Failure Profiling and Reachable Trajectory Selection for Reasoning Distillation

Yijun Zhu[1] **Jianxin Wang**[1] **Chengchao Shen**[1,✉]
[1]School of Computer Science and Engineering, Central South University
zhuyij@csu.edu.cn, jxwang@mail.csu.edu.cn, scc.cs@csu.edu.cn

Abstract

Training smaller language models on reasoning solutions generated by a larger teacher model is the prevailing approach for transferring mathematical reasoning capabilities. However, the student tends to overfit the output distribution of the teacher and generalizes poorly when its own behavior at inference time deviates from the training data. Recent methods attempt to bridge this gap by incorporating error data, yet the corrective trajectories still reflect the reasoning patterns of the teacher, which the student cannot reliably reproduce, leaving the distribution mismatch largely unresolved. We propose a method, Failure Profiling and Reachable Trajectory Selection (FPRT), built on the insight that effective training data should be jointly determined by what the student fails on and what it can realistically learn. Our method first profiles the failure behavior of the student on each problem through repeated sampling, characterizing it in terms of error rate and error consistency to reveal qualitatively distinct failure modes. Each mode demands a fundamentally different training signal. The diagnosed mode then guides candidate generation from the teacher via distinct prompting strategies tailored to each failure type. Among the correct candidates, our method selects the one whose initial reasoning steps after the point of divergence are most reproducible by the student. These first corrective steps constitute the primary bottleneck, while the remaining steps follow an already corrected context and can therefore be continued more readily. This contrasts with standard reranking approaches, which score the trajectory as a whole and thereby dilute the signal at the critical reasoning transition. Experiments on GSM8K, MATH, and three out-of-distribution benchmarks for Llama-3.1-8B and DeepSeek-Math-7B demonstrate that our method achieves the SOTA performance across the five benchmarks on both student backbones. The source code and trained weights are available at https://github.com/visresearch/FPRT.

1 Introduction

Frontier large language models (LLMs), such as GPT-4 [1], DeepSeek-R1 [11], and Qwen2.5 [47], have demonstrated strong mathematical reasoning ability. In contrast, smaller models such as Llama-3.1-8B [9] and DeepSeek-Math-7B [36] still struggle with complex mathematical problems. A growing line of research seeks to bridge this gap by training smaller students on reasoning traces generated by more capable teachers [29, 15, 22, 50]. Despite substantial progress, the resulting student overfits the output distribution of the teacher: trained exclusively on correct demonstrations that mirror the reasoning style of the teacher, the student has no reliable way to recover once its behavior at inference time deviates from that distribution. The cost is especially high in mathematical reasoning, where each step depends on all preceding ones and even a single misstep invalidates the entire subsequent chain.

✉Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

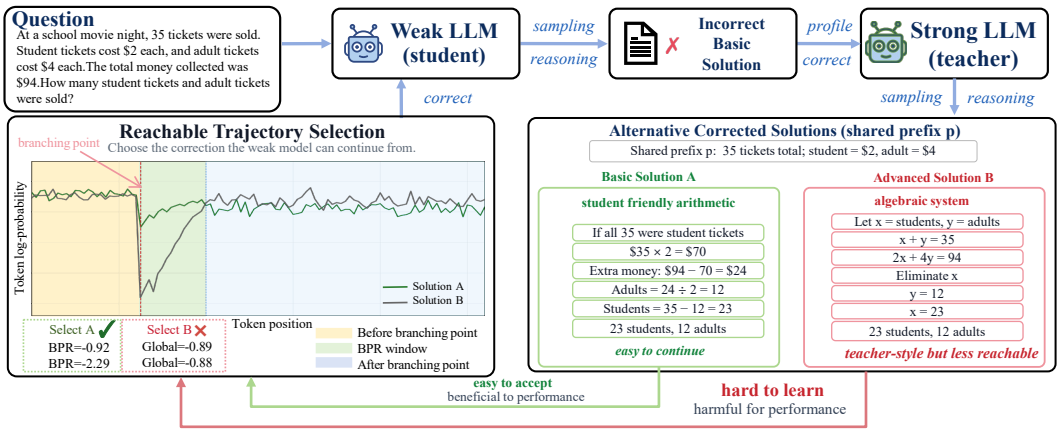


Figure 1: Due to the capability gap between a strong teacher and a weaker student, teacher-generated reasoning trajectories may be difficult for the student to reproduce and may hinder learning. We guide the teacher to generate correction candidates conditioned on the student's errors and select the candidate with the highest branching point reachability score. This score averages student token log-probabilities within a local window immediately after the first divergence, reducing dilution from later tokens.

Recent work has begun to incorporate error data into training to address this gap. LEMMA [32] pairs erroneous solutions with corrective reflections. Self-Correct [41] explores minimal edit corrections that fix mistakes generated by the model. Approaches based on MCTS systematically explore error nodes in the reasoning tree [44, 4]. These methods mark meaningful progress, yet they share two limitations. First, they do not condition on the behavioral failure mode of the student but instead apply the same correction strategy regardless of whether the model is confidently locked onto a wrong answer or producing a different error each time. Specifically, consistently arriving at the same wrong answer reflects a strong but misguided reasoning pattern, whereas producing diverse errors signals the absence of any stable strategy. For the above failure modes, a single correction strategy cannot efficiently fix student errors. Second, even when the failure mode is correctly identified, the corrective trajectories still come from a teacher and exhibit reasoning patterns that the student cannot reliably reproduce, a problem that prior work on knowledge transfer has shown to worsen as the capacity gap between teacher and student grows [31]. Such corrections risk reducing learning to superficial pattern matching rather than genuine skill acquisition.

To address the above limitations, we introduce a novel failure profiling and reachable trajectory selection framework for reasoning distillation. In the failure profiling stage, our method characterizes student errors in terms of error rate and error consistency from multiple sampled responses on each problem to identify the underlying failure mode. Then, we elicit targeted correction candidates from an advanced teacher using prompts tailored to each diagnosed mode. When the student is locked onto a single wrong answer, the prompt directs the teacher to construct corrections that break the specific misconception, whereas when the student produces diverse errors, the prompt instead requests structured step by step reasoning to supply the missing problem solving framework.

In the reachable trajectory selection stage, we select the most reproducible candidate trajectory produced by the teacher to correct the student's erroneous reasoning steps, thereby improving the student's reasoning ability. Unlike standard reranking approaches [57] that score the trajectory as a whole, our method evaluates compatibility within a local window immediately after the position where the reasoning of the student first diverged, as shown in Figure 1. We find that the above local window scoring can clearly distinguish the student-friendly correction from other candidates, where later tokens conditioned on a corrected context will not dilute the signal at the critical reasoning transition. Once the student successfully executes the transition away from its error, the remaining steps follow an already corrected context and can therefore be continued more readily.

Overall, our contributions can be summarized as follows:

1. We introduce behavioral profiling based on error rate and error consistency to classify model errors into distinct failure modes and enable targeted correction strategies for qualitatively different error patterns.

2. We propose branching point reachability scoring, a trajectory selection mechanism that evaluates the log-probability under the student within a local window immediately following the point of divergence rather than over the full response, thereby guiding the student to focus on the critical reasoning transition where the student makes mistakes.
3. Extensive experiments on Llama-3.1-8B and DeepSeek-Math-7B across the mathematical reasoning benchmarks GSM8K and MATH, together with the out-of-distribution evaluation sets SVAMP, ASDiv, and MAWPS, demonstrate that our method achieves the highest average accuracy across the five benchmarks on both student backbones.

2 Related Work

Training on Teacher-Generated Reasoning Traces. A dominant approach to improving mathematical reasoning in smaller models is fine-tuning on reasoning traces from more capable models [50, 28, 52, 53, 26, 22]. Rejection sampling fine-tuning [51] trains on correct paths sampled from the student, and DART-Math [39] rebalances this corpus toward harder queries. LLM2LLM [21] iteratively synthesizes new data targeting problems on which the student fails. A separate line tackles the distribution mismatch between teacher data and student capability. GKD [2] trains the student on self-generated sequences with teacher feedback, and Speculative Knowledge Distillation [45] interleaves student and teacher sampling. UNDO [17] iteratively prompts the teacher for rationales targeted at student errors. Local Naturalness [18] ranks responses using student likelihood at the segment level. GRAPE [54] selects the response with the highest normalized probability under the target model. This instruction-tuning method is distinct from GRAPE for robot-policy preference alignment [57]. Rank-Surprisal Ratio [49] combines token rank and surprisal to account for both alignment and informativeness. BPR instead restricts scoring to the initial corrective steps after the student's first divergence, rather than scoring the full trajectory. Our pipeline additionally uses behavioral failure profiling to guide candidate generation.

Error Aware Data Construction. A growing body of work recognizes that incorrect solutions themselves contain valuable learning signals. LeMa [3], LEMMA [32], and Tong et al. [38] pair incorrect reasoning with teacher-generated corrections or error-type annotations. TrajFusion [7] interleaves incorrect trajectories with reflection prompts and correct trajectories. Process supervision approaches [25, 40, 34, 10, 43] score intermediate steps or explore error nodes in reasoning trees. Another line [55, 12, 46] collects erroneous trajectories by sampling at elevated temperatures, which tends to produce nonsensical errors unlikely to arise during normal generation [27, 35]. Across these methods, errors are treated uniformly regardless of the failure pattern, and corrective trajectories follow teacher patterns the student cannot reliably reproduce. Our method addresses both through behavioral profiling that classifies errors into distinct failure modes and branching point reachability scoring that favors corrections more likely to be learnable by the student.

Self-Correction in LLMs. One line of work uses external feedback from verifiers or critic models for self-correction [24, 38, 37, 23, 8]. GLoRe [13] trains global and local refinement models guided by stepwise outcome reward models. Another line focuses on self-correction without external feedback [55, 12, 46, 42, 27], although Huang et al. [16] demonstrate that LLMs cannot reliably self-correct their reasoning without external input. On the training side, SCoRe [20] proposes multi-turn online RL for self-correction and shows that SFT on offline correction traces fails due to distribution mismatch. SuperCorrect [48] uses cross-model collaborative DPO to optimize student self-correction with teacher thought templates. Rather than training the model to correct errors at inference time through RL or DPO, our method constructs corrective training data filtered by branching point reachability under the student, thereby mitigating the distribution mismatch identified by SCoRe while requiring only standard SFT.

3 Method

3.1 Problem Formulation

Let π_T and π_S denote the token-level conditional distributions of the teacher and student models, respectively, and let q denote a mathematical problem. Standard training on teacher-generated traces

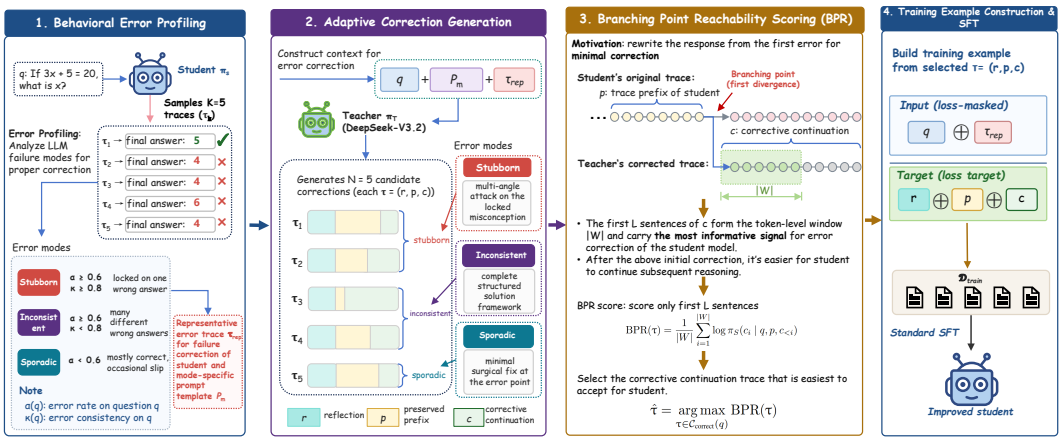


Figure 2: Overview of our pipeline. Stage 1 profiles the student via repeated sampling and classifies each problem as stubborn, inconsistent, or sporadic from error rate α and consistency κ. Stage 2 prompts the teacher with a mode-specific template P_m to generate N candidates, each decomposed into reflection r, preserved prefix p, and corrective continuation c. The three mode-specific branches in Stage 2 illustrate alternative prompting strategies. For each problem, all N candidates are generated using only the prompt corresponding to its diagnosed mode. The sampled-answer example in Stage 1 corresponds to the inconsistent mode. Stage 3 filters by correctness and selects the candidate maximizing branching point reachability over the first L sentences of c.

optimizes the parameters θ of the student by minimizing

$$\mathcal{L}_{\text{SFT}}(\theta) = -\mathbb{E}_{(q,\tau)\sim\mathcal{D}_{\text{train}}}\left[\sum_{t=1}^{\ell} \log \pi_S(s_t \mid q, s_{<t})\right], \tag{1}$$

where $\mathcal{D}_{\text{train}}$ is a dataset of correct reasoning traces generated by the teacher, each trace $\tau = (s_1, \ldots, s_\ell)$ is a sequence of ℓ tokens, and $s_{<t} = (s_1, \ldots, s_{t-1})$ denotes all tokens preceding position t. However, reasoning traces exhibit strong path dependence: the student's ability to reproduce a step s_t depends on the preceding context $(q, s_{<t})$ and on whether the student can reliably generate that context. When the reasoning style of the teacher differs from that of the student, per-step deviations accumulate over long chains, causing the student to encounter states at inference time that were never seen during training. This motivates a shift in the data construction objective: rather than seeking trajectories that are merely correct, we seek trajectories that are simultaneously correct and reachable by the student. Here, *correct* means that the trace leads to the right answer, while *reachable* means that the trace lies within the high-probability region of the distribution of the student, so that each step is a plausible continuation that the student could generate on its own. Let $\mathcal{C}_{\text{correct}}(q)$ denote the set of all correct reasoning traces for a given problem. The ideal selection objective is:

$$\hat{\tau} = \underset{\tau\in\mathcal{C}_{\text{correct}}(q)}{\arg\max}\ \pi_S(\tau \mid q). \tag{2}$$

Our method approximates this objective through a generate-then-select pipeline illustrated in Figure 2, where the teacher proposes candidates and the distribution of the student itself serves as the selection criterion.

3.2 Stage 1: Behavioral Error Profiling

Not all errors are alike: different failure modes demand qualitatively different corrections. Given a dataset $\mathcal{D} = \{(q_i, y_i)\}_{i=1}^{M}$ of M problems with ground-truth answers y, we draw K reasoning traces $\tau_1, \ldots, \tau_K$ from π_S for each problem q and compute two diagnostic statistics.

Error rate. The fraction of the K samples that produce an incorrect final answer:

$$\alpha(q) = \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}\left[\text{answer}(\tau_k) \neq y\right], \tag{3}$$

where $\text{answer}(\tau_k)$ extracts the final answer from the k-th sampled trace.

Table 1: Error mode classification based on (α, κ) and the corresponding correction strategy.

Mode	α	κ	Correction Strategy
Stubborn	≥ 0.6	≥ 0.8	Multi-angle attack on the locked error path
Inconsistent	≥ 0.6	< 0.8	Complete structured solution framework
Sporadic	$0 < \alpha < 0.6$	–	Minimal surgical fix at the error point

Error consistency. Among the incorrect samples, the fraction accounted for by the most frequent wrong answer e:

$$\kappa(q) = \max_{e \neq y} \frac{\sum_k \mathbf{1}[\text{answer}(\tau_k) = e]}{\sum_k \mathbf{1}[\text{answer}(\tau_k) \neq y]}, \tag{4}$$

where the maximization ranges over all incorrect answers e that appear in the K samples. A high κ indicates that the errors of the student concentrate on a single wrong answer, while a low κ indicates scattered, diverse failures.

Problems with $\alpha(q) = 0$ yield no error trajectory to diagnose and are excluded from the corrective pipeline, so the remainder of this section restricts attention to problems with $\alpha(q) > 0$. We do not add a separate pool of problems for which all sampled responses are correct. On this filtered set, (α, κ) partition errors into three distinct modes. Table 1 summarizes the classification criteria and the corresponding correction strategy. This two-dimensional characterization moves beyond existing error taxonomies that classify errors by semantic type, such as calculation error or formula confusion. Our profiling instead captures the behavioral pattern of errors from the perspective of the student, which directly informs how corrections should be structured.

For each profiled problem, we additionally retain a single representative error trajectory τ_{rep}, which serves as the concrete instance of failure to be corrected by the teacher. For stubborn errors, we select τ_{rep} from the incorrect traces whose final answer matches the most frequent wrong answer. For inconsistent and sporadic errors, we randomly select an incorrect trace. After profiling, we obtain the error-annotated dataset:

$$\mathcal{D}_{\text{prof}} = \{(q, y, \{\tau_k\}_{k=1}^K, m, \tau_{\text{rep}}) \mid (q, y) \in \mathcal{D},\ \alpha(q) > 0\}, \tag{5}$$

where $m \in \{\texttt{stubborn}, \texttt{inconsistent}, \texttt{sporadic}\}$ denotes the error mode diagnosed using the criteria in Table 1.

3.3 Stage 2: Adaptive Correction Generation

Given the diagnosed error mode m, the teacher π_T generates N correction candidates using a mode-specific prompt P_m, the problem q, and the representative error trajectory τ_{rep}. For inconsistent errors, `student_reasoning_attempts` is a singleton list containing τ_{rep}. The templates are provided in Appendix D. Each prompt requests N *different* trajectories in one structured call, using mode-specific diversification instructions.

The key design principles differ across modes. For stubborn errors, the prompt requests N trajectories that attack the locked-in reasoning path from different angles, such as a direct fix, an alternative method, and a conceptual reframe. It explicitly asks the teacher to analyze the trap, conceptual pivot, fix, and recurrence-prevention check, and to distill this analysis into a concise first-person reflection. For inconsistent errors, the diverse incorrect answers suggest a lack of a stable problem-solving strategy, so the prompt requests N structured solutions that supply different problem-solving frameworks, such as algebraic equations, step-by-step arithmetic, and work-backwards reasoning. For sporadic errors, the prompt instructs maximal preservation of the correct prefix of the student with minimal modification at the identified error point, and the N trajectories vary only in their local correction style.

Each candidate consists of three parts: a reflection r, a preserved prefix p, and a corrective continuation c. The reflection r is a concise first-person diagnosis that names where the reasoning of the student first went wrong. The preserved prefix p is the verbatim portion of τ_{rep} up to but excluding the divergent step, and is empty when no correct prefix can be retained. With the singleton input used in our experiments, the prefix is the longest correct prefix of τ_{rep}, and the divergence point is its first erroneous step. The corrective continuation c extends p to a correct final answer in a style natural to the student, or starts from scratch when p is empty. Both the localization that yields p and the

diversification across the N candidates are performed jointly by the teacher in a single structured call. Formally, the candidate set is:

$$\mathcal{C}(q) = \{(r_n, p_n, c_n)\}_{n=1}^{N} \sim \pi_T(\cdot \mid P_m, q, \tau_{\text{rep}}). \tag{6}$$

3.4 Stage 3: Branching Point Reachability Scoring

Among the N candidates, we select the one that the student model is most likely to reproduce.

Branching point reachability scoring. We first discard candidates whose final answer does not match the ground truth y, and let $\mathcal{C}_{\text{correct}}(q) \subseteq \mathcal{C}(q)$ denote the resulting filtered set. For each remaining candidate $\tau = (r, p, c) \in \mathcal{C}_{\text{correct}}(q)$, we extract a local window W covering the first L sentences of the corrective continuation c. BPR scores only c, since r is a short reflection used as a training scaffold and p is the output of the student. Let $|W|$ denote the number of tokens in this window, and let c_i denote the i-th token of c, with $c_{<i} = (c_1, \ldots, c_{i-1})$ representing all preceding tokens within the continuation. The branching point reachability (BPR) score is the average log-probability of the window tokens under the student model, conditioned on the problem q and the preserved prefix p:

$$\text{BPR}(\tau) = \frac{1}{|W|} \sum_{i=1}^{|W|} \log \pi_S(c_i \mid q, p, c_{<i}). \tag{7}$$

This score restricts the log-probability computation to the local window immediately following the point where the reasoning of the student diverged. The rationale is that tokens later in the corrective continuation benefit from an already corrected context and receive systematically higher log-probabilities regardless of which candidate generated them. By focusing on the initial corrective steps, BPR isolates the signal that most directly reflects whether the student can execute the critical reasoning transition. The selected trajectory is:

$$\hat{\tau} = \underset{\tau \in \mathcal{C}_{\text{correct}}(q)}{\arg\max}\ \text{BPR}(\tau). \tag{8}$$

In practice, we set $L = 8$ sentences and extract sentence boundaries using newline delimiters, which are common in step by step mathematical reasoning, with a fallback to punctuation-based splitting. The BPR score is computed via a single forward pass through π_S with no gradient computation.

Connection to the ideal objective. The ideal selection criterion in Eq. 2 maximizes $\pi_S(\tau \mid q)$, or equivalently the chain-rule decomposition $\log \pi_S(\tau \mid q) = \sum_{i=1}^{|\tau|} \log \pi_S(\tau_i \mid q, \tau_{<i})$. BPR is a deliberately biased proxy that departs from this objective in two complementary ways. The restriction to a local window W rather than the full continuation is justified above by the dilution argument and confirmed empirically in Figure 7 of Appendix G. The other divergence is that BPR averages rather than sums: dividing by $|W|$ removes the length penalty that would otherwise systematically favor shorter candidates regardless of their per-token compatibility with the student. BPR is therefore not a calibrated estimator of $\log \pi_S(\tau \mid q)$, but instead concentrates the scoring signal on the segment that empirically governs whether the student can execute the critical reasoning transition, which is the property that ultimately matters for selection.

3.5 Training

Within each construction pass, the three stages select at most one training trajectory per retained question. The final SFT corpus uses two input-target formats. For 90% of the examples, the input is $q \oplus \tau_{\text{rep}}$ and the target is $r \oplus p \oplus c$. For the remaining 10%, the input is q alone and the target is the complete corrected solution $p \oplus c$, without the reflection or erroneous trace. These auxiliary examples come from the same $\alpha(q) > 0$ problem pool. Algorithm 1 in Appendix A summarizes the construction stages.

Table 2: Accuracy (%) on in-distribution and out-of-distribution benchmarks. Best results are in **bold** and second-best are underlined. The 88.90K Ours rows report mean ± sample standard deviation across three SFT seeds. Avg statistics are computed across per-seed five-benchmark averages.

			In-Distribution		Out-of-Distribution			
Student Model	**Method**	**# Samples**	GSM8K	MATH	SVAMP	ASDiv	MAWPS	**Avg.**
Llama-3.1-8B	SFT	394.99K	80.3	39.7	77.5	82.4	89.2	73.8
	RFT [51]	86.52K	76.4	40.6	78.6	78.8	83.2	71.5
	ISC [12]	86.78K	72.5	43.5	80.4	81.7	82.7	72.2
	S^3C-Math [46]	927K	83.1	42.8	82.4	–	–	–
	GPTAug	88.62K	73.3	41.6	81.0	81.8	86.3	72.8
	RefAug [56]	29.94K	77.3	42.3	81.7	82.4	88.7	74.5
	LEMMA [32]	88.90K	80.4	**48.8**	83.7	84.5	88.9	77.3
	FPRT (Ours)	62.74K	85.5	44.3	86.4	84.9	**93.9**	79.0
	FPRT (Ours)	88.90K	**86.2±0.4**	46.2±0.5	**86.8±0.2**	**86.1±0.3**	93.3±0.4	**79.7±0.4**
DeepSeek-Math-7B	SFT	394.99K	79.4	42.0	79.4	87.8	90.2	75.8
	RFT	86.52K	77.5	43.9	77.4	86.9	89.6	75.1
	ISC	86.78K	72.2	39.5	75.6	84.4	89.8	72.3
	S^3C-Math	927K	82.8	42.6	83.1	–	–	–
	GPTAug	88.62K	78.6	46.0	74.2	88.9	89.9	75.5
	RefAug	29.94K	77.0	39.8	73.7	83.8	85.3	71.9
	LEMMA	88.90K	81.3	**50.7**	81.7	89.4	91.1	78.8
	FPRT (Ours)	65.76K	83.6	47.1	83.9	**90.1**	**95.3**	80.0
	FPRT (Ours)	88.90K	**84.0±0.2**	48.9±0.3	**85.4±0.2**	89.8±0.4	94.7±0.3	**80.6±0.3**

4 Experiments

4.1 Experimental Setup

We use DeepSeek-V3.2 [6] as teacher and Llama-3.1-8B-Instruct [9] and DeepSeek-Math-7B-Instruct [36] as students. Each construction pass profiles all 394.99K MetaMathQA instances [50] with $K = 5$ student samples and generates $N = 5$ teacher candidates only for instances with $\alpha(q) > 0$, as detailed in Appendix A. The complete first pass yields 62.74K and 65.76K examples for Llama and DeepSeek. A fresh full second pass is run. Part of its output is merged, prioritizing previously unseen questions until reaching 88.90K. SFT, RFT, S^3C-Math, and Ours use MetaMathQA; GPTAug, RefAug, ISC, and LEMMA construct supervision from GSM8K and MATH source problems using their respective recipes. MetaMathQA is also derived from GSM8K and MATH. Sample counts and filtering follow each method's construction, as detailed in Appendix C. Training settings and evaluation benchmarks are detailed in Appendix B.

4.2 Main Results

Consistent Average Lead across Two Student Backbones. Table 2 reports accuracy on the two student models. Our method attains the highest average on both backbones, 79.7% on Llama-3.1-8B and 80.6% on DeepSeek-Math-7B, exceeding the strongest comparison method, LEMMA, by 2.4% and 1.8%, respectively. This lead persists at the smaller 62.74K and 65.76K configurations, where the averages reach 79.0% and 80.0% and still surpass every comparison method with a reported five-benchmark average. The ranking is therefore stable across the two corpus sizes rather than tied to a single training budget. The improvement also holds across a general purpose and a math specialized backbone, indicating that the gain is not tied to any single pretraining distribution. The average-accuracy margins over RFT and ISC range from 5.5% to 8.3%.

Larger Margin under Distribution Shift. The improvement is concentrated on the three out-of-distribution benchmarks, where the averages of 88.7% on Llama-3.1-8B and 90.0% on DeepSeek-Math-7B exceed LEMMA by 3.0% and 2.6%, respectively. On DeepSeek-Math-7B, the in-distribution average rises only modestly from 66.0% for LEMMA to 66.5% for our method, indicating that most of the 1.8% overall lead originates from out-of-distribution gains. MAWPS in particular shows leads over LEMMA of 4.4% and 3.6% on the two student models. This pattern is consistent

Table 3: Accuracy (%) of supervision controls on the same first-pass problem sets, with one correct training trajectory per problem. Random-one-correct and BPR adopt the same correct candidate pools. Direct teacher-trace SFT generates solutions from the question alone. Best accuracy within each student is bold.

Student Model	Method	GSM8K	MATH	SVAMP	ASDiv	MAWPS	Avg.
Llama-3.1-8B	Direct teacher-trace SFT	84.0	42.7	82.0	82.1	88.5	75.9
	Random-one-correct	81.6	42.1	84.7	81.2	93.2	76.6
	BPR selection (Ours)	**85.5**	**44.3**	**86.4**	**84.9**	**93.9**	**79.0**
DeepSeek-Math-7B	Direct teacher-trace SFT	82.6	46.5	81.5	89.2	90.7	78.1
	Random-one-correct	83.0	46.8	82.6	89.6	92.8	79.0
	BPR selection (Ours)	**83.6**	**47.1**	**83.9**	**90.1**	**95.3**	**80.0**

with the hypothesis that anchoring corrections at the first divergence yields trajectories that transfer beyond MetaMathQA style templates.

Trade Off on MATH. On MATH, our method trails LEMMA by 2.6% on Llama-3.1-8B and 1.8% on DeepSeek-Math-7B, while ranking second on both backbones. One possible explanation is that LEMMA's error-type grounded mistake augmentation provides supervision better suited to MATH. Correctness filtering may also limit the available teacher candidates on difficult problems. These hypotheses require further analysis of error-type coverage and candidate retention rates; the current results do not isolate candidate construction from candidate selection as the source of the gap.

Favorable Sample Efficiency. We evaluate our method on two different training corpus sizes with the same pipeline. The 62.74K and 65.76K configurations use the complete first-pass corpora and are subsets of the corresponding 88.90K corpora assembled from two passes. They are used for the ablation and window size studies in Section 4.3. At these smaller subsets, roughly 70% of the LEMMA corpus and about one sixth and one fifteenth of the SFT and S^3C-Math corpora, our method already exceeds every comparison method with a reported five-benchmark average, surpassing LEMMA by 1.7% and 1.2%. Scaling to 88.90K, comparable to the 88.90K of LEMMA, lifts the margin to 2.4% and 1.8%, so the lead does not collapse once both methods operate at the same data budget.

4.3 Ablation Studies

Supervision controls. Table 3 compares two controls on the same first-pass problem sets, with teacher, student, example count, checker, and SFT settings matched to Ours. Direct teacher-trace SFT uses one verified-correct DeepSeek-V3.2 solution per question, generated from the question alone and resampled if incorrect. Our pipeline outperforms this control by 3.1% on Llama and 1.9% on DeepSeek, evaluating the full supervision-construction pipeline.

Random-one-correct follows the same profiling, candidate pools, and correctness filtering as Ours, but uniformly selects one correct candidate per retained problem without scoring. It achieves 76.6% and 79.0%, compared with BPR's 79.0% and 80.0% on Llama and DeepSeek, respectively.

Profiling Drives the Bulk of the Gain, with BPR Refining on Top. To isolate the contributions of behavioral profiling and branching point reachability scoring, we train three reduced variants of our pipeline against the full method, each removing one or both components while holding the rest of the training procedure fixed. These ablations use the Llama-3.1-8B first-pass configuration. The first replaces behavioral profiling with a uniform minimal-correction prompt template applied to all errors, preserving the longest correct prefix and changing only the erroneous reasoning, the second replaces BPR with global log-probability scoring over the full correction, and the third combines both ablations. Results across the five benchmarks are summarized in Table 4. Replacing profiling with a uniform prompt causes the largest single drop, from 79.0% to 72.9% on average, with the largest benchmark-specific drop on GSM8K (8.0%). Replacing BPR with global log-probability scoring reduces the average from 79.0% to 77.3%, a smaller decline in average accuracy that matches the dilution argument in Section 3.4 and Figure 7. MAWPS exhibits a slight reversal of 0.3% in favor of

Table 4: Accuracy (%) of component ablations across five benchmarks: GSM8K, MATH, SVAMP, ASDiv, and MAWPS. "Uniform prompting" replaces behavioral profiling with a single prompt template for all errors. "Global scoring" replaces BPR with global log-probability scoring over the full correction.

Variant	GSM8K	MATH	SVAMP	ASDiv	MAWPS	**Avg.**
Full method	**85.5**	**44.3**	**86.4**	**84.9**	93.9	**79.0**
w/o profiling (uniform prompting)	77.5	38.3	80.4	78.7	89.4	72.9
w/o BPR (global scoring)	82.3	42.5	85.6	81.9	**94.2**	77.3
w/o both (uniform + global)	75.3	37.6	79.8	79.6	89.8	72.4

global scoring, 94.2% against 93.9%. Removing both components yields 72.4%, only 0.5% below the variant that keeps BPR alone. BPR adds 1.7% when profiling supplies a tailored candidate pool, suggesting that its benefit depends on candidate construction. These results suggest complementary contributions from the two components: profiling determines the shape of the corrective trajectory, while BPR ranks the candidates within that pool by reachability under the student. The exact uniform prompt is in Appendix D.4.

Boundary localization. An audit of 50 traces from the 65.76K DeepSeek configuration finds 88% accurate first-error boundaries. Perturbing BPR boundaries on 12% of questions with preserved prefixes changes average accuracy from 80.0% to 79.7% in a single run. Appendix H.2 gives the audit and perturbation protocol.

Robust to Window Size with a Moderate Optimum. To probe the sensitivity of BPR to its only hyperparameter, we sweep the window size L across $\{2, 4, 8, 16, 32, \text{full}\}$ sentences while keeping the rest of the pipeline fixed, where the full setting recovers global scoring over the entire correction. The resulting average accuracy curve is plotted in Figure 3. Average accuracy peaks at 79.0% when $L = 8$, and we adopt this value as the default in all other experiments. The curve is markedly asymmetric around the optimum. For $L < 8$, accuracy is 77.1% and 76.7%, indicating that very short windows do not contain enough tokens to discriminate reachable from unreachable trajectories. For $L > 8$, accuracy decays monotonically, falling to 78.6% at $L = 16$, 78.2% at $L = 32$, and 77.3% at the global setting. This 1.7% drop matches the gap between BPR and global scoring reported in the component ablation. The decay to the right of the optimum is consistent with the dilution argument in Section 3.4 and Figure 7, since enlarging the window past the divergence transition admits tokens whose high probability, inherited from an already corrected context, obscures the discriminative signal at the branching point. The shape of the curve, with both very short and global windows lying below $L = 8$, identifies a moderate window as the regime where BPR captures the intended signal. This regime is broad enough to smooth local noise yet narrow enough to isolate the critical transition. A complementary token-level analysis of where the discriminative signal concentrates is provided in Figure 7 of Appendix G.

Data Scaling. To examine how the size of our constructed corpus translates into student accuracy, we scale the training set from 16.5K to 88.90K examples for both Llama-3.1-8B and DeepSeek-Math-7B and retrain each student under the same recipe. Intermediate sizes uniformly subsample the 88.90K corpus. Each configuration uses one SFT run with the same fixed seed; the 88.90K Ours rows in Table 2 instead report three-seed means. The resulting accuracy curves are plotted in Figure 4. Both student models improve monotonically with more data, yet their slopes differ in a manner that reflects their starting competence. Llama-3.1-8B climbs from 71.8% to 79.7%, a total gain of 7.9%. The gains at the final two plotted sizes are 0.3% and 0.4%, respectively. DeepSeek-Math-7B starts at 77.9% and reaches 80.6%, a smaller gain of 2.7%. The final two increments are 0.5% and 0.1%, respectively. The contrast suggests that the general purpose backbone benefits substantially from targeted corrective trajectories, which supply mathematical patterns under-represented in its pretraining. The math specialized backbone, by contrast, already encodes much of the relevant signal and extracts only incremental value from additional supervision. The flattening at the upper end of both curves indicates that the marginal utility of scaling our corpus diminishes once the dominant failure modes in the training distribution have been absorbed. Within the evaluated range, both curves show smaller marginal gains at the larger training sizes.

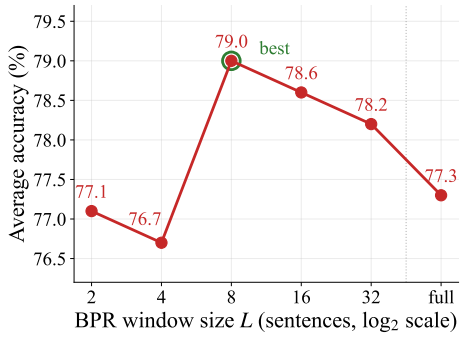


Figure 3: The effect of BPR window size L on average accuracy.

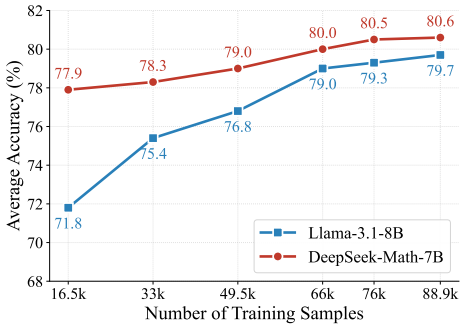


Figure 4: Data scaling: average accuracy across five benchmarks.

Seeds, teachers, and inference behavior. Table 2 reports three-seed results for each fixed 88.90K corpus (the original SFT seed and two additional seeds), with all other settings fixed. In a matched single-run comparison using the same fixed SFT seed, changing only the teacher to DeepSeek-V4-Flash gives 81.0% on DeepSeek, versus 80.6% with DeepSeek-V3.2. Full results and protocols are in Appendices H.3 and H.4.

5 Conclusion

We present a reasoning-distillation pipeline that combines behavioral failure profiling, mode-specific correction generation, and branching point reachability scoring. Profiling tailors supervision to how the student fails, while localized scoring selects correct candidates whose initial corrective steps are more reproducible by the student. Across GSM8K, MATH, and three out-of-distribution benchmarks, the method achieves the highest average accuracy on both Llama-3.1-8B and DeepSeek-Math-7B. Ablations support the contributions of profiling and localized scoring. The results highlight the value of jointly considering student failure patterns and corrective-trajectory reachability when constructing distillation data.

6 Limitations

Dependence on a strong teacher model. Our pipeline relies on a capable teacher π_T, instantiated as DeepSeek-V3.2 in the main experiments, both to generate correction candidates and to localize the divergent step in the error trajectory of the student. The quality and diversity of the constructed training data are therefore upper-bounded by the competence of the teacher on the target problem distribution. When the teacher itself fails on a problem, no usable candidate enters $\mathcal{C}_{\text{correct}}(q)$ and the problem is dropped from the final training set.

Sensitivity to the sampling budget K. Behavioral profiling estimates (α, κ) from $K = 5$ student samples per problem. With small K, both statistics are coarse-grained: α takes values in $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ and κ takes a similarly small discrete set. This grants robustness to threshold choice, as shown in Appendix F, but also limits the resolution of the diagnosed mode. Larger K would yield finer profiling at proportionally higher sampling cost, but we did not explore this trade-off systematically.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No.62302523) and Natural Science Foundation of Hunan Province (No.2026JJ50231). This work was carried out in part using computing resources at the HPC Center of Central South University.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations (ICLR)*, 2024.

[3] Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes LLM better reasoner. *arXiv preprint arXiv:2310.20689*, 2023.

[4] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. AlphaMath almost zero: Process supervision without process. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[5] 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. *arXiv preprint arXiv:2110.14168*, 2021.

[6] DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025.

[7] Jie Deng, Hanshuang Tong, Jun Li, Shining Liang, Ning Wu, Hongzhi Li, and Yutao Xie. Beyond rejection sampling: Trajectory fusion for scaling mathematical reasoning. *arXiv preprint arXiv:2602.04391*, 2026.

[8] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 11733–11763. PMLR, 2024.

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.

[11] Daya Guo, Dejian Yang, He Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[12] Haixia Han, Jiaqing Liang, Jie Shi, Qianyu He, and Yanghua Xiao. Small language model can self-correct. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18162–18170, 2024.

[13] Alexander Havrilla, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. GLoRe: When, where, and how to improve LLM reasoning via global and local refinements. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pages 17719–17733. PMLR, 2024.

[14] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[15] Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14852–14882, 2023.

[16] Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068, 2023.

[17] Kushal Jain, Piyushi Goyal, and Kumar Shridhar. UNDO: Understanding distillation as optimization. *arXiv preprint arXiv:2504.02521*, 2025.

[18] Hoang Anh Just, Myeongseob Ko, and Ruoxi Jia. The signal is in the steps: Local scoring for reasoning data selection. *arXiv preprint arXiv:2510.03988*, 2025.

[19] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. Mawps: A math word problem repository. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies*, pages 1152–1157, 2016.

[20] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2025.

[21] Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael Mahoney, Kurt Keutzer, and Amir Gholami. Llm2llm: Boosting llms with novel iterative data enhancement. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6498–6526, 2024.

[22] Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. Common 7b language models already possess strong math capabilities. *arXiv preprint arXiv:2403.04706*, 2024.

[23] Chengpeng Li, Guanting Dong, Mingfeng Xue, Ru Peng, Xiang Wang, and Dayiheng Liu. DotaMath: Decomposition of thought with code assistance and self-correction for mathematical reasoning. *arXiv preprint arXiv:2407.04078*, 2024.

[24] Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, and Tianyi Zhou. Reflection-tuning: Recycling data for better instruction-tuning. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[25] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri 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.

[26] Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. MathGenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2732–2747, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[27] Zimu Lu, Aojun Zhou, Ke Wang, Houxing Ren, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. Step-controlled DPO: Leveraging stepwise error for enhanced mathematical reasoning. *arXiv preprint arXiv:2407.00782*, 2024.

[28] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, Yansong Tang, and Dongmei Zhang. WizardMath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.

[29] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, 2023.

[30] Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online, July 2020. Association for Computational Linguistics.

[31] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI Conference on Artificial Intelligence*, 2020.

[32] 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, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11615–11639, Vienna, Austria, July 2025. Association for Computational Linguistics.

[33] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, 2021.

[34] Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. Mutual reasoning makes smaller LLMs stronger problem-solvers. In *International Conference on Learning Representations (ICLR)*, 2025.

[35] Matthew Renze. The effect of sampling temperature on problem solving in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7346–7356, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[36] Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[37] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

[38] Yongqi Tong et al. Can LLMs learn from previous mistakes? investigating LLMs' errors to boost for reasoning. *arXiv preprint arXiv:2403.20046*, 2024.

[39] Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-Math: Difficulty-aware rejection tuning for mathematical problem-solving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[40] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, 2024.

[41] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *International Conference on Learning Representations (ICLR)*, 2023.

[42] Zhenyu Wu, Qingkai Zeng, Zhihan Zhang, Zhaoxuan Tan, Chao Shen, and Meng Jiang. Large language models can self-correct with key condition verification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12846–12867, 2024.

[43] Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, et al. Enhancing llm reasoning via critique models with test-time and training-time supervision. *arXiv preprint arXiv:2411.16579*, 2024.

[44] Yuxi Xie et al. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

[45] Wenda Xu, Rujun Han, Zifeng Wang, Long T. Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *International Conference on Learning Representations (ICLR)*, 2025.

[46] Yuchen Yan, Jin Jiang, Yang Liu, Yixin Cao, Xin Xu, Mengdi Zhang, Xunliang Cai, and Jian Shao. S^3c-Math: Spontaneous step-level self-correction makes large language models better mathematical reasoners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 25588–25596, 2025.

[47] An Yang, Beichen Zhang, Binyuan Hui, et al. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

[48] 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 *International Conference on Learning Representations (ICLR)*, 2025.

[49] Yuming Yang, Mingyoung Lai, Wanxu Zhao, Xiaoran Fan, Zhiheng Xi, Mingqi Wu, Chiyue Huang, Jun Zhao, Haijun Lv, Jian Tong, Yunhua Zhou, Yicheng Zou, Qipeng Guo, Tao Gui, Qi Zhang, and Xuanjing Huang. Which reasoning trajectories teach students to reason better? a simple metric of informative alignment. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 42123–42150. Association for Computational Linguistics, 2026.

[50] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[51] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

[52] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *International Conference on Learning Representations (ICLR)*, 2024.

[53] Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. MAmmoTH2: Scaling instructions from the web. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[54] Dylan Zhang, Qirun Dai, and Hao Peng. The best instruction-tuning data are those that fit, 2025.

[55] Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. Small language models need strong verifiers to self-correct reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15637–15653, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[56] 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, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14720–14738, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[57] Zijian Zhang, Kaiyuan Zheng, Zhaorun Chen, Joel Jang, Yi Li, Siwei Han, Chaoqi Wang, Mingyu Ding, Dieter Fox, and Huaxiu Yao. GRAPE: Generalizing robot policy via preference alignment, 2024.

[58] Yaowei Zheng, Richong Zhang, Junhao Zhang, et al. LLaMA-Factory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.

A Full Pipeline Algorithm

Algorithm 1 shows one construction pass followed by SFT. For the two-pass setting, both passes use the same original student checkpoint, and SFT is performed after the final corpus is assembled.

Algorithm 1 Full Pipeline

Require: Dataset $\mathcal{D} = \{(q_i, y_i)\}_{i=1}^{M}$, teacher π_T, student π_S, hyperparameters K, N, L
Ensure: Fine-tuned student $\hat{\pi}_S$

$\mathcal{D}_{\text{selected}} \leftarrow \emptyset$
for each $(q, y) \in \mathcal{D}$ **do**
  Sample K traces $\{\tau_k\}_{k=1}^{K} \sim \pi_S(\cdot \mid q)$
  Compute $\alpha(q)$
  if $\alpha(q) = 0$ **then**
    continue
  end if
  Compute $\kappa(q)$
  Classify error mode m by Table 1
  Select representative error trace τ_{rep}
  Generate N candidate trajectories in one teacher call: $\{(r_n, p_n, c_n)\}_{n=1}^{N} \sim \pi_T(\cdot \mid P_m, q, \tau_{\text{rep}})$
  $\mathcal{C}_{\text{correct}} \leftarrow \{(r_n, p_n, c_n) : \text{answer}(c_n) = y\}$
  if $\mathcal{C}_{\text{correct}} = \emptyset$ **then**
    continue
  end if
  for each $\tau = (r, p, c) \in \mathcal{C}_{\text{correct}}$ **do**
    $W \leftarrow$ first L sentences of c
    $\text{BPR}(\tau) \leftarrow \dfrac{1}{|W|} \displaystyle\sum_{i=1}^{|W|} \log \pi_S(c_i \mid q, p, c_{<i})$
  end for
  $\hat{\tau} \leftarrow \arg\max_{\tau \in \mathcal{C}_{\text{correct}}} \text{BPR}(\tau)$, with components $(\hat{r}, \hat{p}, \hat{c})$
  Append $(q, \tau_{\text{rep}}, \hat{r}, \hat{p}, \hat{c})$ to $\mathcal{D}_{\text{selected}}$
end for
Construct $\mathcal{D}_{\text{train}}$: 90% use $(q \oplus \tau_{\text{rep}},\ \hat{r} \oplus \hat{p} \oplus \hat{c})$; 10% use $(q,\ \hat{p} \oplus \hat{c})$
Fine-tune π_S on $\mathcal{D}_{\text{train}}$ with loss on target tokens only
return $\hat{\pi}_S$

B Training and Evaluation Details

Training formats. For 90% of the final examples, the input is $q \oplus \tau_{\text{rep}}$ and the target is $\hat{r} \oplus \hat{p} \oplus \hat{c}$. For the remaining 10%, the input is q alone and the target is the complete corrected solution $\hat{p} \oplus \hat{c}$, without reflection. These auxiliary examples come from the same $\alpha(q) > 0$ pool, not a separate $\alpha(q) = 0$ pool. The input is loss-masked in both formats.

Training settings. We use full-parameter SFT in LLaMA-Factory [58] for 2–3 epochs with a learning rate between 1×10^{-5} and 2×10^{-5}, a cosine schedule with 3% warmup, an effective batch size of 128, and a maximum sequence length of 2048. Each SFT run uses one NVIDIA H200 or eight NVIDIA GeForce RTX 3090 GPUs.

Evaluation. We evaluate on GSM8K [5] and MATH [14] for in-distribution, and SVAMP [33], ASDiv [30], and MAWPS [19] for out-of-distribution, using greedy decoding with exact-match accuracy after answer extraction.

C Comparison Method Details

We compare against SFT, RFT [51], ISC [12], S^{3}C-Math [46], GPTAug, RefAug [56], and LEMMA [32]. All baselines draw from the same training data source for fair comparison: the GSM8K and MATH training problems. They do not necessarily use identical surface questions or constructed instances.

Methods applied to MetaMathQA. SFT uses the complete 394.99K MetaMathQA corpus of correct solutions. RFT rejection-samples the student and retains correct self-generated traces, with at most one per distinct question. S^{3}C-Math injects intermediate mistakes via step-level sampling and constructs reflective corrections, adding approximately 532K instances to MetaMathQA for approximately 927K samples in total. Our method profiles the student on MetaMathQA and constructs corrections as in Appendix A.

Methods constructing supervision from source problems. GPTAug generates direct teacher solutions, ISC constructs self-correction sequences, RefAug enriches solutions with reflection phases, and LEMMA performs error-type grounded mistake augmentation. These methods follow their respective construction recipes based on the GSM8K and MATH source problems rather than using our MetaMathQA variants. The original RefAug configuration contains 29.94K samples; LEMMA retains 88.90K trajectories after correctness filtering. The other sample counts follow the methods' original construction settings.

Controls on the same first-pass problems. The direct teacher-trace control uses exactly the problems in our 62.74K Llama and 65.76K DeepSeek first-pass sets. DeepSeek-V3.2 generates a solution conditioned only on the question; its final answer is checked with the same checker as our pipeline, and incorrect generations are resampled until a correct trace is obtained. Each problem contributes one verified-correct trajectory. Teacher, student, and SFT settings match our method. This compares complete supervision-construction pipelines rather than isolating BPR.

D Prompt Templates for Adaptive Correction Generation

This appendix provides the full prompt templates used by the teacher model π_T for each diagnosed error mode in Stage 2 of our pipeline, described in Section 3.3. All three templates share the same JSON output schema, a list of N corrective trajectories with four fields: `error_step` for a first-person diagnosis, `preserved_prefix` for the verbatim correct portion of the reasoning of the student that may be empty, `correction` for the continuation in the style of the student ending with "The answer is: [value]", and `final_answer` for the final numerical or symbolic answer. The templates differ in their input slots and in the strategies used to diversify the N candidates, as detailed below. The slot `{trajectory_examples}` contains a fixed block of six examples demonstrating the reflection r, preserved prefix p, and corrective continuation c. The same block is used across all three templates.

The slot `{task_data}` fills the per-problem inputs into P_m: the problem, correct answer, profiling statistics, answer-level summary, and one representative incorrect trace. Stubborn and sporadic prompts use `student_reasoning`; the inconsistent prompt uses a singleton `student_reasoning_attempts` list.

D.1 Prompt for Stubborn Errors

For stubborn errors, the majority of the K sampled attempts of the student produce the same wrong answer, indicating a deep-rooted conceptual flaw rather than a random slip. The prompt therefore instructs the teacher to attack the locked error path from multiple angles, while still preserving any correct portion of the reasoning of the student that precedes the divergence.

Prompt Template for Stubborn Errors

You are an expert Math Corrector. You specialize in diagnosing SYSTEMATIC MISCONCEPTIONS.
The incorrect outcomes concentrate on one wrong answer. Attack the locked-in reasoning path from different angles: direct correction, an alternative method, conceptual reframing, work backwards, or substitution-based verification. **Diagnose the trap**, the conceptual pivot, the fix, and a check that prevents recurrence. Distill that analysis into a concise first-person `error_step`.
Correction-angle examples: trajectory 1 applies a direct fix to the specific algebraic/logical error; trajectory 2 solves the problem via an alternative method entirely; trajectory 3 reframes the underlying concept before solving; further trajectories explore additional creative variations such as work-backwards or substitution-based verification.
Generate exactly `{n_candidates}` DIFFERENT corrective trajectories in ONE JSON array. Each object must contain these string fields:

1. **error_step:** a concise first-person diagnosis naming where and why the first error occurred. For example, "I assumed that…" or "I forgot to multiply by…".
2. **preserved_prefix:** the student's LONGEST correct prefix, copied VERBATIM from the beginning up to but excluding the first erroneous step; empty if none.
3. **correction:** continue from `preserved_prefix` to a complete correct solution. Match the student's vocabulary, formatting, and tone. End with exactly "The answer is: [value]".
4. **final_answer:** only the final numerical or symbolic answer.

Localize the first error and diversify the `{n_candidates}` continuations jointly in this single call. **Never invent an error in correct reasoning.** Avoid apologies, meta-commentary such as "Let me reconsider" or "I made an error", and explanations outside the array.
Reference output-format examples: `{trajectory_examples}`
Task data: `{task_data}`

D.2 Prompt for Inconsistent Errors

For inconsistent errors, the prompt emphasizes *structured reasoning* to provide a stable problem-solving framework. The slot `student_reasoning_attempts` supports one or multiple trajectories. With multiple trajectories, the teacher identifies their shared correct prefix and where their reasoning diverges; with one, it preserves that trajectory's longest correct prefix and identifies its first erroneous step. In our experiments, the slot is a singleton list containing only τ_{rep}. The answer-level summary covers all K samples, without supplying their full reasoning chains.

Prompt Template for Inconsistent Errors

You are an expert Math Corrector. You specialize in STABILIZING chaotic reasoning patterns.
The sampled outcomes are unstable. Supply a structured framework: state the givens and goal, define variables and equations, calculate step by step, and verify the result. **Vary frameworks:** equations, verifiable arithmetic, working backwards, dimensional analysis, or unit tracking. The answer summary describes all sampled outcomes; `student_reasoning_attempts` may contain one or more trajectories to be analyzed. With multiple trajectories, preserve any shared correct prefix and identify where their reasoning diverges; with one, preserve that trajectory's longest correct prefix and diagnose its first erroneous step. If no correct prefix exists, leave `preserved_prefix` empty. Do not infer a shared prefix from unprovided trajectories.
Framework examples: trajectory 1 sets up and solves algebraic equations; trajectory 2 breaks the problem into small verifiable arithmetic sub-calculations; trajectory 3 works backwards from the answer structure; further trajectories use dimensional analysis, unit tracking, or diagram-based reasoning.
Generate exactly `{n_candidates}` DIFFERENT corrective trajectories in ONE JSON array. Each object must contain these string fields:

1. **error_step:** a concise first-person diagnosis naming where and why the first error occurred. With multiple trajectories, describe the common pattern of confusion; with one, describe its concrete first error.
2. **preserved_prefix:** with one trajectory, its LONGEST correct prefix, copied VERBATIM from the beginning up to but excluding the first erroneous step; with multiple trajectories, any shared correct prefix; empty if none.

3. **correction:** continue from `preserved_prefix` to a complete correct solution. Match the student's vocabulary, formatting, and tone. End with exactly "The answer is: [value]".
4. **final_answer:** only the final numerical or symbolic answer.

The correction is intentionally more structured than the student's attempts because the student needs scaffolding, but still matches the student's vocabulary and tone. **Localize the first error** and diversify the `{n_candidates}` continuations jointly in this single call. **Never invent an error in correct reasoning.** Avoid apologies, meta-commentary such as "Let me reconsider", and explanations outside the array.
Reference output-format examples: `{trajectory_examples}`
Task data: `{task_data}`

D.3 Prompt for Sporadic Errors

For sporadic errors, the student answers most attempts correctly and only occasionally fails. The student already possesses the relevant skills, so the prompt instructs the teacher to preserve the longest possible correct prefix p and apply the smallest possible edit at the divergence point.

Prompt Template for Sporadic Errors

You are an expert Math Corrector. You specialize in MINIMAL, PRECISE corrections.
The student usually answers correctly. Preserve as much of the reasoning as possible and change only the localized error. **Vary the local repair:** a direct fix, brief inline verification, a slightly different computation avoiding the same error, or further minimal variations. **Do not restructure correct steps.** The localized slip may be an arithmetic mistake, sign error, misapplied formula, or off-by-one in an otherwise reasonable solution. The overall problem-solving approach and structure are sound; only the erroneous step needs to be repaired.
Local-repair examples: trajectory 1 directly replaces the wrong step with the correct computation; trajectory 2 corrects with a brief inline verification; trajectory 3 takes a slightly different computational path that avoids the specific error; further trajectories explore additional minimal variations.
Generate exactly `{n_candidates}` DIFFERENT corrective trajectories in ONE JSON array. Each object must contain these string fields:

1. **error_step:** a concise first-person diagnosis naming where and why the first error occurred. Describe the specific localized slip, for example "I made an arithmetic error when adding…".
2. **preserved_prefix:** the student's LONGEST correct prefix, copied VERBATIM from the beginning up to but excluding the first erroneous step; empty if none.
3. **correction:** continue from `preserved_prefix` to a complete correct solution. Match the student's vocabulary, formatting, and tone. End with exactly "The answer is: [value]".
4. **final_answer:** only the final numerical or symbolic answer.

The correction should feel like a seamless continuation of the student's own work. Keep the correction short. **Localize the first error** and diversify the `{n_candidates}` continuations jointly in this single call. **Never invent an error in correct reasoning.** Avoid apologies, meta-commentary such as "I made a mistake here" or "Let me reconsider", and explanations outside the array.
Reference output-format examples: `{trajectory_examples}`
Task data: `{task_data}`

D.4 Uniform Minimal-Correction Prompt

The ablation without profiling uses the same minimal-correction policy for every error, instead of the three mode-specific prompts. The uniform prompt receives only the problem, representative incorrect trace, and correct answer, without profiling metadata.

Uniform Minimal-Correction Prompt

[System] You are an expert Math Corrector specializing in minimal, precise corrections.
Context. The student attempted this problem and made a mistake in an otherwise reasonable solution. Preserve as much of the student's original reasoning as possible and fix only what is broken.
Your Task. Generate exactly `{n_candidates}` DIFFERENT corrective trajectories. Each trajectory must:

1. Preserve the maximum correct prefix: copy the student's reasoning VERBATIM up to the point of error, the longest correct prefix possible.
2. Pinpoint the exact error: identify the specific calculation, substitution, or logical misstep in a first-person `error_step`, rather than stating only "I was wrong".
3. Apply a minimal correction: fix ONLY what is wrong; do not restructure or rewrite correct steps.
4. Vary the correction style across the `{n_candidates}` trajectories.

Style Requirements.
- Preserve the student's writing style, formatting, and tone.
- No meta-commentary such as "I made a mistake here" or "Let me reconsider".
- Every trajectory MUST end with exactly: "The answer is: [value]".

Output Format. A JSON array of `{n_candidates}` objects with fields `{error_step, preserved_prefix, correction, final_answer}`.

Reference Examples. `{trajectory_examples}`

Task Data. `{problem}` / `{student_reasoning}` / `{correct_answer}`

E Error Mode Distribution

Figure 5 shows the distribution of diagnosed error modes among 63,062 problems before teacher-candidate correctness filtering in the Llama-3.1-8B first construction pass. Of these problems, 62,740 are retained after filtering to form the first-pass training corpus used in the ablation and window-size studies.

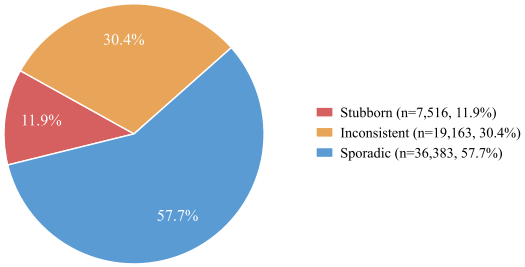


Figure 5: Distribution of diagnosed error modes among 63,062 problems before teacher-candidate correctness filtering in the Llama-3.1-8B first construction pass.

F Profiling Threshold Sensitivity

We evaluate threshold sensitivity on the same 63,062 problems from the Llama-3.1-8B first construction pass shown in Figure 5, before teacher-candidate correctness filtering. The problem set and the five sampled student responses per problem are held fixed across threshold settings.

Our behavioral profiling classifies each problem based on two thresholds: the error rate threshold α_t and the consistency threshold κ_t. With $K = 5$ samples per problem, α takes values in $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ and κ takes a similarly discrete set of values determined by the number of incorrect samples. This discretization means that the classification is invariant to any threshold perturbation that stays within the same discrete interval, so several threshold pairs reproduce the default classification exactly.

Figure 6 reports the agreement with the default thresholds $(\alpha_t, \kappa_t) = (0.6, 0.8)$, measured as the fraction of problems whose diagnosed mode matches the default. The neighboring setting $(\alpha_t, \kappa_t) = (0.5, 0.8)$ reproduces the default classification exactly (100%), and all cells in $\alpha_t \in \{0.5, 0.6\}$ with $\kappa_t \in \{0.7, 0.9, 1.0\}$ remain at 95.1%–97.9% agreement. Agreement degrades smoothly as the thresholds move further from the default: lowering α_t to 0.4 drops agreement to 67.6%–77.8%, because problems with $\alpha = 0.4$ are reclassified from sporadic to stubborn or inconsistent, while raising α_t to 0.7–0.8 yields 80.0%–85.5%. These represent substantial shifts in the classification criteria rather than fine perturbations. In practice, the discrete structure of α and κ under small K provides a natural robustness to threshold selection: the method does not require fine-grained tuning to achieve stable classifications.

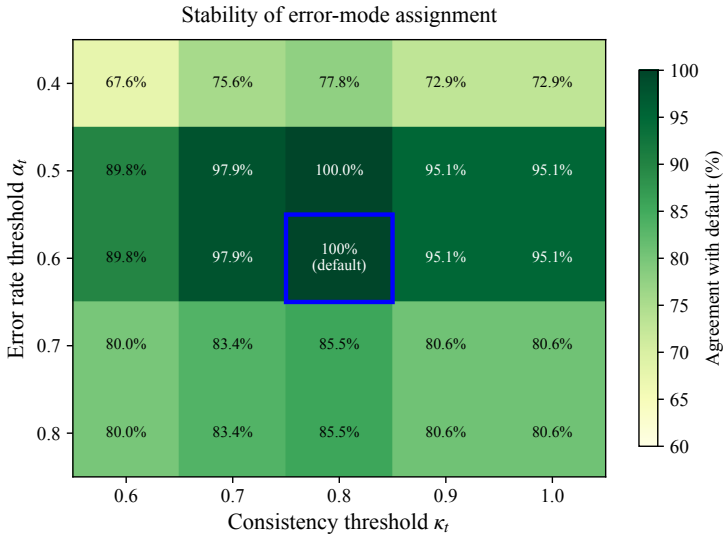


Figure 6: Agreement (%) with the default thresholds $(\alpha_t, \kappa_t) = (0.6, 0.8)$ on the same 63,062-problem set as Figure 5. Each cell shows the fraction of problems whose error mode matches the classification under the default setting, with the default cell at 100%. Cells adjacent to the default exhibit high agreement due to the discrete nature of α and κ under $K = 5$ sampling.

G Token-Level Log-Probability Curve

To verify the motivation behind branching point reachability scoring, we plot the per-token log-probability of the student against position relative to the branching point, averaged over 200 training examples. Figure 7 shows the preserved prefix sitting near the global average of -0.85, dropping sharply to about -3.5 at the branching point, and recovering within roughly 25 tokens. Global scoring averages over the long high-probability tail and washes out the few low-probability tokens that distinguish reachable from unreachable continuations, whereas restricting evaluation to the shaded BPR window concentrates the signal on the positions where candidates diverge, matching the 1.7% gain over global scoring in both ablation studies.

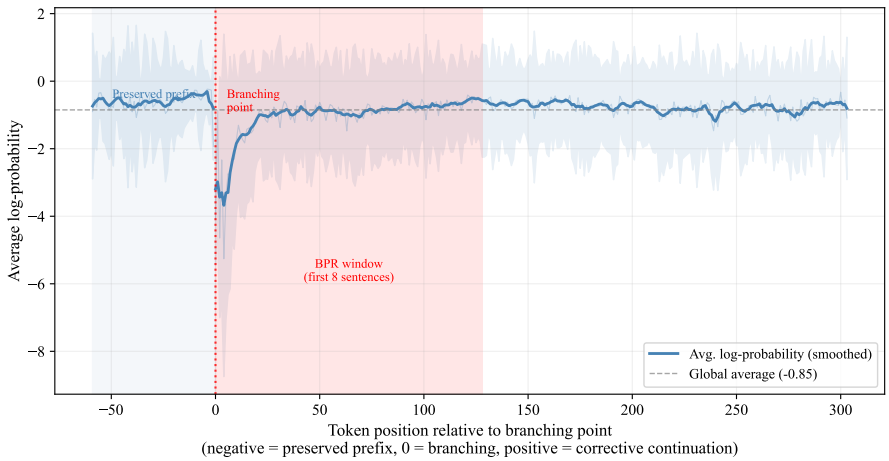


Figure 7: Average per-token log-probability of the Llama-3.1-8B student against token position relative to the branching point, over 200 examples. Both the preserved prefix and the recovered tail of the corrective continuation sit near the global average of -0.85, while only a narrow trough around the branching point falls to about -3.5. The shaded red region marks the BPR scoring window covering the first eight sentences.

H Additional Experimental Results

This section provides the complete supplementary comparisons and analyses summarized in Section 4.

H.1 Shared-Source Coverage Control

We map Ours and LEMMA to their original GSM8K and MATH problems and restrict both methods to the same 12,217 shared source problems, retaining one training example per source problem for each method. This controls source-problem coverage and example count while retaining each method's supervision format. Table 5 reports the DeepSeek-Math-7B results. Ours achieves 77.1% versus LEMMA's 75.7%, with higher OOD results while LEMMA remains stronger on MATH. This is a coverage control, rather than a causal test of BPR alone.

Table 5: Accuracy (%) on DeepSeek-Math-7B after restricting each method to the same 12,217 source problems, with one training example per source problem. This controls source coverage and sample count while retaining each method's supervision format.

Method	GSM8K	MATH	SVAMP	ASDiv	MAWPS	Avg.
Ours	**80.3**	45.2	**81.4**	**87.3**	**91.2**	**77.1**
LEMMA	77.5	**47.7**	78.1	86.4	88.6	75.7

H.2 First-Divergence Localization

Boundary localization. A manual audit of 50 randomly sampled traces from the 65.76K DeepSeek configuration finds that 44 boundaries are accurate, four are too early, and two are too late. Keeping teacher candidate pools fixed, we perturb BPR boundaries on 12% of questions with a preserved prefix: 8% move one reasoning step earlier and 4% one step later. We recompute scores under the shifted prefixes and windows, reselect trajectories, and retrain. Table 6 reports this single run, in which average accuracy changes from 80.0% to 79.7%. This checks the sensitivity of BPR scoring and candidate reselection to boundary localization.

Table 6: Accuracy (%) after boundary perturbation for the 65.76K DeepSeek-Math-7B configuration. Teacher candidate pools are fixed; BPR is recomputed under shifted boundaries before trajectory reselection and retraining. This is a single-run selection-sensitivity experiment.

Setting	GSM8K	MATH	SVAMP	ASDiv	MAWPS	Avg.
Original	83.6	**47.1**	**83.9**	**90.1**	**95.3**	**80.0**
Perturbed	**83.8**	46.6	83.6	89.4	94.9	79.7

H.3 Teacher Variation

Table 7 compares teachers in single SFT runs using the same fixed seed. Replacing DeepSeek-V3.2 with DeepSeek-V4-Flash gives 81.0% average accuracy, versus 80.6%, with all other settings held fixed.

Table 7: Accuracy (%) with different teachers for DeepSeek-Math-7B using 88.90K constructed examples. Both rows report single SFT runs using the same fixed seed. All settings other than the teacher are held fixed.

Teacher	GSM8K	MATH	SVAMP	ASDiv	MAWPS	Avg.
DeepSeek-V3.2	84.1	48.6	85.3	90.1	**95.0**	80.6
DeepSeek-V4-Flash	**84.7**	**49.1**	**85.7**	**90.8**	94.9	**81.0**

H.4 Inference Behavior

Table 8 reports an audit of 300 test outputs per benchmark for each 88.90K student, totaling 1,500 outputs per student. An LLM judge labels explicit revision of an earlier step as REFLECTION and

other solutions as DIRECT. DIRECT outputs account for 90.5% and 95.3% of Llama and DeepSeek outputs, with accuracies of 81.9% and 82.1%; REFLECTION accuracies are 33.1% and 35.7%. The share-weighted accuracies are 77.3% and 79.9%. This provides indirect evidence that corrective training is compatible with question-only inference without routinely triggering explicit revision.

Table 8: Test-output audit for the 88.90K students: 300 outputs per benchmark, totaling 1,500 per student. REFLECTION denotes explicit revision within a test-time solution; DIRECT denotes no such revision. Shares and accuracies are percentages.

	DIRECT		REFLECTION	
Student	**Share**	**Accuracy**	**Share**	**Accuracy**
Llama-3.1-8B	90.5	**81.9**	9.5	33.1
DeepSeek-Math-7B	95.3	**82.1**	4.7	35.7

I Case Examples of Token-Level Log-Probability

Beyond the average curve in Figure 7, we present three individual training examples in Figures 8, 9, and 10 that illustrate the same pattern at the level of single problems. Each figure displays the problem statement, the preserved prefix, the BPR window covering the first eight sentences after the branching point, and a portion of the later continuation, alongside the corresponding per-token log-probability curve under the Llama-3.1-8B student.

In all three cases, the preserved prefix and the later continuation sit close to the global average, while the BPR window concentrates a sequence of sharp dips that pull its mean roughly 1.2 to 1.7 nats below the tail average. This gap is consistent with the averaged behavior in Figure 7 and illustrates the concentration of low-probability tokens near the branching point. The first example introduces an equation-based framework that identifies given quantities, defines variables, and sets up earnings equations for a marble pricing word problem, the second rewrites both sides to a common base before equating exponents on an exponential equation, and the third lays out grouped variables for hourly wage, working days, and total hours on a monthly income word problem. These examples illustrate the probability cost of the initial corrective steps. The performance benefit of using $L = 8$ is evaluated in the component ablation and window-size studies in Section 4.3.

Case Example 1: Marble Pricing Word Problem

Problem. Alvin is selling his marble set. He has 100 marbles. 20% are white, 30% are black, and the rest are different colors. He sells the white ones for a nickel each, the black ones for a dime each, and the colors for $x each. How much does he earn? If we know the answer to the above question is 14, what is the value of unknown variable x?
Preserved prefix. Alvin has 100 marbles. 20% are white: 0.20 x 100 = 20 white marbles 30% are black: 0.30 x 100 = 30 black marbles The rest are different colors: 100 - 20 - 30 = 50 different colored marbles
BPR window. Let's solve this using a clear equation-based framework, sticking to dollars to avoid unit confusion. Step 1: Identify what's given and what's asked. Given: Total marbles = 100. Percentages and prices for white and black marbles. Total earnings = $14. Need to find x (price per colored marble in dollars).
Later continuation. Step 2: Define variables. Let x = price in dollars for one colored marble. Step 3: Set up equations for earnings from each type. Earnings from white marbles = 20 marbles * $0.05 = $1 Earnings from black marbles = 30 marbles * $0.10 = $3 Earnings from colored marbles = 50 marbles * x dollars = 50x dollars Step 4: Write the total earnings equation. Total earnings = $1 + $3 + 50x = $14 Step 5: Solve for x. Combine constants: $4 + 50x = $14 Subtract $4: 50x = $10 Divide by 50: x = $10 / 50 = $0.20 Step 6: Verify. Check: White: 20*0.05=1, Black: 30*0.10=3, Colored: 50*0.20=10. Sum = 1+3+10=14. Correct. The answer is: 0.20

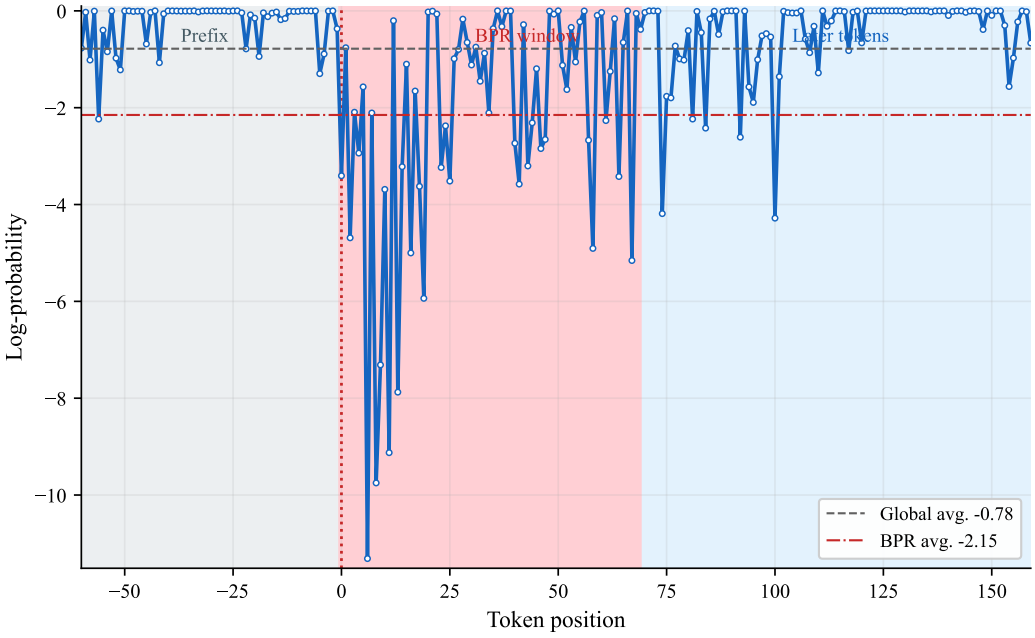


Figure 8: Per-token log-probability for an individual training example on a marble pricing word problem. The preserved prefix and later continuation track the global average, while the BPR window absorbs sharp dips that lower its mean to -2.15 against a tail average of -0.45, yielding a gap of 1.70 nats.

Case Example 2: Exponential Equation

Problem. Solve for x, if $8^{4x-6} = \left(\frac{1}{2}\right)^{x+5}$.
Preserved prefix. To solve for x, we first need to express both sides of the equation with the same base. We can rewrite 8 as 2^3 and $\frac{1}{2}$ as 2^{-1}.
BPR window. Let's follow a clear, equation-based framework. Step 1 – Identify the goal: Solve for x in $8^{4x-6} = \left(\frac{1}{2}\right)^{x+5}$. Step 2 – Choose a common base. The easiest is base 2 because 8 = 2^3 and 1/2 = 2^{-1}. Step 3 – Rewrite each side using base 2.
Later continuation. Left side: $8^{4x-6} = (2^3)^{4x-6} = 2^{3(4x-6)} = 2^{12x-18}$. Right side: $\left(\frac{1}{2}\right)^{x+5} = (2^{-1})^{x+5} = 2^{-1(x+5)} = 2^{-x-5}$. Step 4 – Set the exponents equal since the bases are the same and the equation is an equality. $12x - 18 = -x - 5$ Step 5 – Solve the linear equation systematically. Add x to both sides: $12x + x - 18 = -5 \rightarrow 13x - 18 = -5$. Add 18 to both sides: $13x = 13$. Divide both sides by 13: $x = 1$. Step 6 – Quick check: Plug x=1 back into the original equation. Left: $8^{4(1)-6} = 8^{-2} = 1/64$. Right: $(1/2)^{1+5} = (1/2)^6 = 1/64$. It matches. The answer is: 1

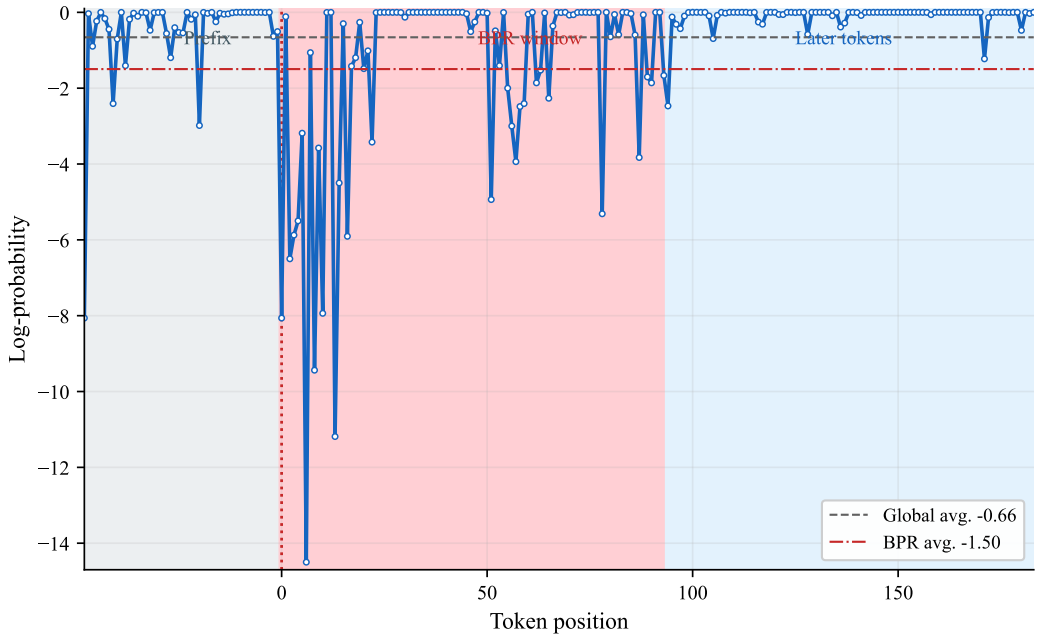


Figure 9: Per-token log-probability for an exponential equation. The BPR window contains the framework setup tokens that rewrite both sides to a common base before equating exponents, lowering the BPR mean to -1.50 against a tail average of -0.09, with a gap of 1.41 nats.

Case Example 3: Monthly Income Word Problem

Problem. If John works for 12 hours every other day and receives a 30% raise from his previous $20 per hour job, what is his monthly income for a 30-day month?
Preserved prefix. First, let's calculate John's new hourly wage: $20 * 1.30 = $26 per hour.
BPR window. Step 1: Understand the quantities given. Previous wage: $20 per hour. Raise: 30%. Hours per workday: 12 hours. Work schedule: every other day. Month length: 30 days. We are asked: monthly income. Step 2: Set up variables and equations.
Later continuation. Let new_wage = old_wage * (1 + raise_percent/100) = 20 * 1.30 = $26. Let work_days = total_days / 2 = 30 / 2 = 15 days. Let total_hours = work_days * hours_per_day = 15 * 12 = 180 hours. Let monthly_income = new_wage * total_hours = 26 * 180. Step 3: Execute computation step by step. 26 * 180 = 26 * (100 + 80) = (26*100) + (26*80) = 2600 + 2080 = 4680. Step 4: Verify. Half of 30 days is 15 days, which is reasonable. 15 days * 12 hours = 180 hours. $26 per hour * 180 hours = $4680, which is a plausible monthly income. The answer is: 4680

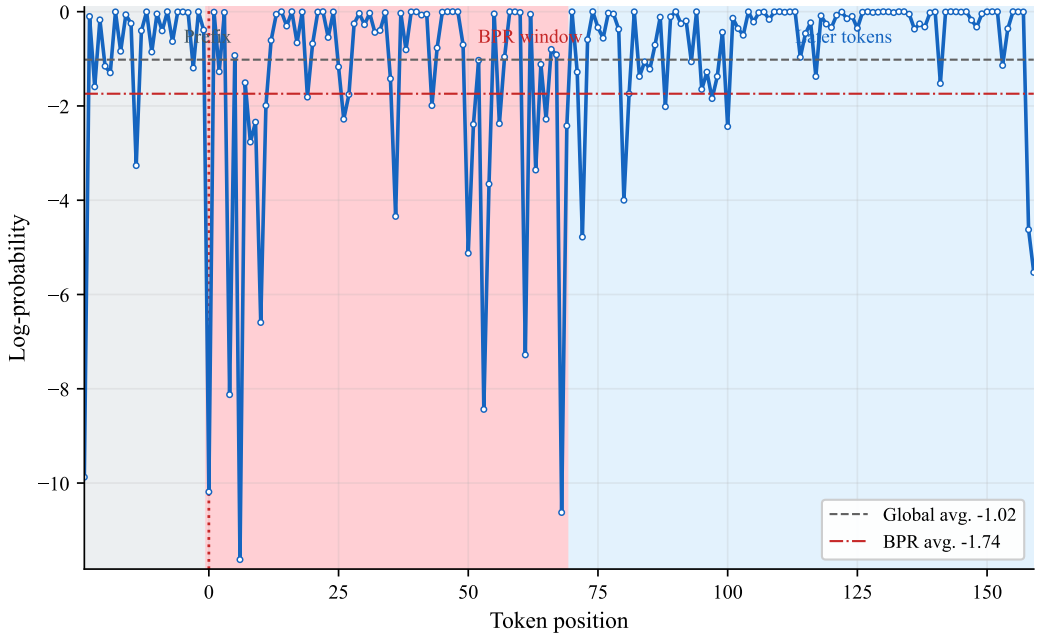


Figure 10: Per-token log-probability for a monthly income word problem. The BPR window summarizes the given quantities and introduces the variable-setting stage, with a mean of -1.74 against a tail average of -0.57, yielding a gap of 1.17 nats.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim three contributions: (i) a behavioral profiling scheme based on error frequency and consistency, (ii) branching point reachability (BPR) scoring for trajectory selection, and (iii) the highest reported five-benchmark average accuracy on both student backbones. (i) and (ii) correspond to Section 3.2 and Section 3.4 respectively, and (iii) is supported by Table 2 and Table 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses dependence on a strong teacher and the coarse resolution of profiling with $K = 5$. It notes that larger sampling budgets could improve resolution at higher cost, but this trade-off has not been systematically explored.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not state any formal theorems or lemmas. Section 3.4 contains a paragraph relating BPR to the ideal selection objective in Eq. 2, but this is presented as an interpretive connection rather than a formal theoretical result requiring assumptions or a proof.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sections 3.2–3.5 describe profiling, candidate construction, BPR, and the two training formats. Section 4.1 reports models, and Appendix B reports SFT settings. Algorithm 1 summarizes a single construction pass, and Appendix B specifies the training formats; Appendix D documents the mode-specific and uniform prompts, including the singleton input used for inconsistent errors. Appendix C describes baseline sources and controls.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The public repository linked in the abstract provides the code, constructed datasets, trained weights, and instructions for reproducing the main experimental results, including environment setup, data preparation, training and evaluation commands, and the corresponding configurations. The construction procedure and prompt templates are also documented in Appendices A and D.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Appendix B reports full-parameter SFT in LLaMA-Factory, 2–3 epochs, a learning rate between 1×10^{-5} and 2×10^{-5}, cosine scheduling with 3% warmup, effective batch size 128, maximum length 2048, and target-only loss. Section 4.1 specifies $K = 5$ and $N = 5$, and Section 3.4 specifies $L = 8$. Appendix B specifies the 90/10 training mixture. Appendix B specifies greedy decoding and final-answer exact match, and Appendix H.4 discusses question-only inference.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 2 reports means and sample standard deviations across three SFT seeds for each fixed 88.90K corpus, including the original SFT seed and two additional seeds. The corpus and other settings are held fixed. Standard deviations use denominator $n - 1$; Avg statistics are computed across the three per-seed five-benchmark averages. These quantify SFT-seed variation, not a significance test against the baselines. Other table entries are point estimates.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Appendix B reports one NVIDIA H200 or eight NVIDIA GeForce RTX 3090 GPUs per SFT run. Section 4.1 describes the construction passes and sampling budgets, Algorithm 1 describes the teacher requests, and Section 3.4 describes gradient-free BPR scoring. The manuscript does not provide complete per-stage runtimes or total project compute, so the resource accounting is partial.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research uses existing student models, API-accessed teachers, and existing mathematical datasets. No human-subject study or collection of personal data is involved. The work follows the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The main paper and appendix do not include a broader-impact discussion. The reported contribution and evaluation concern mathematical reasoning distillation; potential societal benefits and risks have not been systematically assessed in this work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work studies mathematical reasoning distillation using existing models and datasets. It does not introduce a new foundation model or a newly scraped corpus requiring a separate high-risk release protocol. We do not infer safety guarantees for the fine-tuned students from those of the base models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The manuscript cites the model, dataset, and training-framework sources. A complete account of licenses and API terms for every asset, including the additional teacher, is not provided. We therefore do not claim that a full per-asset licensing statement has been supplied.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public repository linked in the abstract releases the code, constructed datasets, and trained weights with accompanying documentation of their intended use, data fields and examples, provenance and construction, model loading, licenses, usage restrictions, and known limitations. The first-pass corpora contain 62.74K and 65.76K examples, and the final corpus contains 88.90K examples per student. Construction details and the 90/10 training mixture are also described in Appendices B, C, and D.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human subjects research; IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: DeepSeek-V3.2 generates correction candidates and localizes student errors; Llama-3.1-8B-Instruct and DeepSeek-Math-7B-Instruct are the students. The additional teacher experiment uses DeepSeek-V4-Flash, and an LLM judge labels inference outputs for the DIRECT/REFLECTION audit. These roles are described in Sections 3.3 and 4.1, Appendices B, H.3, and H.4, and the prompt templates in Appendix D.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.