RECOMATH: BENCHMARKING AND IMPROVING SELF-CORRECTION IN MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-reflection has emerged as a promising approach for improving mathematical reasoning in large language models (LLMs). A central capability of self-reflection is self-correction—the ability of models to identify and recover from errors in their own reasoning. However, existing mathematical reasoning benchmarks primarily evaluate whether models can solve problems from a clean starting point, providing limited insight into whether they can recover once an intermediate reasoning error has occurred. To address this gap, we introduce RECOMATH, a step-level benchmark for evaluating self-correction in mathematical reasoning. Each example pairs a problem with a reasoning prefix ending at the first erroneous step and asks the model to continue the reasoning without being told that the prefix contains an error. Our evaluation across several open-weight LLMs reveals that strong mathematical problem-solving performance does not necessarily translate into reliable self-correction. Using RECOMATH, we conduct a controlled study to isolate the effect of training-trajectory construction, showing that retaining intermediate reflection or repair steps consistently improves self-correction compared with trajectories that omit these processes. Moreover, correct-trace reflection yields more stable gains on easier instances whereas error-retaining repair is stronger on harder ones. Motivated by these findings, we propose *Calibrated Reflection Training* (CRT), which asymmetrically combines correct-trace reflection with a smaller proportion of error-retaining repair. Across three model backbones, CRT achieves a better overall balance between self-correction and mathematical problem-solving performance than either supervision signal alone. Code, data, and benchmark resources are available at https://anonymous.4open.science/r/RecoMath-4323.

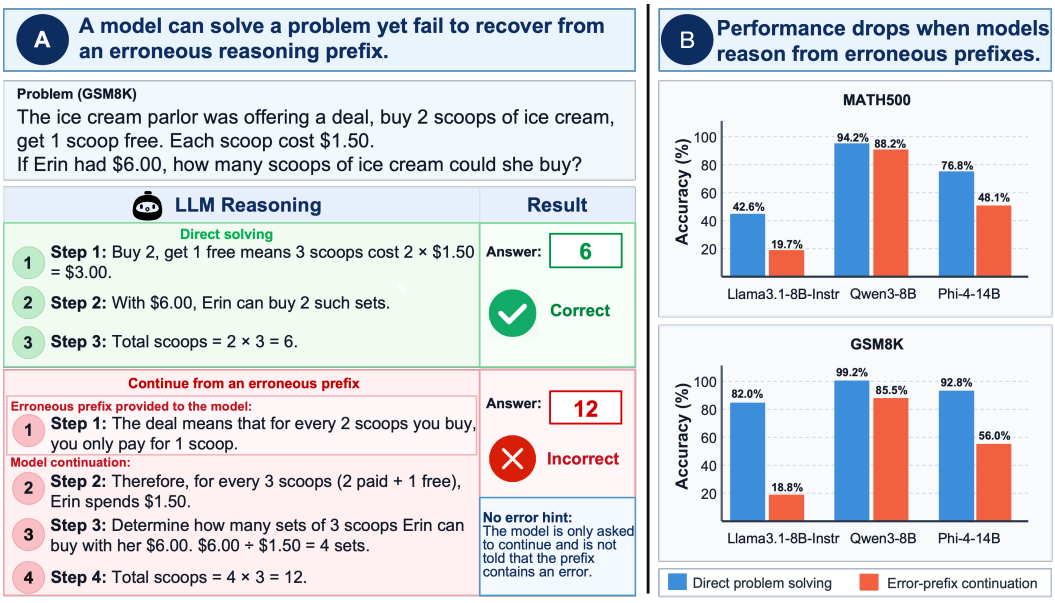

Figure 1: Standard problem solving performance does not fully reflect self-correction ability. **(A)** Phi4-14B solves a GSM8K problem directly but fails from an unhinted erroneous prefix. **(B)** Error-prefix continuation underperforms direct solving across three LLMs on MATH500 and GSM8K.

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.

1 INTRODUCTION

Self-reflection has emerged as an important approach to improving mathematical reasoning in large language models (LLMs), enabling models to revisit intermediate reasoning, identify potential errors, and revise their own solutions (Saunders et al., 2022; Madaan et al., 2023; Kim et al., 2023; Mu et al., 2026). A central capability of self-reflection is self-correction, namely the ability to recover from mistakes made in preceding reasoning (Yan et al., 2025; Huang et al., 2024; Welleck et al., 2023; Tie et al., 2025). Despite growing interest in self-reflection, existing mathematical reasoning benchmarks remain largely centered on problem-solving performance (He et al., 2024; Cobbe et al., 2021; Hendrycks et al., 2021). Existing reflection-oriented benchmarks have begun to assess complementary aspects of reflective reasoning, including error diagnosis, localization, and prompted refinement, but provide limited insight into whether models can recover from erroneous intermediate reasoning during ongoing mathematical problem solving (Hou et al., 2026; Zheng et al., 2025; Qiu et al., 2025). Consequently, self-correction ability in mathematical reasoning remains insufficiently characterized by current benchmarks.

Standard problem-solving benchmarks primarily measure clean-start solving, leaving open whether strong solvers can recover once an intermediate reasoning error has occurred. Figure 1(A) illustrates this distinction: Phi4 solves a GSM8K problem correctly from scratch, but fails when continuing from an erroneous reasoning prefix without an explicit error cue. Figure 1(B) shows the same pattern across models, with substantial performance gaps between direct solving and error-prefix continuation across multiple models on GSM8K and MATH500. These results indicate that strong clean-start problem-solving performance does not necessarily translate into reliable self-correction, motivating a dedicated benchmark for evaluating self-correction in mathematical reasoning.

To this end, we introduce RECOMATH, a step-level benchmark that operationalizes mathematical self-correction as recovery from an erroneous intermediate reasoning state without an explicit error cue. Each instance provides the original problem together with a reasoning prefix ending at the first erroneous step, with the erroneous step itself retained, and asks the model to continue the reasoning toward the correct answer. This construction fixes the model's starting error state and enables error recovery to be evaluated separately from clean-start problem solving, directly measuring whether a model can recover from an existing reasoning error and still reach the correct solution.

Using RECOMATH, we conduct a controlled study of how training trajectory construction affects self-correction and mathematical reasoning. We first find that retaining reflective or repair steps in training trajectories (Pan et al., 2025; Zhang et al., 2024; Deng et al., 2026) leads to stronger self-correction than training only on correct or corrected trajectories that omit such processes (Yang et al., 2025; Yuan et al., 2023; Zelikman et al., 2022). We further compare two ways of retaining this reasoning process: *correct-trace reflection*, which introduces reflective steps into otherwise correct reasoning trajectories, and *error-retaining repair*, which preserves an erroneous reasoning step together with its subsequent repair. Two ways both improve self-correction, but with different difficulty-dependent profiles. Correct-trace reflection performs better on easier recovery instances, whereas error-retaining repair is more effective on difficult ones. On standard mathematical reasoning benchmarks, error-retaining repair can also yield larger gains on more challenging problems, although these gains are less consistent across benchmarks.

Motivated by these complementary performance profiles, we propose *Calibrated Reflection Training* (CRT), an asymmetric composition of the two supervision signals. CRT anchors training in correct-trace reflection while retaining targeted exposure to explicit error-repair trajectories, thereby combining the former's more consistent gains with the latter's strength on difficult recovery instances. Across three backbones, CRT improves mathematical problem-solving performance over the base models while preserving strong self-correction, yielding a better overall balance between the two objectives. Our main contributions are as follows:

- We introduce RECOMATH, a step-level benchmark for evaluating self-correction in mathematical reasoning, where models must recover from erroneous reasoning without being told that an error is present.
- We reveal that retaining self-reflection trajectories substantially improves self-correction, while correct-trace reflection and error-retaining repair play complementary roles in general mathematical reasoning.

- Based on these findings, we propose CRT, an asymmetric mixture of correct-trace reflection and error-retaining repair, and show that it improves RECOMATH accuracy by 4.66 points and the five-benchmark mathematical problem-solving average by 3.89 points over the base models, averaged across three backbones.

2 RELATED WORK

Evaluating Mathematical Reasoning and Self-Correction. Recent work has moved beyond final-answer accuracy toward evaluating the quality, diagnosis, and correction of intermediate reasoning (Lin et al., 2024; Li et al., 2024b; Xia et al., 2025; Zeng et al., 2025). CriticBench (Lin et al., 2024) evaluates critique and correction across reasoning domains, while MWP-MISTAKE (Singh et al., 2025) and MR-GSM8K (Zeng et al., 2025) assess mistake detection, localization, and diagnosis in mathematical reasoning. ProcessBench (Zheng et al., 2025) and Hard2Verify (Pandit et al., 2026) further emphasize step-level verification and first-error localization. Related studies also show that models may correct an error when its location is provided (Tyen et al., 2024), while intrinsic self-correction without external feedback remains unreliable (Huang et al., 2024). In contrast, RECOMATH evaluates whether a model can recover from an erroneous intermediate state under ordinary continuation, without being told that an error has occurred.

Training for Reflection and Error Recovery. Prior work improves reasoning through correction-oriented trajectories and objectives. Positive-only methods such as STaR and rejection-sampling fine-tuning learn from successful trajectories (Zelikman et al., 2022; Yuan et al., 2023). Other work trains on mistakes, critiques, or intermediate reflection (An et al., 2023; Tong et al., 2024; Zhang et al., 2024; Yan et al., 2025). LEMMA (Pan et al., 2025) retains erroneous prefixes followed by reflection and repair, whereas InT (Yang et al., 2026) removes erroneous steps from corrected trajectories. Recent work also constructs recovery supervision from failed reasoning prefixes (Chen et al., 2026). Beyond SFT, SuperCorrect (Yang et al., 2025) uses teacher-guided correction with cross-model preference optimization, SCoRe (Kumar et al., 2025) uses online RL over self-generated correction traces, and S^2R (Ma et al., 2025) combines behavioral initialization with outcome- and process-level RL. Self-rewarding correction (Xiong et al., 2025) further combines self-generated correction trajectories with reinforcement learning. These methods vary in trajectory form, data generation, correction source, and optimization. We instead reinstantiate representative constructions, including the LEMMA-style error-retaining pattern, under a common SFT setting to isolate trajectory construction as the controlled variable.

3 RECOMATH: BENCHMARKING SELF-CORRECTION IN MATHEMATICAL REASONING

3.1 TASK DEFINITION

We formulate mathematical self-correction as an error-recovery task over intermediate reasoning states. Given a mathematical problem x and a reasoning prefix $r_{1:k} = (r_1, \ldots, r_k)$ ending at the first erroneous step r_k, the model is asked to generate a continuation $r_{k+1:T}$ that recovers from the erroneous reasoning state and reaches the correct final answer. Crucially, the model is not informed that the provided prefix contains an error. Formally, given x and $r_{1:k}$, the model generates

$$r_{k+1:T} \sim p_\theta(r_{k+1:T} \mid x, r_{1:k}), \tag{1}$$

and self-correction is considered successful if the resulting complete reasoning trajectory reaches the correct final answer:

$$\text{Success}(x, r_{1:T}) = \mathbb{I}[\text{Ans}(r_{1:T}) = y^*], \tag{2}$$

where y^* denotes the ground-truth answer.

3.2 DATA CONSTRUCTION

Constructing a reliable self-correction benchmark requires addressing three challenges. First, automatic answer verification may misclassify mathematically equivalent expressions, such as 4^n and 2^{2n}, as different answers. Second, reference answers in a large and challenging source dataset may

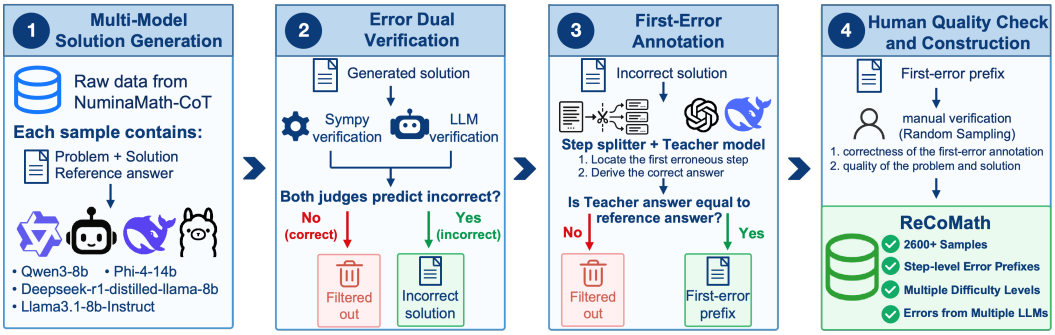

Figure 2: Overview of the cross-model error construction pipeline for RECOMATH. We generate reasoning trajectories from multiple LLMs, apply dual verification using SymPy and an LLM verifier, localize the first erroneous step with a teacher model, and perform manual quality checks before constructing the final benchmark.

occasionally be unreliable and require further validation. Third, an incorrect final answer alone cannot reveal where the reasoning first goes wrong. Accordingly, we analyze verification errors arising from mathematically equivalent answer forms and reference-answer noise in Appendix A.1, with representative symbolic-verification failures provided in Appendix A.2. To address these verification issues and identify where reasoning first goes wrong, we construct RECOMATH through a multi-stage pipeline that verifies genuine model errors, filters unreliable references, and localizes the first erroneous reasoning step (Figure 2).

Multi-Model Solution Generation. We use the last 20,000 examples from NuminaMath-CoT (Li et al., 2024a) as the problem source for constructing model-generated reasoning errors. For each problem, we independently prompt four LLMs—Qwen3-8B, Phi4-14B, DeepSeek-R1-Distill-Llama-8B, and Llama3.1-8B-Instruct—to generate a complete reasoning trajectory and final answer. Using multiple generator models allows us to collect diverse reasoning failures rather than relying on the error patterns of a single model family.

Error Dual Verification. We first compare each generated final answer $\hat{y}$ against the reference answer y^* using SymPy. Since symbolic matching may fail to recognize mathematically equivalent answers expressed in different forms, we additionally employ an LLM-based verifier. Let $V_{\text{sym}}(\hat{y}, y^*)$ and $V_{\text{llm}}(\hat{y}, y^*)$ denote the two verification results, where 1 indicates that the generated answer is judged correct. We retain a trajectory as a verified error only when

$$V_{\text{sym}}(\hat{y}, y^*) = 0 \quad \wedge \quad V_{\text{llm}}(\hat{y}, y^*) = 0. \tag{3}$$

This dual-verification procedure reduces false labels caused by equivalent answer representations. Appendix A.1 further analyzes verification errors on semantically equivalent answers and reference-answer noise, while Appendix A.2 presents representative symbolic-verification failures.

First-Error Annotation. For each verified incorrect trajectory, we segment the complete reasoning trace into individual reasoning steps using a model-aware segmentation procedure following Li et al. (2026). Details of the model-specific step segmentation rules are provided in Appendix A.3. Given the problem and the resulting step-level trajectory, we use a stronger teacher model, either DeepSeek-V4-Flash or GPT-5.4, to identify the earliest erroneous step, explain the error, and independently derive the correct final answer. To reduce noise from unreliable or ambiguous references, we retain the instance only when the teacher-derived answer is consistent with the reference answer. If r_k is identified as the first erroneous step in a trajectory $(r_1, \ldots, r_T)$, we construct the benchmark reasoning prefix as

$$r_{\text{prefix}} = r_{1:k} = (r_1, \ldots, r_k), \tag{4}$$

thereby retaining the erroneous step itself as part of the model input.

Human Quality Check and Construction. As a final quality check, we randomly sample a subset of the constructed instances for manual review. For each sampled instance, we verify the correctness of the first-error annotation, the validity of the error explanation, and the overall quality of the problem and reasoning trajectory. This manual review provides an additional quality check on the automatically constructed benchmark.

3.3 Benchmark Composition and Augmentation

To broaden the coverage of ReCoMath, we augment our constructed dataset with instances derived from ProcessBench (Zheng et al., 2025), using its GSM8K, MATH, OlympiadBench, and OmniMath subsets. For each valid error trajectory, we retain the reasoning prefix ending with the first erroneous step and apply the same unhinted continuation setting. The resulting benchmark contains 4,857 instances, including 2,645 constructed error instances collected from multiple models and 2,212 ProcessBench-derived instances for broader evaluation coverage. Details of the ProcessBench preprocessing procedure are provided in Appendix A.4.

Table 1: Composition of ReCoMath by difficulty split.

Split	Instances	Avg Error Step
Easy	2,380	5.42
Hard	2,477	7.77
Total	4,857	6.62

Before model evaluation, we define a fixed difficulty split based on instance source. The Easy subset split contains error trajectories generated by Llama3.1-8B-Instruct and DeepSeek-R1-Distill-Llama-8B in our constructed data, together with ProcessBench instances from GSM8K and MATH; the Hard subset consists of Qwen3-8B and Phi4-14B error trajectories, together with OlympiadBench and OmniMath. As shown in Table 1, the two subsets are approximately balanced, while error steps in the Hard subset occur later in the reasoning trajectory. Source-level benchmark statistics are reported in Appendix B.1.

3.4 Evaluation Protocol

Self-Correction Evaluation. For each benchmark instance, the model is given the original problem and a reasoning prefix ending at the first erroneous step, and is instructed to continue the reasoning and finish the solution without any indication that the prefix contains an error. The exact evaluation prompt is provided in Appendix A.6.

Error-Indication Analysis. To examine how explicit error information affects recovery, we additionally consider a hinted setting in which the model is explicitly told that the reasoning prefix contains an error and instructed to correct it before completing the solution.

We compare the hinted setting with unhinted continuation from the same erroneous prefix and direct solving from scratch using Llama3.1-8B-Instruct. The former isolates the effect of explicit error information, while the latter reflects baseline problem-solving ability. As shown in Figure 3, error indication improves recovery across all four benchmarks, but its effect varies substantially by dataset: accuracy increases from 18.8% to 52.1% on GSM8K and from 19.7% to 31.1% on MATH500, versus gains of only 1.4 and 1.9 points on OlympiadBench and Omni-MATH. Since explicit error cues systematically alter measured recovery, we use unhinted continuation as the standard evaluation protocol of ReCoMath.

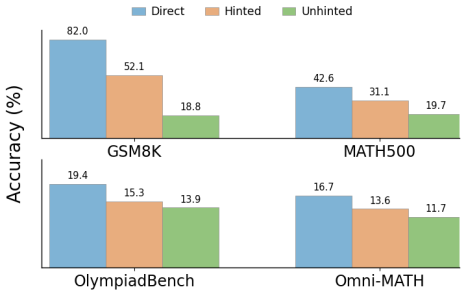

Figure 3: Effect of explicit error indication on accuracy (%) for Llama3.1-8B-Instruct.

Evaluation Metric. We use final-answer accuracy as the primary evaluation metric. A recovery is considered successful if the model reaches the correct final answer after continuing from the provided erroneous prefix, regardless of the specific correction strategy used in the continuation.

4 A Controlled Study for Self-Correction

4.1 Experimental Setup

Prior work on self-correction couples training-trajectory design with many other experimental choices, including the base model, data source, teacher model, supervision objective, and optimization procedure. Consequently, directly comparing their reported performance makes it difficult to

determine which design choices are responsible for the observed gains. To disentangle these effects, we conduct a controlled study in which all trajectory variants are trained under a unified framework, with the training configuration held fixed and only the trajectory construction varied. This allows us to examine how different representations of correct reasoning, errors, reflection, and repair affect both error recovery and general mathematical reasoning. Appendix C provides a detailed comparison of the training scale, supervision, and optimization used by representative prior methods.

To operationalize this controlled comparison, we abstract four representative trajectory-construction families from prior work and implement them within a common supervised fine-tuning framework. As summarized in Table 2, we organize the resulting trajectories into four types. These strategies differ in what information from the correction process is retained in the final training trajectory: successful reasoning only, a corrected continuation, explicit reflection, or an error followed by repair. *Correct-only* retains only successful reasoning, while *error-removed correction* uses a corrected continuation after an error but exposes the model to neither the erroneous step nor an explicit repair process. In contrast, *correct-trace reflection* retains explicit reflection within an otherwise correct trajectory, whereas *error-retaining repair* preserves the erroneous step together with its subsequent repair. We compare strategic and verification reflection in an ablation study. All variants use the same base checkpoint, supervised fine-tuning objective, and optimization configuration, enabling a controlled comparison of how trajectory construction affects model performance.

Table 2: Training trajectory variants compared under our unified SFT framework. C and C' denote correct reasoning segments before and after the reflection or correction point, R denotes reflective or repair reasoning, and E denotes an erroneous reasoning step.

Variant Name	**Construction Strategy**	**Trajectory Form**	**Representative Prior Work**
Correct-Only	Correct-only	C	RFT / STaR (Zelikman et al., 2022)
Error-Removed	Error-removed correction	$C \rightarrow C'$	InT (Yang et al., 2026)
Reflection-Retained	Correct-trace reflection	$C \rightarrow R \rightarrow C'$	RefAug (Zhang et al., 2024)
Error-Retained	Error-retaining repair	$C \rightarrow E \rightarrow R \rightarrow C'$	LEMMA (Pan et al., 2025)

Training and Evaluation Setup. We construct all training variants from the first 100,000 examples of NuminaMath-CoT (Li et al., 2024a), after removing problems originating from our evaluation benchmarks. To ensure a controlled comparison, each variant contains exactly 15,000 training instances. We conduct experiments on three backbone models: Qwen3-8B (Team, 2025), Llama3.1-8B-Instruct (Team, 2024), and Phi4-14B (Abdin et al., 2024). All models are fine-tuned with LoRA for two epochs using a learning rate of 1×10^{-4}, while keeping the remaining training configuration fixed across trajectory variants for the same backbone. We evaluate standard mathematical reasoning on AIME 2025, AIME 2026 (Balunović et al., 2025), AIMO-AMC (AI-MO, 2024), Omni-MATH (Gao et al., 2025), and MATH500 (Lightman et al., 2024), and evaluate self-correction on RECOMATH. For AIME 2025 and AIME 2026, we repeat the evaluation six times with independent sampling and report the mean accuracy. We report accuracy on the full RECOMATH benchmark and on the fixed source-based Easy and Hard subsets defined in Section 3.3. We follow the Qwen2.5 evaluation framework (Yang et al., 2024) with LLM-assisted answer verification for all evaluations.

4.2 RESULTS AND ANALYSIS

Retaining Reflective or Repair Reasoning Improves Self-Correction. Table 3 shows that trajectories retaining intermediate reflection or repair consistently improve self-correction across all three backbones. On the full RECOMATH benchmark, both Reflection-Retained and Error-Retained outperform their corresponding base models, whereas Correct-Only yields only modest gains and Error-Removed shows no consistent benefit. Figure 4 summarizes this contrast: retaining constructions improve RECOMATH accuracy by 2.02–4.90 points across the three backbones, while non-retaining variants remain near the base models or even degrade performance. These results suggest that the gains arise not merely from exposure to correct outcomes, but from explicitly preserving the intermediate reflection or repair process as supervision.

Reflection and Repair Exhibit Complementary Effects. Although both retaining constructions are effective, their relative advantages vary with recovery difficulty. Reflection-Retained performs best on the Easy subset across all three backbones, whereas Error-Retained is strongest on the Hard

Table 3: Self-correction and mathematical problem-solving accuracy (%) under different trajectory constructions. Avg. denotes the average across mathematical problem-solving benchmarks.

Method	RECOMATH			Mathematical Problem Solving					
	All	Easy	Hard	AIME25	AIME26	AIMO-AMC	MATH500	Omni-MATH	Avg.
Qwen3-8B									
Base	53.28	60.10	46.61	64.17	50.00	89.68	94.23	51.70	69.96
Correct-Only	53.94	60.88	47.27	65.00	56.67	87.24	94.08	54.24	71.45
Error-Removed	50.52	55.18	45.97	58.83	53.33	87.61	90.68	50.90	68.27
Reflection-Retained	58.02	65.78	49.29	61.67	58.33	92.49	94.89	55.30	72.54
Error-Retained	58.34	63.25	53.22	71.67	64.17	83.61	92.25	57.40	73.82
Llama3.1-8B-Instruct									
Base	12.71	14.10	11.34	3.33	0.83	21.40	45.42	12.71	16.74
Correct-Only	13.89	15.06	12.75	3.33	1.67	18.63	47.67	13.42	16.94
Error-Removed	13.05	13.40	12.71	5.00	5.83	23.29	44.83	12.03	18.20
Reflection-Retained	14.58	16.41	12.79	4.17	3.33	29.27	48.80	14.60	20.03
Error-Retained	14.88	14.97	14.80	7.50	6.67	21.29	46.87	15.01	19.47
Phi4-14B									
Base	31.19	37.90	24.61	10.00	10.00	53.82	76.83	26.40	35.41
Correct-Only	33.44	40.53	26.87	14.44	10.83	53.64	78.06	26.31	36.66
Error-Removed	30.00	36.68	23.45	8.89	6.67	52.59	73.28	26.40	33.57
Reflection-Retained	35.58	44.51	26.52	14.44	9.16	56.17	78.81	26.08	36.93
Error-Retained	34.52	40.34	28.83	23.33	16.67	51.38	71.82	28.90	38.42

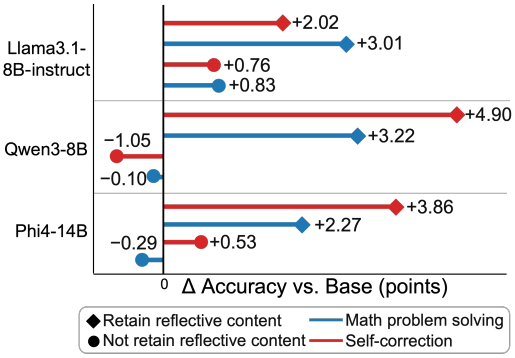

Figure 4: Effect of retaining reflective content across three model backbones. Values denote accuracy changes relative to the corresponding base model after fine-tuning.

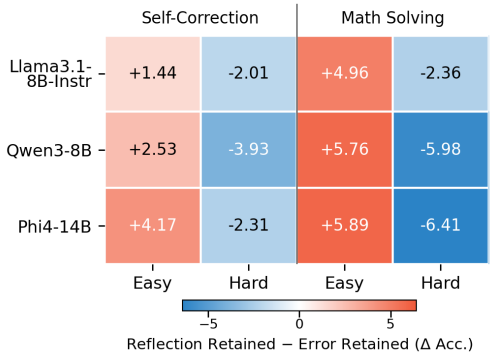

Figure 5: Complementary effects of correct-trace reflection and error-retaining repair. Positive values favor correct-trace reflection, while negative values favor error-retaining repair.

subset. For example, on Qwen3-8B, the former achieves 65.78% versus 63.25% on Easy, while the ordering reverses on Hard (49.29% vs. 53.22%). Figure 5 shows the same difficulty-dependent pattern more broadly, suggesting that reflection is more effective for easier recovery, while explicit error-to-repair supervision becomes more useful as recovery becomes harder. The two constructions also favor different mathematical reasoning benchmarks: Error-Retained is stronger on AIME25, AIME26, and Omni-MATH, whereas Reflection-Retained performs better on AIMO-AMC and MATH500. Together, these results indicate that the two forms of supervision are complementary rather than interchangeable.

5 CALIBRATED REFLECTION TRAINING

5.1 METHOD

The controlled study in Section 4.2 reveals that correct-trace reflection and error-retaining repair provide complementary training signals. Correct-trace reflection yields more consistent gains, whereas

error-retaining repair is particularly beneficial for difficult recovery and challenging mathematical problems. Motivated by this finding, we propose *Calibrated Reflection Training*, which uses correct-trace reflection as the primary supervision signal and incorporates a smaller amount of error-retaining repair as complementary supervision.

Let $\mathcal{D}_{\text{ref}}$ denote the set of correct-trace reflection trajectories, where reflective steps are retained within otherwise correct reasoning, and let $\mathcal{D}_{\text{rep}}$ denote error-retaining repair trajectories, where an erroneous reasoning step is preserved together with its subsequent repair. We construct the training set by sampling from the two sources with

$$\mathcal{D}_{\text{CRT}} \sim \lambda \mathcal{D}_{\text{ref}} + (1-\lambda)\mathcal{D}_{\text{rep}}, \qquad \lambda > 0.5, \tag{5}$$

where λ controls the proportion of correct-trace reflection. This asymmetric composition treats correct-trace reflection as the primary signal, while using error-retaining repair to expose the model to explicit recovery from erroneous reasoning states.

We train on the resulting trajectories using the standard autoregressive language-modeling objective,

$$\mathcal{L}_{\text{CRT}} = -\sum_{(x,y)\in\mathcal{D}_{\text{CRT}}} \log p_\theta(y \mid x), \tag{6}$$

without introducing an additional optimization objective. We keep the total number of training examples fixed relative to the controlled variants in Section 4.1, so that improvements cannot be attributed simply to using more training data. In our main experiments, we set $\lambda = 0.8$, corresponding to an 80:20 mixture of correct-trace reflection and error-retaining repair.

Training Setup. We evaluate Calibrated Reflection Training on Qwen3-8B, Llama3.1-8B-Instruct, and Phi4-14B using the same training and evaluation configuration as in Section 4.1. We keep the training data budget fixed across methods to isolate the effect of combining the two supervision signals.

5.2 Main Results

Table 4: Performance of Calibrated Reflection Training across three backbones. **Bold** and underline denote the best and second-best results, respectively, when compared with the controlled trajectory constructions in Table 3.

Model	ReCoMath			Mathematical Problem Solving					
	All	Easy	Hard	AIME25	AIME26	AIMO-AMC	MATH500	Omni-MATH	Avg.
Qwen3-8B	<u>58.24</u>	63.10	**53.47**	69.17	62.50	89.68	**95.01**	56.51	**74.57**
Llama3.1-8B-Instruct	**15.89**	**17.69**	14.09	5.83	5.83	24.51	47.27	14.35	<u>19.56</u>
Phi4-14B	**37.04**	**45.27**	<u>28.81</u>	<u>22.22</u>	<u>14.17</u>	<u>53.93</u>	**79.47**	<u>28.40</u>	**39.64**

As shown in Table 4, Calibrated Reflection Training achieves strong performance across both self-correction and mathematical problem solving. On Phi4-14B, CRT surpasses all individual trajectory constructions on both overall ReCoMath accuracy and average problem-solving accuracy. On Qwen3-8B, it achieves the highest problem-solving average while remaining within 0.10 points of the best self-correction result. On Llama3.1-8B-Instruct, it achieves the best self-correction result while remaining within 0.47 points of the best problem-solving average.

These results indicate that combining correct-trace reflection with error-retaining repair effectively preserves their complementary strengths, providing a better balance between self-correction and mathematical problem solving than relying on either signal alone.

5.3 Ablation Study

Effect of Constituent Construction. We further examine alternative constructions of the two supervision signals. For correct-trace reflection, *strategic reflection* revisits the reasoning strategy or solution approach, whereas *verification reflection* checks intermediate arithmetic computations and numerical consistency. For error-retaining repair, *teacher repair* retains the model-generated erroneous step but uses a stronger teacher model to generate the subsequent correction, while *self repair* retains both the error and the repair generated by the same student model.

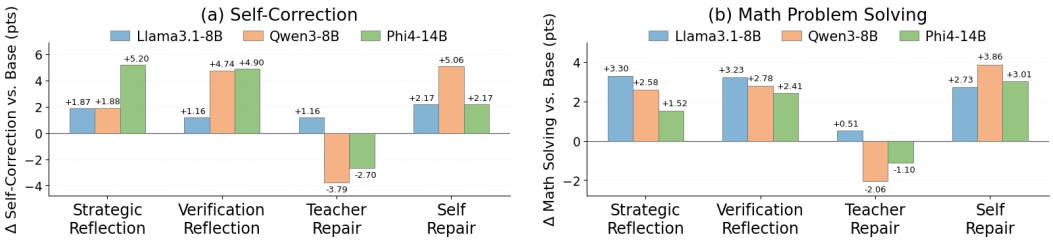

Figure 6: Ablation on the construction of reflection and repair supervision. **(a)** Changes in self-correction performance relative to the base model. **(b)** Changes in mathematical problem-solving performance. Strategic and verification reflection both provide positive gains, while self-generated repair consistently outperforms teacher-generated repair across the three backbones.

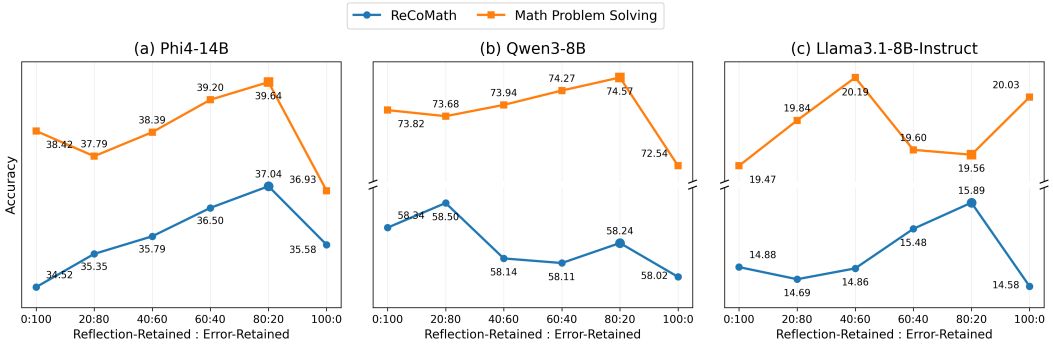

Figure 7: Effect of the reflection–repair mixture ratio across Phi4-14B, Qwen3-8B, and Llama3.1-8B-Instruct. Panels (b) and (c) use broken y-axes to display the two accuracy ranges.

As shown in Figure 6, both forms of reflection consistently improve self-correction and mathematical problem solving, although their relative advantages vary across backbones. The distinction is clearer for repair supervision: self repair consistently yields positive gains, whereas teacher repair provides limited benefits and can even degrade performance on Qwen3-8B and Phi4-14B. These results suggest that preserving the model's own error-to-repair process provides a more effective repair signal than replacing the recovery with a teacher-generated continuation.

Effect of the Reflection–Repair Ratio. We examine how the mixture ratio between correct-trace reflection and error-retaining repair affects model performance. As shown in Figure 7, the optimal ratio varies across models and evaluation objectives, but reflection-dominant mixtures generally provide a stronger balance between self-correction and mathematical problem solving. In particular, the 80:20 mixture achieves the strongest overall performance across the three backbones, although individual metrics may peak at other ratios. Performance also tends to degrade when error-retaining repair is removed entirely, indicating that repair supervision remains complementary even when reflection serves as the primary training signal.

6 Conclusion

In this paper, we introduced ReCoMath, a step-level benchmark for evaluating self-correction in mathematical reasoning by requiring models to recover from erroneous intermediate reasoning without explicit error hints. Our study reveals that strong mathematical problem-solving performance does not necessarily translate into reliable self-correction. Through a controlled study of training trajectory construction, we further find that retaining intermediate reflection or repair provides more effective supervision than using only correct or corrected trajectories, while correct-trace reflection and error-retaining repair exhibit complementary strengths across recovery difficulty and mathematical problem solving. Based on these findings, we proposed Calibrated Reflection Training, which combines the two signals asymmetrically and achieves a better balance between self-correction and mathematical problem-solving performance across multiple backbones. These results highlight the importance of explicitly modeling error recovery and provide a practical direction for improving self-correction in mathematical reasoning.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing, literature search, figure-generation code, and discussion of experimental presentation and result interpretation. Large language models were also used within the research pipeline for reasoning-trajectory generation, answer verification, error annotation, and supervision construction, as described in the paper. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the final content and reported results.

REFERENCES

Marah I Abdin, Jyoti Aneja, Harkirat S. Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report. *CoRR*, abs/2412.08905, 2024. doi: 10.48550/ARXIV.2412.08905. URL https://doi.org/10.48550/arXiv.2412.08905.

AI-MO. Aimo validation amc. Hugging Face Dataset, 2024. URL https://huggingface.co/datasets/AI-MO/aimo-validation-amc.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes LLM better reasoner. *CoRR*, abs/2310.20689, 2023. doi: 10.48550/ARXIV.2310.20689. URL https://doi.org/10.48550/arXiv.2310.20689.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *arXiv preprint arXiv:2505.23281*, 2025. URL https://arxiv.org/abs/2505.23281.

Qirui Chen, Renjie Pi, Jiahui Gao, and Lingpeng Kong. Reflective recovery: A self-supervised method for reasoning by learning from mistakes, 2026. URL https://arxiv.org/abs/2609.19156.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021. URL https://arxiv.org/abs/2110.14168.

Jie Deng, Hanshuang Tong, Jun Li, Shining Liang, Ning Wu, Hongzhi Li, and Yutao Xie. Beyond rejection sampling: Trajectory fusion for scaling mathematical reasoning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 7943–7959. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.390. URL https://doi.org/10.18653/v1/2026.findings-acl.390.

Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=yaqPf0KAlN.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 3828–3850. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.211. URL https://doi.org/10.18653/v1/2024.acl-long.211.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Yujie Hou, Mei Wang, Yaoyao Zhong, Ting Zhang, Xuetao Ma, and Hua Huang. SMART: evaluating llms' mathematical reasoning via a human cognitive process-inspired benchmark. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 35426–35452. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.1638. URL https://doi.org/10.18653/v1/2026.acl-long.1638.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/7cc1005ec73cfbaac9fa21192b622507-Abstract-Conference.html.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal M. P. Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. https://github.com/project-numina/aimo-progress-prize, 2024a. URL https://github.com/project-numina/aimo-progress-prize.

Jiakun Li, Xingwei He, Kefan Li, Hongzheng Chai, Hongyue Yu, and Yuan Yuan. Efficient test-time scaling via temporal reasoning aggregation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics, ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 13304–13318. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.FINDINGS-ACL.651. URL https://doi.org/10.18653/v1/2026.findings-acl.651.

Xiaoyuan Li, Wenjie Wang, Moxin Li, Junrong Guo, Yang Zhang, and Fuli Feng. Evaluating mathematical reasoning of large language models: A focus on error identification and correction. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 11316–11360. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.FINDINGS-ACL.673. URL https://doi.org/10.18653/v1/2024.findings-acl.673.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Zicheng Lin, Zhibin Gou, Tian Liang, Ruilin Luo, Haowei Liu, and Yujiu Yang. Criticbench: Benchmarking llms for critique-correct reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 1552–1587. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.91. URL https://doi.org/10.18653/v1/2024.findings-acl.91.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. S^2r: Teaching llms to self-verify and self-correct via reinforcement learning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 22632–22654. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.1104. URL https://doi.org/10.18653/v1/2025.acl-long.1104.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Jian Mu, Qixin Zhang, Zhiyong Wang, Menglin Yang, Shuang Qiu, Chengwei Qin, Zhongxiang Dai, and Yao Shu. Self-reflective generation at test time. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 10221–10239. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.465. URL https://doi.org/10.18653/v1/2026.acl-long.465.

Zhuoshi Pan, Yu Li, Honglin Lin, Qizhi Pei, Zinan Tang, Wei Wu, Chenlin Ming, H. Vicky Zhao, Conghui He, and Lijun Wu. LEMMA: learning from errors for mathematical advancement in llms. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, volume ACL 2025 of *Findings of ACL*, pp. 11615–11639. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.FINDINGS-ACL.605. URL https://doi.org/10.18653/v1/2025.findings-acl.605.

Shrey Pandit, Austin Xu, Xuan-Phi Nguyen, Yifei Ming, Caiming Xiong, and Shafiq Joty. Hard2verify: A step-level verification benchmark for open-ended frontier math. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 22502–22517. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.1031. URL https://doi.org/10.18653/v1/2026.acl-long.1031.

Ruidi Qiu, Grace Li Zhang, Rolf Drechsler, Ulf Schlichtmann, and Bing Li. Correctbench: Automatic testbench generation with functional self-correction using llms for HDL design. In *Design, Automation & Test in Europe Conference, DATE 2025, Lyon, France, March 31 - April 2, 2025*, pp. 1–7. IEEE, 2025. doi: 10.23919/DATE64628.2025.10992873. URL https://doi.org/10.23919/DATE64628.2025.10992873.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *CoRR*, abs/2206.05802, 2022. doi: 10.48550/ARXIV.2206.05802. URL https://doi.org/10.48550/arXiv.2206.05802.

Joykirat Singh, Akshay Uttama Nambi, and Vibhav Vineet. Exposing the achilles' heel: Evaluating llms ability to handle mistakes in mathematical reasoning. In Wanxiang Che, Joyce Nabende,

Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 27044–27065. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.1313. URL https://doi.org/10.18653/v1/2025.acl-long.1313.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Guiyao Tie, Zenghui Yuan, Zeli Zhao, Chaoran Hu, Tianhe Gu, Ruihang Zhang, Sizhe Zhang, Junran Wu, Xiaoyue Tu, Ming Jin, Qingsong Wen, Lixing Chen, Pan Zhou, and Lichao Sun. Can llms correct themselves? A benchmark of self-correction in llms. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/ec07904adc847a45f53dceb44078f8f0-Abstract-Datasets_and_Benchmarks_Track.html.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. Can llms learn from previous mistakes? investigating llms' errors to boost for reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 3065–3080. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.169. URL https://doi.org/10.18653/v1/2024.acl-long.169.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, volume ACL 2024 of *Findings of ACL*, pp. 13894–13908. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.826. URL https://doi.org/10.18653/v1/2024.findings-acl.826.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=hH36JeQZDaO.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. Evaluating mathematical reasoning beyond accuracy. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pp. 27723–27730. AAAI Press, 2025. doi: 10.1609/AAAI.V39I26.34987. URL https://doi.org/10.1609/aaai.v39i26.34987.

Wei Xiong, Hanning Zhang, Chenlu Ye, Lichang Chen, Nan Jiang, and Tong Zhang. Self-rewarding correction for mathematical reasoning. *CoRR*, abs/2502.19613, 2025. doi: 10.48550/ARXIV.2502.19613. URL https://doi.org/10.48550/arXiv.2502.19613.

Yuchen Yan, Jin Jiang, Yang Liu, Yixin Cao, Xin Xu, Mengdi Zhang, Xunliang Cai, and Jian Shao. S^3cmath: Spontaneous step-level self-correction makes large language models better mathematical reasoners. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pp. 25588–25596. AAAI Press, 2025.

doi: 10.1609/AAAI.V39I24.34749. URL https://doi.org/10.1609/aaai.v39i24.34749.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *CoRR*, abs/2409.12122, 2024. doi: 10.48550/ARXIV.2409.12122. URL https://doi.org/10.48550/arXiv.2409.12122.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Minkai Xu, Joseph E. Gonzalez, Bin Cui, and Shuicheng Yan. Supercorrect: Advancing small LLM reasoning with thought template distillation and self-correction. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=PyjZO7oSw2.

Matthew Y. R. Yang, Hao Bai, Ian Wu, Gene Yang, Amrith Setlur, and Aviral Kumar. Int: Self-proposed interventions enable credit assignment in LLM reasoning. *CoRR*, abs/2601.14209, 2026. doi: 10.48550/ARXIV.2601.14209. URL https://doi.org/10.48550/arXiv.2601.14209.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825, 2023. doi: 10.48550/ARXIV.2308.01825. URL https://doi.org/10.48550/arXiv.2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/639a9a172c044fbb64175b5fad42e9a5-Abstract-Conference.html.

Zhongshen Zeng, Pengguang Chen, Shu Liu, Haiyun Jiang, and Jiaya Jia. MR-GSM8K: A meta-reasoning benchmark for large language model evaluation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=br4H61LOoI.

Zhihan Zhang, Tao Ge, Zhenwen Liang, Wenhao Yu, Dian Yu, Mengzhao Jia, Dong Yu, and Meng Jiang. Learn beyond the answer: Training language models with reflection for mathematical reasoning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 14720–14738. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.817. URL https://doi.org/10.18653/v1/2024.emnlp-main.817.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 1009–1024. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-LONG.50. URL https://doi.org/10.18653/v1/2025.acl-long.50.

A Additional Details of ReCoMath Construction

A.1 Answer Verification and Reference Noise

Our benchmark construction relies on a multi-stage filtering pipeline, making it important to examine both the limitations of symbolic answer matching and the reliability of reference answers. Figure 8 summarizes these two sources of noise.

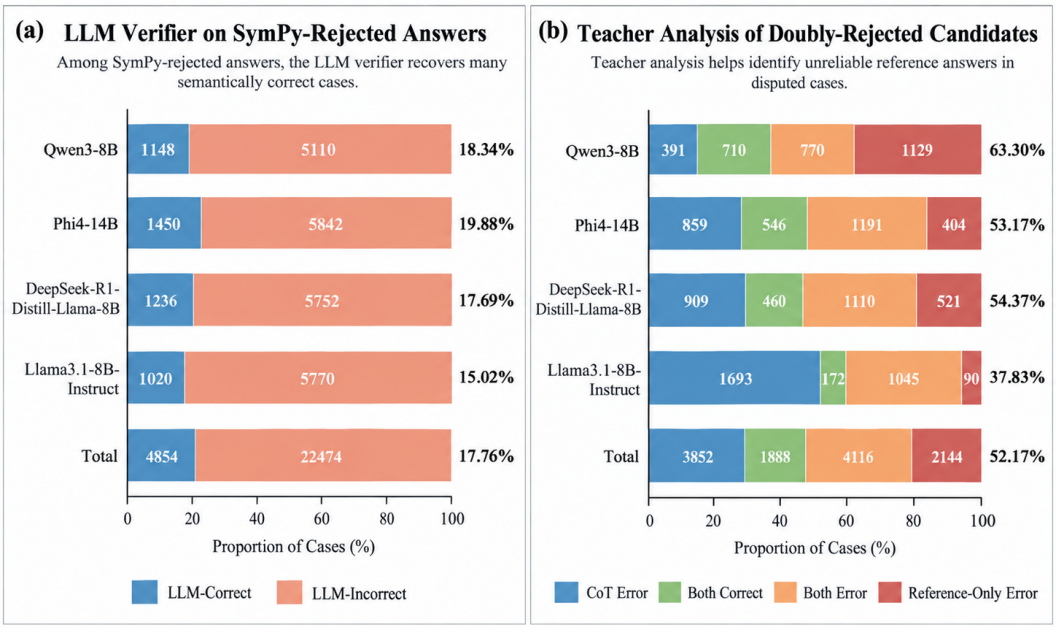

Figure 8: Analysis of answer verification and reference noise during benchmark construction. **(a)** Among answers rejected by SymPy, the LLM verifier recovers a substantial portion of semantically correct cases. **(b)** Teacher analysis of candidates rejected by both SymPy and the LLM verifier. *Both Error* and *Reference-Only Error* indicate cases where the reference answer is flagged as unreliable.

LLM Verification Complements Symbolic Matching. We first examine answers rejected by SymPy. As shown in Figure 8(a), the LLM verifier judges a substantial portion of these answers to be correct. Across the four generator models, 4,854 of 27,328 SymPy-rejected answers (17.76%) are recovered by the LLM verifier. This proportion is relatively consistent across models, ranging from 15.02% for Llama3.1-8B-Instruct to 19.88% for Phi4-14B. These results indicate that symbolic matching alone can incorrectly reject many semantically equivalent answers. Representative failure cases are provided in Appendix A.2.

To assess whether the LLM verifier reliably resolves these disagreements, we randomly sample 500 cases rejected by SymPy but judged correct by the LLM verifier. As summarized in Table 5, human inspection confirms 495 of the 500 cases, corresponding to 99.0% precision. Thus, the LLM verifier reliably recovers semantically correct answers that symbolic matching fails to recognize.

Table 5: Manual validation of model-based filtering.

Stage	Sampled	Confirmed	Precision
LLM verifier	500	495	99.0%
Teacher filter	50	41	82.0%

Teacher Analysis Helps Filter Unreliable References. We next consider candidates rejected by both SymPy and the LLM verifier. For these cases, a stronger teacher model independently solves the problem and analyzes the consistency between the generated reasoning and the reference answer. The resulting analysis contains four categories: *CoT Error*, *Both Correct*, *Both Error*, and *Reference-Only Error*. We regard the latter two categories as cases in which the reference answer is flagged as potentially unreliable.

As shown in Figure 8(b), 4,116 candidates are classified as *Both Error* and 2,144 as *Reference-Only Error*, totaling 6,260 of the 12,000 analyzed candidates (52.17%). The proportion varies across generator models, from 37.83% for Llama3.1-8B-Instruct to 63.30% for Qwen3-8B. Importantly, this statistic is computed over the already filtered pool for which both SymPy and the LLM verifier reject the generated answer, and therefore should not be interpreted as the error rate of the original source dataset.

We further manually inspect 50 randomly sampled cases in which the teacher flags the reference answer as erroneous. Human review confirms 41 cases, yielding 82.0% precision (Table 5). Although less precise than the LLM answer verifier, the teacher analysis provides an effective additional filter for identifying reference-side noise.

Overall, these results support the multi-stage verification procedure used in constructing RECOMATH. LLM verification corrects many false negatives introduced by symbolic matching, while teacher-based consistency checking further removes candidates affected by unreliable reference answers.

A.2 EXAMPLES OF SYMBOLIC VERIFICATION FAILURES

Symbolic answer matching can fail even when the generated answer is semantically equivalent to the reference answer. Table 6 shows representative cases observed during benchmark construction. These failures mainly arise from equivalent algebraic forms, alternative representations of solution sets, and differences in multiple-choice answer formatting.

Table 6: Representative examples where SymPy rejects a generated answer despite semantic equivalence with the reference answer.

Reference Answer	**Generated Answer**	**Failure Type**
4^n	2^{2n}	Equivalent algebraic expression
≤ -1	$(-\infty, -1]$	Equivalent solution-set notation
(C) 126	C	Multiple-choice formatting

For example, 4^n and 2^{2n} are algebraically identical, while ≤ -1 and $(-\infty, -1]$ describe the same solution set. In the multiple-choice example, the generated answer provides only the option label `C`, whereas the reference includes both the label and its corresponding value, `(C) 126`. Such cases illustrate why exact or symbolic matching alone is insufficient for reliable answer verification and motivate the additional LLM-based verification stage used in our pipeline.

A.3 REASONING STEP SEGMENTATION

To localize the first erroneous reasoning step, we first segment each model-generated reasoning trace into a sequence of intermediate reasoning steps. Following the model-aware segmentation principle of Li et al. (2026), we use different boundary rules according to the formatting characteristics of each generator model.

For Qwen3-8B and DeepSeek-R1-Distill-Llama-8B, reasoning steps are identified using discourse-level transition markers that typically indicate a change, revision, or alternative direction in the reasoning process. We use markers such as `Wait`, `But`, `Let me think`, `</think>`, and `Alternatively` as step-boundary cues. Such markers are particularly useful for long-form reasoning models whose outputs are not consistently organized into explicitly separated lines. This follows the discourse-marker-based segmentation strategy of prior work (Li et al., 2026).

For Phi4-14B and Llama3.1-8B-Instruct, the generated reasoning is more consistently separated into line-level reasoning units. We therefore split their trajectories directly at newline characters (`\n`), treating each non-empty segment as an individual reasoning step. Using model-specific segmentation avoids imposing the same formatting assumption on substantially different reasoning styles.

After segmentation, each trajectory is represented as an ordered sequence

$$(r_1, r_2, \ldots, r_T).$$

The problem together with this step-level trajectory is then provided to the teacher model for first-error annotation. If r_k is identified as the earliest erroneous step, we retain

$$r_{1:k} = (r_1, \ldots, r_k)$$

as the benchmark prefix, thereby including the erroneous step itself in the model input.

A.4 PROCESSBENCH PREPROCESSING

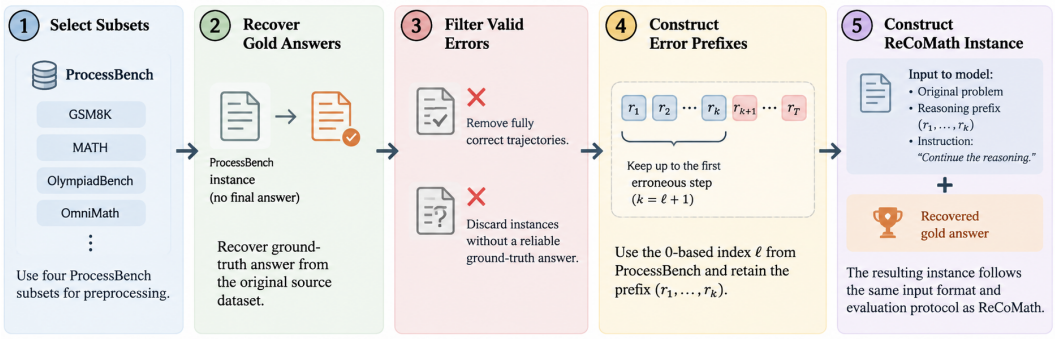

Figure 9: Preprocessing pipeline for incorporating ProcessBench into RECOMATH. We recover gold answers from the original source datasets, filter invalid or fully correct trajectories, construct prefixes up to and including the annotated first erroneous step, and convert the retained examples into the same unhinted recovery format used in RECOMATH.

ProcessBench (Zheng et al., 2025) provides step-level reasoning trajectories together with annotations of the earliest erroneous step, but its released instances do not directly provide the gold final answers required by our recovery-based evaluation. We therefore perform additional preprocessing before incorporating ProcessBench into RECOMATH.

Recovering Gold Answers. We use four ProcessBench subsets: GSM8K, MATH, OlympiadBench, and OmniMath. For each instance, we recover the corresponding ground-truth answer from its original source dataset. This step is necessary because RECOMATH evaluates whether a model can recover from an erroneous reasoning state and eventually reach the correct final answer, rather than only whether it can identify the erroneous step.

Filtering Valid Error Trajectories. We retain only instances that correspond to genuine erroneous reasoning trajectories. In particular, examples whose reasoning process is annotated as fully correct are removed, together with instances for which a reliable reference answer cannot be recovered. This filtering ensures that each retained example contains both a valid first-error annotation and a ground-truth answer for final recovery evaluation.

Constructing Error Prefixes. ProcessBench represents the location of the earliest erroneous step using a 0-based index. For an instance with reasoning steps $(r_1, \ldots, r_T)$ and ProcessBench label ℓ, we therefore define the first erroneous step as

$$k = \ell + 1.$$

Following the task definition of RECOMATH, we retain the prefix up to and including this step,

$$r_{\text{prefix}} = (r_1, \ldots, r_k).$$

The model is then given the original problem together with this prefix and is instructed only to *"Continue the reasoning from where it left off."* No information about the presence or location of the error is provided.

After preprocessing, all retained ProcessBench examples follow the same input format and evaluation protocol as our cross-model constructed data. This yields 2,212 ProcessBench-derived instances in the final RECOMATH benchmark, complementing the model-generated errors with additional failures from existing mathematical reasoning datasets.

A.5 BENCHMARK CONSTRUCTION ALGORITHM

Algorithm 1 summarizes the construction of RECOMATH.

Algorithm 1 Construction of RECOMATH

Require: Problem source $\mathcal{D}_{\text{src}}$; generator models $\mathcal{M}$; symbolic verifier V_{sym}; LLM verifier V_{llm}; teacher model T; ProcessBench subsets $\mathcal{D}_{\text{PB}}$
Ensure: RECOMATH benchmark $\mathcal{D}_{\text{ReCoMath}}$

$\mathcal{D}_{\text{cross}} \leftarrow \emptyset$
$\mathcal{D}_{\text{proc}} \leftarrow \emptyset$
for all $(x, y^*) \in \mathcal{D}_{\text{src}}$ **do**
  for all $M \in \mathcal{M}$ **do**
    $(R, \hat{y}) \leftarrow M(x)$ ▷ Generate reasoning and final answer
    if $V_{\text{sym}}(\hat{y}, y^*) = 1$ **then**
      continue
    end if
    if $V_{\text{llm}}(\hat{y}, y^*) = 1$ **then**
      continue
    end if
    $(r_1, \ldots, r_n) \leftarrow \textsc{SplitSteps}(R, M)$
    $(k, \tilde{y}) \leftarrow T(x, r_{1:n}, y^*)$ ▷ Locate first error and solve independently
    if k is invalid **then**
      continue
    end if
    if $\tilde{y} \not\equiv y^*$ **then**
      continue ▷ Filter unreliable references
    end if
    $r_{\text{prefix}} \leftarrow (r_1, \ldots, r_k)$
    Add $(x, r_{\text{prefix}}, y^*, M)$ to $\mathcal{D}_{\text{cross}}$
  end for
end for

A.6 PROMPT TEMPLATES

We use fixed task instructions throughout benchmark construction and evaluation. Below we report the main prompt templates used for self-correction evaluation and standard mathematical problem solving.

Self-Correction Evaluation. For RECOMATH, the model receives the original problem together with the erroneous reasoning prefix and is instructed to continue without any explicit indication that an error is present. We use the following instruction:

```
Continue the reasoning from where it left off.
Finish solving the problem carefully, and put the final
answer
within \boxed{}.
```

The erroneous prefix is provided as part of the model input immediately before this instruction. Importantly, the prompt does not mention the presence or location of an error.

Mathematical Problem-Solving Evaluation. For standard problem-solving benchmarks, where the model solves each problem from scratch, we use:

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

Using separate continuation and direct-solving prompts preserves the natural distinction between the two evaluation settings while standardizing the final answer format.

B RECOMATH STATISTICS AND ADDITIONAL ANALYSIS

B.1 SOURCE-LEVEL STATISTICS

RECOMATH contains 4,857 instances, including 2,645 newly constructed cross-model errors and 2,212 ProcessBench-derived examples. Table 7 provides the source-level breakdown.

Table 7: Source-level composition of RECOMATH. Error Step denotes the average position of the earliest erroneous step.

Source	# Inst.	Error Step	Split
Cross-model constructed			
DeepSeek-R1-Distill-Llama-8B	671	7.97	Easy
Llama3.1-8B-Instruct	908	5.54	Easy
Qwen3-8B	386	11.33	Hard
Phi4-14B	680	13.26	Hard
ProcessBench-derived			
GSM8K	207	2.86	Easy
MATH	594	3.25	Easy
OlympiadBench	653	4.37	Hard
OmniMath	758	3.96	Hard
Total	**4,857**	**6.62**	–

The fixed Easy/Hard partition is defined by data source rather than by the performance of any evaluated model. The Easy split contains errors from DeepSeek-R1-Distill-Llama-8B and Llama3.1-8B-Instruct together with ProcessBench GSM8K and MATH, while the Hard split contains errors from Qwen3-8B and Phi4-14B together with ProcessBench OlympiadBench and OmniMath. This yields 2,380 Easy and 2,477 Hard instances, with average first-error positions of 5.42 and 7.77, respectively.

B.2 SOURCE-LEVEL RECOVERY ANALYSIS

We further analyze self-correction performance on the cross-model constructed portion of RECOMATH by grouping examples according to the model that generated the erroneous reasoning trajectory. Figure 10 reports the resulting source-level recovery accuracy for the three evaluated backbones.

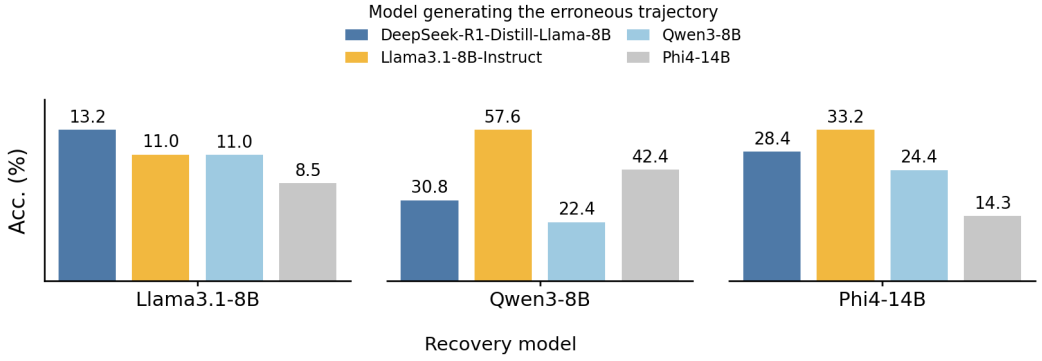

Figure 10: Source-level self-correction accuracy on the cross-model constructed portion of RECOMATH. The x-axis denotes the recovery model, while bar colors indicate the model that generated the erroneous trajectory. Bar heights are independently normalized within each recovery model for readability; numerical labels report the actual accuracy (%).

Table 8: Representative training configurations of prior methods related to self-reflection and self-correction. Reported scales follow the corresponding papers and may count different units of supervision. Unlike these methods, our controlled study fixes the data budget, backbone, and optimization recipe while varying only trajectory construction.

Method	Reported Scale	Training Supervision	Optimization	Additional Signal
STaR / RFT	Task-dependent	Correct self-generated reasoning trajectories retained by outcome filtering	Iterative SFT	Gold-answer correctness
RefAug	15K	Correct solutions augmented with reflective reasoning	SFT	GPT-generated reflection
LEMMA	88.9K	Error-containing trajectories followed by correction	SFT	Teacher-generated errors / corrections
InT	1,076 SFT examples	On-policy correct prefix with a targeted intervention replacing the first erroneous step	SFT $\rightarrow$ RL	Reference solutions + self-proposed interventions
SuperCorrect	100K SFT + 10K preference	Hierarchical thought templates and paired self-/cross-model correction traces	SFT $\rightarrow$ DPO	Strong teacher models
SCoRe	$\sim$12K math problems	On-policy, self-generated multi-turn correction trajectories	Two-stage multi-turn RL	Outcome reward + reward shaping
S^2R	3.1K SFT initialization	Iterative self-verification and self-correction trajectories	SFT $\rightarrow$ outcome/process RL	Outcome- and process-level rewards
Self-Rewarding Correction	32K IFT trajectories	Self-generated trajectories containing self-evaluation and correction	IFT $\rightarrow$ DPO/PPO	Self-rewarding + correctness signals
Our Controlled Study	**15K per variant**	Correct-only, error-removed, reflection-retained, or error-retained trajectories	**Unified LoRA SFT**	**Trajectory form only**

Recovery performance varies substantially across error sources. For example, Qwen3-8B achieves 57.62% accuracy on errors generated by Llama3.1-8B-Instruct, but only 22.44% on Qwen3-8B-generated errors. Similarly, Phi4-14B reaches 33.22% on Llama3.1-8B-Instruct-generated errors, while recovering only 14.26% of Phi4-14B-generated errors.

Notably, errors generated by the same backbone are not consistently easier to correct. Llama3.1-8B-Instruct, Qwen3-8B, and Phi4-14B achieve 11.04%, 22.44%, and 14.26%, respectively, on their generator-matched subsets, none of which is the best-performing source for that recovery model. These results suggest that self-correction difficulty depends strongly on the underlying error distribution rather than simply on whether the recovery model matches the model that produced the erroneous trajectory.

C Comparison with Prior Training Methods

Existing approaches for improving self-correction differ along several dimensions beyond the form of the training trajectory, including training-data scale, teacher supervision, and optimization objective. Table 8 summarizes representative configurations reported in prior work. The reported data sizes should be interpreted as approximate training scales rather than directly comparable budgets, since different studies count problems, trajectories, preference pairs, or online RL prompts differently.

These methods therefore answer a different question from our controlled study. Their performance reflects the combined effects of trajectory construction, data scale, teacher supervision, and optimization strategy. Our goal is instead to isolate the effect of trajectory construction itself: all variants use the same 15K-example budget and the same SFT recipe for each backbone, with only the supervision trajectory changed.

D Additional Behavioral Analysis

D.1 Generation Length Analysis

We further examine whether retaining reflection or repair processes during training changes the amount of reasoning generated at inference time. Figure 11 reports the average number of generated tokens across ReCoMath and the mathematical problem-solving benchmarks for the main training variants.

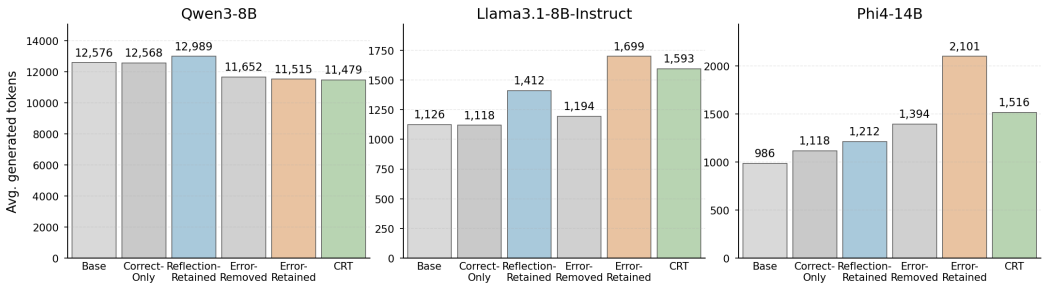

Figure 11: Average generation length across training variants. Each panel corresponds to one backbone and uses an independent vertical scale. Reflection-Retained, Error-Retained, and CRT preserve intermediate reflection or repair behavior to different degrees.

For Llama3.1-8B-Instruct and Phi4-14B, training with retained reflection or repair generally leads to longer generations. On Llama3.1-8B-Instruct, Reflection-Retained increases the average output length from 1,118 tokens under Correct-Only to 1,412, while Error-Retained further increases it to 1,699. A similar pattern is more pronounced on Phi4-14B, where Error-Retained reaches 2,101 tokens compared with 1,118 for Correct-Only. CRT also produces longer generations than Correct-Only on both backbones.

In contrast, this pattern is much weaker on Qwen3-8B. Reflection-Retained increases the average length only modestly, from 12,568 to 12,989 tokens, while Error-Retained and CRT do not increase generation length. This suggests that retaining reflective behavior can encourage additional reasoning for models that otherwise generate relatively short solutions, whereas its effect on reasoning length is limited for a backbone that already exhibits long-form reasoning behavior. More importantly, the improvements from reflection- and repair-based supervision cannot be explained simply by uniformly increasing the number of generated tokens.