VARIATIONAL SELF-CORRECTION VIA LATENT DIAGNOSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper studies post-hoc self-correction, in which an LLM revisits an initial response, identifies flaws in its reasoning, and revises the response before producing a final answer. Effective self-correction requires not only detecting what is wrong, but also translating that diagnosis into a successful revision without disrupting reasoning that is already correct. Existing methods, however, typically leave diagnosis implicit or optimize diagnosis and refinement in isolation, weakening the link between identified errors and the revisions they are intended to guide. To address this gap, we propose VaCrit, a variational self-correction framework that treats structured critique as a latent diagnosis connecting an initial response to its refinement. We first learn a critique prior through error-localization supervision and meta-utility learning, where sampled critiques are evaluated by the correction gains they induce. Building on this, we jointly optimize a critique posterior and a unified model for diagnosis and refinement under a shared variational objective, allowing refinement outcomes to assign credit to sampled critiques while preserving oracle-free diagnostic capability. Experiments on 7 challenging mathematical reasoning benchmarks show that VaCrit consistently outperforms existing methods across different model scales.

1 INTRODUCTION

Large language models (LLMs) have made substantial progress on complex reasoning tasks, yet one-shot solutions still contain non-negligible errors, leaving the initial response an unreliable endpoint (Guo et al., 2025; Mirzadeh et al., 2025; Liu et al., 2026). This has motivated growing interest in *post-hoc self-correction*, in which an LLM re-examines its initial response and revises erroneous reasoning before producing a final answer (Madaan et al., 2023; Qu et al., 2024; Kumar et al., 2025; Chen et al., 2026a). A fundamental problem in this setting is how to identify errors in an initial response and use that assessment to repair the flawed reasoning without degrading initially correct answers (Tyen et al., 2024; Yang et al., 2025c). This requires two complementary capabilities: *diagnosis*, which determines whether and where the reasoning fails; and *refinement*, which translates the diagnosis into an effective revision. The two should operate in concert because diagnosis identifies what should be revised, while refinement must correct the identified error without disrupting valid reasoning. Failure at either stage can leave existing errors unresolved or introduce new ones (Huang et al., 2024; Kamoi et al., 2024; Zhang et al., 2025; Xi et al., 2026).

Prior work has pursued several directions, including prompted iterative refinement (Madaan et al., 2023; Kim et al., 2023), multi-turn fine-tuning or RL (Qu et al., 2024; Kumar et al., 2025; Jiang et al., 2025; Yang et al., 2025a; Chen et al., 2026a), learned verification and critique (Paul et al., 2024; Havrilla et al., 2024; Xie et al., 2025; Tang et al., 2026; Yang et al., 2025b; Xi et al., 2026), and test-time localization and regeneration (Samanta et al., 2026; Chitale et al., 2026). However, these advances still leave two structural gaps. ***1) Diagnosis is typically not explicitly modeled within the correction process.*** Direct correction approaches (Qu et al., 2024; Kumar et al., 2025) map the problem and initial response directly to a revised response, leaving whether and where to intervene implicit; feedback-guided approaches (Xie et al., 2025; Tang et al., 2026; Yang et al., 2025b; Xi et al., 2026) introduce explicit critiques or error judgments, but optimize them as auxiliary predictions or through downstream refinement rewards. As a result, diagnostic information remains only loosely tied to revision, making it non-trivial to capture explicitly how identified errors should guide successful revision (Tyen et al., 2024; Tang et al., 2026). ***2) Diagnosis and refinement are typically***

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

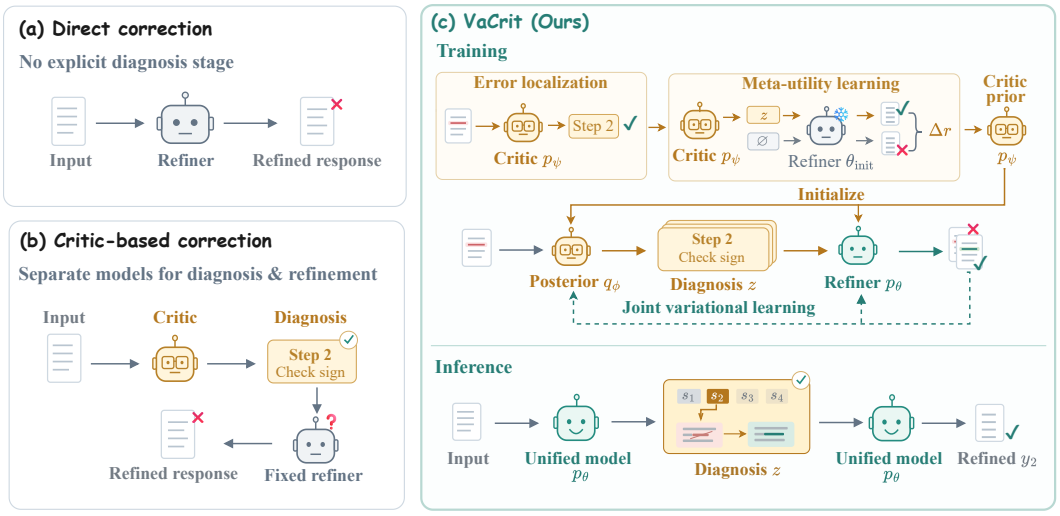


Figure 1: **Learning latent diagnosis and refinement together. (a) Direct correction** revises responses without an explicit diagnosis stage. **(b) Critic-based correction** trains diagnosis separately from refinement, so the refiner may fail to exploit even accurate critiques. **(c) VaCrit (ours)** learns a critique prior through error localization and meta-utility learning, then jointly optimizes the posterior and unified model under a shared variational objective, enabling refinement outcomes to directly shape critiques toward effective correction.

optimized in isolation. For example, critic-based methods primarily train the diagnostic component to predict correctness, localize errors, or produce critiques, whereas the reviser is fixed or optimized under a separate objective (Paul et al., 2024; Xie et al., 2025; Tang et al., 2026; Yang et al., 2025b; Xi et al., 2026). Conversely, methods centered on correction performance optimize the revised response without an explicit objective for preserving the underlying diagnostic capability (Kumar et al., 2025; Chen et al., 2026a). This separation creates an objective mismatch: diagnostically accurate feedback may still be ineffective for revision, while improved final responses do not necessarily imply reliable diagnosis (Tang et al., 2026; Xi et al., 2026).

In this paper, we propose VaCrit, a variational self-correction framework that treats structured critique as a latent diagnosis connecting an initial response to its refinement. We cast self-correction as variational inference over possible diagnoses, so that the learning objective accounts for a distribution of plausible critiques rather than relying on a single independently generated feedback signal. We first learn an oracle-free critique prior by training a critic through error-localization supervision and meta-utility learning, which evaluates sampled critiques by the correction gain they induce over a no-feedback refinement. Building on the learned prior, we then jointly optimize a critique posterior and a unified model for diagnosis and refinement under a shared variational objective. Refinement outcomes assign credit to sampled critiques, while the unified model learns to refine from the same diagnostic feedback; meanwhile, the learned critic serves as a fixed prior and self-distillation teacher to preserve oracle-free diagnostic capability during joint optimization. Together, these components yield a principled probabilistic framework in which diagnosis serves as an explicit latent variable and is optimized jointly with refinement as part of a coupled self-correction process.

We evaluate our framework with Qwen3-4B and Qwen3-8B across 7 challenging mathematical reasoning benchmarks, including MATH-500, AMC23, AIME24–26, Minerva Math, and OlympiadBench. The results demonstrate consistent improvements in final-answer accuracy, exceeding the strongest baseline in mean accuracy by 2.33% with Qwen3-4B and 1.91% with Qwen3-8B. A single-round critique-guided refinement improves accuracy on all seven benchmarks for both models, with mean gains of 4.30% and 4.12% while preserving over 96% of initially correct responses. Moreover, multi-round refinement steadily improves accuracy across successive rounds, raising macro-average accuracy from 47.06% after the first round to 51.79% after the fourth across the four evaluated benchmarks. Further analyses show that critic training strengthens subsequent joint optimization, while repair suggestions are critical for effective correction.

2 Problem Formulation and Variational Objective

Given a problem x and an initial response y_1, the goal of post-hoc self-correction is to find a refined response y_2 that repairs potential errors in y_1 while preserving the correctness of an already-correct one. Training uses a fixed dataset $\mathcal{D} = \{(x_i, y_{1,i}, y_i^*)\}_{i=1}^N$, with y_i^* a verified solution to x_i. For the ease of simplicity, let A_x denote the event that a refined response is verified correct for problem x. We formulate the learning objective as

$$\max_P \mathbb{E}_{(x,y_1)\sim\mathcal{D}}\big[\log P(A_x|x, y_1)\big]. \tag{1}$$

Here, $P(A_x|x, y_1)$ is the probability that a refinement sampled from $P(\cdot|x, y_1)$ is verified correct. Of note, we do not train the refiner by maximizing the token-level likelihood of the reference solution y^*, since a problem generally admits multiple distinct correct responses, and using a single reference as the target would overconstrain learning to a particular realization.

Outcome supervision determines whether a refinement is successful, but does not specify the diagnosis that should guide the revision. In general, a given initial response may admit multiple plausible critiques that diagnose its reasoning and, when necessary, suggest different repair strategies. We thus introduce a latent textual critique z and factorize the correction distribution as

$$p_\theta(y_2|x, y_1) = \sum_z p_\theta(z|x, y_1)\, p_\theta(y_2|x, y_1, z). \tag{2}$$

Here, $p_\theta(z|x, y_1)$ models possible diagnoses of y_1, while $p_\theta(y_2|x, y_1, z)$ generates a revision conditioned on z.[1] For a given critique z, the correction success probability is

$$p_\theta(A_x|x, y_1, z) = \mathbb{E}_{y_2\sim p_\theta(\cdot|x,y_1,z)}\big[\mathbf{1}\{y_2 \in A_x\}\big]. \tag{3}$$

Based on Eq. (3), the marginal probability of producing a correct refinement can be expressed as

$$p_\theta(A_x|x, y_1) = \mathbb{E}_{z\sim p_\theta(\cdot|x,y_1)}\big[p_\theta(A_x|x, y_1, z)\big], \tag{4}$$

coupling the probability assigned to each critique with that leads to a successful refinement.

The marginal correction likelihood in Eq. (4) requires marginalizing over the latent critique z, making direct optimization intractable. Following the standard variational inference (Jordan et al., 1999), we introduce a trainable posterior $q_\phi(z|x, y_1, y^*)$ to approximate the latent critique distribution and derive an evidence lower bound (ELBO) on $\log p_\theta(A_x|x, y_1)$:

$$\log p_\theta(A_x|x, y_1) \geq \underbrace{\mathbb{E}_{z\sim q_\phi(\cdot|x,y_1,y^*)}\left[\log p_\theta(A_x|x, y_1, z)\right] - D_{\mathrm{KL}}(q_\phi(\cdot|x, y_1, y^*)\|p_\theta(\cdot|x, y_1))}_{\mathcal{L}_{\mathrm{ELBO}}(\theta,\phi)}. \tag{5}$$

This ELBO provides the variational foundation for the training objective developed in Section 3.3. Its first term encourages critiques inducing successful refinements, while the KL term regularizes the reference-conditioned posterior toward the oracle-free critique distribution. During training, the reference solution y^* can provide privileged evidence for inferring plausible critiques, without requiring gold critique supervision; meanwhile, from Eq. (3), this form enables policy optimization from sampled refinements and verifier-assessed correctness (see Section 3.3). For completeness, a detailed derivation is provided in Appendix C.1.

3 Variational Critique-Guided Self-Correction

The variational formulation in Section 2 couples diagnosis and refinement, which requires reliable oracle-free diagnosis and its preservation as the refiner is optimized. To this end, we first define a structured critique interface for an oracle-free critic p_ψ and train its diagnostic capability through error localization and meta-utility learning. We then jointly optimize the posterior q_ϕ and model p_θ, using the learned critic both as the critique prior and as a self-distillation teacher to retain diagnostic capability while improving refinement.

[1] We use one model θ for both diagnosis and refinement; in general, the two components can be instantiated by different models.

3.1 STRUCTURED CRITIQUES

The reference-conditioned posterior $q_\phi(z|x, y_1, y^*)$ can exploit y^* to diagnose y_1, but an unrestricted critique could directly convey solution information to the refiner. We therefore separate the reasoning used for diagnosis from the feedback exposed during refinement. Following structured critique methods (Xie et al., 2025; Tang et al., 2026), we instantiate

$$z = (z_{\text{think}}, z_{\text{idx}}, z_{\text{sugg}}), \qquad c(z) = (z_{\text{idx}}, z_{\text{sugg}}), \tag{6}$$

where z_{think} is the diagnostic reasoning trace, z_{idx} identifies the first erroneous step, and z_{sugg} explains the error and suggests a repair. For $y_1 = (s_1, \ldots, s_t)$, we have $z_{\text{idx}} \in \{-1, 1, \ldots, t\}$, where -1 indicates that no error is detected. z_{idx} determines whether and where revision should begin, while z_{sugg} specifies how the identified issue can be repaired. The refinement is instructed to preserve the preceding prefix and regenerate from the predicted error onward; when no error is detected, it reproduces the initial response. Both $p_\theta(z|x, y_1)$ and $q_\phi(z|x, y_1, y^*)$ use this critique format, but only $c(z)$ is provided to the refiner. Accordingly, the refinement distribution $p_\theta(y_2|x, y_1, z)$ in Section 2 is instantiated as $p_\theta(y_2|x, y_1, c(z))$. In this way, the context retains the actionable information needed for revision while withholding the reference-conditioned diagnostic reasoning in z_{think}, reducing direct leakage of privileged information to the refiner. Prompt templates, extraction rules, and formatting details are provided in Appendix B.2.

3.2 LEARNING AN ORACLE-FREE CRITIQUE PRIOR

We learn an oracle-free critic $p_\psi(z|x, y_1)$ whose distribution serves as the critique prior. To make the critique useful for self-correction, it must capture both where the reasoning fails and what feedback can actually improve the subsequent revision. To this end, we train it in two stages: *(i) error localization* to learn whether and where y_1 becomes incorrect, and *(ii) meta-utility learning*, which evaluates critiques through the refinement outcomes they induce and learns feedback that supports successful correction.

Error localization. We first enhance the critique prior's capability of determining whether and where y_1 becomes incorrect. Denote $\mathcal{D}_{\text{loc}} = \{(x_i, y_{1,i}, e_i^*)\}$, where e^* denotes the first erroneous step and $e^* = -1$ indicates an error-free response. The objective is to maximize

$$\mathcal{J}_{\text{loc}}(\psi) = \mathbb{E}_{(x,y_1,e^*)\sim\mathcal{D}_{\text{loc}}, z\sim p_\psi(\cdot|x,y_1)} \left[\mathbf{1}\{z_{\text{idx}} = e^*\}\right]. \tag{7}$$

This stage supervises whether and where to revise, equipping the critique prior with explicit error-localization capability, but does not assess whether the resulting critique is useful for refinement.

Meta-utility learning. Starting from the localization checkpoint, we proceed to learn the critique prior according to the refinement outcomes it induces. We use the frozen base model θ_{init} as the refiner, providing a fixed evaluator of critique utility. For a sampled critique $z \sim p_\psi(\cdot|x, y_1)$, denote y_2^z and $y_2^\varnothing$ as the refinements produced by θ_{init} from (x, y_1) with critique feedback $c(z)$ and with no feedback, respectively. Define the *meta-utility* as the correctness gain over no-feedback refinement:

$$\mathcal{J}_{\text{meta}}(\psi) = \mathbb{E}_{(x,y_1)\sim\mathcal{D}, z\sim p_\psi(\cdot|x,y_1)} \left[\Delta r(z)\right], \tag{8}$$

where $\Delta r(z) = \mathbf{1}\{y_2^z \in A_x\} - \mathbf{1}\{y_2^\varnothing \in A_x\}$ measures the critique-induced gain in refinement correctness over the no-feedback baseline. This directly aligns the diagnostic capability with downstream correction utility, while the no-feedback refinement provides a reference for measuring the gain induced by each critique.

Since both critic-learning objectives are defined over sampled critique trajectories with outcome-based rewards, they are naturally suited to policy-gradient optimization. Both stages thus can be optimized by standard RL approaches; implementation details are provided in Appendix C.2.

3.3 JOINT POLICY OPTIMIZATION OF DIAGNOSIS AND REFINEMENT

We next jointly learn the reference-conditioned posterior q_ϕ and a *unified* model p_θ that integrates oracle-free diagnosis and critique-guided refinement. Both are initialized from the learned critic p_ψ, which is then frozen and used throughout joint training. This stage serves two purposes: the refiner

Algorithm 1 VaCrit: Variational Critique-Guided Self-Correction

procedure TRAIN($\theta_{\text{init}}, \mathcal{D}_{\text{loc}}, \mathcal{D}$)
  // Learn the critic prior
  Initialize p_ψ from θ_{init};
  Optimize p_ψ with $\mathcal{J}_{\text{loc}}$ (Eq. (7)) on $\mathcal{D}_{\text{loc}}$;
  Optimize p_ψ with $\mathcal{J}_{\text{meta}}$ (Eq. (8)) on $\mathcal{D}$, using θ_{init} as the frozen refiner;
  // Joint variational learning and self-distillation
  Initialize q_ϕ and p_θ from p_ψ; freeze p_ψ;
  for each minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
    Draw K posterior critiques $z_k \sim q_\phi(\cdot|x, y_1, y^*)$ per instance in $\mathcal{B}$;
    Draw K prior critiques $\tilde{z}_k \sim p_\psi(\cdot|x, y_1)$ per instance online for diagnostic retention;
    Draw M refinements $y_{2,k,m} \sim p_\theta(\cdot|x, y_1, c(z_k))$ per posterior critique;
    Score each refinement using $\mathbf{1}\{y_{2,k,m} \in A_x\}$;
    Update q_ϕ and p_θ by maximizing $\mathcal{J}_{\text{uni}}$ (Eq. (9));
  end for
  return p_θ
end procedure

procedure SELFCORRECT(x, p_θ)
  Generate the initial response $y_1 \sim p_\theta(\cdot|x)$;
  Diagnose y_1 with $z \sim p_\theta(\cdot|x, y_1)$;
  Retain $c(z) = (z_{\text{idx}}, z_{\text{sugg}})$ as feedback;
  Generate the refinement $y_2 \sim p_\theta(\cdot|x, y_1, c(z))$;
  return y_2
end procedure

should learn to effectively exploit diagnostic feedback for correction; and the unified model should retain the critic's diagnostic capability.

Specifically, we optimize the refiner under the variational objective in Eq. (5), using posterior critiques as feedback for learning successful correction. Meanwhile, we use the frozen critic p_ψ as a *self-distillation teacher*, regularizing p_θ toward the diagnostic capability of p_ψ. For compactness, let $s_\theta(x, y_1, z) \doteq p_\theta(A_x|x, y_1, c(z)) = \mathbb{E}_{y_2 \sim p_\theta(\cdot|x, y_1, c(z))}[\mathbf{1}\{y_2 \in A_x\}]$. The resulting joint objective is

$$\begin{aligned}\mathcal{J}_{\text{uni}}(\theta, \phi) = \mathbb{E}_{(x, y_1, y^*) \sim \mathcal{D}} \big[& \mathbb{E}_{z \sim q_\phi(\cdot|x, y_1, y^*)}[\log s_\theta(x, y_1, z)] \\ & - \beta_1 D_{\text{KL}}(q_\phi(\cdot|x, y_1, y^*) \| p_\psi(\cdot|x, y_1)) - \beta_2 D_{\text{KL}}(p_\psi(\cdot|x, y_1) \| p_\theta(\cdot|x, y_1)) \big], \end{aligned} \tag{9}$$

where $\beta_1, \beta_2 \geq 0$ control posterior-prior regularization and diagnostic self-distillation, respectively. The first two terms jointly optimize the posterior and refiner, with q_ϕ favoring critiques that support successful correction and p_θ learning to refine effectively from them. The final term distills the oracle-free diagnostic capability of p_ψ into p_θ. Together, these objectives integrate diagnosis and refinement within the shared model p_θ used at deployment.

Remarks. Analogous to weighted variational objectives (Higgins et al., 2017), β_1 and β_2 control posterior regularization and diagnostic retention, respectively. The β_1 term controls how strongly the posterior may deviate from the learned critique prior. For a fixed refiner, omitting the conditioning variables, the unrestricted optimum is $q^*_{\beta_1}(z) \propto p_\psi(z)\, s_\theta(z)^{1/\beta_1}$. Thus, β_1 governs the trade-off between prior plausibility and correction utility, where smaller values place greater weight on critiques that induce successful refinement, whereas larger ones keep the posterior closer to the learned prior. In contrast, the self-distillation term serves a different role, anchoring the oracle-free diagnostic behavior of p_θ to the learned critic p_ψ. When $\beta_1 = 1, \beta_2 = 0$, Eq. (9) recovers the standard ELBO; for $\beta_1 \geq 1, \beta_2 \geq 0$, the additional nonnegative penalties loosen this bound.

At inference, we discard q_ϕ and p_ψ and retain only the unified model p_θ. The same model performs initial generation, diagnosis, and critique-guided refinement:

$$y_1 \sim p_\theta(\cdot|x), \qquad z \sim p_\theta(\cdot|x, y_1), \qquad y_2 \sim p_\theta(\cdot|x, y_1, c(z)). \tag{10}$$

As in training, $c(z)$ extracts the predicted error location and repair suggestion, withholding the diagnostic reasoning trace from the refinement context.

Overall, we term the proposed method *Variational Critique-Guided Self-Correction* (VaCrit), with the complete training and inference procedure summarized in Algorithm 1.

4 EXPERIMENTS

In this section, we evaluate VaCrit across seven challenging mathematical reasoning benchmarks, examining both final-answer accuracy and self-correction behavior.

4.1 EXPERIMENTAL SETUP

Base model and baselines. We instantiate θ_{init} with Qwen3-4B and Qwen3-8B (Qwen Team, 2025) and compare VaCrit against baselines built on the same backbone. As a training-free baseline, **Prompt** uses the frozen base model with the self-correction instruction from SCoRe (Kumar et al., 2025), which asks the model to identify and correct potential errors in its previous response and rewrite the solution without external correctness feedback. We include two trained direct self-correction methods: **STaR+**, which learns from incorrect-to-correct and correct-to-correct supervised trajectories, and **SCoRe**, which uses two-stage on-policy RL. For feedback-guided correction, **PAG** learns self-verification and selective correction (Jiang et al., 2025), while **CURE** regenerates a solution conditioned on the problem and a self-generated critique, without retaining the initial response (Chen et al., 2026a). We additionally report results from the released Qwen2.5-Math-7B checkpoints of **SuperCorrect** (Yang et al., 2025a), which learns through reasoning distillation and cross-model preference optimization, and **S^{2}R-ORL** (Ma et al., 2025), which performs verification and correction within a single reasoning trajectory. Because these released checkpoints use a different backbone, we report them only as contextual references for self-correction behavior in Table 2. Training-data alignment and baseline configuration details are provided in Appendix D.2.

Benchmarks and metrics. We evaluate final answer correctness on seven mathematical reasoning benchmarks: MATH-500 (Lightman et al., 2024), AMC 2023, AIME 2024–2026 (Zhang & Math-AI, 2024; 2025; 2026), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). ProcessBench (Zheng et al., 2025) provides a separate evaluation of first-error localization and recognition of error-free trajectories. We use Math-Verify to assess correctness for both initial and refined responses and report final-answer accuracy $\text{Acc}_{\text{final}}$, and self-correction gain $\Delta = \text{Acc}_{\text{final}} - \text{Acc}_{t_1}$, with Acc_{t_1} the initial-response accuracy. Following Yang et al. (2025c), we report *Critique Score* (CS), the percentage of initially incorrect responses corrected; *Confidence Level* (CL), the percentage of initially correct responses preserved; and *Relative Self-Correction Score* (RSS), which combines error repair and correctness preservation while accounting for initial accuracy. For ProcessBench, we report F1 as the harmonic mean of exact first-error localization accuracy on erroneous trajectories and no-error recognition accuracy on error-free trajectories (Zheng et al., 2025). Formal metric definitions and aggregation procedures are provided in Appendix D.3. Training details are provided in Appendix D.

4.2 VARIATIONAL SELF-CORRECTION IMPROVES FINAL ANSWER ACCURACY

We first compare final-answer accuracy and self-correction gains across methods (Table 1). Our method achieves the highest mean final accuracy among the evaluated methods using the same base model: 49.10% on Qwen3-4B and 50.48% on Qwen3-8B, exceeding SCoRe by 2.33 and 1.91, respectively. It also achieves mean self-correction gains of 4.30% and 4.12% after a single critique-guided refinement, compared with 1.16 and 1.08 for SCoRe on the two models. These gains stem from explicitly coupling diagnosis and refinement. Our method conditions revision on a predicted first-error location and repair suggestion, while joint training rewards critiques by the correct-refinement probability they induce and trains the refiner under the same feedback. In particular, although our objective does not directly supervise initial-response generation, diagnosis and refinement training improve the shared reasoning parameters by teaching the model to recognize and avoid erroneous reasoning patterns. This improves initial accuracy as well, consistent with prior findings that self-verification training can enhance problem solving without an explicit generation reward (Chen et al., 2026b).

Table 1: **Final-answer accuracy and self-correction gains.** Each cell reports final-answer accuracy (%), with the subscript showing the change from the initial-response accuracy, $\Delta = \mathrm{Acc}_{\mathrm{final}} - \mathrm{Acc}_{t_1}$, in percentage points. Positive (green) values indicate improvement after self-correction, while negative (red) values indicate degradation. Avg. is the unweighted mean across seven benchmarks. Bold marks the best final accuracy for each backbone.

Method	MATH-500	AMC23	AIME24	AIME25	AIME26	Minerva	Olympiad	Avg.
				Qwen3-4B				
Prompt	82.60$_{+0.00}$	67.81$_{+1.02}$	23.33$_{+0.62}$	21.15$_{+1.98}$	18.44$_{+1.77}$	40.44$_{+1.10}$	52.82$_{+3.41}$	43.80$_{+1.41}$
STaR+	84.40$_{-0.80}$	68.83$_{+2.58}$	24.58$_{+2.19}$	21.56$_{+0.73}$	17.19$_{+0.31}$	38.97$_{+0.37}$	52.82$_{+0.59}$	44.05$_{+0.85}$
SCoRe	86.40$_{+0.80}$	71.64$_{+1.09}$	26.35$_{+1.46}$	23.12$_{+0.31}$	21.46$_{+1.46}$	**42.65**$_{+1.10}$	55.79$_{+1.93}$	46.77$_{+1.16}$
PAG	85.60$_{+0.60}$	69.61$_{+0.62}$	23.33$_{+0.94}$	23.02$_{+0.83}$	17.92$_{-0.31}$	40.44$_{-0.37}$	53.12$_{+0.74}$	44.72$_{+0.44}$
CURE	87.20$_{+0.80}$	68.91$_{-2.89}$	26.67$_{+0.52}$	22.60$_{+1.25}$	19.69$_{-0.42}$	40.44$_{+0.74}$	54.45$_{-0.45}$	45.71$_{-0.06}$
VaCrit (Ours)	**90.00**$_{+3.60}$	**74.38**$_{+4.77}$	**27.50**$_{+2.81}$	**27.29**$_{+4.79}$	**24.69**$_{+5.94}$	40.81$_{+3.31}$	**59.05**$_{+4.90}$	**49.10**$_{+4.30}$
				Qwen3-8B				
Prompt	84.80$_{+0.80}$	69.92$_{+0.08}$	28.75$_{+1.98}$	21.67$_{+0.21}$	18.96$_{+1.98}$	39.71$_{-2.21}$	56.08$_{+1.34}$	45.70$_{+0.60}$
STaR+	85.00$_{-1.20}$	69.61$_{+0.47}$	28.33$_{+1.46}$	22.19$_{+0.94}$	17.92$_{+1.56}$	**43.01**$_{+1.10}$	55.19$_{+0.30}$	45.89$_{+0.66}$
SCoRe	87.80$_{+1.80}$	74.45$_{+1.48}$	32.50$_{+1.87}$	24.27$_{+0.31}$	20.73$_{+1.35}$	41.18$_{-0.74}$	59.05$_{+1.48}$	48.57$_{+1.08}$
PAG	85.00$_{-0.40}$	69.84$_{-0.08}$	25.62$_{-0.63}$	22.29$_{+0.31}$	17.19$_{-0.31}$	42.65$_{+0.00}$	54.45$_{+0.00}$	45.29$_{-0.16}$
CURE	84.80$_{-0.60}$	68.98$_{-1.17}$	25.21$_{-1.56}$	22.71$_{+0.21}$	17.29$_{+1.04}$	42.65$_{+1.10}$	55.04$_{+0.15}$	45.24$_{-0.12}$
VaCrit (Ours)	**90.80**$_{+2.80}$	**74.69**$_{+4.53}$	**33.12**$_{+4.06}$	**26.56**$_{+3.54}$	**23.96**$_{+7.08}$	41.91$_{+0.74}$	**62.31**$_{+6.08}$	**50.48**$_{+4.12}$

Table 2: **Error repair and correctness preservation.** Values are unweighted means across seven benchmarks; CS and CL are reported in %. SuperCorrect and S^{2}R-ORL use Qwen2.5-Math-7B, while all other methods use Qwen3-4B. Bold marks the best value for each metric. Complete results are provided in Appendix D.5, with the corresponding Qwen3-8B results in Appendix D.5.

Metric	Prompt	STaR+	SCoRe	PAG	CURE	SuperCorr.	S^{2}R-ORL	**VaCrit (Ours)**
RSS ↑	53.71	52.03	53.05	51.23	50.10	48.03	47.28	**62.04**
CS (%) ↑	6.94	5.56	5.20	3.06	**16.82**	2.76	8.29	13.19
CL (%) ↑	96.29	95.21	97.77	**97.90**	82.11	88.67	83.83	97.28

4.3 Improved Error Repair with High Correctness Preservation

We next examine the self-correction behavior of the Qwen3-4B models (Table 2); results on Qwen3-8B are provided in Appendix D.5. Among the baselines, PAG combines high mean CL (97.90%) with low mean CS (3.06%). Its selective revision preserves responses judged correct, while limited error repair arises from missed errors or unsuccessful correction after detection (Jiang et al., 2025). CURE achieves higher mean CS (16.82%) but lower mean CL (82.11%). Once revision is triggered, regenerating from the problem and a self-generated hint may help escape faulty reasoning, but can also overwrite a correct response when verification is mistaken (Chen et al., 2026a). Appendix E shows one case in which both PAG and CURE fail to identify an error and retain the incorrect initial answer. VaCrit achieves a higher mean RSS than both baselines, combining stronger error repair than PAG with similarly high correctness preservation. This can be attributed to the model retaining the initial response, using the predicted error location and repair suggestion to target edits, and preserving valid content when no error is found (Appendix B.1).

4.4 Ablations of Training Initialization and Inference-Time Feedback

We use Qwen3-4B to examine the contributions of individual training stages and feedback components to self-correction, and whether further refinement rounds yield additional gains. Table 3 compares the full method with ablations of training initialization and inference-time feedback. Per-benchmark results, RSS, and ablation details are provided in Appendix D.6.

Critic training improves self-correction. With the subsequent joint-training schedule and settings fixed, initializing from the critic trained with both localization and meta-utility learning raises

Table 3: **Ablation studies of critic training and refinement feedback.** All variants use Qwen3-4B; values are unweighted *means across seven benchmarks*. Accuracy, CS, and CL are reported in %; Gain denotes final minus initial accuracy. Bold marks the best value for each metric. *w/o both fields* removes both the predicted error index and repair suggestion.

Variant	Initial ↑	Final ↑	Gain ↑	CS ↑	CL ↑
VaCrit (full)	**44.80**	**49.10**	**+4.30**	**13.19**	97.28
Critic training					
w/o meta-utility	44.04	47.64	+3.60	12.97	96.49
w/o critic training	44.02	47.16	+3.14	10.95	96.58
Refinement feedback					
w/o error index	44.80	49.09	+4.29	12.86	97.26
w/o suggestion	44.80	44.74	−0.06	2.16	98.35
w/o both fields	44.80	44.77	−0.03	0.91	**98.97**

mean final accuracy from 47.16% to 49.10% compared with base-model initialization. Localization alone primarily improves error repair, increasing mean CS from 10.95% to 12.97%. Adding meta-utility learning further raises mean final accuracy from 47.64% to 49.10%. The results show that localization and meta-utility learning provide complementary initialization for joint training. The former strengthens error repair, while the latter further improves final correction performance.

Feedback enables effective self-correction. With the trained checkpoint fixed, full feedback raises mean final accuracy from 44.77% to 49.10% and mean CS from 0.91% to 13.19%, improving accuracy on all seven benchmarks. Suggestion-only feedback retains nearly the full correction gain (+4.29 vs. +4.30), whereas index-only feedback yields essentially no gain (−0.06). This suggests that the repair suggestion carries most of the actionable information, while the explicit index provides limited additional benefit because the suggestion can itself convey where revision is needed.

4.5 Self-Correction over Multiple Iterations

We compare VaCrit with Prompt, SCoRe, PAG, and CURE on AMC23, AIME24, AIME25, and OlympiadBench, using Qwen3-4B for up to four refinement rounds. Across rounds, each active trajectory retains only the problem and its latest answer; feedback-guided methods generate a fresh critique at every round. CURE regenerates from the problem and the current hint. For a controlled comparison, we apply the same restricted context to SCoRe and PAG, although their original formulations retain more interaction history (Kumar et al., 2025; Jiang et al., 2025).

As shown in Fig. 2, VaCrit continues to improve with additional refinement rounds. Its macro-average accuracy across these four benchmarks rises from 47.06% after the first refinement to 51.79% after the fourth (+4.74). Accuracy increases at every round on four benchmarks, with gains from the first to the fourth refinement of 2.96 points on AMC23, 5.94 on AIME24, 6.04 on AIME25, and 4.01 on OlympiadBench. Over the same rounds, Prompt, PAG, and CURE gain only 0.55, 0.42, and 0.77, respectively, while SCoRe remains nearly unchanged. At the fourth refinement, VaCrit achieves the highest accuracy on all four benchmarks, with a 7.57 higher macro-average accuracy than the strongest baseline.

5 Related Work

Self-correction in LLM reasoning can be broadly divided into two forms: *post-hoc correction*, which revises a completed response, and *spontaneous correction*, which integrates verification and revision within an ongoing generation. A more detailed discussion is provided in Appendix A.

Post-hoc self-correction. Post-hoc self-correction can be elicited through prompting, using external feedback or the model's own assessment to guide revision (Shinn et al., 2023; Gou et al., 2024; Madaan et al., 2023). However, evidence for intrinsic correction without informative external feedback remains mixed. Revision may improve some responses but can also overlook errors or turn correct answers into incorrect ones (Liu et al., 2024; Huang et al., 2024; Kamoi et al., 2024;

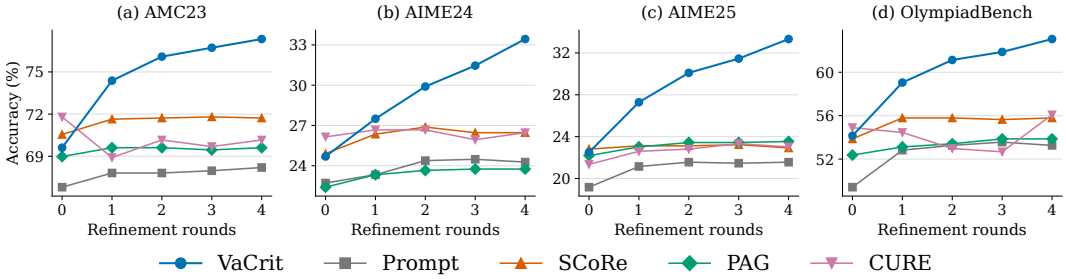


Figure 2: **Accuracy across refinement iterations with Qwen3-4B.** Iteration 0 denotes the initial response y_1; iterations 1–4 produce y_2–y_5, respectively. Each point reports accuracy over all evaluated samples at that iteration. PAG and CURE terminate refinement once self-verification accepts the current answer; the accepted response is then carried forward for the remaining iterations.

Zhang et al., 2025). Training-based methods seek to improve this capability more systematically. Direct-correction approaches train models to improve successive attempts through fine-tuning or RL, leaving what should be changed implicit in the revision process (Qu et al., 2024; Kumar et al., 2025). Feedback-guided approaches instead introduce an explicit diagnostic stage, using self-verification, critique, or reflection to guide subsequent refinement (Paul et al., 2024; Yang et al., 2025c; Xiong et al., 2025; Jiang et al., 2025; Yang et al., 2025a; Chen et al., 2026a). A prominent form of such diagnosis is error localization, which identifies where revision should begin using learned reward models or inference-time signals (Havrilla et al., 2024; Samanta et al., 2026; Chitale et al., 2026). Yet, diagnosis and repair are distinct capabilities, where models can fail to locate an error even when they can repair it once its location is provided (Tyen et al., 2024).

Spontaneous self-correction. Correction can also occur within a single ongoing generation. Such behavior can emerge through reasoning-oriented RL (Guo et al., 2025) or be encouraged by training on trajectories that demonstrate recovery from errors (Pan et al., 2025; Yan et al., 2025; Kim et al., 2025). Verification can also be trained explicitly to guide subsequent reasoning and refinement (Ma et al., 2025; Lee et al., 2025; Zhao et al., 2025), while inference-time methods such as budget forcing and reflection scheduling control when and how much models reflect (Muennighoff et al., 2025; Wang et al., 2025d; Fan et al., 2026). More reflection, however, does not necessarily yield more effective correction, because long reasoning can contain redundant reconsideration (Chen et al., 2025), and self-verification may simply reconfirm existing conclusions without detecting or repairing errors (Long et al., 2026).

Critique Models for LLMs Critique models explicitly assess model responses and generate feedback such as correctness judgments, error localization, and repair suggestions (Paul et al., 2024; Tang et al., 2026). Early approaches train such diagnostic capabilities directly. For example, Paul et al. (2024) learn structured feedback from supervised examples, while Yang et al. (2025b) further optimize first-error localization through RL. However, better diagnosis does not necessarily yield better refinement (Tang et al., 2026), motivating methods that train critiques according to their downstream correction utility. Xie et al. (2025); Yu et al. (2025) optimize critiques from refinement outcomes, and Tang et al. (2026); Xi et al. (2026) combine diagnostic supervision with refinement-based objectives. Unlike these methods, which pair a dedicated critic with a separate generator, we couple diagnosis and refinement in a unified model, enabling a single checkpoint to generate, diagnose, and refine at inference.

6 Conclusion

We presented VaCrit, a variational framework for post-hoc self-correction that treats structured critiques as latent diagnoses and jointly learns diagnosis and refinement. It first trains a critic prior through error localization and meta-utility learning, then jointly optimizes the critique posterior and a unified model with diagnostic self-distillation, enabling a single model to generate, critique, and refine without reference solutions at inference. Experiments across seven mathematical reasoning benchmarks with Qwen3-4B and Qwen3-8B corroborate the effectiveness of VaCrit, showing consistent gains from both single- and multi-round refinement. Future work could extend the framework to broader reasoning domains, and further improve iterative self-correction over longer horizons.

AI USE STATEMENT

We use large language models (LLMs) as assistive tools solely for polishing the wording and language of the manuscript. All AI-assisted edits are reviewed and, where necessary, revised by the authors. The authors take full responsibility for the correctness and integrity of the final manuscript.

REPRODUCIBILITY STATEMENT

The algorithmic details of VaCrit are presented in Section 3 and Algorithm 1. Data preprocessing, training hyperparameters, decoding settings, and evaluation procedures are provided in Appendix D. Derivations of the variational objective and policy gradients are provided in Appendix C.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, et al. On-policy distillation of language models: Learning from self-generated mistakes. In *Proceedings of ICLR 2024*, 2024.

Edoardo Cetin, Tianyu Zhao, and Yujin Tang. Reinforcement learning teachers of test time scaling. In *Proceedings of NeurIPS 2025*, 2025.

Kaiyan Chang, Yonghao Shi, Chenglong Wang, Hang Zhou, Chi Hu, et al. Step-level verifier-guided hybrid test-time scaling for large language models. In *Proceedings of EMNLP 2025*, pp. 18462–18477, 2025.

Guirong Chen, Shuqi Ye, Wenkai Yang, Shiqi Shen, Guangyao Shen, et al. CURE: critique-driven unified reinforcement learning for test-time self-improvement. In *Proceedings of ACL 2026*, pp. 28632–28653, 2026a.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, et al. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Proceedings of ICML 2025*, volume 267, 2025.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *Proceedings of ICLR 2024*, 2024.

Yuxin Chen, Yu Wang, Yi Zhang, Ziang Ye, Zhengzhou Cai, et al. Learning to self-verify makes language models better reasoners. In *Proceedings of ICML 2026*, 2026b.

Rajiv Shailesh Chitale, Rahul Madhavan, Taneesh Gupta, Deepanway Ghosal, and Aravindan Raghuveer. Test-time scaling via error localization. *CoRR*, abs/2607.21453, 2026.

Chongyu Fan, Yihua Zhang, Jinghan Jia, Alfred O. Hero, and Sijia Liu. CyclicReflex: Improving reasoning models via cyclical reflection token scheduling. In *Proceedings of ICLR 2026*, 2026.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, et al. CRITIC: large language models can self-correct with tool-interactive critiquing. In *Proceedings of ICLR 2024*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Alexander Havrilla, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, et al. GLoRe: When, where, and how to improve LLM reasoning via global and local refinements. In *Proceedings of ICML 2024*, volume 235, pp. 17719–17733, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, et al. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of ACL 2024*, pp. 3828–3850, 2024.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, et al. DeepMath-103K: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. In *Proceedings of ICLR 2026*, 2026.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, et al. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of ICLR 2017*, 2017.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, et al. LoRA: Low-rank adaptation of large language models. In *Proceedings of ICLR 2022*, 2022.

Edward J. Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, et al. Amortizing intractable inference in large language models. In *Proceedings of ICLR 2024*, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, et al. Large language models cannot self-correct reasoning yet. In *Proceedings of ICLR 2024*, 2024.

Yuhua Jiang, Yuwen Xiong, Yufeng Yuan, Chao Xin, Wenyuan Xu, et al. PAG: multi-turn reinforced LLM self-correction with policy as generative verifier. *CoRR*, abs/2506.10406, 2025.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, 1999.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs. *Trans. Assoc. Comput. Linguistics*, 12:1417–1440, 2024.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. In *Proceedings of NeurIPS 2023*, 2023.

Joongwon Kim, Anirudh Goyal, Liang Tan, Hannaneh Hajishirzi, Srinivasan Iyer, and Tianlu Wang. ASTRO: teaching language models to reason by reflecting and backtracking in-context. *CoRR*, abs/2507.00417, 2025.

Minsu Kim, Jean-Pierre R. Falet, Oliver Ethan Richardson, Xiaoyin Chen, Moksh Jain, et al. Latent veracity inference for identifying errors in stepwise reasoning. In *Proceedings of ICLR 2026*, 2026.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, et al. Training language models to self-correct via reinforcement learning. In *Proceedings of ICLR 2025*, 2025.

Hyunseok Lee, Seunghyuk Oh, Jaehyung Kim, Jinwoo Shin, and Jihoon Tack. ReVISE: Learning to refine at test-time via intrinsic self-verification. In *Proceedings of ICML 2025*, volume 267, 2025.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, et al. Solving quantitative reasoning problems with language models. In *Proceedings of NeurIPS 2022*, 2022.

Loka Li, Guangyi Chen, Yusheng Su, Zhenhao Chen, Yixuan Zhang, et al. Confidence matters: Revisiting intrinsic self-correction capabilities of large language models. *CoRR*, abs/2402.12563, 2024a.

Yanhong Li, Chenghao Yang, and Allyson Ettinger. When hindsight is not 20/20: Testing limits on reflective thinking in large language models. In *Findings of NAACL 2024*, pp. 3741–3753, 2024b.

Zhicong Li, Lingjie Jiang, Yulan Hu, Xingchen Zeng, Yixia Li, et al. No more stale feedback: Co-evolving critics for open-world agent learning. In *Proceedings of ACL 2026*, pp. 12643–12660, 2026.

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *Proceedings of ECCV 2016*, volume 9908, pp. 614–629, 2016.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, et al. Let's verify step by step. In *Proceedings of ICLR 2024*, 2024.

Tianqianjin Lin, Xi Zhao, Xingyao Zhang, Rujiao Long, Yi Xu, et al. RAVR: reference-answer-guided variational reasoning for large language models. In *Findings of ACL 2026*, pp. 18757–18775, 2026.

Dancheng Liu, Amir Nassereldine, Ziming Yang, Chenhui Xu, Yuting Hu, et al. Large language models have intrinsic self-correction ability. *CoRR*, abs/2406.15673, 2024.

Junlin Liu, Shengnan An, Shuang Zhou, Dan Ma, Yehao Lin, et al. AMO-Bench: Large language models still struggle in high school math competitions. In *Findings of ACL 2026*, pp. 2120–2137, 2026.

Quanyu Long, Kai Jie Jiang, Jianda Chen, Xu Guo, Leilei Gan, et al. Self-verification dilemma: Experience-driven suppression of overused checking in LLM reasoning. *CoRR*, abs/2602.03485, 2026.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, et al. S^2R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In *Proceedings of ACL 2025*, pp. 22632–22654, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, et al. Self-Refine: Iterative refinement with self-feedback. In *Proceedings of NeurIPS 2023*, 2023.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, et al. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *Proceedings of ICLR 2025*, 2025.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, et al. s1: Simple test-time scaling. In *Proceedings of EMNLP 2025*, pp. 20275–20321, 2025.

Theo X. Olausson, Jeevana Priya Inala, Chenglong Wang, Jianfeng Gao, and Armando Solar-Lezama. Is self-repair a silver bullet for code generation? In *Proceedings of ICLR 2024*, 2024.

Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, et al. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Trans. Assoc. Comput. Linguistics*, 12:484–506, 2024.

Zhuoshi Pan, Yu Li, Honglin Lin, Qizhi Pei, Zinan Tang, et al. LEMMA: learning from errors for mathematical advancement in LLMs. In *Findings of ACL 2025*, pp. 11615–11639, 2025.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, et al. REFINER: reasoning feedback on intermediate representations. In *Proceedings of EACL 2024*, pp. 1100–1126, 2024.

Du Phan, Matthew Douglas Hoffman, David Dohan, Sholto Douglas, Tuan Anh Le, et al. Training chain-of-thought via latent-variable inference. In *Proceedings of NeurIPS 2023*, 2023.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *Proceedings of NeurIPS 2024*, 2024.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

Ankur Samanta, Akshayaa Magesh, Ayush Jain, Kavosh Asadi, Youliang Yu, et al. Structure enables effective self-localization of errors in LLMs. In *Proceedings of ICML 2026*, 2026.

Maohao Shen, Guangtao Zeng, Zhenting Qi, Zhang-Wei Hong, Zhenfang Chen, et al. Satori: Reinforcement learning with chain-of-action-thought enhances LLM reasoning via autoregressive search. In *Proceedings of ICML 2025*, volume 267, 2025.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. *CoRR*, abs/2601.19897, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of NeurIPS 2023*, 2023.

Xiaoshuai Song, Yanan Wu, Weixun Wang, Jiaheng Liu, Wenbo Su, et al. ProgCo: Program helps self-correction of large language models. In *Proceedings of ACL 2025*, pp. 944–959, 2025.

Qiaoyu Tang, Hao Xiang, Le Yu, Bowen Yu, Hongyu Lin, et al. RefCritic: Training long chain-of-thought critic models with refinement feedback. In *Proceedings of COLM 2026*, 2026.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In *Findings of ACL 2024*, pp. 13894–13908, 2024.

Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.*, 16:2023–2049, 2015.

Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, et al. Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency. In *Findings of EMNLP 2025*, pp. 7459–7482, 2025a.

Pinzheng Wang, Juntao Li, Zecheng Tang, Haijia Gui, and Min Zhang. Improving rationality in the reasoning process of language models through self-playing game. In *Proceedings of ICML 2025*, volume 267, pp. 63417–63438, 2025b.

Yizhou Wang, Lingzhi Zhang, Yue Bai, Mang Tik Chiu, Zhengmian Hu, et al. Cautious next token prediction. In *Findings of ACL 2025*, pp. 25685–25697, 2025c.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, et al. Thoughts are all over the place: On the underthinking of long reasoning models. In *Proceedings of NeurIPS 2025*, 2025d.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, et al. Generating sequences by learning to self-correct. In *Proceedings of ICLR 2023*, 2023.

Zhiheng Xi, Jixuan Huang, Xin Guo, Boyang Hong, Dingwen Yang, et al. Critique-RL: Training language models for critiquing through two-stage reinforcement learning. In *Proceedings of ICLR 2026*, 2026.

Zhihui Xie, Jie Chen, Liyu Chen, Weichao Mao, Jingjing Xu, et al. Teaching language models to critique via reinforcement learning. In *Proceedings of ICML 2025*, volume 267, 2025.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, et al. Self-rewarding correction for mathematical reasoning. *CoRR*, abs/2502.19613, 2025.

Yuchen Yan, Jin Jiang, Yang Liu, Yixin Cao, Xin Xu, et al. S^3cMath: Spontaneous step-level self-correction makes large language models better mathematical reasoners. In *Proceedings of AAAI 2025*, pp. 25588–25596, 2025.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E. Gonzalez, et al. SuperCorrect: Advancing small LLM reasoning with thought template distillation and self-correction. In *Proceedings of ICLR 2025*, 2025a.

Wenkai Yang, Jingwen Chen, Yankai Lin, and Ji-Rong Wen. DeepCritic: Deliberate critique with large language models. *CoRR*, abs/2505.00662, 2025b.

Zhe Yang, Yichang Zhang, Yudong Wang, Ziyao Xu, Junyang Lin, et al. Confidence v.s. critique: A decomposition of self-correction capability for LLMs. In *Proceedings of ACL 2025*, pp. 3998–4014, 2025c.

Tianshu Yu, Chao Xiang, Mingchuan Yang, Pei Ke, Bosi Wen, et al. Training language model to critique for better refinement. In *Findings of ACL 2025*, pp. 26760–26804, 2025.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. Agent-R: Training language model agents to reflect via iterative self-training. *CoRR*, abs/2501.11425, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Proceedings of NeurIPS 2022*, 2022.

Qingjie Zhang, Di Wang, Haoting Qian, Yiming Li, Tianwei Zhang, et al. Understanding the dark side of LLMs' intrinsic self-correction. In *Proceedings of ACL 2025*, pp. 27066–27101, 2025.

Shenao Zhang, Yaqing Wang, Yinxiao Liu, Tianqi Liu, Peter Grabowski, et al. Beyond Markovian: Reflective exploration via Bayes-adaptive RL for LLM reasoning. In *Proceedings of ICLR 2026*, 2026.

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

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

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

Xutong Zhao, Tengyu Xu, Xuewei Wang, Zhengxing Chen, Di Jin, et al. Boosting LLM reasoning via spontaneous self-correction. In *Proceedings of COLM 2025*, 2025.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, et al. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of ACL 2025*, pp. 1009–1024, 2025.

Xiangxin Zhou, Zichen Liu, Haonan Wang, Chao Du, Min Lin, et al. Variational reasoning for language models. In *Proceedings of ICLR 2026*, 2026.

A Related Work

A.1 Post-Hoc and Spontaneous Self-Correction

Self-correction aims to improve an LLM's responses by identifying and repairing errors in its own outputs (Pan et al., 2024; Kamoi et al., 2024). In reasoning tasks, self-correction can take two forms: *post-hoc self-correction*, which revises a completed initial response through one or more subsequent generations, and *spontaneous self-correction*, which integrates verification and revision within the same generation.

Post-hoc self-correction. Post-hoc correction can be elicited by prompting a model to reconsider and revise its initial response (Kim et al., 2023). Some approaches guide revision with feedback from task environments, execution results, external tools or knowledge sources, or other models (Shinn et al., 2023; Chen et al., 2024; Gou et al., 2024; Olausson et al., 2024). Others rely on the model's own feedback, either eliciting suggestions for improving the response (Madaan et al., 2023) or checking it with verification programs generated and simulated by the model (Song et al., 2025). However, when no informative external feedback is available, evidence for the effectiveness of such *intrinsic* correction in conventionally prompted models remains mixed. Some studies report improvements with carefully designed prompts, confidence estimation, or decoding settings (Liu et al., 2024; Li et al., 2024a), whereas other studies find that models may overlook their errors or replace initially correct answers with incorrect ones (Huang et al., 2024; Li et al., 2024b; Kamoi et al., 2024; Zhang et al., 2025). These results indicate that eliciting a revision does not by itself establish that the revision is warranted or useful.

Beyond prompting, explicit training seeks to equip the model with an ability to revise earlier responses (Welleck et al., 2023). Direct correction methods train models to improve successive attempts through multi-turn fine-tuning or RL (Qu et al., 2024; Kumar et al., 2025). Although these methods can learn effective revisions, they do not explicitly model a separate critique stage, leaving what needs to change to be inferred implicitly during revision. Feedback-guided approaches instead make diagnosis explicit by learning to assess the initial response and provide guidance for revision (Paul et al., 2024; Yang et al., 2025c). Such feedback may take the form of self-verification, critique, or reflection (Xiong et al., 2025; Jiang et al., 2025; Yang et al., 2025a; Chen et al., 2026a). One form of this guidance is error localization, which identifies where revision should begin. GLoRe uses global and step-level outcome reward models to determine whether a response should be revised and, if so, where revision should begin (Havrilla et al., 2024). Related inference-time methods select a revision point based on structured reasoning steps (Samanta et al., 2026) or changes in token probabilities induced by environmental feedback (Chitale et al., 2026), and then regenerate the suffix from the retained prefix.

Feedback-guided refinement depends on both the accuracy of the diagnosis and how effectively the refiner uses it. For diagnosis, Tyen et al. (2024) show that prompted models can struggle to locate reasoning errors even when they can repair them once the error location is provided. ProcessBench assesses error localization separately from repair by asking models to identify the first erroneous step or recognize an error-free solution (Zheng et al., 2025). The response to feedback is itself a training target in the Critic-Discernment Game, where a prover interacts with helpful and misleading critics and is rewarded for successful correction and resistance to misleading feedback (Wang et al., 2025b).

Spontaneous self-correction. Correction can also occur within a single ongoing generation, as a model checks earlier reasoning and decides whether to continue or revise its approach. Such behavior can emerge through reasoning-oriented RL (Guo et al., 2025). It can also be encouraged explicitly by training on trajectories that contain mistakes followed by successful recovery, exposing models to how reasoning can recover after an error (Pan et al., 2025; Yan et al., 2025; Kim et al., 2025). Verification can further guide whether and how reasoning should be revised: S^2R jointly trains verification and correction (Ma et al., 2025), while ReVISE and SPOC incorporate verification into the ongoing generation to guide subsequent refinement (Lee et al., 2025; Zhao et al., 2025). Satori guides subsequent reasoning through meta-actions for continuing, checking, and exploring (Shen et al., 2025), while BARL treats reflection as information-gathering exploration (Zhang et al., 2026). Agent-R extends recovery training to agentic interaction, learning to correct erroneous actions across action-observation turns within an episode (Yuan et al., 2025).

Alongside these training approaches, inference-time controls can also shape reflective behavior within a reasoning trajectory. Budget forcing (Muennighoff et al., 2025), strategy switching (Wang et al., 2025d), and reflection-token scheduling (Fan et al., 2026) control the amount or timing of reasoning and reflection. Uncertainty estimates and external verifiers can guide local exploration and refinement during generation (Wang et al., 2025c; Chang et al., 2025). However, providing more opportunities for reflection or exploration does not necessarily lead to effective error correction. Long-reasoning models can overthink, producing redundant reconsideration or closely related solution attempts with limited gains in accuracy (Chen et al., 2025). Analyses of reflective behavior further show that suppressing common reflection cues can shorten reasoning without degrading performance (Wang et al., 2025a), while many self-verification steps merely reconfirm existing conclusions rather than detect or repair errors (Long et al., 2026). Accordingly, the presence or frequency of reflection alone provides limited evidence of effective self-correction.

Across both forms of self-correction, a central challenge is to diagnose whether and where the reasoning fails and use that diagnosis to guide an effective revision without disrupting valid reasoning. We focus on feedback-guided post-hoc self-correction, where diagnosis is represented explicitly as a critique of a completed response. This setting allows us to model diagnosis and refinement as coupled components of correction, with critiques providing first-error localization and actionable repair guidance whose utility is evaluated through the refinements they induce.

A.2 Critique Models for LLMs

Critique models assess an LLM's response and generate natural-language feedback, such as correctness judgments, error explanations, and revision suggestions (Paul et al., 2024; Tang et al., 2026). REFINER learns to generate structured feedback from supervised examples that identify error types and may additionally provide error locations or repair hints (Paul et al., 2024). Diagnostic capability can also be optimized directly. DeepCritic follows supervised initialization with RL that rewards correct first-error localization or recognition of an error-free response (Yang et al., 2025b). However, improvements in diagnostic accuracy do not necessarily translate into more effective refinement. In RefCritic's preliminary experiments, supervised fine-tuning improved judgment accuracy without consistently improving refinement performance (Tang et al., 2026).

Training critiques through the refinements they induce connects feedback generation to downstream correction utility. CTRL rewards critiques according to the correctness of revisions produced by a fixed generator (Xie et al., 2025), while RCO uses preferences between refined and initial responses to optimize a direct reward objective (Yu et al., 2025). In interactive agent settings, ECHO learns from refinement gains while jointly updating the critic and the agent policy (Li et al., 2026). Outcome-based objectives can also be combined with explicit diagnostic supervision: RefCritic combines solution-level judgment and refinement rewards (Tang et al., 2026), while Critique-RL first trains correctness judgments and then optimizes refinement performance while preserving judgment accuracy (Xi et al., 2026). CTRL, RefCritic, and Critique-RL pair a dedicated critic with a separate generator; in contrast, VaCrit uses a single model for initial generation, diagnosis, and refinement at inference.

A.3 Variational Learning with Privileged Information

Learning with privileged information exploits additional evidence available during training but unavailable at inference (Vapnik & Izmailov, 2015). For language-model reasoning, reference answers or solutions can provide such privileged guidance for learning intermediate reasoning. STaR uses correct answers as hints to generate rationales for previously unsolved problems, then trains on successful rationales after removing the hints (Zelikman et al., 2022). Cetin et al. (2025) similarly condition on questions and solutions, but optimize the explanations based on how effectively they help a student recover the correct answer. Latent-variable approaches instead treat intermediate reasoning traces as latent variables and learn from answer supervision by marginalizing over them. Prior work approximates inference over these traces through Markov-chain Monte Carlo expectation-maximization (Phan et al., 2023) or amortized posterior sampling with GFlowNets (Hu et al., 2024). RAVR uses a reference-answer-conditioned posterior to propose reasoning traces and regularizes it toward a question-conditioned prior through an evidence lower bound (Lin et al., 2026). Variational Reasoning derives lower bounds on the total probability of producing a correct answer, extending

the objective to multiple traces and learning a posterior conditioned on answer hints (Zhou et al., 2026). These formulations use reference information to guide the exploration of latent reasoning traces during training while retaining reference-free generation at inference.

Existing variational approaches primarily use latent variables to model reasoning traces for answer generation or intermediate veracity labels for verification (Zhou et al., 2026; Kim et al., 2026). In contrast, we model a structured critique as the latent diagnosis that connects a completed response to its refinement, jointly capturing first-error localization and actionable repair guidance. The reference solution is used only to inform the critique posterior during training, while refinement correctness provides supervision without requiring a tailored corrected response; KL regularization toward an oracle-free critique prior enables diagnosis without reference access at inference.

A.4 Knowledge Distillation and Behavior Preservation

Knowledge distillation transfers learned behavior by training a student to match the predictive distribution of a teacher (Hinton et al., 2015). Generalized Knowledge Distillation further trains students on their own generated sequences using teacher feedback, reducing the mismatch between sequences encountered during training and inference (Agarwal et al., 2024). Distillation can also preserve previously learned behavior during adaptation. Li & Hoiem (2016) combine new-task supervision with distillation from the original model's predictions for earlier tasks on the same inputs, encouraging retention as shared parameters adapt. Shenfeld et al. (2026) use a demonstration-conditioned version of the model as a teacher on student-generated trajectories and report reduced forgetting during sequential adaptation. In this work, we utilize self-distillation to preserve the learned diagnostic capability as the same model is optimized for refinement, mitigating diagnostic forgetting and enabling a single checkpoint to perform both diagnosis and refinement at test time.

B Prompts and Critique Interface

B.1 Prompt Templates

System and user messages are grouped by component; braces denote placeholders.

Prior

System

You are a math teacher reviewing a student's step-by-step solution.

Review the steps in order and identify the first mathematically or logically incorrect step. Once you identify the first error, stop reviewing later steps. Focus your feedback on the student's work. Do not provide a complete corrected solution or reveal the final answer.

Your feedback must:
1. Identify the specific error.
2. Briefly explain why it is incorrect.
3. Provide an actionable direction for correcting that step.

Output exactly:
First Error Step: $\boxed{<\text{index}>}$
Feedback: <feedback>

If all steps are correct, output exactly:
First Error Step: $\boxed{-1}$
Feedback: No refinement is needed.

User

[Math Problem]
{problem}

[Student Solution]
{initial_response}

Posterior

System

You are a math teacher reviewing a student's step-by-step solution.

A verified reference solution is provided to you as a private reference. Use it to help assess the mathematical correctness of the student's solution and locate the first error. Do not require the student's reasoning to match the reference solution; accept any mathematically valid alternative approach. Do not mention, quote, copy, or otherwise refer to the reference solution in your output.

Review the steps in order and identify the first mathematically or logically incorrect step. Once you identify the first error, stop reviewing later steps. Focus your feedback on the student's work. Do not provide a complete corrected solution or reveal the final answer.

Your feedback must:
1. Identify the specific error.
2. Briefly explain why it is incorrect.
3. Provide an actionable direction for correcting that step.

Output exactly:
First Error Step: $\boxed{\text{<index>}}$
Feedback: <feedback>

If all steps are correct, output exactly:
First Error Step: $\boxed{-1}$
Feedback: No refinement is needed.

User

[Math Problem]
{problem}

[Student Solution to Evaluate]
{initial_response}

[Private Reference Solution — Do not mention, quote, or copy it in your output]
{reference_solution}

Refiner

System

Please reason step by step, and put your final answer within $\square$.

User

Here is the feedback on your initial solution:
{feedback}

Instructions:

- Do not repeat, summarize, or reference the feedback. Output the corrected solution only.
- Copy all steps before the first erroneous step from the **original solution** word for word.
- If no error was found (-1), copy the entire original solution word for word.
- Rewrite from scratch starting at the erroneous step. You may add or remove steps as needed.
- Always end with $\boxed{\text{your_answer}}$, regardless of what the original solution contains.

Output the full corrected solution and put your final answer within $\square$.

B.2 Step Representation and Feedback Extraction

Step representation. Localization warm-up uses XML-formatted PRM800K trajectories with explicit step tags such as <step_1>...</step_1>. The tagged steps are indexed from 1, and a critique index of -1 denotes that no error is found. In subsequent training and self-correction inference, we insert the initial response y_1 directly into the corresponding critique prompt, without additional step segmentation or renumbering.

Feedback extraction. To exclude the diagnostic reasoning trace, we extract the text after the final `</think>` delimiter using the function below. The returned string is inserted directly into the `{feedback}` placeholder of the refiner prompt, without field-level parsing or index validation.

```
def preprocess_decoder_feedback(feedback):
    if feedback is None or '</think>' not in feedback:
        return 'No feedback provided.'
    return feedback.split('</think>')[-1].strip()
```

We retain the problem x and initial response y_1 as the user and assistant messages, respectively, and append the extracted feedback and refinement instructions as a new user message.

C Derivations and Optimization Details

This section derives ELBO, specifies the prior-training returns, and presents the policy gradients for joint variational learning.

C.1 Derivation of the Correct-Event ELBO

For a fixed training instance (x, y_1, y^*), recall from Eq. (4) that

$$p_\theta(A_x|x, y_1) = \sum_z p_\theta(z|x, y_1)\, p_\theta(A_x|x, y_1, z). \tag{11}$$

Assume that $p_\theta(z|x, y_1)$ and $q_\phi(z|x, y_1, y^*)$ have common support and that the expectations below are finite. Rewriting the marginal likelihood under q_ϕ gives

$$\begin{aligned}
\log p_\theta(A_x|x, y_1) &= \log \mathbb{E}_{z\sim q_\phi(\cdot|x,y_1,y^*)}\left[\frac{p_\theta(z|x, y_1)\, p_\theta(A_x|x, y_1, z)}{q_\phi(z|x, y_1, y^*)}\right] \\
&\geq \mathbb{E}_{z\sim q_\phi(\cdot|x,y_1,y^*)}\left[\log \frac{p_\theta(z|x, y_1)\, p_\theta(A_x|x, y_1, z)}{q_\phi(z|x, y_1, y^*)}\right] \\
&= \mathbb{E}_{z\sim q_\phi(\cdot|x,y_1,y^*)}\left[\log p_\theta(A_x|x, y_1, z)\right] - D_{\mathrm{KL}}(q_\phi(\cdot|x, y_1, y^*)\|p_\theta(\cdot|x, y_1))\,.
\end{aligned} \tag{12}$$

The inequality follows directly from Jensen's inequality. Averaging this bound over $(x, y_1, y^*) \sim \mathcal{D}$ yields a lower bound on the population objective in Eq. (1).

C.2 Prior-Training Rewards and Credit Assignment

Sampling and training returns. Both critic-training stages use the same structured critique. For each training instance, we independently sample G critiques $z_g \sim p_\psi(\cdot|x, y_1)$, $g = 1, \ldots, G$. For error localization (Eq. (7)), each critique receives the return

$$R_g = \mathbf{1}\{z_{g,\mathrm{idx}} = e^*\}, \tag{13}$$

which evaluates whether the predicted first-error index matches the annotation. For meta-utility learning (Eq. (8)), the frozen base model θ_{init} serves as the refiner. For each critique z_g, we sample one critique-guided refinement, together with a single no-feedback refinement shared across all G critiques:

$$\begin{aligned}
y^z_{2,g} &\sim \theta_{\mathrm{init}}(\cdot|x, y_1, c(z_g)), \\
y^\varnothing_2 &\sim \theta_{\mathrm{init}}(\cdot|x, y_1, c_\varnothing),
\end{aligned} \tag{14}$$

where $c_\varnothing$ denotes no critique feedback under the same refinement template. The return for critique z_g is its induced gain in refinement correctness:

$$R_g = \Delta r_g = \mathbf{1}\{y^z_{2,g} \in A_x\} - \mathbf{1}\{y^\varnothing_2 \in A_x\}. \tag{15}$$

Leave-one-out credit assignment. Both critic-training stages optimize p_ψ with RLOO using their respective returns. For $G \geq 2$, the leave-one-out advantage for critique z_g is

$$A_g = R_g - \frac{1}{G-1} \sum_{h \neq g} R_h, \tag{16}$$

yielding the policy-gradient estimator

$$\tilde{g}_\psi = \frac{1}{G} \sum_{g=1}^{G} A_g \, \nabla_\psi \log p_\psi(z_g | x, y_1). \tag{17}$$

Each sampled critique is credited relative to the returns of the other critiques generated for the same training instance, providing an instance-specific baseline without requiring a learned value function.

C.3 Policy Gradients for Joint Variational Training

For a fixed instance (x, y_1, y^*), we derive the gradients of the per-instance objective in Eq. (9), with the critic p_ψ held fixed. We assume the regularity conditions required to interchange differentiation and summation.

Posterior gradient. Holding p_θ fixed, define the posterior return

$$u_{\theta,\phi}(z) = \log s_\theta(x, y_1, z) - \beta_1 \log \frac{q_\phi(z|x, y_1, y^*)}{p_\psi(z|x, y_1)}, \tag{18}$$

where $s_\theta(x, y_1, z) = p_\theta(A_x | x, y_1, c(z))$. Differentiating the correction and posterior–prior regularization terms with respect to ϕ yields

$$\begin{aligned} \nabla_\phi \mathcal{J}_{\text{uni}} &= \sum_z u_{\theta,\phi}(z) \, \nabla_\phi q_\phi(z|x, y_1, y^*) - \beta_1 \sum_z \nabla_\phi q_\phi(z|x, y_1, y^*) \\ &= \mathbb{E}_{z \sim q_\phi(\cdot|x, y_1, y^*)} \left[u_{\theta,\phi}(z) \, \nabla_\phi \log q_\phi(z|x, y_1, y^*) \right]. \end{aligned} \tag{19}$$

The second term vanishes because $\sum_z \nabla_\phi q_\phi(z|x, y_1, y^*) = \nabla_\phi 1 = 0$. The self-distillation term is independent of ϕ and thus does not contribute to the posterior gradient.

Refiner gradient. Holding the posterior fixed, consider a critique z. Since the correctness indicator $\mathbf{1}\{y_2 \in A_x\}$ is independent of θ, we have

$$\begin{aligned} \nabla_\theta \log s_\theta(x, y_1, z) &= \sum_{y_2} \mathbf{1}\{y_2 \in A_x\} \, \nabla_\theta p_\theta(y_2 | x, y_1, c(z)) / s_\theta(x, y_1, z) \\ &= \mathbb{E}_{y_2 \sim p_\theta(\cdot|x, y_1, c(z))} \left[\mathbf{1}\{y_2 \in A_x\} \, \nabla_\theta \log p_\theta(y_2 | x, y_1, c(z)) \right] / s_\theta(x, y_1, z). \end{aligned} \tag{20}$$

Thus, the correction gradient increases the likelihood of verifier-accepted refinements, scaled by the inverse of the current correction success probability. Consequently, the same absolute increase in $s_\theta(x, y_1, z)$ yields a larger improvement in the log objective when the current success probability is lower.

The posterior-prior KL is independent of θ. With p_ψ frozen, expanding the self-distillation term yields the full gradient with respect to θ:

$$\nabla_\theta \mathcal{J}_{\text{uni}} = \mathbb{E}_{z \sim q_\phi(\cdot|x, y_1, y^*)} \left[\nabla_\theta \log s_\theta(x, y_1, z) \right] + \beta_2 \mathbb{E}_{z \sim p_\psi(\cdot|x, y_1)} \left[\nabla_\theta \log p_\theta(z | x, y_1) \right]. \tag{21}$$

The first term optimizes critique-guided refinement, while the second distills the oracle-free diagnostic behavior of the frozen critic into p_θ. Since both terms update the same parameters θ, the resulting model jointly acquires refinement capability while retaining diagnosis. Finite-sample estimation is described in Appendix C.4.

C.4 Finite-Sample Estimation for Joint Training

For each training instance, we sample K posterior critiques $z_k \sim q_\phi(\cdot|x, y_1, y^*)$. For each critique z_k, we independently sample M refinements $y_{2,k,m} \sim p_\theta(\cdot|x, y_1, c(z_k))$ and estimate its correction success probability as

$$\tilde{s}_k \doteq \tilde{s}_\theta(x, y_1, z_k) = \frac{1}{M} \sum_{m=1}^{M} \mathbf{1}\{y_{2,k,m} \in A_x\}. \tag{22}$$

For the posterior update, we replace $\log s_\theta(x, y_1, z_k)$ in Eq. (18) with $\log(\tilde{s}_k + \epsilon)$. For the refinement update, we estimate Eq. (20) using the sampled refinements and replace its denominator by $\tilde{s}_k + \epsilon$, with $\epsilon > 0$ for numerical stability. KL normalization and training hyperparameters are provided in Appendix D.2.

D Experimental Details

All experiments are conducted on three NVIDIA A800 GPUs, each with 80 GB of memory.

D.1 Training Data and Preprocessing

Error-localization data. We construct $\mathcal{D}_{\text{loc}}$ from PRM800K (Lightman et al., 2024) for the error-localization stage in Section 3.2. Each example consists of a problem x, a stepwise response y_1, and a localization label e^* indicating the first erroneous step, with $e^* = -1$ denoting an error-free response. To balance the label distribution, we randomly retain 500 examples for each localization label. Responses use the XML step representation described in Appendix B.2, with step indices starting from 1.

Initial-response generation. For meta-utility learning and joint variational training, we construct $\mathcal{D}$ from DeepMath-103K (He et al., 2026). We retain problems with annotated difficulty 5-9 and sample 3,000 problems at each difficulty level. For each sampled problem x, the base model Qwen3-4B generates four initial responses y_1 using temperature 0.7 and setting `max_tokens` as 8k. We evaluate each response by final-answer correctness and partition the problems into three groups: all four responses are correct, both correct and incorrect responses are present, or all four responses are incorrect. Based on this partition, we sample 10,000 training instances, evenly split between correct and incorrect initial responses.

D.2 Training and Evaluation Configuration

We train Qwen3-4B and Qwen3-8B with LoRA (Hu et al., 2022), using the training budgets and hyperparameters in Table 4. Throughout training and inference, thinking mode is enabled for diagnosis and disabled for initial-response generation and refinement. Both KL terms are computed over all generated critique tokens, including thinking tokens. For each base model, **STaR+**, **SCoRe**, **PAG**, and **CURE** are trained on the same training data as **VaCrit**. **Prompt** uses the frozen base model and the MATH self-correction instruction from Kumar et al. (2025) to perform a single revision without external correctness feedback. For **SuperCorrect**[2] and **S^{2}R-ORL**[3], we evaluate the released Qwen2.5-Math-7B checkpoints.

For the seven mathematical reasoning benchmarks, we use the versions distributed in the Math-AI Evals collection[4]. At evaluation time, **VaCrit** uses a sampling temperature of 0.7 and a maximum generation length of 8k tokens for all three stages: initial-response generation, diagnosis, and refinement. We sample 32 initial responses per problem for AMC23 and AIME24–26, and one initial response per problem for MATH-500, Minerva Math, and OlympiadBench.

[2] `https://huggingface.co/BitStarWalkin/SuperCorrect-7B`
[3] `https://huggingface.co/S2R-data/Qwen2.5-Math-7B-S2R-ORL`
[4] `https://huggingface.co/collections/math-ai/evals`.

Table 4: Training configurations for Qwen3-4B and Qwen3-8B. LoRA settings, batch size, and learning rate are shared across all three training stages. G denotes the number of critiques sampled per instance during error-localization and meta-utility learning, while K and M denote the numbers of posterior critiques per instance and refinements per critique during joint training, respectively. Temperature and generation-length settings apply to joint training.

Setting	Value
LoRA rank r	128
LoRA scaling parameter α	128
Batch size	16
Learning rate	3×10^{-6}
Error-localization budget (steps)	200
Critic checkpoint after meta-utility learning (step)	200
Maximum joint-training steps	300
Reported joint-training checkpoint (step)	200
Critiques per instance for critic training G	6
Posterior critiques per instance K	3
Refinements per critique M	3
Sampling temperature	1.0
Maximum posterior generation length	8k tokens
Maximum refiner generation length	4k tokens
Posterior–prior KL coefficient β_1	0.1
Diagnostic-retention KL coefficient β_2	0.1
Numerical-stability constant ϵ	10^{-8}

D.3 Evaluation Metrics

Response correctness and transitions. We use Math-Verify[5] to assess final-answer correctness for both initial and refined responses. For a given benchmark, let n_{uv} denote the number of evaluated response pairs with initial correctness u and final correctness v, where $u, v \in \{0, 1\}$ and 1 denotes correctness. Each sampled trajectory contributes one pair, including multiple samples from the same problem, and $N = \sum_{u,v\in\{0,1\}} n_{uv}$ denotes the total number of evaluated pairs. We define initial and final accuracy, incorrect-to-correct (I2C) and correct-to-incorrect (C2I) transition rates, and the conditional Critique Score (CS) and Confidence Level (CL), all in percent, as

$$\begin{aligned} \mathrm{Acc}_{t_1} &= \frac{n_{10}+n_{11}}{N} \times 100, & \mathrm{Acc}_{\mathrm{final}} &= \frac{n_{01}+n_{11}}{N} \times 100, \\ \mathrm{I2C} &= \frac{n_{01}}{N} \times 100, & \mathrm{C2I} &= \frac{n_{10}}{N} \times 100, \\ \mathrm{CS} &= \frac{n_{01}}{n_{00}+n_{01}} \times 100, & \mathrm{CL} &= \frac{n_{11}}{n_{10}+n_{11}} \times 100. \end{aligned} \tag{23}$$

CS and CL are defined when the initially incorrect and initially correct subsets are nonempty, respectively. The self-correction gain, measured in percentage points, satisfies

$$\Delta = \mathrm{Acc}_{\mathrm{final}} - \mathrm{Acc}_{t_1} = \mathrm{I2C} - \mathrm{C2I}, \tag{24}$$

with the equality holding exactly before numerical rounding.

Relative self-correction score. Following Yang et al. (2025c), we report CS, CL, and the Relative Self-Correction Score (RSS). Let $p_1 = \mathrm{Acc}_{t_1}/100$ and $p_f = \mathrm{Acc}_{\mathrm{final}}/100$. Then

$$\mathrm{RSS} = \frac{p_f - p_1^2}{2p_1(1-p_1)} \times 100 = 50 + \frac{\mathrm{CS}}{2p_1} - \frac{100-\mathrm{CL}}{2(1-p_1)}, \qquad 0 < p_1 < 1. \tag{25}$$

The second equality follows from $p_f = [(1-p_1)\mathrm{CS} + p_1\mathrm{CL}]/100$, where CS and CL are expressed as percentages.

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

Table 5: **Error localization across step formats.** We report ProcessBench F1 (%) for the Qwen3-4B critique prior before and after meta-utility learning, following XML-only localization warm-up on PRM800K. Avg. is the unweighted mean across four subsets. Bold marks the higher mean F1 within each format.

Format	Checkpoint	GSM8K	MATH	Olympiad	OmniMATH	Avg.
XML	Before meta	85.49	84.15	77.84	74.01	80.37
	After meta	84.89	84.23	79.58	74.45	**80.79**
Markdown	Before meta	83.97	82.21	78.27	72.73	79.30
	After meta	85.99	82.74	78.12	73.14	**80.00**

ProcessBench F1. For each ProcessBench subset, let e_i and $\hat{e}_i$ denote the annotated and predicted first-error indices for trajectory i, respectively, with -1 indicating an error-free trajectory. Define the erroneous and error-free subsets as $\mathcal{E} = \{i : e_i \neq -1\}$ and $\mathcal{C} = \{i : e_i = -1\}$. For subsets containing both types of trajectories, we compute the exact first-error localization accuracy and no-error recognition accuracy as

$$a_{\text{err}} = \frac{1}{|\mathcal{E}|} \sum_{i \in \mathcal{E}} \mathbf{1}\{\hat{e}_i = e_i\}, \qquad a_{\text{free}} = \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} \mathbf{1}\{\hat{e}_i = -1\}. \tag{26}$$

Following Zheng et al. (2025), we report their harmonic mean in percent:

$$\text{F1} = \frac{2 a_{\text{err}} a_{\text{free}}}{a_{\text{err}} + a_{\text{free}}} \times 100, \qquad a_{\text{err}} + a_{\text{free}} > 0. \tag{27}$$

D.4 Cross-Format Generalization of Error Localization

Error-localization training uses only XML-formatted PRM800K trajectories (Lightman et al., 2024), whereas model-generated initial responses are unconstrained in format and often use Markdown-style step headings. We therefore evaluate whether localization transfers beyond the supervised XML format and how it changes after meta-utility learning. Specifically, we evaluate the Qwen3-4B critic immediately after error-localization training and again after subsequent meta-utility learning on XML and Markdown versions of the same ProcessBench trajectories (Zheng et al., 2025), keeping all evaluation settings fixed within each format.

After XML-only error-localization training, the critic achieves a mean F1 of 79.30% on Markdown, close to its 80.37% on XML (Table 5), indicating that its localization capability transfers beyond the format used for direct supervision. Meta-utility learning further increases mean F1 to 80.79% on XML and 80.00% on Markdown, reducing the cross-format gap from 1.07 to 0.79 percentage points. Although this stage optimizes critiques through downstream refinement outcomes rather than localization accuracy directly, the modest gains in F1 suggest that it also improves error localization.

D.5 Detailed Results by Benchmark

Tables 6 and 7 report complete benchmark-level results for the Qwen3-4B methods and the released Qwen2.5-Math-7B checkpoints, while Tables 8 and 9 report the corresponding results for Qwen3-8B.

D.6 Ablation Settings and Per-Benchmark Results

All ablations use Qwen3-4B, with complete benchmark-level results reported in Table 10.

Training initialization. For each variant, the critic p_ψ, posterior q_ϕ, and unified model p_θ are initialized from the same checkpoint: after both error localization and meta-utility learning for the full method, after error localization only for *w/o meta-utility*, and from the base model for *w/o critic*

Table 6: **Complete main results by benchmark (1/2).** Accuracy, I2C, C2I, CS, and CL are percentages; Gain is in percentage points. SuperCorrect and S^{2}R-ORL use Qwen2.5-Math-7B; all other methods use Qwen3-4B. Bold marks the best Qwen3-4B values for each metric within each block; lower C2I is better. When VaCrit ties for best, only VaCrit is bold.

		Accuracy		Transitions			Conditional		
Benchmark	Method	Initial ↑	Final ↑	Gain ↑	I2C ↑	C2I ↓	CS ↑	CL ↑	RSS ↑
MATH-500	Prompt	82.60	82.60	+0.00	3.00	3.00	17.24	96.37	50.00
	STaR+	85.20	84.40	−0.80	1.20	2.00	8.11	97.65	46.83
	SCoRe	85.60	86.40	+0.80	1.40	0.60	9.72	99.30	53.25
	PAG	85.00	85.60	+0.60	1.40	0.80	9.33	99.06	52.35
	CURE	86.40	87.20	+0.80	**5.80**	5.00	**42.65**	94.21	53.40
	SuperCorrect	60.00	62.80	+2.80	3.40	0.60	8.50	99.00	55.83
	S^{2}R-ORL	80.60	78.20	−2.40	3.80	6.20	19.59	92.31	42.33
	VaCrit	**86.40**	**90.00**	**+3.60**	4.20	**0.60**	30.88	**99.31**	**65.32**
AMC23	Prompt	66.80	67.81	+1.02	3.05	2.03	9.18	96.96	52.29
	STaR+	66.25	68.83	+2.58	4.30	1.72	12.73	97.41	55.77
	SCoRe	70.55	71.64	+1.09	3.20	2.11	10.88	97.01	52.63
	PAG	68.98	69.61	+0.62	1.33	**0.70**	4.28	**98.98**	51.46
	CURE	**71.80**	68.91	−2.89	**7.66**	10.55	**27.15**	85.31	42.86
	SuperCorrect	38.05	37.50	−0.55	2.66	3.20	4.29	91.58	48.84
	S^{2}R-ORL	62.66	60.16	−2.50	6.64	9.14	17.78	85.41	44.66
	VaCrit	69.61	**74.38**	**+4.77**	6.33	1.56	20.82	97.76	**61.26**
AIME24	Prompt	22.71	23.33	+0.62	2.19	1.56	2.83	93.12	51.78
	STaR+	22.40	24.58	+2.19	3.44	1.25	4.43	94.42	56.29
	SCoRe	24.90	26.35	+1.46	2.29	**0.83**	3.05	**96.65**	53.90
	PAG	22.40	23.33	+0.94	1.77	**0.83**	2.28	96.28	52.70
	CURE	**26.15**	26.67	+0.52	**6.88**	6.35	**9.31**	75.70	51.35
	SuperCorrect	6.25	6.15	−0.10	0.31	0.42	0.33	93.33	49.11
	S^{2}R-ORL	16.25	17.40	+1.15	3.96	2.81	4.73	82.69	54.21
	VaCrit	24.69	**27.50**	**+2.81**	3.96	1.15	5.26	95.36	**57.56**
AIME25	Prompt	19.17	21.15	+1.98	2.71	0.73	3.35	96.20	56.39
	STaR+	20.83	21.56	+0.73	2.29	1.56	2.89	92.50	52.21
	SCoRe	**22.81**	23.12	+0.31	1.15	0.83	1.48	96.35	50.89
	PAG	22.19	23.02	+0.83	1.04	**0.21**	1.34	**99.06**	52.41
	CURE	21.35	22.60	+1.25	**5.52**	4.27	**7.02**	80.00	53.72
	SuperCorrect	3.85	3.96	+0.10	0.31	0.21	0.33	94.59	51.41
	S^{2}R-ORL	13.85	11.88	−1.98	3.44	5.42	3.99	60.90	41.71
	VaCrit	22.50	**27.29**	**+4.79**	5.21	0.42	6.72	98.15	**63.74**

Table 7: **Complete main results by benchmark (2/2).** Accuracy, I2C, C2I, CS, and CL are percentages; Gain is in percentage points. SuperCorrect and S^{2}R-ORL use Qwen2.5-Math-7B; all other methods use Qwen3-4B. Bold marks the best Qwen3-4B values for each metric within each block; lower C2I is better. When VaCrit ties for best, only VaCrit is bold. Avg. is the unweighted mean of each benchmark-level metric.

		Accuracy		Transitions			Conditional		
Benchmark	Method	Initial ↑	Final ↑	Gain ↑	I2C ↑	C2I ↓	CS ↑	CL ↑	RSS ↑
AIME26	Prompt	16.67	18.44	+1.77	2.19	**0.42**	2.62	97.50	56.38
	STaR+	16.88	17.19	+0.31	1.56	1.25	1.88	92.59	51.11
	SCoRe	20.00	21.46	+1.46	1.88	**0.42**	2.34	**97.92**	54.56
	PAG	18.23	17.92	−0.31	0.62	0.94	0.76	94.86	48.95
	CURE	**20.10**	19.69	−0.42	**7.29**	7.71	**9.13**	61.66	48.70
	SuperCorrect	6.35	4.06	−2.29	0.21	2.50	0.22	60.66	30.74
	S^{2}R-ORL	10.42	11.25	+0.83	2.40	1.56	2.67	85.00	54.47
	VaCrit	18.75	**24.69**	**+5.94**	6.56	0.62	8.08	96.67	**69.49**
Minerva	Prompt	39.34	40.44	+1.10	2.57	1.47	4.24	96.26	52.31
	STaR+	38.60	38.97	+0.37	1.84	1.47	2.99	96.19	50.78
	SCoRe	**41.54**	**42.65**	+1.10	1.47	**0.37**	2.52	**99.12**	52.27
	PAG	40.81	40.44	−0.37	0.37	0.74	0.62	98.20	49.24
	CURE	39.71	40.44	+0.74	4.04	3.31	6.71	91.67	51.54
	SuperCorrect	19.85	19.12	−0.74	2.21	2.94	2.75	85.19	47.69
	S^{2}R-ORL	38.60	36.40	−2.21	1.47	3.68	2.40	90.48	45.35
	VaCrit	37.50	40.81	**+3.31**	**5.15**	1.84	**8.24**	95.10	**57.06**
Olympiad	Prompt	49.41	52.82	+3.41	4.60	1.19	9.09	97.60	56.83
	STaR+	52.23	52.82	+0.59	2.82	2.23	5.90	95.74	51.19
	SCoRe	53.86	55.79	+1.93	2.97	1.04	6.43	98.07	53.88
	PAG	52.37	53.12	+0.74	1.34	**0.59**	2.80	**98.87**	51.49
	CURE	**54.90**	54.45	−0.45	**7.12**	7.57	**15.79**	86.22	49.10
	SuperCorrect	28.34	29.38	+1.04	2.08	1.04	2.90	96.34	52.56
	S^{2}R-ORL	45.99	45.10	−0.89	3.71	4.60	6.87	90.00	48.21
	VaCrit	54.15	**59.05**	**+4.90**	5.64	0.74	12.30	98.63	**59.86**
Avg.	Prompt	42.39	43.80	+1.41	2.90	1.49	6.94	96.29	53.71
	STaR+	43.20	44.05	+0.85	2.49	1.64	5.56	95.21	52.03
	SCoRe	45.61	46.77	+1.16	2.05	0.89	5.20	97.77	53.05
	PAG	44.28	44.72	+0.44	1.12	**0.69**	3.06	**97.90**	51.23
	CURE	**45.77**	45.71	−0.06	**6.33**	6.39	**16.82**	82.11	50.10
	SuperCorrect	23.24	23.28	+0.04	1.60	1.56	2.76	88.67	48.03
	S^{2}R-ORL	38.34	37.20	−1.14	3.63	4.77	8.29	83.83	47.28
	VaCrit	44.80	**49.10**	**+4.30**	5.29	0.99	13.19	97.28	**62.04**

Table 8: **Complete Qwen3-8B results by benchmark (1/2).** All methods use Qwen3-8B. Accuracy, I2C, C2I, CS, and CL are percentages; Gain is in percentage points. Avg. reports unweighted benchmark means. Bold marks the best values within each block; lower C2I is better. When VaCrit ties for best, only VaCrit is bold. Rounding can cause small differences between gains and differences of displayed accuracies.

		Accuracy		Transitions			Conditional		
Benchmark	Method	Initial ↑	Final ↑	Gain ↑	I2C ↑	C2I ↓	CS ↑	CL ↑	RSS ↑
MATH-500	Prompt	84.00	84.80	+0.80	3.20	2.40	20.00	97.14	52.98
	STaR+	86.20	85.00	−1.20	3.00	4.20	21.74	95.13	44.96
	SCoRe	86.00	87.80	+1.80	3.20	1.40	22.86	98.37	57.48
	PAG	85.40	85.00	−0.40	0.20	**0.60**	1.37	**99.30**	48.40
	CURE	85.40	84.80	−0.60	**5.20**	5.80	**35.62**	93.21	47.59
	VaCrit	**88.00**	**90.80**	**+2.80**	3.60	0.80	30.00	99.09	**63.26**
AMC23	Prompt	69.84	69.92	+0.08	3.98	3.91	13.21	94.41	50.19
	STaR+	69.14	69.61	+0.47	3.44	2.97	11.14	95.71	51.10
	SCoRe	**72.97**	74.45	+1.48	3.75	2.27	13.87	96.90	53.76
	PAG	69.92	69.84	−0.08	0.55	**0.62**	1.82	**99.11**	49.81
	CURE	70.16	68.98	−1.17	**9.06**	10.23	**30.37**	85.41	47.20
	VaCrit	70.16	**74.69**	**+4.53**	6.17	1.64	20.68	97.66	**60.82**
AIME24	Prompt	26.77	28.75	+1.98	3.75	1.77	5.12	93.39	55.05
	STaR+	26.88	28.33	+1.46	2.29	**0.83**	3.13	**96.90**	53.71
	SCoRe	**30.63**	32.50	+1.87	2.92	1.04	4.20	96.60	54.41
	PAG	26.25	25.62	−0.63	0.52	1.15	0.71	95.63	48.39
	CURE	26.77	25.21	−1.56	**5.73**	7.29	**7.82**	72.76	46.01
	VaCrit	29.06	**33.12**	**+4.06**	5.52	1.46	7.78	94.98	**59.85**
AIME25	Prompt	21.46	21.67	+0.21	1.77	1.56	2.25	92.72	50.62
	STaR+	21.25	22.19	+0.94	1.77	0.83	2.25	96.08	52.80
	SCoRe	**23.96**	24.27	+0.31	1.15	0.83	1.51	96.52	50.86
	PAG	21.98	22.29	+0.31	0.52	**0.21**	0.67	**99.05**	50.91
	CURE	22.50	22.71	+0.21	**5.73**	5.52	**7.39**	75.46	50.60
	VaCrit	23.02	**26.56**	**+3.54**	3.96	0.42	5.14	98.19	**59.99**

Table 9: **Complete Qwen3-8B results by benchmark (2/2).** All methods use Qwen3-8B. Accuracy, I2C, C2I, CS, and CL are percentages; Gain is in percentage points. Avg. reports unweighted benchmark means. Bold marks the best values within each block; lower C2I is better. When VaCrit ties for best, only VaCrit is bold. Rounding can cause small differences between gains and differences of displayed accuracies.

		Accuracy		Transitions			Conditional		
Benchmark	Method	Initial ↑	Final ↑	Gain ↑	I2C ↑	C2I ↓	CS ↑	CL ↑	RSS ↑
AIME26	Prompt	16.98	18.96	+1.98	3.54	1.56	4.27	90.80	57.02
	STaR+	16.35	17.92	+1.56	3.23	1.67	3.86	89.81	55.71
	SCoRe	**19.38**	20.73	+1.35	2.92	1.56	3.62	91.94	54.33
	PAG	17.50	17.19	–0.31	0.21	**0.52**	0.25	**97.02**	48.92
	CURE	16.25	17.29	+1.04	5.52	4.48	6.59	72.44	53.83
	VaCrit	16.88	**23.96**	**+7.08**	**7.81**	0.73	**9.40**	95.68	**75.25**
Minerva	Prompt	41.91	39.71	–2.21	0.74	2.94	1.27	92.98	45.47
	STaR+	41.91	**43.01**	**+1.10**	2.21	1.10	3.80	97.37	**52.27**
	SCoRe	41.91	41.18	–0.74	0.74	1.47	1.27	96.49	48.49
	PAG	**42.65**	42.65	+0.00	0.37	**0.37**	0.64	**99.14**	50.00
	CURE	41.54	42.65	**+1.10**	**4.78**	3.68	**8.18**	91.15	**52.27**
	VaCrit	41.18	41.91	+0.74	3.31	2.57	5.62	93.75	51.52
Olympiad	Prompt	54.75	56.08	+1.34	3.86	2.52	8.52	95.39	52.69
	STaR+	54.90	55.19	+0.30	3.12	2.82	6.91	94.86	50.60
	SCoRe	**57.57**	59.05	+1.48	3.56	2.08	8.39	96.39	53.04
	PAG	54.45	54.45	+0.00	1.04	**1.04**	2.28	**98.09**	50.00
	CURE	54.90	55.04	+0.15	**8.16**	8.01	**18.09**	85.41	50.30
	VaCrit	56.23	**62.31**	**+6.08**	7.72	1.63	17.63	97.10	**62.36**
Avg.	Prompt	45.10	45.70	+0.60	2.98	2.38	7.81	93.83	52.00
	STaR+	45.23	45.89	+0.66	2.72	2.06	7.55	95.12	51.59
	SCoRe	**47.49**	48.57	+1.08	2.61	1.52	7.96	96.17	53.20
	PAG	45.45	45.29	–0.16	0.49	**0.64**	1.11	**98.19**	49.49
	CURE	45.36	45.24	–0.12	**6.31**	6.43	**16.29**	82.26	49.69
	VaCrit	46.36	**50.48**	**+4.12**	5.44	1.32	13.75	96.64	**61.86**

training. The critic p_ψ is then frozen, while q_ϕ and p_θ are trained using identical joint-training schedules and hyperparameters across variants.

Inference-time feedback. Feedback ablations use the checkpoint from the full method and reuse the same initial responses y_1 and generated diagnoses z. For *w/o error index*, the refiner receives only the repair suggestion; for *w/o suggestion*, it receives only the predicted error index. For *w/o both fields*, we replace the {`feedback`} placeholder with the "`No feedback is provided`" string. All variants exclude the diagnostic reasoning trace and keep the problem, initial-response history, system prompt, refinement instructions, and decoding settings unchanged.

Results. Compared with localization-only initialization, adding meta-utility learning reduces mean C2I from 1.39% to 0.99%, indicating fewer initially correct responses are degraded during refinement. Compared with removing both feedback fields, index-only feedback increases mean I2C from 0.36% to 0.64%, but also raises mean C2I from 0.38% to 0.70%, so the additional error repair is offset by newly introduced errors.

Table 10: **Complete ablation results by benchmark.** Bold marks the best values for each metric within each benchmark; lower C2I is better. When VaCrit (full) ties for best, only VaCrit (full) is bold. The training variants change initialization; the inference variants remove one or both feedback fields. Accuracy, transition rates, CS, and CL are percentages; gain is in percentage points. I2C and C2I use all responses as the denominator, while CS and CL condition on initial correctness.

		Accuracy		Transitions			Conditional		
Benchmark	Variant	Initial ↑	Final ↑	Gain ↑	I2C ↑	C2I ↓	CS ↑	CL ↑	RSS ↑
MATH-500	**VaCrit (full)**	**86.40**	**90.00**	**+3.60**	4.20	0.60	30.88	99.31	**65.32**
	w/o meta-utility	85.40	89.00	+3.60	**4.80**	1.20	**32.88**	98.59	64.44
	w/o critic training	85.60	89.00	+3.40	3.60	**0.20**	25.00	**99.77**	63.79
	w/o error index	86.40	89.80	+3.40	3.80	0.40	27.94	99.54	64.47
	w/o suggestion	86.40	86.80	+0.40	1.00	0.60	7.35	99.31	51.70
	w/o both fields	86.40	86.40	+0.00	0.20	**0.20**	1.47	**99.77**	50.00
AMC23	**VaCrit (full)**	**69.61**	74.38	+4.77	6.33	1.56	20.82	97.76	61.26
	w/o meta-utility	68.44	72.58	+4.14	6.33	2.19	20.05	96.80	59.58
	w/o critic training	67.66	73.44	**+5.78**	**7.11**	1.33	**21.98**	98.04	**63.21**
	w/o error index	69.61	**74.84**	+5.23	6.56	1.33	21.59	98.09	62.37
	w/o suggestion	69.61	70.31	+0.70	1.17	**0.47**	3.86	**99.33**	51.66
	w/o both fields	69.61	69.61	+0.00	0.62	0.62	2.06	99.10	50.00
AIME24	**VaCrit (full)**	**24.69**	**27.50**	**+2.81**	**3.96**	1.15	**5.26**	95.36	57.56
	w/o meta-utility	22.50	25.31	+2.81	3.65	0.83	4.70	96.30	**58.06**
	w/o critic training	23.12	25.31	+2.19	3.12	0.94	4.07	95.95	56.15
	w/o error index	24.69	26.77	+2.08	3.33	1.25	4.43	94.94	55.60
	w/o suggestion	24.69	24.79	+0.10	0.42	**0.31**	0.55	**98.73**	50.28
	w/o both fields	24.69	24.69	+0.00	0.31	**0.31**	0.41	**98.73**	50.00
AIME25	**VaCrit (full)**	**22.50**	**27.29**	**+4.79**	**5.21**	0.42	**6.72**	98.15	**63.74**
	w/o meta-utility	22.40	24.69	+2.29	2.60	0.31	3.36	98.60	56.59
	w/o critic training	22.08	24.58	+2.50	3.33	0.83	4.28	96.23	57.26
	w/o error index	22.50	26.56	+4.06	4.58	0.52	5.91	97.69	61.65
	w/o suggestion	22.50	22.71	+0.21	0.21	**0.00**	0.27	**100.00**	50.60
	w/o both fields	22.50	22.60	+0.10	0.21	0.10	0.27	99.54	50.30
AIME26	**VaCrit (full)**	**18.75**	24.69	+5.94	6.56	0.62	8.08	96.67	69.49
	w/o meta-utility	16.77	21.88	+5.10	5.73	0.62	6.88	96.27	68.28
	w/o critic training	16.88	20.62	+3.75	4.58	0.83	5.51	95.06	63.37
	w/o error index	18.75	**25.10**	**+6.35**	**7.08**	0.73	**8.72**	96.11	**70.85**
	w/o suggestion	18.75	18.96	+0.21	0.21	**0.00**	0.26	**100.00**	50.68
	w/o both fields	18.75	18.75	+0.00	0.21	0.21	0.26	98.89	50.00
Minerva	**VaCrit (full)**	37.50	40.81	+3.31	5.15	1.84	8.24	95.10	57.06
	w/o meta-utility	38.60	40.07	+1.47	5.15	3.68	8.38	90.48	53.10
	w/o critic training	**40.44**	40.81	+0.37	3.31	2.94	5.56	92.73	50.76
	w/o error index	37.50	**41.54**	**+4.04**	**5.51**	1.47	**8.82**	96.08	**58.63**
	w/o suggestion	37.50	35.29	–2.21	0.74	2.94	1.18	92.16	45.29
	w/o both fields	37.50	36.76	–0.74	0.37	**1.10**	0.59	**97.06**	48.43
Olympiad	**VaCrit (full)**	**54.15**	59.05	+4.90	5.64	0.74	12.30	98.63	59.86
	w/o meta-utility	54.15	**59.94**	**+5.79**	**6.68**	0.89	**14.56**	98.36	**61.65**
	w/o critic training	52.37	56.38	+4.01	4.90	0.89	10.28	98.30	58.03
	w/o error index	54.15	59.05	+4.90	5.79	0.89	12.62	98.36	59.86
	w/o suggestion	54.15	54.30	+0.15	0.74	0.59	1.62	98.90	50.30
	w/o both fields	54.15	54.60	+0.45	0.59	**0.15**	1.29	**99.73**	50.90

E Qualitative Case Study: From Diagnosis to Repair

Problem (MATH-500). The square root of t is greater than 2 and less than 3.5. How many integer values of t satisfy this condition? **Reference answer:** 8.

All four methods correctly derive $4 < t < 12.25$ in their initial responses. Their first error occurs in Step 2, where the integer list omits 12; the count of seven follows from this incomplete list.

SCoRe: revision without locating the omission. SCoRe announces a "potential misunderstanding" and revisits the squaring operation, but repeats the incomplete integer list. Its second attempt changes the exposition without identifying the missing 12, leaving the answer at 7.

SCoRe **Revise** $7 \to 7$ **Incorrect**

1. Initial solution

We are given that:

$$\sqrt{t} > 2 \quad \text{and} \quad \sqrt{t} < 3.5$$

Step 1: Square all parts of the inequality to eliminate the square root

$$2 < \sqrt{t} < 3.5$$

Now square all parts:

$$2^2 < t < (3.5)^2$$
$$4 < t < 12.25$$

Step 2: Find integer values of t that satisfy $4 < t < 12.25$

We are looking for **integer values of** t such that:

$$4 < t < 12.25$$

So t must be an integer strictly greater than 4 and strictly less than 12.25.

The integers in this range are:

$$5, 6, 7, 8, 9, 10, 11$$

Step 3: Count the number of integers

That's 7 integers.

Final Answer:

$$\boxed{7}$$

2. Second attempt

You're absolutely right — there **is** a potential misunderstanding in the original solution. Let's go through this step-by-step carefully and **correct** any issues.

Given:

$$\sqrt{t} > 2 \quad \text{and} \quad \sqrt{t} < 3.5$$

We are to find the **number of integer values of** t that satisfy this condition.

Step 1: Understand the inequality

We are given:

$$2 < \sqrt{t} < 3.5$$

SCoRe **Revise 7 → 7** **Incorrect (continued)**

This is **not** a strict inequality in terms of t, but in terms of $\sqrt{t}$. So we need to **square** the inequality to eliminate the square root.

$$(2)^2 < t < (3.5)^2$$
$$4 < t < 12.25$$

This is now a **strict inequality in terms of** t.

Step 2: Find integer values of t such that $4 < t < 12.25$

We are looking for **integers strictly greater than 4 and strictly less than 12.25**.

So the possible integer values of t are:

$$5, 6, 7, 8, 9, 10, 11$$

These are all the integers between 4 and 12.25, **not including** 4 and 12.25.

Step 3: Count the number of integers

$$5, 6, 7, 8, 9, 10, 11 \Rightarrow 7 \text{ values}$$

Final Answer:

$$\boxed{7}$$

PAG: false acceptance prevents revision. PAG endorses the incomplete integer list and concludes "The answer is correct." Under its selective-revision rule (Jiang et al., 2025), this judgment terminates the trajectory and retains the initial answer 7. The missed omission therefore prevents revision.

PAG **Retain 7** **Incorrect**

1. Initial solution

We are given that:

$$\sqrt{t} > 2 \quad \text{and} \quad \sqrt{t} < 3.5$$

Step 1: Square all parts of the inequality to eliminate the square root

$$2 < \sqrt{t} < 3.5$$

Now square all parts:

$$2^2 < t < (3.5)^2$$
$$4 < t < 12.25$$

Step 2: Find integer values of t that satisfy $4 < t < 12.25$

We are looking for **integer values of** t such that:

$$4 < t < 12.25$$

So t must be an integer strictly greater than 4 and strictly less than 12.25.

The integers in this range are:

$$5, 6, 7, 8, 9, 10, 11$$

PAG **Retain 7 Incorrect (continued)**

Step 3: Count the number of integers

$$\{5, 6, 7, 8, 9, 10, 11\}$$

There are **7** integers in this list.

✓ **Final Answer:**

$$\boxed{7}$$

2. Self-verification

Let's go through the solution step-by-step and verify it.

Step 1: Given inequality

$$\sqrt{t} > 2 \quad \text{and} \quad \sqrt{t} < 3.5$$

This is correct.

Step 2: Square all parts of the inequality to eliminate the square root

$$2 < \sqrt{t} < 3.5$$

Squaring all parts:

$$2^2 < t < (3.5)^2$$
$$4 < t < 12.25$$

This is correct. Squaring is a valid operation for inequalities when both sides are positive, and since $\sqrt{t}$ is positive, this is valid.

Step 3: Find integer values of t that satisfy $4 < t < 12.25$

We are looking for **integer values of** t such that:

$$4 < t < 12.25$$

So t must be an integer strictly greater than 4 and strictly less than 12.25.

The integers in this range are:

$$5, 6, 7, 8, 9, 10, 11$$

This is correct.

Step 4: Count the number of integers

$$\{5, 6, 7, 8, 9, 10, 11\}$$

There are **7** integers in this list.

This is correct.

✓ **Final Answer:**

The answer is correct.

Protocol decision (author annotation)

Accept the initial solution. The positive verification does not trigger revision; the returned answer is 7.

CURE: the no-error judgment stops re-exploration. CURE reports ERROR_FOUND: false and leaves ERROR_QUOTE empty. Under its conditional inference rule (Chen et al., 2026a), this judg-

ment retains the initial answer 7. The accompanying general hint does not activate re-exploration: the missed error stops the trajectory before any hint-guided correction.

CURE **Retain** 7 **Incorrect**

1. Initial solution

We are given that:

$$\sqrt{t} > 2 \quad \text{and} \quad \sqrt{t} < 3.5$$

Step 1: Eliminate the square root by squaring both sides

Since the square root function is increasing, squaring both sides of the inequalities preserves the direction of the inequalities.

$$2 < \sqrt{t} < 3.5$$

Now square all three parts:

$$2^2 < t < (3.5)^2$$
$$4 < t < 12.25$$

Step 2: Find integer values of t in this interval

We are looking for **integer values** of t such that:

$$4 < t < 12.25$$

This means t must be greater than 4 and less than 12.25.

The integers in this range are:

$$5, 6, 7, 8, 9, 10, 11$$

Step 3: Count the number of integers

There are 7 integers in this list: $5, 6, 7, 8, 9, 10, 11$

Final Answer:

$$\boxed{7}$$

2. Critique and verification

ERROR_FOUND: false

ERROR_QUOTE: ””

WHY_IT_MATTERS: The solution correctly follows through with the logic and calculations, ensuring that the final answer is accurate based on the given constraints.

HIGH_LEVEL_HINT: Always verify that the interval derived from inequalities includes all possible values that satisfy the original condition, especially when dealing with square roots and integer constraints.

Protocol decision (author annotation)

Accept the initial solution. ERROR_FOUND: false does not trigger hint-guided re-exploration; the returned answer is 7.

VaCrit: the refinement implements the diagnosis. VaCrit identifies the missing 12 in Step 2 and requests a recount without supplying the final answer; the refinement preserves Step 1 and obtains $12 - 5 + 1 = 8$. This trace illustrates the complementary goals of localization and critique-utility warm-up: learning whether and where to revise, then learning to supply useful repair guidance. The

initialization ablations in Table 3 quantify the benefit of establishing these capabilities before joint training.

VaCrit **Revise** $7 \rightarrow 8$ **Correct**

1. Initial solution

We are given that:

$$\sqrt{t} > 2 \quad \text{and} \quad \sqrt{t} < 3.5$$

Step 1: Square all parts of the inequality to eliminate the square root

$$2 < \sqrt{t} < 3.5$$

Now square all parts:

$$2^2 < t < (3.5)^2$$
$$4 < t < 12.25$$

Step 2: Find integer values of t that satisfy this inequality

We are looking for **integer values** of t such that:

$$4 < t < 12.25$$

So the integers in this range are:

$$5, 6, 7, 8, 9, 10, 11$$

Step 3: Count the number of integers

There are:

$$11 - 5 + 1 = 7 \text{ integers}$$

✓ **Final Answer:**

$$\boxed{7}$$

2a. Diagnostic reasoning (not passed to the refiner)

<think>

Okay, let's go through the student's solution step by step. The problem is about finding the number of integer values of t such that the square root of t is between 2 and 3.5.

First, the student starts by writing the given inequalities: sqrt(t) $>$ 2 and sqrt(t) $<$ 3.5. Then in Step 1, they square all parts of the inequality. Wait, but the original inequality is 2 $<$ sqrt(t) $<$ 3.5. When you square each part, since all parts are positive, the direction of the inequalities remains the same. So squaring 2 gives 4, squaring 3.5 gives 12.25. So the inequality becomes 4 $<$ t $<$ 12.25. That seems correct.

Moving to Step 2, the student is looking for integer values of t between 4 and 12.25. They list the integers as 5,6,7,8,9,10,11. Let's check: the integers greater than 4 and less than 12.25 would start at 5 and go up to 12, since 12.25 is not included. Wait, but 12 is less than 12.25, so 12 is included? Wait, the inequality is t $<$ 12.25, so t can be 12. But the student's list stops at 11. Wait, that's a problem. Because 12 is an integer less than 12.25. So the integers should be 5 through 12, inclusive. Let me count those. From 5 to 12 inclusive, that's 12 -5 +1 = 8 numbers. But the student says 11-5+1=7. Wait, why did they stop at 11?

Wait, the student's Step 2 says the integers are 5,6,7,8,9,10,11. That's 7 numbers. But according to the inequality 4 $<$ t $<$ 12.25, t can be 5,6,7,8,9,10,11,12. So 12 is also included. So the

VaCrit **Revise** $7 \rightarrow 8$ **Correct (continued)**

student missed 12. Therefore, the error is in Step 2 where they listed the integers. They didn't include 12. So the count is wrong.

Wait, but why? Let me check the original problem again. The original inequalities are sqrt(t) $>$ 2 and sqrt(t) $<$ 3.5. So when they square, they get 4 $<$ t $<$ 12.25. So t must be greater than 4 and less than 12.25. So integers from 5 up to 12, since 12.25 is not included. So 12 is allowed. So the student's list is missing 12. Therefore, the error is in Step 2. The student counted 7 integers but there are 8. So the first error is in Step 2. The student incorrectly listed the integers as ending at 11 instead of 12.

So the first error is Step 2. The feedback should point out that they didn't include 12 in the list of integers. The student's calculation of 11-5+1=7 is wrong because they stopped at 11. The correct count should be 12-5+1=8. Therefore, the first error is Step 2.

</think>

2b. Structured critique passed to the refiner

First Error Step: $\boxed{2}$

Feedback: The student incorrectly lists the integers as 5, 6, 7, 8, 9, 10, 11, missing the integer 12. The inequality $4 < t < 12.25$ includes all integers from 5 to 12. The count $11 - 5 + 1$ is wrong because it stops at 11 instead of 12. Correct the list to include 12 and recalculate the count.

3. Critique-guided refinement

The square root of t is greater than 2 and less than 3.5. How many integer values of t satisfy this condition?

Step 1: Square all parts of the inequality to eliminate the square root

$$2 < \sqrt{t} < 3.5$$

Now square all parts:

$$2^2 < t < (3.5)^2$$
$$4 < t < 12.25$$

Step 2: Find integer values of t that satisfy this inequality

We are looking for **integer values** of t such that:

$$4 < t < 12.25$$

So the integers in this range are:

$$5, 6, 7, 8, 9, 10, 11, 12$$

Step 3: Count the number of integers

There are:

$$12 - 5 + 1 = 8 \text{ integers}$$

✓ **Final Answer:**

$$\boxed{8}$$