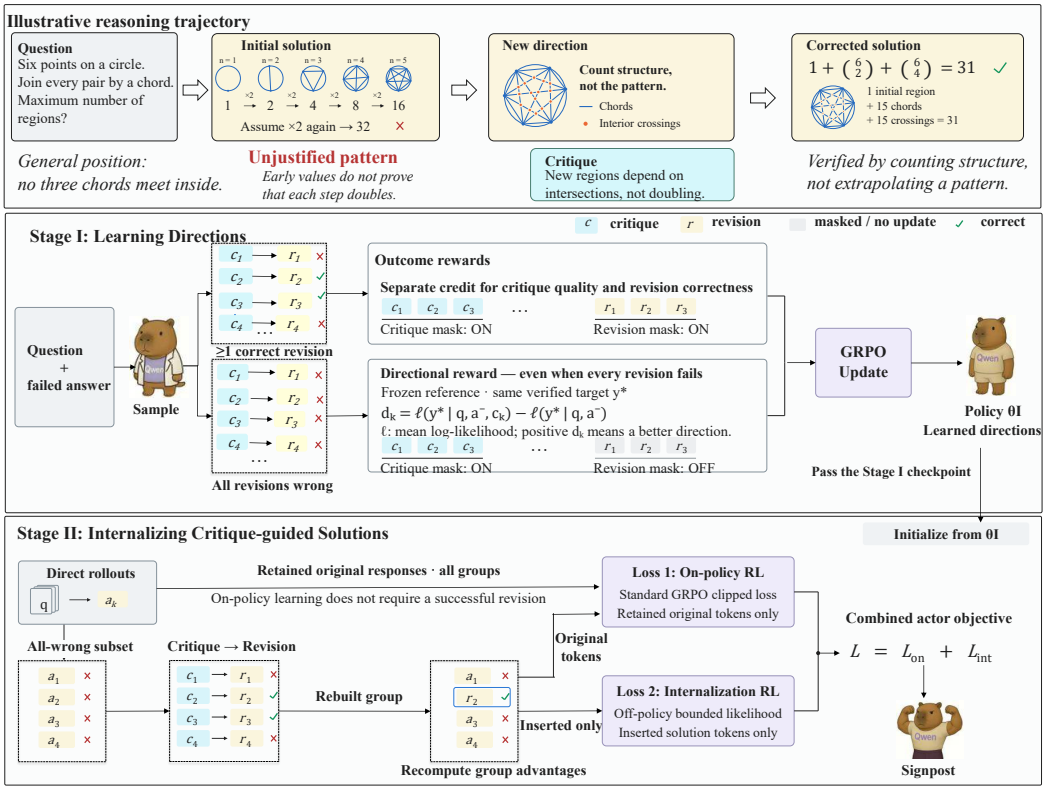


Figure 1: Overview of Signpost. Stage I *learns directions*: critiques in all-wrong refinement groups are rewarded by how much they raise a frozen reference model's likelihood of the reference target. Stage II *internalizes critique-guided solutions*: each verified solution is stripped of its critique context and used to train the solver on the original problem.

3 Signpost: Critique as Direction

Signpost trains a single policy π_θ in two roles through role-specific prompts: a solver π_θ^{S} that answers a problem and a critic π_θ^{C} that critiques a failed answer. Following the principle of *Critique as Direction*, Signpost first *learns directions* by rewarding critiques that move the model toward a correct solution, even when no revision succeeds (Stage I, Section 3.2). It then trains the solver to produce each verified critique-guided solution directly from the original problem (Stage II, Section 3.3). The two stages are run sequentially, which outperforms optimizing them jointly (Section 4.3). Critiques are used only during training; at inference, the solver answers from the problem alone. Section 3.4 describes an optional extension that spends extra training compute to follow directions over multiple rounds.

3.1 Background: The Learning Signal Vanishes Twice

Self-critique and refinement. Problems whose direct group has a low empirical success rate, including all-wrong groups, are routed to refinement. Given a failed trajectory τ^-, the critic writes a critique $c \sim \pi_\theta^{\mathrm{C}}(\cdot \mid q, \tau^-)$, and the solver revises the answer into $\tilde{\tau} \sim \pi_\theta^{\mathrm{S}}(\cdot \mid q, \tau^-, c)$. For each selected failure, we sample a refinement group $\mathcal{G}_{\mathrm{R}}(q, \tau^-) = \{(c_k, \tilde{\tau}_k)\}_{k=1}^{K}$ and mark critique and revision tokens with masks $m_{k,t}^{\mathrm{C}}$ and $m_{k,t}^{\mathrm{R}}$, so that each span can receive its own reward.

Two levels of vanishing. Under binary rewards, if $r(\tau_i) = 0$ for every $\tau_i \in \mathcal{G}(q)$, Equation 8 gives $\widehat{A}_i = 0$ for all i, and the problem contributes no gradient. Refinement is meant to recover signal from such problems, but the same failure recurs one level deeper. If $r(\tilde{\tau}_k) = 0$ for all k, any

outcome-based reward is constant across the refinement group and yields zero advantage for both revisions and critiques. A critique that identifies the key error but whose single revision still fails is then indistinguishable from an irrelevant one. Signpost is designed to address this second failure.

3.2 Learning Directions (Stage I)

Directional reward for all-wrong refinement groups. When all K revisions fail, we score each critique by the change it induces in a frozen reference model's mean teacher-forced log-likelihood of a reference target $y^\star = (y_1^\star, \ldots, y_T^\star)$ available during training. In the default scoring template, this target is the serialized ground-truth field and may contain only the final answer rather than a complete derivation. The score provides target-conditioned supervision for critique learning; it does not verify intermediate reasoning steps.

$$r_k^{\mathrm{D}} = \frac{1}{T}\sum_{t=1}^{T}\left[\log \pi_{\mathrm{ref}}(y_t^\star \mid q, \tau^-, c_k, y_{<t}^\star) - \log \pi_{\mathrm{ref}}(y_t^\star \mid q, \tau^-, y_{<t}^\star)\right]. \tag{1}$$

A positive r_k^{D} means that the critique makes the target more likely than the failed answer alone does, even though its sampled revision is still wrong.

Why this design. Target likelihood can vary across critiques even when all sampled revisions are incorrect. Freezing the reference model keeps the scoring function fixed as the trainable policy changes. The critique generator receives only the problem and failed answer, without the reference target. For an all-wrong refinement group, the directional advantage is applied only to critique tokens; the incorrect revision tokens receive no direct actor-loss contribution from this reward. If all critiques receive the same directional score, their centered directional advantages remain zero. The second term in Equation 1 is shared by all critiques of the same failure and therefore leaves group-relative advantages unchanged; it makes the sign of r_k^{D} interpretable as helping or hurting relative to no critique. Since r_k^{D} is a target-conditioned proxy for direction rather than a guarantee of correction, Table 6 tests whether critiques with higher directional reward also lead to more successful independent corrections.

Outcome supervision for successful refinement groups. When at least one revision is correct, outcomes are informative and supervise both spans. The revision span receives $r_k^{\mathrm{R}} = r(\tilde{\tau}_k)$. The critique span receives a correction-based reward

$$r_k^{\mathrm{C}} = \left[\lambda_{\mathrm{mode}} d_k^{\mathrm{mode}} + \lambda_{\mathrm{succ}} r_k^{\mathrm{R}} + \lambda_{\mathrm{align}} r_k^{\mathrm{R}} a_k - \lambda_{\mathrm{leak}} \ell_k\right]_+, \tag{2}$$

where d_k^{mode} indicates whether the revision departs from the modal initial answer, $a_k \in [0,1]$ measures critique–revision alignment, and $\ell_k \in \{0,1\}$ flags direct answer leakage. To balance directional and outcome supervision, we retain at most $\lfloor \eta N_{\mathrm{batch}}^{+} \rfloor$ all-wrong groups per batch, where N_{batch}^{+} is the number of groups containing a correct revision.

Span-wise advantage routing. Because critique, revision, and directional rewards have different semantics, we normalize them separately. For $g \in \{\mathrm{C}, \mathrm{R}, \mathrm{D}\}$, let $\mathcal{G}_g$ denote samples sharing the same prompt and reward type, and define $\widehat{A}_k^g = (r_k^g - \mathrm{mean}_{j\in\mathcal{G}_g} r_j^g)/(\mathrm{std}_{j\in\mathcal{G}_g} r_j^g + \delta)$. We then route each advantage only to the span it evaluates:

$$\widehat{A}_{k,t} = \begin{cases} m_{k,t}^{\mathrm{C}} \widehat{A}_k^{\mathrm{C}} + m_{k,t}^{\mathrm{R}} \widehat{A}_k^{\mathrm{R}}, & \text{if the refinement group succeeds,} \\ \alpha_{\mathrm{D}} m_{k,t}^{\mathrm{C}} \widehat{A}_k^{\mathrm{D}}, & \text{if the refinement group is all-wrong,} \end{cases} \tag{3}$$

where $\alpha_{\mathrm{D}} > 0$ weights the directional signal. Substituting $\widehat{A}_{k,t}$ into Equation **??** updates both spans in successful groups, but only the critique span in all-wrong groups.

3.3 Internalizing Critique-Guided Solutions (Stage II)

Stage I produces correct revisions, but the model writes each of them after seeing a failed answer and a critique. Neither is available at inference, where the solver sees only the problem. Stage II closes this gap by training the solver to produce these solutions directly from the problem.

Recontextualization. We use *recontextualization* to mean placing a solution found under one context into another: here, from the critique-guided context (q, τ^-, c) to the problem q alone. Concretely, consider a verified revision $\tau^+ \sim \pi^{\mathrm{S}}_{\theta_{\mathrm{roll}}}(\cdot \mid q, \tau^-, c)$. We keep the solution but remove the failed answer τ^- and the critique c, so that τ^+ is treated as an answer to q alone. We then place it in the direct group $\mathcal{G}(q)$ in place of one failed answer, so that the group now contains a correct solution the solver could not find by itself; replacing more failed answers is ablated in Section 4.3. This rebuilding is part of data construction and happens before any gradient step.

Training objective. We optimize the rebuilt group, which contains both the original direct responses and the inserted solution:

$$J_{\mathrm{II}}(\theta) = J_{\mathrm{direct}}(\theta) + J_{\mathrm{int}}(\theta). \tag{4}$$

A common group-relative advantage $\widehat{A}^{\mathrm{T}}_j$ is computed over the complete group using Equation **??**. For original responses, the behavior-policy ratio remains valid, and we use the standard GRPO objective

$$J_{\mathrm{direct}}(\theta) = \mathbb{E}_{j,t}\Big[(1 - M^{\mathrm{int}}_{j,t}) \min\Big(\rho_{j,t}(\theta)\widehat{A}^{\mathrm{T}}_j, \mathrm{clip}(\rho_{j,t}(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}^{\mathrm{T}}_j\Big)\Big]. \tag{5}$$

The inserted solution, however, was sampled under critique-conditioned context, so evaluating it under the original problem prompt does not recover its actual behavior probability. We treat it as an off-policy training example and use a bounded objective to modulate its contribution according to its likelihood under the critique-free prompt. We therefore adopt the policy-shaping transform of LUFFY (Yan et al., 2025): for each token, we evaluate $p_{j,t}(\theta) = \pi^{\mathrm{S}}_\theta(y_{j,t} \mid q, y_{j,<t})$ under the critique-free prompt and define $\psi_{j,t}(\theta) = p_{j,t}(\theta)/(p_{j,t}(\theta) + \beta)$ with $\beta > 0$.

The transform is bounded in $(0, 1)$ and requires no behavior probability. We optimize the clipped internalization objective

$$\begin{aligned} J_{\mathrm{int}}(\theta) = \mathbb{E}_{j,t}\Big[M^{\mathrm{int}}_{j,t} \min\big(&\psi_{j,t}(\theta)\widehat{A}^{\mathrm{T}}_j, \\ &\mathrm{clip}(\psi_{j,t}(\theta), 1-\epsilon_{\mathrm{off}}, 1+\epsilon_{\mathrm{off}})\widehat{A}^{\mathrm{T}}_j\big)\Big], \end{aligned} \tag{6}$$

where $M^{\mathrm{int}}_{j,t}$ marks tokens from the inserted solution. Min–clip is applied directly to ψ, without dual clipping or the on-policy ratio ρ. Thus, J_{direct} preserves learning from on-policy responses, while J_{int} transfers critique-guided discoveries to the original problem prompt, so that the solver can answer directly at inference.

In the default configuration, inserted solutions are verified correct and have nonnegative group-relative advantages. Since $0 < \psi < 1$, the min-clip expression reduces to $\widehat{A}\psi$ on these tokens, so the modulation comes from the probability transform rather than active clipping. For $z = \log p$, the transform satisfies $\partial\psi/\partial z = \psi(1-\psi) \le 1/4$; for the weighted term $\widehat{A}\psi$, the corresponding bound is $|\widehat{A}|/4$. These are scalar sensitivity bounds, not a bound on the full parameter gradient or a trust-region guarantee.

3.4 Optional Extension: Sequential Dynamic Sampling

A single round of refinement can leave every revision wrong even when the critiques point in useful directions, because some problems need several rounds of correction. When extra training compute is available, *sequential dynamic sampling* follows these directions further on unresolved branches only. For each failed branch of an all-wrong refinement group, let $\tilde{\tau}^{(1)}$ be its first-round revision. Each active branch generates $(c^{(i)}, \tilde{\tau}^{(i+1)}) \sim \pi_{\theta_{\mathrm{roll}}}(\cdot \mid q, \tilde{\tau}^{(i)})$ and stops once $r(\tilde{\tau}^{(i+1)}) = 1$; otherwise, $\tilde{\tau}^{(i+1)}$ becomes the failed answer for the next round, up to K_{extra} additional rounds. Verified terminal answers are recontextualized in the same way, while unresolved branches leave the direct group unchanged. The main configuration uses $K_{\mathrm{extra}} = 0$; optional settings with $K_{\mathrm{extra}} > 0$ are evaluated separately in Section 4.3. This extension changes training only; direct evaluation uses the same problem-only prompting protocol as the main configuration.

Table 1: Mathematical reasoning performance (%, higher is better). AIME24/25/26 and AMC23 report mean@8 (m@8) and pass@8 (p@8); MinervaMath, MATH-500, and OlympiadBench report pass@1 (p@1). **Bold** indicates the highest value in each metric column within the same backbone, including ties. Avg. is the equal-weight mean over seven benchmarks, using m@8 for AIME24/25/26 and AMC23 and p@1 for MinervaMath, MATH-500, and OlympiadBench; p@8 is excluded. Signpost uses $K_{\text{extra}} = 0$.

Method	AIME24		AIME25		AIME26		AMC23		Minerva	MATH-500	Olympiad	Avg.
	m@8	p@8	m@8	p@8	m@8	p@8	m@8	p@8	p@1	p@1	p@1	
Qwen3-8B												
Base model	24.6	43.3	19.6	33.3	18.8	43.3	67.8	90.0	42.6	81.6	53.0	44.0
GRPO (Shao et al., 2024)	57.5	76.7	40.0	56.7	46.2	63.3	91.5	97.5	46.6	95.0	68.0	63.5
DAPO (Yu et al., 2025)	53.8	73.3	49.1	60.0	49.5	70.0	92.1	95.0	47.0	95.0	69.2	65.1
Scaf-GRPO (Zhang et al., 2026c)	50.7	76.7	41.6	53.3	48.0	63.3	88.7	97.5	47.4	93.4	65.8	62.2
Critique-GRPO (Zhang et al., 2026b)	45.9	70.0	36.5	53.3	40.9	60.0	85.8	95.0	48.1	95.2	69.5	60.3
Signpost (Ours)	**58.8**	**83.3**	**51.6**	**66.7**	**52.1**	**73.3**	**94.0**	**100.0**	**49.2**	**96.6**	**72.1**	**67.8**
Qwen3-14B												
Base model	26.7	46.7	25.0	43.3	20.4	40.0	71.9	95.0	45.2	86.4	56.4	51.1
GRPO (Shao et al., 2024)	55.0	73.1	46.2	60.7	48.3	66.6	92.0	97.2	47.4	94.2	68.7	64.5
DAPO (Yu et al., 2025)	58.1	76.7	46.7	67.3	50.1	70.0	**95.8**	**100.0**	48.1	87.2	69.8	65.1
Scaf-GRPO (Zhang et al., 2026c)	59.0	76.6	49.1	60.7	49.5	63.3	94.6	97.8	47.4	93.4	65.8	65.5
Critique-GRPO (Zhang et al., 2026b)	55.0	73.1	43.4	63.3	43.3	63.3	93.4	95.6	47.4	95.2	69.5	63.9
Signpost (Ours)	**64.1**	**81.7**	**54.8**	**77.3**	**55.6**	**76.6**	95.2	**100.0**	**48.9**	**97.2**	**73.2**	**69.9**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and Evaluation Metrics. We evaluate Signpost on seven mathematical reasoning benchmarks: AIME24, AIME25, AIME26, AMC23, MinervaMath, MATH-500, and OlympiadBench. To examine transfer beyond mathematics, we also evaluate on ARC-C (Clark et al., 2018), MMLU-Pro (Wang et al., 2024), and GPQA Diamond (Rein et al., 2023) without target-domain fine-tuning. We report mean@8 and pass@8 on AIME24/25/26 and AMC23, measuring average response correctness and problem coverage with eight sampled responses, respectively. All remaining benchmarks use pass@1 with one sampled response per problem. Dataset descriptions and evaluation protocols are provided in Appendices B and I.

Baselines. Our comparison includes two groups of reinforcement-learning baselines: outcome-based optimization methods, GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025), and guided-exploration methods, Scaf-GRPO (Zhang et al., 2026c) and Critique-GRPO (Zhang et al., 2026b). All methods use the same backbone within each comparison and are evaluated as direct solvers without critiques or test-time refinement. Further descriptions of the baselines are available in Appendix C.

Implementation Details. We use Qwen3-8B and Qwen3-14B as backbones in non-thinking mode and implement our two-stage training pipeline in verl. Both stages use an 8K-problem subset of DeepScaleR (Luo et al., 2025). The main configuration performs one-step refinement during training, with no additional sequential attempts ($K_{\text{extra}} = 0$). Further training configurations and evaluation details are provided in Appendices D and I.

Table 2: Out-of-distribution performance (pass@1, %). Models are trained on mathematics without target-domain fine-tuning. **Bold** numbers mark the best result per benchmark within each backbone, including ties.

Method	ARC-C	MMLU-Pro	GPQA Diamond
Qwen3-8B			
Base model	86.6	27.2	47.4
GRPO (Shao et al., 2024)	90.7	36.9	60.0
DAPO (Yu et al., 2025)	91.4	42.4	64.6
Scaf-GRPO (Zhang et al., 2026c)	92.3	45.1	66.7
Critique-GRPO (Zhang et al., 2026b)	91.4	46.0	65.4
Signpost (Ours)	**94.1**	**52.5**	**68.6**
Qwen3-14B			
Base model	88.3	30.7	50.8
GRPO (Shao et al., 2024)	91.4	40.0	64.0
DAPO (Yu et al., 2025)	**95.2**	46.5	**70.1**
Scaf-GRPO (Zhang et al., 2026c)	92.8	43.4	66.3
Critique-GRPO (Zhang et al., 2026b)	92.0	44.6	65.7
Signpost (Ours)	94.8	**53.0**	69.5

4.2 Main Results

Improved reasoning without critique at inference. Signpost achieves the highest average mathematical reasoning scores on both backbones (Table 1): 67.8% on Qwen3-8B and 69.9% on Qwen3-14B, exceeding the strongest baseline averages by 2.7 and 4.4 percentage points, respectively. It outperforms Scaf-GRPO and Critique-GRPO across all seven benchmarks and improves both mean@8 and pass@8 over GRPO on all three AIME benchmarks, demonstrating gains in response correctness and solution coverage. These benefits also extend beyond mathematics (Table 2): Signpost leads on all three out-of-distribution benchmarks with Qwen3-8B and improves over GRPO on every benchmark with both backbones. Most notably, it surpasses the strongest MMLU-Pro baseline by 6.5 percentage points at both model sizes, while remaining competitive with DAPO on Qwen3-14B ARC-C and GPQA Diamond, with gaps of 0.4 and 0.6 points. Since all evaluations use direct responses without critiques or refinement, and the out-of-distribution tasks receive no target-domain fine-tuning, these results support the effectiveness of learning to explore through self-critique and transferring its benefits to unassisted reasoning across domains. Section 4.3 examines the contributions of directional critique learning and solution internalization separately.

4.3 Ablation Studies

Ablation study of core components. Removing directional rewards reduces both correction accuracy and direct-solving performance (Table 3), supporting the effectiveness of our proposed directional critique learning. Shuffling these rewards further degrades performance, indicating that the gains depend on distinguishing the utility of individual critiques even when all revisions fail, rather than merely introducing additional reward variation. Removing answer supervision further degrades performance, supporting its complementary role. In contrast, inserting more verified solutions preserves correction accuracy but weakens direct solving. Thus, successful discovery alone is insufficient: internalization benefits from selectively incorporating guided solutions while retaining the policy's own direct-solving trajectories.

Ablation study of refinement depth. Additional refinement rounds improve both correction accuracy and direct-solving coverage (Table 4). With two extra rounds, correction accuracy increases from 10.8% to 15.2%, and pass@8 improves on all three AIME benchmarks. Because evaluation uses no test-time refinement, these gains show that deeper training-time exploration can translate into stronger unassisted reasoning, rather than merely improving correction with feedback.

Ablation study of the training schedule. Sequential Stage-I→Stage-II training outperforms Stage I alone, Stage I alone, joint optimization, and Stage-I→GRPO on all three benchmarks (Table 5). The comparison with continued GRPO indicates that further outcome-based training does

Table 3: Component ablations on Qwen3-8B with $K_{\text{extra}} = 0$ and default $m = 1$. Refinement success is measured on all-wrong rollout groups at entry to Stage II; AIME scores are post-training QA pass@8. All scores are percentages. In the "Shuffled directional rewards" ablation, we multiply the directional rewards by -1 before computing the advantages, thereby reversing the direction of the reward signal.

Variant	Refinement success	QA evaluation (p@8)		
		AIME24	AIME25	AIME26
Signpost (Ours)	**10.8**	**83.3**	**66.6**	**73.3**
w/o directional reward	8.7	80.0	60.0	66.6
Shuffled directional rewards	6.0	73.3	60.0	63.3
w/o answer supervision	5.3	70.0	63.3	60.0
Replace up to 2 ($m = 2$)	**10.8**	80.0	63.3	63.3
Replace up to 4 ($m = 4$)	**10.8**	76.6	60.0	63.3

Table 4: Refinement-depth ablations on Qwen3-8B ($m = 1$). K_{extra} counts extra rounds after the initial revision. Refinement success is measured on all-wrong rollout groups at entry to Stage II; AIME scores are post-training QA pass@8. All scores are percentages.

Variant	Refine	AIME24 p@8	AIME25 p@8	AIME26 p@8
Signpost + refinement ($K_{\text{extra}} = 2$)	**15.2**	**86.6**	**76.6**	**80.0**
Signpost + refinement ($K_{\text{extra}} = 1$)	12.4	83.3	73.3	76.6
Signpost ($K_{\text{extra}} = 0$)	10.8	83.3	66.6	73.3

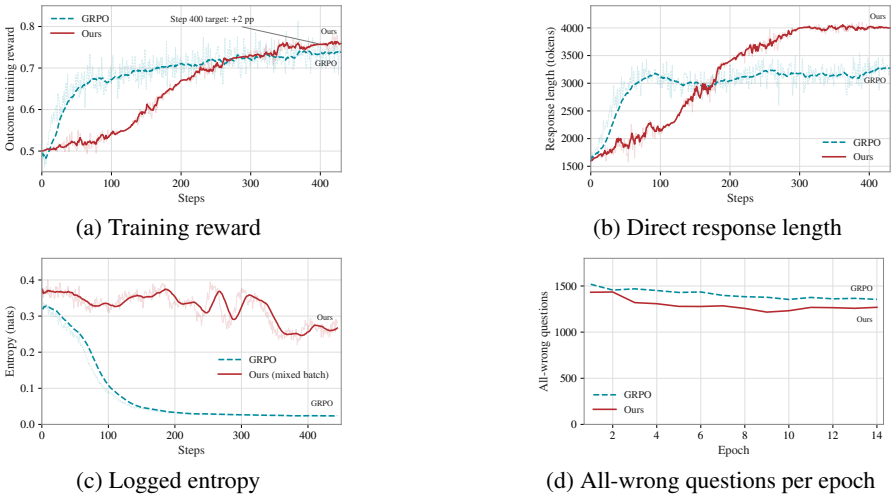
Table 5: Training-schedule ablations on Qwen3-8B. AIME24, AIME25, and AIME26 report direct-solving mean@8 (m@8) and pass@8 (p@8). All scores are percentages. All schedules use a total training budget of 14 epochs. Stage I $\rightarrow$ Stage II and Stage I $\rightarrow$ GRPO each allocate 4 epochs to Stage I (*Learning Directions*), followed by 10 epochs of Stage II or GRPO, respectively. Without Stage I, the model trains with Stage II alone for all 14 epochs. Mixed training jointly optimizes the Stage-I and Stage-II losses with equal weights throughout the 14 epochs. **Bold** numbers mark the best result per benchmark within each benchmark based on the different training schedule.

Training schedule	AIME24		AIME25		AIME26	
	m@8	p@8	m@8	p@8	m@8	p@8
Signpost(Stage I $\rightarrow$ Stage II)	**58.8**	**83.3**	**51.6**	**66.6**	**52.1**	**73.3**
Without Stage I	44.2	73.3	40.0	56.7	45.0	60.0
Without Stage II	38.7	66.7	27.5	50.0	30.4	56.7
Mixed Stage-I/II losses	53.3	70.0	40.0	53.3	44.3	56.7
Stage I $\rightarrow$ GRPO	55.4	76.8	42.0	59.1	46.3	60.0

not fully substitute for explicitly internalizing critique-guided solutions. The advantage over joint optimization further supports separating direction learning from solution internalization: learning useful guidance first, then transferring its discoveries to direct solving.

4.4 Training Dynamics

Learning from Self-Critique and Maintaining Exploration Potential. Figure 2 shows that our method initially has both lower direct-answer reward and shorter responses than GRPO. After entering Stage II, both metrics rise rapidly and eventually stabilize above GRPO (Figures 2(a) and 2(b)). Meanwhile, the number of all-wrong question groups remains below GRPO through most of the observed epochs (Figure 2(d)) and figure 2(c) shows that our method consistently maintains higher entropy than GRPO model. It indicates that signpost enabling the solver to draw on reasoning paths that were previously difficult to generate independently encourageing model to maintain exploration potential. Overall, compared with online GRPO, our method achieves higher direct-answer reward, fewer all-wrong groups, longer responses and higher entropy. Together, these trends suggest that our framework promotes more extensive reasoning exploration while enabling the model to discover correct solutions more effectively.



(a) Training reward

(b) Direct response length

(c) Logged entropy

(d) All-wrong questions per epoch

Figure 2: Training dynamics of GRPO and Signpost on Qwen3-8B: (a) direct-answer reward, (b) response length, (c) policy entropy, and (d) all-wrong question groups per epoch. For Signpost, entropy is averaged over direct question-to-answer ($q \to a$) responses only, excluding critiques and revisions.

Table 6: Directional reward versus independent correction success on screened all-wrong inputs. Five equal-count bins are defined by reward rank within each failed input, from lowest (B1) to highest (B5). Statistics are aggregated with equal problem weight.

Reward-rank bin	B1	B2	B3	B4	B5
Correction rate (%)	1.67	4.17	5.83	5.00	5.00

Diagnostic	Estimate [95% CI]
Within-input ROC-AUC	0.563 [0.510, 0.681]

Protocol. Correction rates use fresh revision trials, averaging within each problem and then across problems. Reward ties use outcome-independent ordering for bins. Within-input ROC-AUC uses $r^{\mathrm{D}}_{x,k}$ to score each fresh trial's binary outcome; inputs without both outcome classes are excluded, and AUC ties count as one half. The confidence interval resamples whole problems.

4.5 Analysis of Directional Rewards

Ranking quality of directional rewards. Table 6 shows that fresh correction rates increase from 1.67% in B1 to 4.17% in B2 and peak at 5.83% in B3, while B4 and B5 both achieve 5.00%; the relationship is therefore positive overall but not strictly monotonic. The within-input ROC-AUC is 0.563 (95% CI [0.510, 0.681]), indicating modest discrimination above chance. These results support that this pattern is consistent with target-likelihood rewards capturing some information about a critique's ability to guide correction even when all screening revisions fail, using directional rewards as a proxy, which can provide differentiated supervision for critique utility in all-wrong refinement groups.

5 Conclusion

We introduced Signpost, a training framework based on Critique as Direction that enables learning beyond failed answers and revisions. By rewarding critiques for increasing the likelihood of a target solution under a frozen reference model, Signpost provides a learning signal when outcome-based relative advantages vanish, without rewarding incorrect answers. It then transfers verified critique-guided discoveries into question-only solving through a bounded-likelihood objective. Experiments on mathematical reasoning show that Signpost outperforms GRPO, DAPO, Scaf-GRPO, and Critique-GRPO, improving Qwen3-8B pass@8 over GRPO by 6.6–10.0 percentage points across AIME 2024–2026.

6 AI USE STATEMENT

We used LLMs (GPT and Claude) as tools to support this work. Firstly, they were used for polishing English wording of the manuscript, checking grammar, and assisting with LaTeX formatting and the preparation of comparison tables. Every technical content, including the research ideas, the experimental design, and the analysis of results, was conducted solely by the authors. Sometimes,author will use deepresearch to analysis articles. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the content of this paper.

REFERENCES

Priyank Agrawal, Ankur Samanta, Shervin Ghasemlou, Boris Vidolov, Jalaj Bhandari, Kavosh Asadi, Daniel Jiang, and Aditya Modi. Off-context GRPO: Learning to reason on hard problems using privileged information. *arXiv preprint arXiv:2607.19313*, 2026. doi: 10.48550/arXiv.2607.19313. URL https://arxiv.org/abs/2607.19313.

Bin Chen, Xinye Liao, Yiming Liu, Xin Liao, and Chonghan Liu. CAPF: Guiding search-agent rollouts with credit-attenuated privileged feedback. *arXiv preprint arXiv:2606.01830*, 2026. doi: 10.48550/arXiv.2606.01830. URL https://arxiv.org/abs/2606.01830.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. doi: 10.48550/arXiv.1803.05457. URL https://arxiv.org/abs/1803.05457.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025. doi: 10.48550/arXiv.2502.01456. URL https://arxiv.org/abs/2502.01456.

Ken Ding. HDPO: Hybrid distillation policy optimization via privileged self-distillation. *arXiv preprint arXiv:2603.23871*, 2026. doi: 10.48550/arXiv.2603.23871. URL https://arxiv.org/abs/2603.23871.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Difan Jiao, Qianfeng Wen, Blair Yang, Zhenwei Tang, and Ashton Anderson. ThinkTwice: Jointly optimizing large language models for reasoning and self-refinement. *arXiv preprint arXiv:2604.01591*, 2026. doi: 10.48550/arXiv.2604.01591. URL https://arxiv.org/abs/2604.01591.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/871ac99fdc5282d0301934d23945ebaa-Abstract-Conference.html.

Ariel Kwiatkowski, Natasha Butt, Ismail Labiad, Julia Kempe, and Yann Ollivier. Likelihood-based reward designs for general LLM reasoning. *arXiv preprint arXiv:2602.03979*, 2026. doi: 10.48550/arXiv.2602.03979. URL https://arxiv.org/abs/2602.03979.

Jianbo Lin, Xiaomin Yu, Yi Xin, Yifu Guo, Zhuosong Jiang, Zhongqi Yue, Weishi Wang, Heqing Zou, Chengwei Qin, and Hui Xiong. ICRL: Learning to internalize self-critique with reinforcement learning. *arXiv preprint arXiv:2605.15224*, 2026. doi: 10.48550/arXiv.2605.15224. URL https://arxiv.org/abs/2605.15224.

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. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2503.20783.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. DeepScaleR: Surpassing O1-Preview with a 1.5B model by scaling RL. Notion blog, 2025. URL https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

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. *arXiv preprint arXiv:2311.12022*, 2023. doi: 10.48550/arXiv.2311.12022. URL https://arxiv.org/abs/2311.12022.

Mengjie Ren, Jie Lou, Boxi Cao, Xueru Wen, Hongyu Lin, Xianpei Han, Le Sun, Xing Yu, and Yaojie Lu. Learning from failures: Correction-oriented policy optimization with verifiable rewards. *arXiv preprint arXiv:2605.14539*, 2026. doi: 10.48550/arXiv.2605.14539. URL https://arxiv.org/abs/2605.14539.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300. URL https://arxiv.org/abs/2402.03300.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (PLE): A novel multi-task learning (MTL) model for personalized recommendations. In *Proceedings of the Fourteenth ACM Conference on Recommender Systems*, pp. 269–278, 2020. doi: 10.1145/3383313.3412236. URL https://doi.org/10.1145/3383313.3412236.

Vu Duc Anh, Nhat M. Hoang, Do Xuan Long, Cong-Duy Nguyen, Ponhvoan Srey, and Luu Anh Tuan. Reinforcing step-level reasoning for effective self-correction in LLMs. *arXiv preprint arXiv:2608.11573*, 2026. doi: 10.48550/arXiv.2608.11573. URL https://arxiv.org/abs/2608.11573.

Jingyi Wang, Lei Zhu, Tengjin Weng, Song-Li Wu, Haochen Tan, Jierun Chen, Chaofan Tao, Haoli Bai, Lu Hou, Lifeng Shang, and Xiao-Ping Zhang. GRPO-VPS: Enhancing group relative policy optimization with verifiable process supervision for effective reasoning. *arXiv preprint arXiv:2604.20659*, 2026. doi: 10.48550/arXiv.2604.20659. URL https://arxiv.org/abs/2604.20659.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024. doi: 10.48550/arXiv.2406.01574. URL https://arxiv.org/abs/2406.01574.

Yubo Wang, Xiang Yue, and Wenhu Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate. *arXiv preprint arXiv:2501.17703*, 2025. doi: 10.48550/arXiv.2501.17703. URL https://arxiv.org/abs/2501.17703.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, Xiao Wang, Rui Zheng, Tao Ji, Xiaowei Shi, Yitao Zhai, Rongxiang Weng, Jingang Wang, Xunliang Cai, Tao Gui, Zuxuan Wu, Qi Zhang, Xipeng Qiu, Xuanjing Huang, and Yu-Gang Jiang. Enhancing LLM reasoning via critique models with test-time and training-time supervision. *arXiv preprint arXiv:2411.16579*, 2024. doi: 10.48550/arXiv.2411.16579. URL https://arxiv.org/abs/2411.16579.

Changyi Xiao, Caijun Xu, and Yixin Cao. Reinforcement learning with conditional expectation reward. *arXiv preprint arXiv:2603.10624*, 2026. doi: 10.48550/arXiv.2603.10624. URL https://arxiv.org/abs/2603.10624.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. In *Advances in Neural Information Processing*

Systems, volume 38, pp. 129799–129828, 2025. doi: 10.52202/085713-3904. URL https://openreview.net/forum?id=vO8LLoNWWk.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. doi: 10.48550/arXiv.2503.14476. URL https://arxiv.org/abs/2503.14476.

Junjie Zhang, Guozheng Ma, Shunyu Liu, Zetian Hu, Yongcheng Jing, Ting-En Lin, Yongbin Li, and Dacheng Tao. STRIDE: Learnable stepwise language feedback for LLM reasoning. *arXiv preprint arXiv:2605.18851*, 2026a. doi: 10.48550/arXiv.2605.18851. URL https://arxiv.org/abs/2605.18851.

Xiaoying Zhang, Yipeng Zhang, Hao Sun, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen M. Meng. Advancing LLM reasoning with natural language and numerical feedback. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=gz7hVnrRWq. Spotlight; preprint arXiv:2506.03106 first posted in 2025.

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, 2026c.

A PRELIMINARY

We first introduce the reasoning setting, group-relative optimization, and the self-critique formulation used in Section 3.

Problem formulation. Given a problem q, a language-model policy π_θ generates a reasoning trajectory $\tau = (y_1, \ldots, y_T)$ autoregressively. A verifier compares its extracted final answer with the reference answer $a^\star$ and assigns a binary outcome reward $r(\tau) \in \{0, 1\}$. The training objective is to improve the expected correctness of direct answers:

$$J(\theta) = \mathbb{E}_{q\sim\mathcal{D},\, \tau\sim\pi_\theta(\cdot|q)}\big[r(\tau)\big], \qquad \pi_\theta(\tau \mid q) = \prod_{t=1}^{T} \pi_\theta(y_t \mid q, y_{<t}). \tag{7}$$

The verifier evaluates the final answer rather than assigning credit to individual reasoning steps. Consequently, two incorrect trajectories receive the same outcome reward even if their intermediate reasoning differs substantially.

Group relative policy optimization. GRPO (Shao et al., 2024) samples a group $\mathcal{G}(q) = \{\tau_i\}_{i=1}^{G}$ from a behavior policy $\pi_{\theta_{\text{old}}}$ and computes a group-relative advantage:

$$\widehat{A}_i = \frac{r(\tau_i) - \overline{r}_q}{\sigma_q + \delta}, \qquad \overline{r}_q = \frac{1}{G}\sum_{j=1}^{G} r(\tau_j), \qquad \sigma_q = \mathrm{std}_{j\in\{1,\ldots,G\}}\, r(\tau_j), \tag{8}$$

where $\delta > 0$ ensures numerical stability. The trajectory-level advantage is shared by its response tokens. With the token-wise probability ratio

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid q, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid q, y_{i,<t})}, \tag{9}$$

the clipped policy objective is

$$J_{\text{GRPO}}(\theta) = \mathbb{E}_{i,t}\Big[\min\Big(\rho_{i,t}(\theta)\widehat{A}_i, \mathrm{clip}\,\big(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon\big)\widehat{A}_i\Big)\Big], \tag{10}$$

where ϵ controls clipping and the expectation is over sampled response tokens. Group-relative comparison supplies credit without a separately learned value function, but requires variation in the rewards being compared.

Self-critique and refinement. The same policy serves as a solver π_θ^{S} and a critic π_θ^{C} through different prompts. For a failed answer τ^-, the critic generates $c \sim \pi_\theta^{\mathrm{C}}(\cdot \mid q, \tau^-)$, and the solver generates a revision $\tilde{\tau} \sim \pi_\theta^{\mathrm{S}}(\cdot \mid q, \tau^-, c)$. Sampling K such pairs gives a refinement group $\mathcal{G}_{\mathrm{R}}(q, \tau^-) = \{(c_k, \tilde{\tau}_k)\}_{k=1}^{K}$. Critique and revision tokens are distinguished by masks $m_{k,t}^{\mathrm{C}}$ and $m_{k,t}^{\mathrm{R}}$, allowing separate credit assignment despite shared parameters.

Two levels of vanishing outcome supervision. If all direct answers are wrong, their rewards are identical and Equation 8 yields zero advantages. Refinement can recover a learning signal when its outcomes differ, but centered group-relative normalization encounters the same limitation when every revision also fails. In this second case, binary correctness cannot distinguish a useful critique from an unhelpful one. The vanishing statement concerns the outcome-based surrogate, not any separate regularization term. Signpost addresses these two failures by learning critique directions beyond sampled correctness and internalizing verified revisions under the original problem prompt, as detailed next.

B DATASETS

DeepScaleR (Training). The DeepScaleR dataset (Luo et al., 2025) contains approximately 40,000 distinct mathematical question–answer pairs collected from AIME examinations from 1984–2023, AMC examinations before 2023, Omni-MATH, and Still. It provides final-answer annotations and reference solutions where available, supporting verifiable-reward training on mathematical reasoning tasks. We use an 8K-problem subset for both stages of Signpost training; the datasets below serve as evaluation benchmarks.

AIME24, AIME25, and AIME26. AIME24, AIME25, and AIME26 contain problems from the 2024, 2025, and 2026 American Invitational Mathematics Examinations, respectively. These competition problems require multi-step mathematical reasoning and produce short integer answers. The three editions provide separate evaluations of a model's ability to construct a correct solution rather than select an answer from a list of candidates.

AMC23. AMC23 is a competition-mathematics benchmark drawn from the 2023 American Mathematics Competitions. Its problems test mathematical problem solving through quantitative reasoning and the application of mathematical concepts. In our evaluation, models generate solutions in free form, and correctness is determined from the extracted final answer.

MinervaMath. MinervaMath evaluates quantitative reasoning on mathematical and scientific problems. Unlike competition-only benchmarks, it includes questions that require applying mathematical tools to scientific settings, linking domain knowledge with calculation and symbolic reasoning. Each response is assessed using its final answer, rather than a direct evaluation of every intermediate reasoning step.

MATH-500. MATH-500 is a 500-problem subset of the MATH benchmark, spanning subjects such as algebra, geometry, number theory, counting and probability, and precalculus. It includes problems at different difficulty levels and provides reference answers and worked solutions. This breadth supports evaluation across mathematical topics rather than within a single competition or subject area.

OlympiadBench. OlympiadBench is a challenging benchmark built from olympiad and university-entrance examination problems, with mathematical and physical reasoning tasks in its full collection. The problems are accompanied by detailed expert solutions and often require several connected reasoning steps. We use its mathematical-reasoning evaluation set and assess final-answer correctness, rather than treating the benchmark as a test of automatically verified proofs.

ARC-C. ARC-C is the Challenge subset of the AI2 Reasoning Challenge (Clark et al., 2018), consisting of multiple-choice science questions at the grade-school level. The subset was designed to contain questions that are difficult for retrieval and word-co-occurrence methods, encouraging reasoning beyond surface-level matching. We use the ARC-Challenge test split to examine generalization from mathematical training to science question answering.

MMLU-Pro. MMLU-Pro (Wang et al., 2024) is a broad, multidisciplinary benchmark covering areas including science, engineering, humanities, and social sciences. Compared with the original MMLU, it emphasizes more demanding reasoning questions and expands the answer-choice set to increase the difficulty of identifying the correct option. It complements the mathematical benchmarks by testing knowledge-intensive reasoning across different subject areas.

GPQA Diamond. GPQA Diamond is a subset of GPQA (Rein et al., 2023), a graduate-level multiple-choice question-answering benchmark developed by domain experts. Its questions concern biology, physics, and chemistry, combining specialized scientific knowledge with reasoning. It provides a demanding setting for assessing transfer beyond competition mathematics, where a correct answer may depend on both domain understanding and multi-step inference.

C Baselines

We compare Signpost with four representative reinforcement-learning methods for improving language-model reasoning: GRPO, DAPO, Scaf-GRPO, and Critique-GRPO. These methods cover outcome-based optimization, scaffold-assisted exploration, and critique-guided refinement. Our comparisons use Qwen3-8B and Qwen3-14B, with all methods evaluated as direct solvers without critiques or test-time refinement.

GRPO. Group Relative Policy Optimization (GRPO) (Shao et al., 2024) samples multiple responses to the same problem and estimates their advantages by normalizing outcome rewards within the group. It optimizes a clipped policy objective without a separately trained value model. This provides an outcome-based learning baseline, but groups with identical rewards yield no centered group-relative learning signal.

DAPO. DAPO (Yu et al., 2025) is a reinforcement-learning framework designed to improve the effectiveness of large-scale reasoning training. Its dynamic sampling mechanism filters out prompt groups with uniformly correct or uniformly incorrect responses and samples additional groups to maintain informative reward variation in the training batch. This approach sustains outcome-based optimization by selecting useful rollout groups, rather than assigning differentiated credit to the unsuccessful responses within an all-wrong group.

Scaf-GRPO. Scaf-GRPO (Zhang et al., 2026c) introduces reasoning scaffolds to help the policy solve difficult problems for which the initial response group contains no correct answer. It progressively supplies hints until a successful hint-conditioned trajectory can replace a failed group member, allowing the reconstructed group to provide a learning signal. This design uses guided exploration to recover successful trajectories that direct sampling alone fails to discover.

Critique-GRPO. Critique-GRPO (Zhang et al., 2026b) combines natural-language feedback with numerical correctness rewards in an online reinforcement-learning framework. When an initial response group contains no correct answer, the policy generates critiques and uses them to revise its failed responses. Initial and revised responses are then combined for group-relative optimization, enabling learning from critique-guided corrections while retaining direct answering at inference.

D Detailed Experimental Setup

Our main comparisons use Qwen3-8B and Qwen3-14B in non-thinking (`no_think`) mode during both training and evaluation, with `enable_thinking=False` in the chat template. The two-stage reinforcement-learning procedure is implemented in verl and trained on an 8K-problem subset of DeepScaleR. Tables 8 and 9 summarize the reproducible Qwen3-8B configurations supplied by the Stage-I frozen-reference directional launcher and the Stage-II off-policy min–clip launcher, respectively. Values resolve shell defaults and inherited configuration fields; environment variables and command-line overrides can change them.

Data and training stages. Both launchers use a filtered DeepScaleR training file and AIME24/AIME25 for in-training validation. Stage I initializes the policy from the local Qwen3-8B model, samples eight direct answers, and generates eight critique–revision pairs for one selected failure per eligible problem. It updates critique and revision spans separately, using a frozen reference scorer for all-wrong directional rewards. Stage II selects all-wrong direct groups, generates one

Table 7: Extended GRPO training versus Signpost on Qwen3-8B. AIME scores are percentages. All-wrong counts initial question groups with no correct direct response within one training epoch (lower is better). **Bold** marks the best value in each performance column.

Method	Steps	AIME24		AIME25		All-wrong / epoch
		m@8	p@8	m@8	p@8	
GRPO	300	53.8	76.7	41.6	56.7	1,355
GRPO	600	56.4	80.0	49.1	63.3	1,308
Signpost (Ours)	420	**58.8**	**83.3**	**51.6**	**66.7**	**1,218**

refinement per selected failed answer, and replaces at most one failed slot with a verified solution. The inserted tokens use $\beta = 0.1$ and $\epsilon_{\text{off}} = 0.2$; critique-conditioned generation does not contribute a separate actor loss. The Stage-II launcher resumes the checkpoint specified by its resume-path setting at global step 120, including optimizer and scheduler state.

Evaluation and reporting. Checkpoints are saved every 10 steps and validated every 50 steps in both recipes. In-training validation uses eight samples at temperature 0.6. Final benchmark reporting follows Appendix I: mean@8/pass@8 on AIME24/25/26 and AMC23, and pass@1 on MinervaMath, MATH-500, and OlympiadBench, without critiques or test-time refinement.

Training subset, checkpoint selection, and compute. Taking Qwen3-8B as an example, we use an 8K-problem subset of DeepScaleR selected from problems on which Qwen3-8B achieves a mean@16 accuracy of 0.5 with a maximum generation length of 8,192 tokens. We select the checkpoint at step 120 of Stage I to initialize Stage II and train for an additional 300 steps to obtain the final checkpoint. On NVIDIA H100 GPUs, our method requires 912.6 GPU-hours of training, compared with 616 GPU-hours for GRPO.

E Comparison with Extended GRPO Training

Experimental setup. We examine whether extending outcome-based training closes the gap to Signpost. Using Qwen3-8B, we evaluate GRPO at 300 and 600 training steps and compare it with Signpost at 420 total steps (120 steps of Stage I followed by 300 steps of Stage II). We report direct-solving mean@8 and pass@8 on AIME24 and AIME25, together with the number of all-wrong training question groups within one epoch. An all-wrong group contains no correct answer among its eight initial direct responses, before any critique-guided refinement or answer replacement.

Results. Doubling GRPO training from 300 to 600 steps improves both accuracy and coverage on AIME24 and AIME25 and reduces the number of all-wrong groups from 1,355 to 1,308 (Table 7). Nevertheless, Signpost at 420 steps outperforms the 600-step GRPO checkpoint on all four evaluation metrics: mean@8 improves by 2.4 and 2.5 percentage points on AIME24 and AIME25, respectively, while pass@8 improves by 3.3 and 3.4 points. Signpost also has 90 fewer all-wrong groups per epoch, a reduction of 6.9% relative to the longer GRPO run.

Thus, extending GRPO training improves its performance but does not close the gap to Signpost in this comparison. Signpost achieves stronger direct solving and fewer all-wrong groups with 30% fewer training steps than 600-step GRPO. This comparison concerns training-step budgets rather than matched wall-clock time or total computation.

Table 8: Default hyperparameters and training configuration for Stage I (Qwen3-8B), resolved from the frozen-reference directional launcher and its inherited configuration.

Name	Value	Description
RL Training		
Initial policy	Qwen3-8B	Local model checkpoint; fresh optimizer state.
Training data	DeepScaleR	Filtered first-eight-response training file.
Total epochs	4	
Prompt batch size	256	Number of input problems per training batch.
PPO mini-batch size	128	Configured mini-batch before rollout expansion.
Micro-batch size / GPU	2	Nominal setting; dynamic batching is enabled.
PPO epochs	1	Optimization passes per collected batch.
Learning rate	10^{-6}	Policy optimizer learning rate.
LR schedule / warmup	Constant / 0	Warmup ratio is zero.
Weight decay	0	Optimizer weight-decay coefficient.
PPO clip (lower / upper)	0.2 / 0.2	On-policy probability-ratio clipping.
Dual-clip constant	3	Additional clipping for negative advantages.
KL loss / reward penalty	Off / Off	KL loss coefficient is zero.
Entropy coefficient	0	No entropy regularization term.
Save / evaluation interval	10 / 50	Checkpoint and validation frequency in steps.
Sampling and Directional Critique Learning		
Initial responses G	8	Direct samples for each problem.
Selected failures / problem	1	Maximum failed initial answers sent to refinement.
Initial-mean threshold	0.5	Refine groups with mean reward at most this value.
Refinements / failure K	8	Inherits the rollout sample count.
Critique / revision updates	On / On	Separate advantages for the two response spans.
Direct-answer update	Off	No separate initial-answer actor update.
Reference scorer	Frozen policy	Likelihood source is `ref`.
Directional scale / clipping	1 / None	Raw target log-likelihood difference.
Directional weight α_{D}	0.5	Multiplier after group normalization.
All-wrong quota η	0.5	Relative to successful groups; schedule disabled.
λ_{mode} / λ_{succ}	0.1 / 0.4	Mode-change and correction-success weights.
λ_{align} / λ_{leak}	0.1 / 0.1	Notion-alignment bonus and leakage penalty.
Selected-answer weight	0	The optional selected-answer-change term is off.
Rollout and System Configuration		
Max prompt / response	12,000 / 8,192	Maximum token lengths.
Training temperature / top-p	1.0 / 1.0	Stochastic rollout sampling.
Rollout engine / TP size	vLLM / 2	Tensor parallelism for generation.
GPU memory utilization	0.8	vLLM memory-utilization setting.
Nodes $\times$ GPUs / node	1×8	GPU model is not specified by the script.
Training strategy	FSDP2	Ulysses sequence-parallel size is 2.
Gradient checkpointing	On	Activation offloading is also enabled.
Actor parameter / optimizer offload	Off / On	Reference-policy parameter offload is enabled.
Overlong filtering	On	Overlong prompts are filtered; truncation errors.

Table 9: Default hyperparameters and training configuration for Stage II (Qwen3-8B), resolved from the off-policy min–clip launcher resumed at step 120. This is the single-replacement, non-serial configuration.

Name	Value	Description
RL Training		
Policy backbone	Qwen3-8B	Same architecture as the resumed policy.
Resume checkpoint	Step 120	Full training state, not weights-only initialization.
Training data	DeepScaleR	Filtered first-eight-response training file.
Total epochs	10	
Prompt batch size	256	Number of input problems per training batch.
PPO mini-batch size	128	Configured mini-batch before rollout expansion.
Micro-batch size / GPU	2	Nominal setting; dynamic batching is enabled.
PPO epochs	1	Optimization passes per collected batch.
Learning rate	10^{-6}	Configured value; optimizer state is resumed.
LR schedule / warmup	Constant / 0	Scheduler state is resumed from the checkpoint.
Weight decay	0	Configured optimizer weight decay.
PPO clip (lower / upper)	0.2 / 0.2	Applies to original direct-response tokens.
Dual-clip constant	3	Used only on the on-policy route.
KL loss / reward penalty	Off / Off	KL loss coefficient is zero.
Entropy coefficient	0	No entropy regularization term.
Save / evaluation interval	10 / 50	Checkpoint and validation frequency in steps.
Refinement and Solution Internalization		
Initial responses G	8	Direct samples for each problem.
Selected failures / problem	Up to 8	Failed responses eligible for refinement.
Initial-mean threshold	0	Only all-wrong initial groups are selected.
Refinements / failure K	1	One critique–revision pair per selected failure.
Verified replacements / group	1	Replace one wrong slot; keep the group size.
Internalization transform	$p/(p+0.1)$	$\beta = 0.1$, inherited actor configuration.
Off-policy clip ϵ_{off}	0.2	Min–clip on inserted-token probabilities.
Replacement-only loss	Off	Train original and inserted response tokens.
Critique / revision actor loss	Off / Off	Generate and verify refinements without this loss.
Additional answer distillation	Off	No separate distillation branch.
Extra rounds K_{extra}	0	Sequential dynamic sampling is disabled.
Success-rate loss scaling	Off	No extra scaling of the replacement objective.
Rollout and System Configuration		
Max prompt / response	12,000 / 8,192	Maximum token lengths.
Training temperature / top-p	1.0 / 1.0	Stochastic rollout sampling.
Rollout engine / TP size	vLLM / 2	Tensor parallelism for generation.
GPU memory utilization	0.8	vLLM memory-utilization setting.
Nodes $\times$ GPUs / node	1×8	GPU model is not specified by the script.
Training strategy	FSDP2	Ulysses sequence-parallel size is 2.
Gradient checkpointing	On	Activation offloading is also enabled.
Actor parameter / optimizer offload	Off / On	Reference-policy parameter offload is enabled.
Overlong filtering	On	Overlong prompts are filtered; truncation errors.

E.1 Reward Design and Implementation

The reward design distinguishes revision correctness, outcome-based critique utility, and directional critique utility. These are not summed into a single trajectory reward: the refinement group's outcomes determine which critique signal is used, and token masks keep critique credit separate from revision credit. The definitions below make the terms in Equations 2 and 3 explicit.

Outcome reward. Let $\mathcal{V}(q, \tau, a^\star) \in \{0, 1\}$ denote the task verifier's final-answer decision. Direct responses and revisions receive

$$r(\tau) = \mathcal{V}(q, \tau, a^\star), \qquad r_k^{\mathrm{R}} = \mathcal{V}(q, \tilde{\tau}_k, a^\star). \tag{11}$$

This reward evaluates the extracted answer, not the validity of every reasoning step. In the implementation, the success feature used in the critique reward is $b_k = \mathbf{1}\{s_k > 0\}$, where s_k is the sum of the revision's reward tensor. For the binary verifier in Equation 11, $b_k = r_k^{\mathrm{R}}$. Group routing separately tests whether any s_k exceeds its configured success threshold; that comparison is strict. With threshold zero, it agrees with the binary-success rule above.

Departure from the modal initial answer. Let $E(\tau)$ extract the final boxed answer, returning $\perp$ when extraction fails, and let $\mathcal{A}_q$ contain the successfully extracted initial answer strings. The modal answer and its change indicator are

$$\begin{aligned} a_q^{\mathrm{mode}} &= \begin{cases} \arg\max_{a \in \mathcal{A}_q} \sum_{i=1}^{G} \mathbf{1}\{E(\tau_i) = a\}, & \mathcal{A}_q \neq \varnothing, \\ \perp, & \mathcal{A}_q = \varnothing, \end{cases} \\ d_k^{\mathrm{mode}} &= \mathbf{1}\big\{E(\tilde{\tau}_k) \neq \perp,\ a_q^{\mathrm{mode}} \neq \perp,\ E(\tilde{\tau}_k) \not\equiv a_q^{\mathrm{mode}}\big\}. \end{aligned} \tag{12}$$

The mode is computed by exact extracted-string counts, not by first merging mathematical equivalence classes. Ties follow first occurrence in the initial group. The subsequent $\equiv$ test uses mathematical equivalence, falling back to stripped-string equality if that test raises an exception. If no answer is extractable, the mode is missing and the change indicator is zero. The feature rewards departure from the initial consensus, but is not itself a correctness test.

Critique–revision alignment. The structured critique proposes a list of concepts in its `correction_notion` field. Normalize each entry by lowercasing, replacing characters other than letters, digits, whitespace, and the arithmetic symbols `+ - * / ^` with spaces, collapsing whitespace, and applying the implementation's suffix-based singularization. The suffix rule maps `ies` to `y`, removes `es` after `ch/sh/x/z/o`, and otherwise removes a final `s` except for `ss` and `series`. Let $\mathcal{N}_k$ be the list of normalized entries after discarding empty entries, entries shorter than four characters, normalized entries occurring as substrings of the raw lowercased question (without applying the notion normalization to the question), and generic entries from

$$\begin{aligned} \mathcal{U} = \{&\text{math}, \text{mathematics}, \text{problem}, \text{answer}, \text{solution}, \text{calculation}, \\ &\text{formula}, \text{step}, \text{reasoning}, \text{logic}, \text{algebra}\}. \end{aligned} \tag{13}$$

Write $\mathrm{occ}(v, x)$ for a substring-occurrence indicator, $\tilde{x}_k$ for the lowercased revised answer, and $W(v)$ for the non-generic whitespace-separated words of notion v. Define

$$\begin{aligned} \mathrm{match}(v, \tilde{x}_k) &= \mathbf{1}\left\{ \mathrm{occ}(v, \tilde{x}_k) = 1 \ \vee \ \left(|W(v)| > 0 \ \wedge \sum_{w \in W(v)} \mathrm{occ}(w, \tilde{x}_k) \geq \tfrac{1}{2} |W(v)| \right) \right\}, \\ a_k &= \begin{cases} |\mathcal{N}_k|^{-1} \sum_{v \in \mathcal{N}_k} \mathrm{match}(v, \tilde{x}_k), & |\mathcal{N}_k| > 0, \\ 0, & |\mathcal{N}_k| = 0. \end{cases} \end{aligned} \tag{14}$$

Lists retain duplicate entries, as does the implementation. Missing or unparsable notion fields give zero alignment. Matching uses substrings, not semantic entailment, so a_k is only a lexical proxy for following the proposed correction. Multiplying it by revision success prevents an incorrect revision from receiving the alignment bonus.

Answer-leakage penalty. Define $B(c)$ to detect a `\boxed` or `\framebox` command followed by optional whitespace and an opening brace. Let $P(c)$ detect, case-insensitively and with word boundaries, any of the phrases *final answer*, *correct answer*, *the answer is*, *therefore the answer*, or *so the answer*. The leakage indicator is

$$\ell_k = \mathbf{1}\{B(c_k) \vee P(c_k)\}. \tag{15}$$

Only the extracted critique is checked; an unavailable critique is treated as an empty string for this feature. This heuristic can penalize benign diagnostic wording and miss unmarked answer disclosure. It is distinct from the strict acceptance checks used before directional scoring.

Combined critique reward for successful refinement groups. When the group contains at least one correct revision, the four-term reward in the method is

$$r_k^{\mathrm{C}} = \max\{0, \lambda_{\mathrm{mode}} d_k^{\mathrm{mode}} + \lambda_{\mathrm{succ}} b_k + \lambda_{\mathrm{align}} b_k a_k - \lambda_{\mathrm{leak}} \ell_k\}. \tag{16}$$

The lower truncation is applied after summing all terms. The implementation also exposes an optional selected-answer-change term $\lambda_{\mathrm{sel}} d_k^{\mathrm{sel}}$ inside the same maximum, where

$$d_k^{\mathrm{sel}} = \mathbf{1}\{E(\tilde{\tau}_k) \neq \bot,\ E(\tau^-) \neq \bot,\ E(\tilde{\tau}_k) \not\equiv E(\tau^-)\}. \tag{17}$$

Equation 2 corresponds to $\lambda_{\mathrm{sel}} = 0$; the optional term must not be added when reproducing that equation. The Stage-I launcher in Table 8 sets

$$\lambda_{\mathrm{mode}} = 0.1, \qquad \lambda_{\mathrm{succ}} = 0.4, \qquad \lambda_{\mathrm{align}} = 0.1, \qquad \lambda_{\mathrm{leak}} = 0.1.$$

Directional reward for all-wrong refinement groups. When all revisions fail, critique utility instead comes from the frozen scorer. Let $z = y^\star$ denote the tokenized ground-truth continuation, h_0 the baseline context, and h_{c_k} the critique-conditioned context. For nonempty target tokens with mask u_t, compute

$$\begin{aligned} L(h, z) &= \frac{\sum_t u_t \log \pi_{\mathrm{ref}}(z_t \mid h, z_{<t})}{\sum_t u_t}, \\ r_k^{\mathrm{D}} &= L(h_{c_k}, z) - L(h_0, z). \end{aligned} \tag{18}$$

The two contexts use the same failed answer and target; the baseline places `N/A` in the self-reflection field, while the conditioned context inserts c_k. Prompt and padding tokens are excluded from the mean. The default target template serializes the raw ground-truth field, which may be a short final answer rather than a complete reference derivation. Ground truth is not supplied to the critique-generation policy. This likelihood difference replaces, rather than adds to, Equation 16 on the all-wrong route.

For the structured prompt, directional scoring requires correctly ordered, uniquely occurring tags, nonempty spans, and valid JSON with the declared keys and value types. Boxed-answer commands in either the raw critique or decoded JSON fields are rejected. These strict checks do not reuse the phrase-based penalty $P(c)$: a format violation makes a sample ineligible, rather than assigning a separate format-reward bonus or a numeric zero score. An entire directional group is rejected if any member lacks a valid score or the expected group size is not preserved.

Scaling, quotas, and credit assignment. The implementation can transform raw directional scores before group normalization:

$$\bar{r}_k^{\mathrm{D}} = \mathrm{clip}\left(\gamma_{\mathrm{raw}} r_k^{\mathrm{D}}, v_{\min}, v_{\max}\right), \tag{19}$$

where absent clipping bounds are interpreted as infinite. The Stage-I launcher uses the unmodified reward in Equation 1, with $\gamma_{\mathrm{raw}} = 1$ and no clipping. This transformation is distinct from the post-normalization directional-advantage multiplier, set to $\alpha_{\mathrm{D}} = 0.5$. The all-wrong-to-successful group quota ratio is $\eta = 0.5$, with the quota schedule disabled. If $\mathcal{D}_{\mathrm{valid}}$ is the set of fully scored all-wrong groups and N_{batch}^{+} the number of successful refinement groups, a finite quota retains

$$N_{\mathrm{keep}}^{\mathrm{D}} = \min\{|\mathcal{D}_{\mathrm{valid}}|, \lfloor \eta N_{\mathrm{batch}}^{+} \rfloor\} \tag{20}$$

whole groups in batch group order, not the highest-scoring individual critiques. If a quota schedule is enabled, η denotes its current value; disabling the cap retains all fully valid directional groups. The centered advantages and role masks then follow Equation 3, with the singleton fallback described in Appendix F.1. In the reward tensor, each directional scalar is stored at the last active critique token; after group normalization, its advantage is broadcast over the active critique span. All-wrong revision tokens receive no directional advantage. Stage II does not add another critique-quality reward: its rebuilt direct group uses verified outcome rewards, and the inserted-token transform changes the optimization objective rather than the verifier's correctness label.

F ALGORITHM AND NOTATION

This appendix specifies the data construction and credit assignment behind Signpost. The two stages share a policy but use different training contexts: Stage I evaluates critiques in the context of failed answers, whereas Stage II trains on verified solutions under the original problem alone. Table 10 distinguishes the sampling budgets and masks used below.

Table 10: Notation for training and evaluation. A refinement group is associated with one selected failed answer, not with all problems in a minibatch.

Symbol	Meaning
$q, \tau^-, c, \tilde{\tau}$	Problem, selected failed answer, critique, and revised answer.
G	Number of initial direct answers for a training problem.
K	Number of critique–revision samples for a selected failure.
K_{extra}	Maximum additional sequential rounds after the first refinement.
$y^\star, T$	Scoring target supplied by the training example and its token count.
η, α_{D}	All-wrong group quota ratio and directional-advantage multiplier.
$m^{\text{C}}, m^{\text{R}}$	Critique and revision response-token masks in Stage I.
M^{int}	Inserted-solution token mask in the rebuilt Stage-II direct group.
n	Responses per problem in evaluation; independent of G and K.

F.1 STAGE I: LEARNING DIRECTIONS

Algorithm 1 expands Section 3.2. Generation and reward construction precede the policy update; neither verifier decisions nor reference-model scores are differentiated through.

Algorithm 1 Signpost Stage I: Learning Directions

Require: Training data $\mathcal{D}$, shared policy π_θ, frozen scorer π_{ref}, verifier r
Require: Direct group size G, refinement group size K, quota ratio η, weight α_{D}
Ensure: Stage-I policy parameters θ_{I}

for each training minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
  $\theta_{\text{old}} \leftarrow \theta$; $\mathcal{P} \leftarrow \varnothing$; $\mathcal{Z} \leftarrow \varnothing$
  // Step 1: Collect direct answers and refinement groups
  for each problem $q \in \mathcal{B}$ **do**
    Sample $\mathcal{G}(q) = \{\tau_i\}_{i=1}^{G}$ from $\pi^{\text{S}}_{\theta_{\text{old}}}(\cdot \mid q)$
    Verify all τ_i; select failures $\mathcal{F}_q$ using the configured success threshold
    for each selected failure $\tau^- \in \mathcal{F}_q$ **do**
      Sample $g = \{(c_k, \tilde{\tau}_k)\}_{k=1}^{K}$ conditioned on (q, τ^-)
      Parse masks $m^{\text{C}}, m^{\text{R}}$; compute $r^{\text{R}}_k \leftarrow r(\tilde{\tau}_k)$
      if $\max_k r^{\text{R}}_k > 0$ **then**
        Compute r^{C}_k using Equation 2
        Add g with its two reward streams and masks to $\mathcal{P}$
      else
        Score valid critiques using π_{ref} and Equation 1
        if all K members have valid directional scores and critique spans **then**
          Add the complete group g with r^{D}_k and m^{C} to $\mathcal{Z}$
        end if
      end if
    end for
  end for
  // Step 2: Apply the quota and assign span-specific advantages
  $\mathcal{Z} \leftarrow \textsc{KeepFirstGroups}(\mathcal{Z}, \lfloor \eta |\mathcal{P}| \rfloor)$
  for each group $g \in \mathcal{P}$ **do**
    $\widehat{A}^{\text{C}}, \widehat{A}^{\text{R}} \leftarrow \textsc{NormalizeSeparately}(r^{\text{C}}, r^{\text{R}})$
    $\widehat{A}_{k,t} \leftarrow m^{\text{C}}_{k,t}\widehat{A}^{\text{C}}_k + m^{\text{R}}_{k,t}\widehat{A}^{\text{R}}_k$
  end for
  for each group $g \in \mathcal{Z}$ **do**
    $\widehat{A}^{\text{D}} \leftarrow \textsc{NormalizeByGroup}(r^{\text{D}})$
    $\widehat{A}_{k,t} \leftarrow \alpha_{\text{D}} m^{\text{C}}_{k,t}\widehat{A}^{\text{D}}_k$
  end for
  // Step 3: Update the shared policy; exclude prompt and padding tokens
  Update θ on $\mathcal{P} \cup \mathcal{Z}$ with the masked GRPO surrogate if nonempty
end for
Return $\theta_{\text{I}} \leftarrow \theta$

The quota is applied to whole groups, not individual high-scoring critiques. With a finite quota ratio and no successful groups in a batch, the quota is zero. Directional scoring therefore does not imply that every all-wrong problem contributes to every update. For groups with $K > 1$, centered group-relative normalization still gives zero advantage if all directional scores coincide. The implementation separately supports $K = 1$ by using the raw scalar reward rather than subtracting a singleton mean; this fallback can yield a nonzero advantage and is not the centered estimator. Under the centered estimator, directional scoring removes the requirement for a successful sampled revision, not the requirement for within-group reward variation.

F.2 Stage II: Internalization by Group Reconstruction

Algorithm 2 implements Section 3.3 without treating a critique-conditioned answer as an on-policy sample under the problem-only prompt. The main configuration uses one replacement and $K_{\text{extra}} = 0$.

Algorithm 2 Signpost Stage II: Internalizing Critique-Guided Solutions

Require: Training data $\mathcal{D}$, Stage-I parameters θ_{I}, verifier r, group sizes G, K
Require: Internalization parameters $\beta, \epsilon_{\mathrm{off}}$; configured candidate-validity checks
Ensure: Policy $\pi_\theta^{\mathrm{S}}(\cdot \mid q)$ for direct, critique-free inference

$\theta \leftarrow \theta_{\mathrm{I}}$
for each training minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
  $\theta_{\mathrm{old}} \leftarrow \theta; \mathcal{U} \leftarrow \varnothing$
  // Step 1: Discover and verify critique-guided solutions
  for each problem $q \in \mathcal{B}$ **do**
    Sample $\mathcal{G}(q) = \{\tau_i\}_{i=1}^{G}$ from $\pi_{\theta_{\mathrm{old}}}^{\mathrm{S}}(\cdot \mid q)$
    Verify all τ_i; select eligible failures $\mathcal{F}_q$; set $\mathcal{C}_q \leftarrow \varnothing$
    for each selected failure $\tau^- \in \mathcal{F}_q$ **do**
      Sample K critique–revision pairs conditioned on (q, τ^-)
      Add verified revisions passing span, length, and enabled finish checks to $\mathcal{C}_q$
    end for
    // Step 2: Rebuild the direct group without changing its size
    $\mathcal{G}^{\mathrm{T}}(q) \leftarrow \mathcal{G}(q)$; $M^{\mathrm{int}} \leftarrow 0$
    if $\mathcal{C}_q \neq \varnothing$ and $\mathcal{G}(q)$ contains a failed answer **then**
      $\tau^+ \leftarrow$ SELECTVERIFIED$(\mathcal{C}_q)$
      Retain only the standalone solution text of τ^+ under prompt q
      Replace one failed slot in $\mathcal{G}^{\mathrm{T}}(q)$ with τ^+
      Set $M^{\mathrm{int}} = 1$ on inserted response tokens only
    end if
    Recompute $\widehat{A}^{\mathrm{T}}$ from rewards of the complete group $\mathcal{G}^{\mathrm{T}}(q)$
    Add $(q, \mathcal{G}^{\mathrm{T}}(q), \widehat{A}^{\mathrm{T}}, M^{\mathrm{int}})$ to $\mathcal{U}$
  end for
  // Step 3: Optimize direct and inserted tokens with different objectives
  Compute J_{direct} on original response tokens using Equation 5
  Compute $p_{j,t} \leftarrow \pi_\theta^{\mathrm{S}}(y_{j,t} \mid q, y_{j,<t})$ for inserted tokens
  $\psi_{j,t} \leftarrow p_{j,t}/(p_{j,t} + \beta)$; compute J_{int} using Equation 6
  Update θ on $\mathcal{U}$ with $J_{\mathrm{direct}} + J_{\mathrm{int}}$
end for
Return θ

Replacing multiple slots is a separate ablation, not a change to the group size. The implementation exposes candidate-selection variants; these should not be conflated with the essential reconstruction operation, which requires a verified solution but does not require a particular ranking heuristic.

Optional sequential refinement. For $K_{\mathrm{extra}} > 0$, each unsuccessful branch can use its latest revision as the failed answer for another critique–revision attempt. A branch stops on success or budget exhaustion. Only terminal verified solutions are eligible for insertion; an unresolved branch cannot be converted into a positive training example. The maximum number of rounds on a branch is $1+K_{\mathrm{extra}}$, but early stopping changes the realized token cost. The main configuration uses $K_{\mathrm{extra}} = 0$. This training-time budget is distinct from the sequential test-time diagnostic in Section **??**.

G Reward and Objective Details

G.1 What the Directional Score Measures

Let h_0 be the baseline scoring context and h_c its critique-conditioned counterpart. The scorer computes

$$L(h; y^\star) = \frac{1}{T}\sum_{t=1}^{T} \log \pi_{\mathrm{ref}}(y_t^\star \mid h, y_{<t}^\star), \qquad r^{\mathrm{D}} = L(h_c; y^\star) - L(h_0; y^\star). \tag{21}$$

Only valid target tokens enter the average; prompt and padding tokens do not. Both contexts use the same target and frozen scorer. The provided scoring template serializes the example's ground-truth field as the continuation after the answer-request context shown in Appendix H. Thus the target can be a final-answer string: the score does not require a human-written chain of reasoning or certify that the critique's reasoning is correct. The target must be distinguished from the model-generated, verified revision used for Stage-II internalization.

A shared baseline changes interpretation, not relative ranking. For a fixed failure, write $s_k = L(h_{c_k}; y^\star)$ and $b = L(h_0; y^\star)$. Because b is shared,

$$\frac{(s_k - b) - \text{mean}_j(s_j - b)}{\text{std}_j(s_j - b) + \delta} = \frac{s_k - \text{mean}_j\, s_j}{\text{std}_j\, s_j + \delta}. \tag{22}$$

This identity describes centered normalization for $K > 1$ with unmodified likelihood differences; nonlinear clipping of scores need not preserve it. It does not describe the singleton raw-reward fallback, for which subtracting the baseline changes the advantage. The baseline makes a positive raw score interpretable as an increase relative to the no-critique context. It does not turn that increase into a calibrated probability of successful correction. The reference target is used by the scorer and verifier, never inserted into the critique or revision generation prompt.

G.2 Outcome-Based Critique Rewards

Appendix E.1 gives the explicit component formulas and experimental coefficient fields. The mode-change term in Equation 2 compares the revised final answer with the modal extracted initial answer. The implementation selects the mode among extracted answer strings, then tests the revised answer for mathematical equivalence to that mode. If either required answer cannot be extracted, the difference indicator is zero. Changing an answer is not by itself evidence of correctness; revision success is supplied by the task verifier.

The alignment term is a lexical proxy for using the critique's proposed mathematical notions. It reads the structured `correction_notion` field, normalizes its entries, and excludes generic notions, short entries, and notions copied verbatim from the question. A retained notion matches if its phrase appears in the revision or at least half of its non-generic words do. The alignment score is the fraction of retained notions matched, or zero when none remain. Its contribution is gated by revision correctness. This is not a semantic entailment test.

The leakage penalty detects boxed answers and explicit answer-announcing phrases inside the critique; it is a heuristic rather than a guarantee that no answer information appears there. The complete weighted critique reward is truncated below at zero. A configurable implementation variant additionally rewards departure from the particular selected failed answer. Setting that coefficient to zero gives the four-term reward in Equation 2; it must not be silently added when reproducing that equation. Likewise, coefficient values belong to a specified training configuration, not to the reward definition alone.

G.3 Inserted-Token Objective and Numerical Handling

The inserted answer lacks a behavior probability under the new problem-only context. Its objective therefore uses the current conditional probability p rather than dividing by the likelihood of an answer sampled with a different context. The implementation obtains p by exponentiating log probabilities clipped to $[-20, 0]$, then applies $\psi = p/(p + \beta)$ for $\beta > 0$. The numerical floor avoids underflow; it is distinct from the surrogate's clipping interval.

For a fixed advantage A and $0 < \epsilon_{\text{off}} < 1$, the uncluttered scalar form of Equation 6 is

$$f(\psi, A) = \begin{cases} A\psi, & A \geq 0, \\ A \max\{\psi, 1 - \epsilon_{\text{off}}\}, & A < 0, \end{cases} \qquad 0 < \psi < 1. \tag{23}$$

In particular, clipping does not impose an on-policy trust region around a behavior ratio: ψ is not such a ratio. Away from the numerical clamp, its derivative with respect to the token log probability $z = \log p$ is

$$\frac{\partial \psi}{\partial z} = \frac{\beta p}{(p + \beta)^2} = \psi(1 - \psi) \leq \frac{1}{4}. \tag{24}$$

This is a bound on the transform's scalar sensitivity, not on the full parameter gradient or a convergence guarantee. The inserted-token route does not apply dual clipping. Original responses retain their ordinary GRPO clipping, with the respective token masks preventing the two objectives from being applied to the same response tokens.

H Prompt Templates

Figures 3–9 document the implemented prompt interfaces. Braced placeholders are filled with the problem and model-generated text; line wrapping is typographical. The tokenizer's chat template

supplies the message delimiters. An existing task system instruction is preserved unless explicitly overridden; the provided fallback is `Please reason step by step, and put your final answer within \boxed{}.` Prompt variants below have different roles and are not interchangeable experimental settings.

H.1 Structured Joint Critique and Revision

This user template produces both spans in one response. The JSON field names and the enclosing tags determine the parsing contract; the revised solution must remain understandable after the critique is removed.

Structured Critique and Revision Prompt

Question:
{question}

Initial answer:
{initial_answer}

The initial answer is incorrect. Write a structured critique only inside the critique tags.
The critique must be valid JSON with exactly these keys:
- "error_type": a short natural-language phrase describing the type of error, such as "confusing decimal and base-b units digits", "assuming symmetry without proof", or "missing a boundary case".
- "first_wrong_step": quote or paraphrase the earliest incorrect or unsupported step from the initial answer. Do not include the final answer.
- "why_wrong": explain the flaw in that step, referring to the problem condition or mathematical rule it violates. Do not solve the problem.
- "correction_notion": a list of 1 to 4 mathematical notions, tools, or reasoning skills needed to correct the solution. Do not copy phrases directly from the problem statement. Do not include the final answer.
- "fix_direction": describe the next high-level correction action the refined solution should take. Do not include numeric final results.
- "missing_condition": name the overlooked condition/case/constraint, or "none".
- "do_not_solve": true.

Rules for the critique JSON:
- Do not compute or reveal the final answer.
- Do not include a boxed answer.
- Do not write a full derivation or corrected solution.
- Do not include final numeric results, final choices, or final simplified expressions.
- Keep each string value concise and diagnostic, not a full solution.
- The correction_notion values should be reusable concepts, not copied wording from the problem.

Then inside the refined answer tags, write a complete standalone corrected solution to the original question.
The refined answer must have similar completeness and explanatory detail to the initial answer, but with the incorrect steps repaired according to the critique.
The refined answer should explicitly use the notions listed in correction_notion when they are relevant.
Do not make the refined answer a short conclusion. Do not merely state the answer.
Re-derive the solution step by step, include all necessary intermediate reasoning, and put the final answer in boxed form.
The refined answer must be understandable without reading the critique JSON.

Use exactly this format:
<critique>
{"error_type":"...","first_wrong_step":"...","why_wrong":"...","correction_notion":["...","..."],"fix_direction":"...","missing_condition":"...","do_not_solve":true}
</critique>
<refined_answer>
Your complete standalone corrected solution with similar completeness to the initial answer. Put the final answer in boxed form.
</refined_answer>

Figure 3: The structured joint prompt for generating a critique and a standalone revised solution.

H.2 Independent Critique and Revision for Diagnostics

The independent directional-reward diagnostic separates critique generation from screening and fresh revisions. Its critique-only template shares the structured schema but does not ask for a revision in the same response:

Independent Critique Prompt

Question:
{question}

Initial answer:
{initial_answer}

The initial answer is incorrect. Write a structured critique only inside the critique tags.
The critique must be valid JSON with exactly these keys:
- "error_type": a short natural-language phrase describing the type of error.
- "first_wrong_step": quote or paraphrase the earliest incorrect or unsupported step from the initial answer. Do not include the final answer.
- "why_wrong": explain the flaw in that step, referring to the problem condition or mathematical rule it violates. Do not solve the problem.
- "correction_notion": a list of 1 to 4 mathematical notions, tools, or reasoning skills needed to correct the solution. Do not include the final answer.
- "fix_direction": describe the next high-level correction action. Do not include numeric final results.
- "missing_condition": name the overlooked condition/case/constraint, or "none".
- "do_not_solve": true.

Do not compute or reveal the final answer. Do not include a boxed answer, a full derivation, or a corrected solution.
Keep every string concise and diagnostic. Use exactly this format:
<critique>
{"error_type":"...","first_wrong_step":"...","why_wrong":"...","correction_notion":["..."],"fix_direction":"...","missing_condition":"...","do_not_solve":true}
</critique>

Figure 4: The critique-only prompt used to generate feedback independently of diagnostic revisions.

Each screening or fresh revision then uses the following separate user message. For the no-critique control, only the `Critique of the initial answer:` block and its contents are removed; the failed answer, remaining instructions, and response budget are retained.

Independent Revision Prompt

Question:
{question}

Initial answer:
{initial_answer}

Critique of the initial answer:
{critique}

The initial answer is incorrect. Now solve the original question again from scratch.
Use any feedback above only as guidance. Return a complete standalone corrected solution.
Put the final answer in boxed form.
Return only the refined solution.

Figure 5: The independent revision prompt for screening and fresh correction trials.

H.3 Teacher-Forcing Context

The reference scorer uses the following context, followed by the separately tokenized ground-truth continuation. For the baseline, replace `{critique}` with the literal `N/A`; do not substitute the no-critique generation prompt above. The default target serialization is the raw ground-truth string, with no additional rationale synthesized for scoring.

Teacher-Forcing Scoring Context

Question:
{question}

Previous attempt:
{initial_answer}

Self-reflection:
{critique}

Please directly provide the correct final answer to the problem.

Correct answer:

Figure 6: The critique-conditioned teacher-forcing context; the baseline substitutes `N/A` for the critique.

H.4 Serial Self-Check and Refinement Evaluation

Figures 7–9 give the prompt interface implemented by `self_refine_eval.py`. The initial response uses the dataset's original message sequence when available. Each subsequent round receives the problem and the currently retained answer; the policy first judges whether that answer is correct, rather than being told that it is wrong. A dataset-provided system instruction takes precedence over the fallback in Figure 7.

Serial Evaluation: Fallback System Prompt

You are a helpful assistant. Solve math problems carefully and put the final answer in boxed form.

Figure 7: The fallback system instruction for serial evaluation; an existing dataset system message is preserved instead.

The user prompt is displayed in two consecutive cards for readability. They form *one user message in one generation request*, not separate judging and revision calls. The placeholder `{current_answer}` is updated after each revision. If the parsed decision is `is_correct=true`, the evaluator stops the trajectory and carries its current answer forward. Otherwise, it uses the extracted revision, falling back to the raw generated output when no revised-answer span is available, and continues up to the configured round limit. The external verifier scores answers but does not decide when to stop. This decision-tag protocol is specific to this evaluator and is distinct from the verdict-based interface in `self_refine_aime.py`.

Serial Self-Check and Refinement Prompt (1/2)

Question:
{question}

Initial answer:
{current_answer}

First decide whether the initial answer is mathematically correct. Do not assume it is wrong.

If the initial answer is correct, do not solve the problem again. Use exactly this format:
<decision>{"is_correct":true}</decision>
<critique>
A concise reason why the initial answer is correct. Do not solve the problem here.
</critique>
<refined_answer>
</refined_answer>

If the initial answer is incorrect, write a structured critique only inside the critique tags.
The critique must be valid JSON with exactly these keys:
- "error_type": a short natural-language phrase describing the type of error, such as "confusing decimal and base-b units digits", "assuming symmetry without proof", or "missing a boundary case".
- "first_wrong_step": quote or paraphrase the earliest incorrect or unsupported step from the initial answer. Do not include the final answer.
- "why_wrong": explain the flaw in that step, referring to the problem condition or mathematical rule it violates. Do not solve the problem.
- "correction_notion": a list of 1 to 4 mathematical notions, tools, or reasoning skills needed to correct the solution. Do not copy phrases directly from the problem statement. Do not include the final answer.
- "fix_direction": describe the next high-level correction action the refined solution should take. Do not include numeric final results.
- "missing_condition": name the overlooked condition/case/constraint, or "none".
- "do_not_solve": true.

Rules for the critique JSON:
- Do not compute or reveal the final answer.
- Do not include a boxed answer.
- Do not write a full derivation or corrected solution.
- Do not include final numeric results, final choices, or final simplified expressions.
- Keep each string value concise and diagnostic, not a full solution.
- The correction_notion values should be reusable concepts, not copied wording from the problem.

Figure 8: Serial evaluation user prompt, part 1 of 2: self-assessment, the stop response, and structured critique requirements.

Serial Self-Check and Refinement Prompt (2/2)

Then inside the refined answer tags, write a complete standalone corrected solution to the original question.
The refined answer must have similar completeness and explanatory detail to the initial answer, but with the incorrect steps repaired according to the critique.
The refined answer should explicitly use the notions listed in correction_notion when they are relevant.
Do not make the refined answer a short conclusion. Do not merely state the answer.
Re-derive the solution step by step, include all necessary intermediate reasoning, and put the final answer in boxed form.
The refined answer must be understandable without reading the critique JSON.

Use exactly this format when the initial answer is incorrect:
<decision>{"is_correct":false}</decision>
<critique>
{"error_type":"...","first_wrong_step":"...","why_wrong":"...","correction_notion":["...","..."],"fix_direction":"...","missing_condition":"...","do_not_solve":true}
</critique>
<refined_answer>
Your complete standalone corrected solution with similar completeness to the initial answer. Put the final answer in boxed form.
</refined_answer>

Figure 9: Serial evaluation user prompt, part 2 of 2: revision instructions and the output format for an answer judged incorrect. This text continues the same user message as Figure 8.

I Training and Evaluation Protocols

I.1 Training scope and inference roles

The main comparison uses Qwen3-8B and Qwen3-14B, with Signpost implemented in verl. Stage I trains critique and revision spans with separate supervision, using frozen-reference directional rewards for critiques in all-wrong refinement groups. Stage II trains on verified revisions recontextualized under the original problem prompt. The main configuration uses one refinement step and no additional sequential attempts ($K_{\text{extra}} = 0$); additional-round variants are separate ablations.

Three sampling quantities have distinct roles. In the method, G is the number of initial training responses in a direct group, and K is the number of critique–revision pairs for a selected failure. Below, n is the number of evaluation responses per problem. Neither training group size is defined by the evaluation budget. In particular, one-step refinement does not mean one critique candidate, and $K_{\text{extra}} = 0$ does not remove critique-guided training. The main benchmark comparisons evaluate the resulting policies as direct solvers, without critiques, revision prompts, or test-time refinement.

I.2 Metrics, sampling budgets, and supported evaluation

For a benchmark with N problems and n responses per problem, let $z_{ij} \in \{0, 1\}$ indicate whether response j to problem i passes the task verifier. The reported empirical metrics are

$$\begin{aligned} \text{mean@}n &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n}\sum_{j=1}^{n} z_{ij}, \\ \text{pass@}n &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}\left\{\sum_{j=1}^{n} z_{ij} > 0\right\}. \end{aligned} \tag{25}$$

Both weight problems equally. Mean@n measures the average correctness of sampled responses; pass@n measures oracle coverage by at least one correct response, not accuracy after majority voting or a deployable answer-selection rule. At $n = 1$, they coincide. Scores expressed as percentages multiply these quantities by 100.

AIME24, AIME25, AIME26, and AMC23 use $n = 8$ and report both metrics. MinervaMath, MATH-500, and OlympiadBench use one sampled response and report pass@1. Mathematics-trained checkpoints are additionally evaluated on ARC-C, MMLU-Pro, and GPQA Diamond using pass@1 without target-domain fine-tuning; ARC-C uses the ARC-Challenge test split. The evaluation temperature is 0.6, with task-specific prompts, decoding limits, and verifiers held fixed across methods within a backbone. Multiple-choice prompt templates and demonstration counts remain to be finalized. Pass@1 specifies the response budget, not the number of in-context demonstrations. Benchmarks with different response budgets are reported separately rather than combined into an aggregate score.

The direct-evaluation tooling supports explicit benchmark selection, validation-only generation, and checks of problem identity, sample coverage, and verifier scores. It records resolved configurations and generation artifacts. These capabilities describe a supported protocol, not evidence that every reported experiment used the launcher's current defaults. Its 128-response mode can compute nested-prefix metrics using fixed sample indices at $k \in \{8, 16, 32, 64, 128\}$. Such observed-prefix pass@k is necessarily nondecreasing and differs from an estimator averaged over subsets; this implementation option does not establish the sampling-pool construction of other reported curves.

Figure 11 addresses separate inference questions. Its parallel panel measures coverage as the candidate budget grows. Its sequential panel scores the current retained answer at round 0 and after up to three self-check-and-refine rounds. Stopping follows the model's judgment, stopped answers are carried forward, and the external verifier only scores outputs. Sequential accuracy therefore does not select the best historical answer with an oracle. Equal response counts also need not imply equal token use, latency, or training cost; the two panels are not an equal-compute comparison.

Table 11: Matched-checkpoint comparison at training step 170. Direct solving is mean@8; recovery is conditional on the policy's own initially incorrect responses. Values are percentages; differences are percentage points. No correction evaluation is recorded for QA-only.

	Direct solving		Conditional recovery	
Training strategy	AIME24	AIME25	AIME24	AIME25
QA-only	53.33	42.08	—	—
CA-only	37.50	31.67	26.00	18.90
Mixed	46.67	37.92	21.88	18.79
Mixed minus task specialist	-6.67	-4.17	-4.13	-0.11

J Task Interference under Joint QA–CA Optimization

Research question. We examine whether joint optimization retains the direct-solving and correction performance of task-specialized training. We compare three Qwen3-8B configurations: direct question-to-answer optimization (**QA-only**), critique-conditioned correction without direct-answer updates (**CA-only**), and joint optimization of both tasks (**Mixed**). This diagnostic studies the task trade-off associated with the seesaw phenomenon in multi-task learning (Tang et al., 2020). It complements the full Signpost training-schedule ablation in Table 5 by examining the interaction between QA and CA updates.

Experimental setup. The recorded configurations use Qwen3-8B with DeepScaleR training and AIME24/AIME25 validation. All three specify a training batch of 256 problems, eight rollout samples, prompt/response limits of 12,000/8,192 tokens, learning rate 10^{-6}, one PPO epoch per update, and PPO minibatch size 128. Training and validation temperatures are 1.0 and 0.6, respectively, with eight validation responses per problem. QA-only and CA-only use eight GPUs on one node; Mixed uses sixteen GPUs on two nodes.

CA-only disables direct-answer updates, whereas Mixed combines direct-answer and correction updates. Both correction-enabled configurations supervise critique and revised-answer spans using split-segment advantages. They select one incorrect answer from question groups with initial mean reward at most 0.5 and retain refinement groups containing at least one correct revision. Configured branch loss weights are one. Directional rewards, off-policy answer replacement, and answer distillation are absent from this diagnostic, separating the QA/CA comparison from the complete Signpost procedure.

Comparison window. We compare the 17 shared validation checkpoints at steps $10, 20, \ldots, 170$ and use step 170, the latest common checkpoint, as the primary endpoint. This rule aligns the displayed training horizon without selecting each run's best checkpoint. Figure 10 shows the shared window, and Table 11 reports the endpoint comparison.

Evaluation metrics. For each checkpoint and benchmark, let $b_i \in \{0, 1\}$ denote the verifier outcome of initial response i. Initially correct responses are retained; incorrect responses receive a critique-conditioned correction. Let f_i denote the final outcome of this assisted procedure. Direct accuracy is $B = \mathrm{mean}_i\, b_i$, and assisted accuracy is $F = \mathrm{mean}_i\, f_i$. Since each problem contributes eight responses, both are equal-weight empirical mean@8 scores.

To measure correction separately from initial correctness, we report the fraction of initially incorrect responses that are recovered:

$$R_{\text{recover}} = \frac{\sum_i (1 - b_i) f_i}{\sum_i (1 - b_i)} = \frac{F - B}{1 - B} = \frac{\mathrm{mean}(\texttt{delta_reward})}{\mathrm{mean}(\texttt{critique_initial_wrong})}. \tag{26}$$

The metric is defined for $B < 1$. The logged identities $F - B = \mathrm{mean}(\texttt{delta_reward})$ and $1 - B = \mathrm{mean}(\texttt{critique_initial_wrong})$ hold at every included checkpoint. QA-only has no recorded correction evaluation, so its recovery entries are left blank. Recovery is measured on each checkpoint's own initial failures.

Results. At step 170, Mixed achieves direct-solving mean@8 of 46.67% on AIME24 and 37.92% on AIME25. These scores exceed CA-only's 37.50% and 31.67%, but fall below QA-only's 53.33%



Figure 10: QA/CA training dynamics over 17 shared validation checkpoints. Top: direct-solving mean@8 for QA-only, CA-only, and Mixed. Bottom: conditional recovery of initial failures for CA-only and Mixed. Each configuration has one run; points are validation measurements, connected without smoothing or extrapolation.

and 42.08% by 6.67 and 4.17 percentage points, respectively. Thus, Mixed's direct-solving performance lies between the two specialized configurations on both benchmarks.

For correction, Mixed recovers 21.88% of its initial failures on AIME24, compared with 26.00% for CA-only, a deficit of 4.13 percentage points. On AIME25, recovery is nearly identical: 18.79% for Mixed and 18.90% for CA-only. The observed trade-off is therefore clearest on AIME24: Mixed improves direct solving relative to CA-only while attaining lower recovery, and remains below QA-only in direct solving. AIME25 shows the same ordering in direct performance but little separation in recovery.

Implications and scope. Within these runs, joint QA–CA training does not simultaneously retain the strongest observed direct-solving and correction performance. This pattern is consistent with a task trade-off and motivates separating exploration learning from the internalization of verified solutions. Table 5 separately evaluates the benefit of the complete two-stage schedule. Recovery also depends on each policy's remaining failures, so differences reflect both correction behavior and failure-set difficulty. The results describe an observed task trade-off, which is a measurement of gradient conflict or a general limit of joint optimization.

K QUALITATIVE CASE STUDIES

We present two selected examples that connect critique-guided correction with subsequent *direct* solving. Each example contains an early incorrect response, its critique and successful revision, and a later Stage-II response to the *same problem without the earlier answer or critique*. The later responses have initial-answer verifier reward 1, and their logs explicitly mark critique generation as skipped. They therefore demonstrate more than success after test-time feedback: in these samples, the policy supplies the corrected reasoning structure from the problem alone.

The examples come from one training sequence. Grid-path responses are paired at steps 20 and 220; divisibility responses at steps 10 and 150. We reproduce each complete question, early answer, critique, revision, and later direct answer without excerpting their reasoning. All critique fields are displayed in their original order; mathematical notation and Markdown are typeset for readability. Logger-added response labels and skipped-refinement markers are not part of the model answers and are reported in the box titles and source notes instead. Long responses continue across pages with repeated titles. The analysis outside the boxes is our commentary rather than model output.

K.1 CASE 1: RECOVERING THE ALTERNATING STRUCTURE OF GRID PATHS

Observed progression: early answer 1470 (incorrect) $\rightarrow$ critique-guided answer 294 (correct) $\rightarrow$ Stage-II direct answer 294 (correct, no critique).

Question: Grid Paths

Consider the paths of length 16 that follow the lines from the lower left corner to the upper right corner on an 8×8 grid. Find the number of such paths that change direction exactly four times, as in the examples shown below.

Source note. The logged question refers to example diagrams, but none are included in its text input. The grid dimensions, endpoints, path length, and turn constraint specify the counting problem without those examples.

Early Direct Response, Step 20 — Incorrect

We are given an 8×8 grid, and we are to find the number of paths of length 16 from the **lower left corner** to the **upper right corner** that **change direction exactly four times**. These paths are made by moving only **right (R)** or **up (U)**, and the path must consist of 8 right moves and 8 up moves (since it's an 8×8 grid, the total number of steps is $8 + 8 = 16$).

Step 1: Understand the Problem

- Total number of steps = 16 (8 R's and 8 U's)
- A **path** is a sequence of R's and U's.
- A **change of direction** occurs when the sequence switches from R to U or U to R.
- We are to count the number of such paths that **change direction exactly four times**.

Early Direct Response, Step 20 — Incorrect (continued)

Step 2: Key Observations

- A path of 16 steps that changes direction **exactly four times** must consist of **five segments** (since each change of direction divides the path into a new segment).
- Each segment is a sequence of moves in the same direction (either all R's or all U's).
- Since there are 8 R's and 8 U's, the path must consist of:
  - 3 segments of one direction (say R)
  - 2 segments of the other direction (say U), or vice versa
- This is because the number of segments of each direction must differ by at most 1 (since 8 R's and 8 U's are equal).

So, the only valid way to split the path into 5 segments with exactly 4 direction changes is:

- 3 segments of one direction (say R) and 2 segments of the other (say U)
- Or vice versa: 3 segments of U and 2 segments of R

Let's proceed with the first case: 3 segments of R and 2 segments of U.

Step 3: Count the Number of Ways to Arrange the Segments
We are to arrange 5 segments, 3 of one type (say R) and 2 of the other (say U). The number of such arrangements is:

$$\binom{5}{3} = 10$$

So, there are **10 ways** to choose which segments are R and which are U.

Step 4: Count the Number of Ways to Distribute the Moves
Now, for each such arrangement of segments, we need to distribute the 8 R's and 8 U's into the segments of that type.

- Suppose we have 3 R segments and 2 U segments.
- Let the lengths of the R segments be a, b, c, and the lengths of the U segments be d, e, such that:
$$a + b + c = 8 \quad \text{and} \quad d + e = 8$$
- The number of ways to split 8 R's into 3 segments is the number of **compositions** of 8 into 3 positive integers:
$$\binom{8-1}{3-1} = \binom{7}{2} = 21$$
- Similarly, the number of ways to split 8 U's into 2 segments is:
$$\binom{8-1}{2-1} = \binom{7}{1} = 7$$

So, for each arrangement of segments (there are 10), the number of ways to assign the moves is:

$$21 \times 7 = 147$$

Step 5: Multiply by the Number of Segment Arrangements
We had 10 ways to choose which segments are R and which are U.
So, the total number of such paths is:

$$10 \times 147 = \boxed{1470}$$

Final Answer:

$$\boxed{1470}$$

Critique at Step 20 — All Original Fields

error_type.
incorrect segment distribution assumption

first_wrong_step.
We are to arrange 5 segments, 3 of one type (say R) and 2 of the other (say U). The number of such arrangements is: $\binom{5}{3} = 10$

why_wrong.
This step assumes that the segments are distinguishable only by their direction, not by their position. However, the sequence of direction changes must be considered in order, and the segments must be non-empty. The error is in assuming that all combinations of 3 R-segments and 2 U-segments are valid, without considering that the direction changes must alternate strictly (e.g., R-U-R-U-R or U-R-U-R-U).

correction_notion.

- permutations with alternating directions
- combinatorics of direction sequences
- valid segment sequences
- composition of integers

fix_direction.
Ensure that the sequence of segments alternates strictly between R and U, and that the number of direction changes is exactly four, which implies the path must alternate R-U-R-U-R or U-R-U-R-U.

missing_condition.
The path must alternate strictly between directions, not just have a fixed number of segments of each type.

do_not_solve.
`true`

Critique-Guided Revision, Step 20 — Correct

To find the number of paths of length 16 from the lower-left to the upper-right corner on an 8×8 grid, where the path consists of only right (R) and up (U) moves, and changes direction exactly four times, we must carefully consider the structure of such paths.

Step 1: Understanding the Path

- A path from the lower-left to the upper-right of an 8×8 grid consists of exactly 8 right moves (R) and 8 up moves (U), for a total of 16 moves.
- A **change of direction** occurs when a move from R to U or from U to R happens.
- A path that changes direction exactly four times implies that it consists of **five segments**, each of which consists of moves in the same direction (either all R or all U), and that the segments alternate direction.

Step 2: Valid Segment Sequences
Since the path changes direction exactly four times, it must alternate between R and U. Thus, the only possible sequences of directions are:

- R-U-R-U-R
- U-R-U-R-U

These are the only two valid sequences of 5 segments that alternate direction and have exactly 4 direction changes.

Step 3: Counting the Number of Valid Sequences
There are exactly **2** such valid sequences of direction segments.

Step 4: Distributing Moves into Segments
Now, for each of these two direction sequences, we need to distribute the 8 R's and 8 U's into the segments, ensuring that each segment has at least one move (i.e., the segments are non-empty).

Critique-Guided Revision, Step 20 — Correct (continued)

Case 1: Sequence R-U-R-U-R

- There are 3 R segments and 2 U segments.
- We need to distribute 8 R's into 3 non-empty segments (i.e., a composition of 8 into 3 positive integers).
- The number of compositions of 8 into 3 positive integers is:

$$\binom{8-1}{3-1} = \binom{7}{2} = 21$$

- Similarly, we distribute 8 U's into 2 non-empty segments:

$$\binom{8-1}{2-1} = \binom{7}{1} = 7$$

- So, for this sequence, the number of ways to assign moves is:

$$21 \times 7 = 147$$

Case 2: Sequence U-R-U-R-U

- This is symmetric to the first case.
- There are 3 U segments and 2 R segments.
- The number of ways to assign moves is again:

$$\binom{7}{2} \times \binom{7}{1} = 21 \times 7 = 147$$

Step 5: Total Number of Valid Paths
Since there are two valid direction sequences, and each yields 147 valid paths, the total number of such paths is:

$$2 \times 147 = \boxed{294}$$

Stage-II Direct Response, Step 220 — Correct; No Critique

We are given an 8×8 grid, and we are to find the number of paths of length 16 (i.e., from the lower-left corner to the upper-right corner) that change direction **exactly four times**.

Step 1: Understand the Path
In an 8×8 grid, to go from the lower-left corner to the upper-right corner, you need to move **right** 8 times and **up** 8 times — a total of 16 steps.
So any such path is a sequence of 8 R's (right) and 8 U's (up), with no restrictions on order, but in a grid, you can only move right or up, not left or down.
So the total number of such paths is:

$$\binom{16}{8}$$

But we are only interested in those paths that **change direction exactly four times**.

Stage-II Direct Response, Step 220 — Correct; No Critique (continued)

Step 2: Understand What It Means to Change Direction
Each time the direction changes from R to U or U to R, it counts as a **direction change**.
So, for example, a path like R R R U U U R R R U U U would have **three direction changes** (from R to U, then U to R, then R to U).
We want paths with **exactly four direction changes**.
Let's think about how many direction changes are possible in a path of this kind.
Since the path is made up of alternating runs of R and U, the number of direction changes is one less than the number of runs.
So, for example:

- A path with 2 runs (all R then all U) has 1 direction change.
- A path with 3 runs (e.g., R...R U...U R...R) has 2 direction changes.
- A path with 4 runs has 3 direction changes.
- A path with 5 runs has 4 direction changes.

So, a path that changes direction **exactly four times** has **five runs** — alternating between R and U.
So, we want to count the number of such paths that consist of **five runs** of R and U, alternating, and total 8 R's and 8 U's.

Step 3: Determine the Structure of the Path
Since the path has five runs, and alternates between R and U, the sequence of runs could be:

- R U R U R
- U R U R U

These are the only two possible patterns of five runs, alternating between R and U.
Let's consider each case.

Case 1: R U R U R
This is a sequence of 3 runs of R and 2 runs of U.
Let the number of R's in the runs be: a_1, a_3, a_5, and the number of U's in the runs be b_2, b_4, where:

$$a_1 + a_3 + a_5 = 8 \quad \text{(total R's)}$$
$$b_2 + b_4 = 8 \quad \text{(total U's)}$$

Each $a_i \geq 1$, $b_j \geq 1$
So, we are distributing 8 R's into 3 runs (each $\geq$1) and 8 U's into 2 runs (each $\geq$1).
The number of such distributions is:

$$\binom{8-1}{3-1} = \binom{7}{2} = 21 \quad \text{(for R's)}$$
$$\binom{8-1}{2-1} = \binom{7}{1} = 7 \quad \text{(for U's)}$$

So total number of such paths is:

$$21 \times 7 = 147$$

Stage-II Direct Response, Step 220 — Correct; No Critique (continued)

Case 2: U R U R U
This is a sequence of 2 runs of R and 3 runs of U.
Similar to above:

- R's are in runs a_2, a_4
- U's are in runs b_1, b_3, b_5

So:

$$a_2 + a_4 = 8 \quad \text{(R's)}$$
$$b_1 + b_3 + b_5 = 8 \quad \text{(U's)}$$

Number of such distributions:

$$\binom{8-1}{2-1} = \binom{7}{1} = 7 \quad \text{(for R's)}$$
$$\binom{8-1}{3-1} = \binom{7}{2} = 21 \quad \text{(for U's)}$$

Total number of such paths:

$$7 \times 21 = 147$$

Step 4: Total Number of Paths with Exactly 4 Direction Changes
We have two cases, each contributing 147 paths, so:

$$147 + 147 = \boxed{294}$$

Final Answer:

$$\boxed{294}$$

Analysis. The initial solution correctly recognizes that four turns produce five nonempty direction segments, but overcounts by allowing arbitrary arrangements of three horizontal and two vertical segments. Adjacent segments with the same direction would merge, so only the alternating patterns are admissible. The critique identifies this structural constraint without supplying the final count. The revision then assigns positive segment lengths within the two patterns and obtains 294. In the later direct response, the policy identifies the same two patterns and applies positive-integer compositions *without receiving the earlier critique*. Exhaustive enumeration of the $\binom{16}{8} = 12{,}870$ shortest paths independently confirms that 294 have exactly four turns. The retained combinatorial structure, rather than only the matching final answer, is the qualitative indication of internalization here.

K.2 Case 2: Preserving the Full Divisibility Condition

Observed progression: early answer 11 (incorrect) → critique-guided answer 49 (correct) → Stage-II direct answer 49 (correct, no critique).

Question: Divisibility

Find the sum of all positive integers n such that $n+2$ divides the product $3(n+3)(n^2+9)$.

Notation in the early response. The model sets $k = n+2$ in the response below. The original condition requires $3f(k)$ to be an integer; it does not require $f(k)$ itself to be an integer.

Early Direct Response, Step 10 — Incorrect

We are given the condition:

$$n+2 \mid 3(n+3)(n^2+9)$$

We are to find all **positive integers** n that satisfy this, and then **sum** all such n.

Step 1: Understand the divisibility condition
We are told:

$$n+2 \mid 3(n+3)(n^2+9)$$

This means that $n+2$ divides the product $3(n+3)(n^2+9)$. Since $n+2$ is a linear expression, we can simplify the problem by **letting**:

$$k = n+2 \Rightarrow n = k-2$$

Now, we substitute $n = k-2$ into the expression:

$$3(n+3)(n^2+9) = 3((k-2)+3)((k-2)^2+9)$$

Simplify:

- $n+3 = (k-2)+3 = k+1$
- $n^2+9 = (k-2)^2+9 = k^2-4k+4+9 = k^2-4k+13$

So the expression becomes:

$$3(k+1)(k^2-4k+13)$$

Therefore, the condition becomes:

$$k \mid 3(k+1)(k^2-4k+13)$$

We now want to find all **positive integers** k such that this divisibility holds, and then convert back to $n = k-2$, and sum all such n.

Early Direct Response, Step 10 — Incorrect (continued)

Step 2: Analyze the divisibility condition
We have:

$$k \mid 3(k+1)(k^2-4k+13)$$

We can write:

$$\frac{3(k+1)(k^2-4k+13)}{k}$$

We want this to be an integer. Let's simplify:

$$\frac{3(k+1)(k^2-4k+13)}{k} = 3 \cdot \frac{(k+1)(k^2-4k+13)}{k}$$

Let's denote:

$$f(k) = \frac{(k+1)(k^2-4k+13)}{k}$$

We want $f(k)$ to be an integer.
Let's write this as:

$$f(k) = \frac{(k+1)(k^2-4k+13)}{k}$$

We can perform polynomial division or simplify the expression:
Let's compute the numerator:

$$(k+1)(k^2-4k+13) = k^3-4k^2+13k+k^2-4k+13 = k^3-3k^2+9k+13$$

So:

$$f(k) = \frac{k^3-3k^2+9k+13}{k} = k^2-3k+9+\frac{13}{k}$$

So the expression is:

$$f(k) = k^2-3k+9+\frac{13}{k}$$

We want this to be an integer. That means $\frac{13}{k}$ must be an integer, so $k \mid 13$.

Step 3: Find all divisors of 13
Since $k \mid 13$, and k is a **positive integer**, the possible values of k are the **positive divisors** of 13:

$$k \in \{1, 13\}$$

So $k = 1$ or $k = 13$
Now convert back to $n = k - 2$:

- If $k = 1$, then $n = 1 - 2 = -1 \rightarrow$ **not positive**, discard
- If $k = 13$, then $n = 13 - 2 = 11$

So the only positive integer n satisfying the condition is:

$$\boxed{11}$$

Final Answer:

$$\boxed{11}$$

Critique at Step 10 — All Original Fields

error_type.
incorrect divisibility simplification

first_wrong_step.
We want $k \mid 3(k+1)(k^2-4k+13)$, so we simplified it to $f(k) = k^2 - 3k + 9 + \frac{13}{k}$ and concluded that $\frac{13}{k}$ must be an integer.

why_wrong.
The simplification assumes that k divides the entire expression $3(k+1)(k^2-4k+13)$, but the error arises from incorrectly concluding that only $\frac{13}{k}$ needs to be an integer. The full expression includes more than just $\frac{13}{k}$, and the divisibility condition is not equivalent to that.

correction_notion.

- polynomial division and remainder theorem
- modular arithmetic
- divisibility rules for polynomials

fix_direction.
Revisit the divisibility condition using modular arithmetic or polynomial division to correctly determine when $k \mid 3(k+1)(k^2-4k+13)$

missing_condition.
The divisibility condition must be checked for all values of k, not just those dividing 13

do_not_solve.
`true`

Critique-Guided Revision, Step 10 — Correct

We are given the condition:

$$n + 2 \mid 3(n+3)(n^2+9)$$

Our goal is to find all **positive integers** n that satisfy this, and then **sum** all such n.

Step 1: Substitute and simplify
Let's make a substitution to simplify the expression. Define:

$$k = n + 2 \Rightarrow n = k - 2$$

Substitute $n = k - 2$ into the expression $3(n+3)(n^2+9)$:

- $n + 3 = (k-2) + 3 = k + 1$
- $n^2 + 9 = (k-2)^2 + 9 = k^2 - 4k + 4 + 9 = k^2 - 4k + 13$

So the expression becomes:

$$3(k+1)(k^2-4k+13)$$

Thus, the divisibility condition becomes:

$$k \mid 3(k+1)(k^2-4k+13)$$

We now want to find all **positive integers** k such that this divisibility holds, and then convert back to $n = k - 2$, and sum all such n.

Critique-Guided Revision, Step 10 — Correct (continued)

Step 2: Analyze the divisibility condition
We want:

$$k \mid 3(k+1)(k^2-4k+13)$$

Let's consider this expression modulo k. Since $k \mid 3(k+1)(k^2-4k+13)$, the expression must be divisible by k, which implies that the **remainder** of the expression when divided by k must be zero.
Let's compute the expression modulo k:

- $(k+1) \equiv 1 \pmod{k}$
- $(k^2-4k+13) \equiv 0-0+13 = 13 \pmod{k}$

So:

$$3(k+1)(k^2-4k+13) \equiv 3 \cdot 1 \cdot 13 = 39 \pmod{k}$$

Therefore, the expression is divisible by k if and only if:

$$k \mid 39$$

Step 3: Find all divisors of 39
Since $k \mid 39$, and k is a **positive integer**, the possible values of k are the **positive divisors** of 39:

$$k \in \{1, 3, 13, 39\}$$

Now convert back to $n = k - 2$:

- If $k = 1$, then $n = 1 - 2 = -1 \rightarrow$ **not positive**, discard
- If $k = 3$, then $n = 3 - 2 = 1$
- If $k = 13$, then $n = 13 - 2 = 11$
- If $k = 39$, then $n = 39 - 2 = 37$

So the positive integers n that satisfy the condition are:

$$n = 1, 11, 37$$

Final Answer:

$$\boxed{1 + 11 + 37 = 49}$$

Stage-II Direct Response, Step 150 — Correct; No Critique

We are given the problem:
Find the sum of all **positive integers** n such that $n + 2$ divides the product $3(n+3)(n^2+9)$.

Step 1: Understand the divisibility condition
We are told that:

$$n + 2 \mid 3(n+3)(n^2+9)$$

This means that $n + 2$ divides the expression $3(n+3)(n^2+9)$.
Let's denote:

$$f(n) = 3(n+3)(n^2+9)$$

We are looking for all **positive integers** n such that $n + 2 \mid f(n)$.

Stage-II Direct Response, Step 150 — Correct; No Critique (continued)

Step 2: Use modular arithmetic
Let's consider the condition:

$$n+2 \mid f(n) \Rightarrow f(n) \equiv 0 \pmod{n+2}$$

Let's make the substitution:

$$n \equiv -2 \pmod{n+2}$$

So we can substitute $n \equiv -2$ into $f(n)$, and compute $f(n) \mod (n+2)$.
Let's compute:

$$f(n) = 3(n+3)(n^2+9)$$

We substitute $n = -2$ into this expression:

$$f(-2) = 3(-2+3)((-2)^2+9) = 3(1)(4+9) = 3 \cdot 1 \cdot 13 = 39$$

So the condition becomes:

$$f(n) \equiv 0 \pmod{n+2} \iff f(-2) \equiv 0 \pmod{n+2} \Rightarrow 39 \equiv 0 \pmod{n+2} \Rightarrow n+2 \mid 39$$

So now we have:

$$n+2 \mid 39$$

Step 3: Find all positive integers n such that $n+2 \mid 39$
Let's find all positive divisors of 39.

$$39 = 3 \cdot 13 \Rightarrow \text{Positive divisors of 39: } 1, 3, 13, 39$$

So the possible values of $n+2$ are: 1, 3, 13, 39.
We are looking for **positive integers** n, so:

- $n+2 = 1 \Rightarrow n = -1 \rightarrow$ not positive $\rightarrow$ discard
- $n+2 = 3 \Rightarrow n = 1 \rightarrow$ positive $\rightarrow$ keep
- $n+2 = 13 \Rightarrow n = 11 \rightarrow$ positive $\rightarrow$ keep
- $n+2 = 39 \Rightarrow n = 37 \rightarrow$ positive $\rightarrow$ keep

So the valid values of n are: 1, 11, 37

Step 4: Compute the sum of these values

$$1 + 11 + 37 = \boxed{49}$$

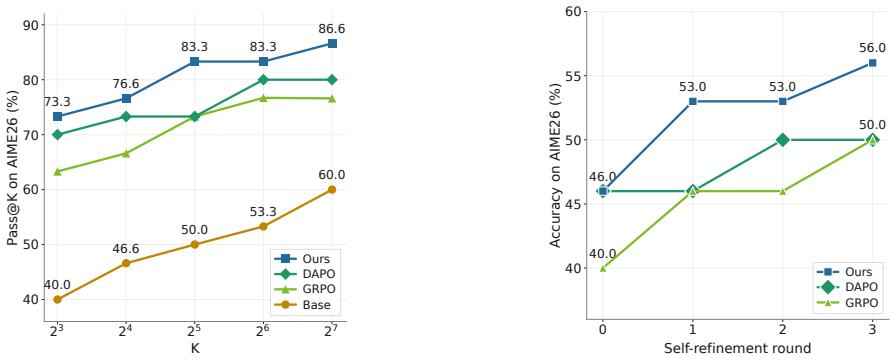
Final Answer:

$$\boxed{49}$$

Analysis. The initial answer incorrectly requires $f(k)$, rather than $3f(k)$, to be integral and therefore misses two valid solutions. The critique questions this equivalence and recommends modular arithmetic, without explicitly stating the missing factor or remainder. The revision obtains $k \mid 39$ and the sum 49. At step 150, the policy directly substitutes $n \equiv -2 \pmod{n+2}$ into the original product and recovers the same relation without feedback. Independent reduction confirms remainder 39 and $n \in \{1, 11, 37\}$. This direct use of the corrected principle complements the grid example; neither selected trajectory alone determines how frequently, or because of which training component, such transfer occurs.

L Parallel and Sequential Exploration

Figure 11 shows that our method improves both parallel sampling and sequential self-refinement. Under parallel sampling, it leads at every budget, with only 32 samples achieving higher coverage than all baselines at 128 samples. Under sequential refinement, our method starts at the same accuracy as DAPO but gains more from subsequent revisions, ultimately outperforming both GRPO and DAPO. These complementary advantages are consistent with the roles of exploration learning and solution internalization: internalization makes critique-guided discoveries accessible in independent attempts, while learned self-critique helps the model identify useful correction directions from ex-



(a) Parallel sampling

(b) Sequential self-refinement

Figure 11: Parallel and sequential exploration on AIME26. (a) Reported pass@k at sampling budgets $k \in \{8, 16, 32, 64, 128\}$, measuring coverage by at least one correct candidate. (b) Accuracy of the current retained answer over rounds 0–3: round 0 is the initial answer, followed by up to three self-check-and-refine rounds. Stopping uses the model's own judgment; stopped answers are carried forward, and the external verifier is used only for scoring. All scores are percentages. The panels use different metrics and budget axes and are not an equal-compute comparison.

isting answers. The benefits therefore extend beyond initial solving performance to more effective use of further exploration opportunities. Overall, our framework strengthens both the discovery of correct solutions across independent attempts and the progressive improvement of answers along a reasoning trajectory, enabling broader problem coverage with fewer samples and further gains through self-generated feedback. Appendix