

Beyond the Answer Key: Robustness Evaluation of Large Language Models for Step-Level Mathematical Verification

Fateme Mazdarani
School of Computing
Clemson University
Clemson, SC, USA
fmazdar@clemson.edu

Carlos Toxtli
School of Computing
Clemson University
Clemson, SC, USA
ctoxtli@clemson.edu

Abstract—Large language models (LLMs) are increasingly used as graders, verifiers, and process auditors, but most mathematical evaluations still emphasize final-answer accuracy. This can obscure whether a model can verify a non-canonical but valid solution trace. We introduce a controlled linear-equation benchmark for evaluating LLMs in the evaluator role. Each instance asks the model to judge final-answer correctness, step-level trace correctness, and the first incorrect step. Our evaluation of state-of-the-art open LLMs reveals a significant robustness gap: models that accurately evaluate canonical solutions often fail when presented with perturbed but logically equivalent variants. Across GPT-OSS 20B, Qwen3-14B, and Phi-4-Reasoning, base models perform well on canonical traces but degrade substantially on perturbed traces, especially for error localization. On valid perturbed traces, base-model false-rejection rates reach 75.6–85.3%, showing strong sensitivity to canonical solution form. Supervised fine-tuning, distillation, and test-time compute improve robustness in some settings, but gains are model dependent and can trade off against canonical performance. The results show that reliable process-level verification remains challenging, and evaluator robustness should be measured separately from solver accuracy, even in a simple algebraic domain with exact ground truth.

Index Terms—Large language models, mathematical reasoning, verifier evaluation, step-level verification, robustness, LoRA

I. INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable capabilities in supporting complex, multi-step tasks across education, software engineering, and scientific research. In correctness-critical applications, the validity of the final result depends entirely on the integrity of the underlying process, and a single error in the process can invalidate the whole result. Therefore, LLMs, when functioning as evaluators, must be capable of distinguishing between valid alternative methods and genuine logical failures. Automated evaluators must be accurate and robust since users may trust explained outputs without necessarily understanding their limitations [1]–[3].

In mathematical contexts, LLMs are not only used to solve problems but also to evaluate solutions produced by humans, tools, or other models. In such evaluator settings, final-answer correctness is insufficient: a solution with the right answer

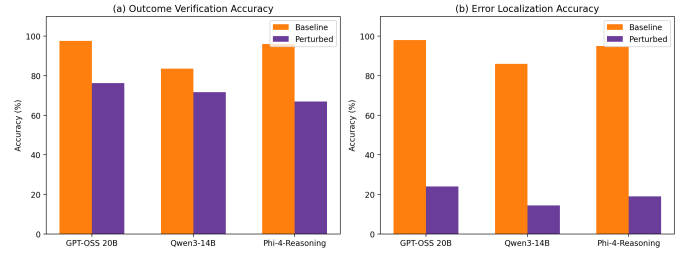


Fig. 1. Robustness gap for base models. Final-answer evaluation is affected by solution perturbations, but the larger degradation occurs in first-error localization, indicating that the models often fail to separate valid non-canonical transformations from true errors.

may contain an invalid step, and a correct solution may use a non-standard path that should not be penalized. Most mathematical LLM benchmarks emphasize the solver role and report outcome-based metrics such as final-answer accuracy on datasets including GSM8K [4] and MATH [5]. This focus can hide a different failure mode: a model may grade canonical worked solutions accurately while rejecting an alternative trace that is mathematically sound but unfamiliar in form. We call this failure mode *template sensitivity*: dependence on a familiar solution template rather than on step-level validity. As shown in Fig. 1, this sensitivity is especially severe for first-error localization, where several models that appear reliable on canonical traces collapse under solution-level perturbations.

We study this problem in linear equations of the form $ax + b = cx + d$. This domain is intentionally narrow: it gives exact ground truth, controllable trace generation, and unambiguous error localization. We do not claim that linear equations capture all mathematical or scientific verification tasks. Instead, we use them as a diagnostic setting in which failures cannot be attributed to ambiguity in the problem statement or to subjective grading criteria. Our contributions are as follows:

- **Controlled trace-perturbation benchmark:** We hold each equation fixed while varying perturbed traces, isolating evaluator robustness with exact labels for correct,

incorrect, and tricky solutions.

- **Three-part evaluation protocol:** We evaluate final-answer correctness (Q_1), step-level correctness (Q_2), and first-error localization (Q_3), enabling separate measurement of answer checking, process checking, and diagnostic feedback.
- **Empirical and adaptation analysis:** We evaluate leading open models GPT-OSS 20B [6], Qwen3-14B [7], and Phi-4-Reasoning [8], quantifying the performance gap between baseline and perturbed alternatives, and identifying common failure modes; then test supervised fine-tuning, Gemini-enhanced distillation, and test-time compute.

II. RELATED WORK

A. Mathematical Reasoning Benchmarks and Verification

A large body of work evaluates LLMs on mathematical reasoning by measuring solving performance on controlled problem sets such as MATH [5]. Such benchmarks primarily assess LLMs as problem solvers rather than evaluators. Related verifier work trains process-based reward models or outcome reward models for scoring generated reasoning traces [2], [9]. More recent work studies step-level verifier training, verifier-guided inference, and formally verified process labels [10]–[12]. Our work complements these efforts by focusing on the grading task itself: whether a model assigns consistent judgments to worked solutions that are procedurally different but algebraically admissible.

B. Robustness under Perturbations

Recent studies show that LLM mathematical reasoning can be fragile under perturbations to problem statements, irrelevant context, symbolic forms, or numerical values [13]–[17]. These studies mainly examine the solver setting. We instead perturb the *solution trace* while holding the original equation fixed. This isolates evaluator robustness: the model is not asked to solve a new problem, but to judge whether a supplied derivation is valid.

C. Evaluating Worked Solutions

Recent work has explored using LLMs as autograders for worked solutions in mathematics by comparing LLM-assigned correctness labels against human grading [1]. Other step-by-step evaluation settings, though not focused on math grading specifically, also emphasize measuring intermediate reasoning behavior rather than only final outputs [18]. Such studies demonstrate the promise of automated evaluation, but they typically do not systematically control solution-trace variability. Our benchmark fills this gap by generating paired canonical and perturbed traces with exact labels for the first invalid step.

III. METHODOLOGY

A. Dataset Generation and Taxonomy

We generate equations of the form $ax + b = cx + d$ with a unique rational solution. A solving engine then emits worked traces under three trace types:

	Baseline	Perturbed
T1(correct)	Question: $4x + 7 = 2x - 5$ [0] $4x + 7 = 2x - 5$ [1] $2x + 7 = -5$ [2] $2x = -12$ [3] $x = -6$	Question: $4x + 7 = 2x - 5$ [0] $4x + 7 = 2x - 5$ [1] $8x + 14 = 4x - 10$ [2] $4x = -24$ [3] $x = -6$
T2(incorrect)	Question: $4x + 7 = 2x - 5$ [0] $4x + 7 = 2x - 5$ [1] $2x + 7 = -5$ [2] $2x = -10$ $\times$ $(-5-7 = -12)$ [3] $x = -5$	Question: $4x + 7 = 2x - 5$ [0] $4x + 7 = 2x - 5$ [1] $8x + 14 = 4x - 10$ [2] $4x = -22$ $\times$ $(-10-14 = -24)$ [3] $x = -5.5$
T3(tricky)	Question: $4x + 7 = 2x - 5$ [0] $4x + 7 = 2x - 5$ [1] $2x + 7 = -5$ [2] $2x = -10$ $\times$ $(-5-7 = -12)$ [3] $x = -6$ σ (reverted)	Question: $4x + 7 = 2x - 5$ [0] $4x + 7 = 2x - 5$ [1] $8x + 14 = 4x - 10$ [2] $4x = -22$ $\times$ $(-10-14 = -24)$ [3] $x = -6$ σ (reverted)

Fig. 2. Benchmark taxonomy. The same equation can appear with a canonical or perturbed trace and as T1, T2, or T3. This design separates final answer from trace correctness; the evaluation prompt does not contain underlined text.

- **T1 (Correct):** Every step maintains mathematical equivalence under the trace semantics in Section III-C, and the final answer is correct. T1 is used to measure false rejection of valid reasoning.
- **T2 (Incorrect):** A single arithmetic or algebraic error is injected at a known step and propagated to an incorrect final answer. T2 tests whether the evaluator detects and localizes the true first error.
- **T3 (Tricky):** A trace contains an invalid intermediate step, but the final answer is manually reverted to the correct value. T3 separates final-answer checking from process checking.

The benchmark taxonomy is illustrated in Fig. 2.

B. Reproducibility Details

Equation instances are generated by sampling integer coefficients a, b, c, d from $[-10, 10]$ and enforcing $a \neq c$, so the solution $x = (d - b)/(a - c)$ is unique. We reject degenerate equations, empty transformations, and traces whose verification obligations cannot be discharged. Train and test splits are instance-disjoint: we hash the tuple (a, b, c, d) before trace generation so that no equation appears in both training and evaluation.

Step indexing is 0-based. Step 0 is the original equation string. The first-error label Q_3 is the index of the first invalid step, and $Q_3 = -1$ when no invalid step exists. All outputs are requested as strict JSON. A malformed response is recorded as a parse error and counted as incorrect for the affected metrics.

C. Trace Semantics and Validation

A central concern in this benchmark is that not every familiar algebraic manipulation is an equivalence over the real line. For example, multiplying both sides by an expression that may be zero or squaring both sides can introduce extraneous candidates. We therefore label traces using a checker semantics rather than treating every syntactic transformation as a bidirectional equivalence.

Let $S_i(x)$ denote the solution set represented by step i , and let O_i denote any active obligation, such as verifying

Question: Solve the equation $5x + 2 = -3$ for real x

Add terms to both sides 0: $5x+2=-3$ 1: $6x+1=x-4$ 2: $5x=-5$ 3: $x=-1$	Move terms to the right-hand side 0: $5x+2=-3$ 1: $3x+1=-2x-4$ 2: $5x=-5$ 3: $x=-1$	Multiply by a constant 0: $5x+2=-3$ 1: $25x+10=-15$ 2: $25x=-25$ 3: $x=-1$
Multiply by a term 0: $5x+2=-3$ 1: $5x^2+2x=-3x$ 2: $5x^2+5x=0$ 3: $x=0$ or $x=-1$ 4: Verification: $x=-1$ valid, $x=0$ invalid	Divide by a constant 0: $5x+2=-3$ 1: $(5/3)x + 2/3 = -1$ 2: $(5/3)x = -5/3$ 3: $x=-1$	Square both sides 0: $5x+2=-3$ 1: $5x=-5$ 2: $25x^2=25$ 3: $x^2=1$ 4: $x=1$ or $x=-1$ 5: Verification: $x=-1$ valid, $x=1$ invalid

Fig. 3. Perturbation families. The first four panels are equivalence-preserving in the generated domain. Multiplication by a term and squaring both sides are candidate-generating transformations; they are accepted only when all resulting candidates are explicitly checked against the original equation.

a candidate or recording a nonzero denominator. A step is accepted as *strictly equivalent* when $S_i(x) \Leftrightarrow S_{i+1}(x)$ over the active domain. This category includes adding the same expression to both sides, moving terms across the equality sign with sign change, and multiplying or dividing by a known nonzero constant. A step is accepted as *candidate-generating* when it preserves all true solutions but may introduce extra candidates, i.e., $S_i(x) \Rightarrow S_{i+1}(x)$, provided that the trace later verifies every candidate against the original equation and discards extraneous roots. Multiplication by a variable term and squaring both sides are handled in this second category. A step is labeled invalid when it contains an arithmetic error, drops a valid candidate, introduces an unsupported restriction, or leaves an obligation unresolved. Accordingly, $Q_2 = 1$ if and only if every transition is either strictly equivalent or candidate-generating and Q_3 is the first transition where this condition fails. A deterministic exact-arithmetic trace checker is used to generate and validate all gold labels.

D. Perturbation Families

A core feature of this benchmark is the inclusion of perturbed solutions through systematic injection of mathematical perturbations—valid but unconventional transformations that deviate from the shortest path to a solution. The benchmark includes six perturbation families, shown in Fig. 3. Four families are strictly equivalence-preserving in our domain: adding terms to both sides, moving terms to the right-hand side, multiplying by a nonzero constant, and dividing by a nonzero constant. Two families are obligation-generating: multiplying by an expression involving x and squaring both sides. For these cases, the generated trace explicitly verifies the resulting candidate set by substitution into the original equation. These controlled stress tests assess whether models distinguish valid alternative paths from errors; they are not intended to represent the full distribution of student work.

E. Evaluation Framework

For each sample, the model receives the original equation and the complete worked trace. It must return a JSON object

with three fields: (i) Q_1 , a Boolean indicating whether the final answer satisfies the original equation; (ii) Q_2 , a Boolean indicating whether the full trace is mathematically valid; and (iii) Q_3 , the integer index of the first invalid step or -1 if no invalid step exists. This format separates outcome verification from process verification and diagnostic localization.

We report accuracy for each query overall and stratified by T1/T2/T3. For T1 traces, $1 - \text{Acc}(Q_2)$ is the false-rejection rate: the evaluator rejects a valid trace. For T2 and T3 traces, a high Q_2 error rate indicates false acceptance of an invalid process. Localization accuracy is intentionally strict and requires exact agreement with the first-error index.

F. Model Selection and Adaptation

We evaluate three open-weight reasoning-oriented models: GPT-OSS 20B [6], Qwen3-14B [7], and Phi-4-Reasoning [8]. We selected these models after preliminary screening to avoid conflating evaluator robustness with a basic inability to follow mathematical notation. All experiments are implemented using the Hugging Face Transformers and PEFT libraries in bf16 precision with gradient checkpointing enabled. Non-TTC evaluations use deterministic decoding; TTC uses seven independent samples at temperature 0.3 followed by majority vote for each query.

- **Fine-Tuning (FT):** We perform parameter-efficient fine-tuning with LoRA ($\text{lr} = 2 \times 10^{-4}$, $r \in \{8, 16\}$, $\alpha \in \{16, 32\}$) for 3 epochs with batch size 16. T1 samples teach acceptance of valid non-canonical traces; T2 and T3 samples teach rejection and localization of invalid transitions.
- **Gemini-Enhanced Distillation (GED-FT):** We refine the programmatic explanations using Gemini 3 Flash [19]. To prevent teacher errors from corrupting the labels, we retain only samples for which the teacher explanation preserves the ground-truth (Q_1, Q_2, Q_3) labels. This filtering yields 4,000 training samples. The goal is to test whether richer natural-language rationales improve verifier behavior beyond label-only supervision.
- **Test-Time Compute (TTC):** We aggregate $k = 7$ sampled responses by majority vote for each output field. TTC tests whether additional reasoning samples correct isolated mistakes or instead amplify systematic template sensitivity.

G. Data Composition and Partitioning

The training set contains 27,000 samples: 9,000 canonical traces, evenly split across T1/T2/T3, and 18,000 perturbed traces, evenly split across trace type and perturbation family. Evaluation uses two held-out sets of 1,080 samples each: one canonical and one perturbed. Each evaluation set has 360 samples per trace type; the perturbed set is balanced across perturbation families. For each sample, the released metadata records the equation tuple, trace type, perturbation family, obligation log, gold labels, and checker explanation. The benchmark, code, and experimental configurations are available at <https://github.com/mazdarani/beyond-answer-key>.

TABLE I
TEST ACCURACY (%) FOR GPT-OSS-20B-BASED MODELS

Model	Outcome Accuracy (Q_1)				Step-Level Accuracy (Q_2)				Localization Accuracy (Q_3)			
	Overall	T1: Correct	T2: Incorrect	T3: Tricky	Overall	T1: Correct	T2: Incorrect	T3: Tricky	Overall	T1: Correct	T2: Incorrect	T3: Tricky
Baseline												
Base	97.69	98.33	98.06	96.67	98.24	98.33	98.06	98.33	98.06	98.33	97.78	98.06
Base + TTC	85.83	100	99.17	58.33	99.35	100	98.89	99.17	80.09	100	77.50	61.94
FT	81.48	96.39	61.67	86.39	82.13	75.00	88.89	82.50	74.17	74.72	70.56	77.22
FT + TTC	68.98	97.50	36.67	72.78	69.72	93.89	51.11	64.17	66.39	96.11	41.11	61.94
GED-FT	77.31	97.78	85.83	48.33	89.63	98.06	86.11	84.72	85.74	96.94	80.28	80.00
GED-FT + TTC	84.63	100	61.94	92.22	76.85	100	72.78	57.78	74.72	100	70.56	52.78
Perturbed												
Base	76.30	72.50	88.61	67.78	54.06	24.44	71.72	72.00	24.07	24.72	23.06	24.44
Base + TTC	32.04	38.06	44.72	13.33	31.11	18.61	45.83	28.89	35.07	95.83	6.94	3.00
FT	73.33	95.00	31.39	93.61	69.35	23.06	92.78	92.22	42.87	23.33	47.22	58.06
FT + TTC	63.89	83.61	35.00	73.06	47.59	74.72	40.28	27.78	43.89	86.94	27.50	16.94
GED-FT	84.35	95.83	73.61	83.61	76.94	94.72	79.44	56.67	58.98	93.61	45.00	38.33
GED-FT + TTC	83.70	99.44	56.11	96.56	66.20	98.33	68.33	31.06	56.50	100	42.50	25.06

IV. RESULTS AND DISCUSSION

A. Base-Model Robustness Gap

Tables I, II, and III show that base models perform well on canonical traces but degrade sharply under perturbation. The largest degradation is in Q_3 , confirming that first-error localization is substantially harder than final-answer checking. The drop is not merely a formatting artifact: for valid perturbed T1 traces, the base models falsely reject correct reasoning at high rates according to Q_2 : 75.56% for GPT-OSS 20B, 83.89% for Qwen3-14B, and 85.28% for Phi-4-Reasoning. These rates clearly quantify template sensitivity.

1) *Template Sensitivity*: Base models perform well on canonical traces but often reject admissible perturbed traces, even when every transition is accepted by the deterministic checker. This is clearest for T1 examples, where the correct judgment is unambiguous: the final answer is correct, the process is valid, and $Q_3 = -1$. On baseline T1 traces, base-model Q_2 accuracies are 98.33%, 93.89%, and 92.22% for GPT-OSS 20B, Qwen3-14B, and Phi-4-Reasoning. On perturbed T1 traces, these drop to 24.44%, 16.11%, and 14.72%, respectively. This gap suggests that the models are not only evaluating algebraic validity. They often treat the first unfamiliar transformation as an error, effectively mistaking procedural variation for incorrect reasoning. We refer to this behavior as *template sensitivity*: the evaluator appears calibrated to common solution forms rather than to the local validity of adjacent steps. This matters beyond the specific models tested here. In any setting where multiple valid procedures can lead to the same result, a verifier that rewards only familiar paths may suppress correct but non-standard reasoning.

2) *Outcome and Process Are Entangled*: T3 examples separate final-answer correctness from process correctness. This design tests whether the model can judge the outcome and the reasoning process independently. Under perturbation, this separation weakens. On T3 traces, base-model Q_1 accuracy drops from 96.67% to 67.78% for GPT-OSS 20B,

from 71.39% to 63.06% for Qwen3-14B, and from 99.44% to 56.94% for Phi-4-Reasoning. Since the final answer remains correct by construction, these drops indicate that an unfamiliar or suspicious-looking derivation can contaminate final-answer judgment. Conversely, because the final answer is correct, models may also overlook the invalid intermediate step. This coupling is undesirable for automated evaluation: a grader should not reject a correct answer merely because the derivation is unusual, nor accept flawed reasoning merely because it reaches the right endpoint.

3) *Localization Remains the Bottleneck*: Error localization is the most fragile part of step-level verification. Detecting that a trace is invalid is easier than identifying the first invalid transition, especially when the trace contains valid detours mixed with actual errors. On perturbed traces, base-model Q_3 accuracy falls to 24.07% for GPT-OSS 20B, 14.35% for Qwen3-14B, and 19.81% for Phi-4-Reasoning. This limitation is important because localization is what turns a judgment into useful feedback. In tutoring, a model that says a solution is wrong but points to the wrong step can mislead the student. In verifier-guided inference, poor localization may cause the generator to revise the wrong part of its reasoning. More generally, process-level evaluation should not stop at binary correctness: reliable evaluators must also identify where a reasoning chain first becomes invalid.

B. Impact of Supervised Fine-Tuning and Distillation

Supervised adaptation improves robustness to perturbed traces, but the effect is model dependent. The results suggest that fine-tuning does not simply improve “mathematical ability” in a uniform way; instead, it shifts the model’s decision boundary between accepting valid procedural variation and rejecting suspicious-looking traces.

For **GPT-OSS 20B**, GED-FT improves perturbed Q_2 from 54.06% to 76.94% and Q_3 from 24.07% to 58.98%. The strongest gain is on valid T1 perturbations, where GED-FT raises Q_2 from 24.44% to 94.72% and Q_3 from 24.72% to

TABLE II
TEST ACCURACY (%) FOR QWEN3-14B-BASED MODELS

Model	Outcome Accuracy (Q_1)				Step-Level Accuracy (Q_2)				Localization Accuracy (Q_3)			
	Overall	T1: Correct	T2: Incorrect	T3: Tricky	Overall	T1: Correct	T2: Incorrect	T3: Tricky	Overall	T1: Correct	T2: Incorrect	T3: Tricky
Baseline												
Base	83.61	96.94	82.50	71.39	86.48	93.89	82.50	83.06	86.02	93.89	82.22	81.94
Base + TTC	86.85	100	99.17	61.39	99.63	100	98.89	100	92.41	100	90.83	86.39
FT	75.65	55.56	75.56	95.83	78.33	49.17	89.17	96.67	77.87	49.17	87.78	96.67
FT + TTC	87.31	80.83	84.44	96.67	93.70	83.33	99.17	93.61	85.00	97.78	97.78	97.78
GED-FT	71.94	86.67	43.61	85.56	56.30	86.67	48.61	33.61	52.50	86.67	40.28	30.56
GED-FT + TTC	73.33	100	23.61	96.39	53.70	100	47.78	13.33	51.28	100	26.67	27.17
Perturbed												
Base	71.67	90.28	61.67	63.06	52.69	16.11	68.06	73.89	14.35	16.11	17.78	9.17
Base + TTC	34.35	28.06	71.94	3.06	53.61	18.33	72.50	70.00	30.09	80.00	9.17	0.00
FT	76.48	70.83	68.89	89.72	77.87	57.50	88.06	88.06	44.91	57.50	45.83	31.39
FT + TTC	78.24	83.33	81.39	70.00	84.54	73.33	91.11	89.17	81.67	81.67	83.33	80.00
GED-FT	59.35	87.78	18.06	72.22	41.85	85.56	29.44	10.56	36.57	85.56	17.78	6.39
GED-FT + TTC	74.91	98.61	32.78	93.33	56.57	90.28	57.78	21.67	44.91	88.89	33.61	12.22

TABLE III
TEST ACCURACY (%) FOR PHI-4-REASONING-BASED MODELS

Model	Outcome Accuracy (Q_1)				Step-Level Accuracy (Q_2)				Localization Accuracy (Q_3)			
	Overall	T1: Correct	T2: Incorrect	T3: Tricky	Overall	T1: Correct	T2: Incorrect	T3: Tricky	Overall	T1: Correct	T2: Incorrect	T3: Tricky
Baseline												
Base	96.48	92.22	97.78	99.44	96.11	92.22	97.78	98.33	95.93	92.22	97.22	98.33
Base + TTC	65.56	98.61	91.11	6.94	90.74	99.74	91.11	82.67	42.78	100	15.83	12.50
FT	68.15	100.00	4.72	99.72	72.96	99.72	48.89	70.28	38.98	99.72	8.06	9.17
FT + TTC	66.67	100	0.00	100	33.33	100	0.00	0.00	33.33	100	0.00	0.00
GED-FT	89.07	98.89	97.22	71.11	98.33	98.61	97.50	98.89	97.87	98.61	96.94	98.06
GED-FT + TTC	93.61	100	85.56	94.28	94.35	100	92.78	89.17	87.13	100	75.28	85.11
Perturbed												
Base	67.87	56.39	90.28	56.94	63.43	14.72	91.39	84.17	19.81	14.72	23.06	21.67
Base + TTC	25.00	14.72	58.33	1.94	38.06	11.94	58.33	43.89	33.89	98.89	3.06	0.00
FT	67.13	100.00	2.22	99.17	61.02	97.50	34.72	50.83	33.70	97.22	2.22	1.67
FT + TTC	66.67	100	0.00	100	33.33	100	0.00	0.00	33.33	100	0.00	0.00
GED-FT	96.39	99.44	95.83	93.89	96.76	93.61	96.94	99.72	93.70	93.61	91.39	96.11
GED-FT + TTC	91.31	100	76.44	97.50	95.93	100	91.39	98.39	85.11	100	78.39	77.50

93.61%. This indicates that distillation helps the model accept non-canonical but valid derivations, although some baseline trade-offs remain.

For **Qwen3-14B**, the best perturbed performance comes from FT+TTC, which raises localization from 14.35% to 81.67%. In contrast, GED-FT underperforms for this model, showing that teacher-rationale distillation is not uniformly beneficial. The usefulness of a distilled explanation appears to depend on the student model and on how well the explanation style matches its learned reasoning behavior.

For **Phi-4-Reasoning**, GED-FT is most effective, reaching 96.39% on perturbed Q_1 , 96.76% on Q_2 , and 93.70% on Q_3 . Standard FT, however, tends to over-accept traces, performing well on T1 while failing on T2. This shows why valid and invalid traces must be balanced: high acceptance of correct examples is not sufficient if the verifier loses its ability to reject flawed reasoning.

Overall, perturbation-oriented supervision can move models

toward process verification, but it must be calibrated carefully. Robust verifier training should preserve canonical competence while also teaching models to accept valid alternatives and reject subtle invalid transitions.

C. Test-Time Compute by Majority Voting

Majority-vote TTC helps only when model errors are noisy rather than systematic. If a model has learned the relevant verification rule but occasionally samples an incorrect judgment, voting can reduce variance. If the model applies the wrong heuristic consistently, voting reinforces that heuristic.

The base-model results show this risk. For GPT-OSS 20B, TTC reduces perturbed Q_1 from 76.30% to 32.04% and Q_2 from 54.06% to 31.11%. Qwen3-14B and Phi-4-Reasoning show similar drops in perturbed Q_1 . This suggests that more samples do not necessarily produce better verification; they can simply make an incorrect judgment more stable.

TTC is more useful after suitable adaptation. For Qwen3-14B, FT+TTC gives the strongest perturbed localization result,

reaching 81.67% on Q_3 . In this case, training appears to provide a better verification policy, and TTC reduces residual uncertainty. Thus, it should be viewed as a calibration tool, not a substitute for learning trace-validity semantics.

D. Limitations and Future Work

This study is intentionally limited to single-variable linear equations with unique rational solutions. This narrow setting enables exact label validation and unambiguous first-error indices, but it does not establish that the same behavior holds for inequalities, systems of equations, proofs, programs, or scientific workflows. The purpose of this benchmark is therefore diagnostic: it isolates process-level verification in a setting where the ground truth can be controlled precisely.

Future work should extend the same procedural-evaluation framework to richer domains. In mathematics, this includes equations with branching conditions, multi-solution problems, and proof-style arguments. Beyond mathematics, the same idea may apply to software debugging, scientific protocols, chemical synthesis plans, and other workflows where the final output is not enough and the validity of intermediate operations matters. Our current analysis also motivates finer-grained studies in future. Per-perturbation-family breakdowns and held-out perturbation tests would help determine whether models learn general trace validity or adapt to specific transformation templates. We also report single-run adaptation results, so small differences between variants should be interpreted cautiously. Finally, Gemini-enhanced distillation depends on the teacher model, prompt design, and filtering policy; future work should evaluate whether the observed gains hold across different teachers and larger verified rationale sets.

V. CONCLUSION

We introduced a controlled benchmark for evaluating LLMs as step-level verifiers rather than only as problem solvers. In a linear-equation setting with exact trace labels, base models that perform well on canonical solutions often fail on admissible non-canonical traces, especially when asked to localize the first invalid step. These failures show that evaluator robustness is a distinct capability from final-answer accuracy.

The broader implication is that trustworthy LLM evaluation requires attention to process, not only outcomes. Many real tasks allow more than one valid path, and an evaluator should be able to accept legitimate procedural variation. Our results show that adaptation can reduce this robustness gap, but its benefits are model dependent, and majority-vote TTC can amplify systematic mistakes. Controlled process-level benchmarks therefore provide a potential tool for measuring and improving LLM reliability in domains where the reasoning path itself must be verified, such as automated software verification, legal reasoning, and clinical diagnostics.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under Award No. 2434704.

REFERENCES

- [1] S. Bhandari and Z. Pardos, “Can language models grade algebra worked solutions? Evaluating LLM-based autograders against human grading,” in *Proceedings of the 18th International Conference on Educational Data Mining*, 2025, pp. 554–558.
- [2] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, “Let’s verify step by step,” in *International Conference on Learning Representations*, 2024. [Online]. Available: <https://openreview.net/forum?id=v8L0pN6EOi>
- [3] F. Mazdarani, A. C. Hernandez, and C. Toxtli, “Worker-centered AI: Transparent explanations for trustworthy task recommendation in crowd work,” in *2026 IEEE Conference on Artificial Intelligence*, 2026, pp. 295–302.
- [4] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, “Training verifiers to solve math word problems,” *CoRR*, vol. abs/2110.14168, 2021. [Online]. Available: <https://arxiv.org/abs/2110.14168>
- [5] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt, “Measuring mathematical problem solving with the MATH dataset,” *CoRR*, vol. abs/2103.03874, 2021. [Online]. Available: <https://arxiv.org/abs/2103.03874>
- [6] OpenAI, “Gpt-oss: Open-weight reasoning models,” OpenAI, Tech. Rep., 2025. [Online]. Available: <https://github.com/openai/gpt-oss>
- [7] Qwen Team, “Qwen3 technical report,” Alibaba Group, Tech. Rep., 2025. [Online]. Available: <https://github.com/QwenLM/Qwen3>
- [8] Microsoft, “Phi-4-reasoning: Redefining small language model logic,” Microsoft Research, Tech. Rep., 2025. [Online]. Available: <https://huggingface.co/microsoft/phi-4-reasoning>
- [9] P. Wang, L. Li, Z. Shao, R. X. Xu, D. Dai, Y. Li, D. Chen, Y. Wu, and Z. 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*, 2024, pp. 9426–9439. [Online]. Available: <https://aclanthology.org/2024.acl-long.510/>
- [10] K. Chang, Y. Shi, C. Wang, H. Zhou, C. Hu, X. Liu, Y. Luo, Y. Ge, T. Xiao, and J. Zhu, “Step-level verifier-guided hybrid test-time scaling for large language models,” in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 18 462–18 477.
- [11] R. Kamoi, Y. Zhang, N. Zhang, S. S. D. Sarkar, and R. Zhang, “Generalizable process reward models via formally verified training data,” *arXiv preprint arXiv:2505.15960*, 2025.
- [12] X. Li, X. Li, S. Hu, Y. Guo, and W. Zhang, “Verifybench: A systematic benchmark for evaluating reasoning verifiers across domains,” *arXiv preprint arXiv:2507.09884*, 2025.
- [13] Q. Li, L. Cui, X. Zhao, L. Kong, and W. Bi, “GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024, pp. 2961–2984. [Online]. Available: <https://aclanthology.org/2024.acl-long.163/>
- [14] I. Mirzadeh, K. Alizadeh, H. Sedghi, S. Bengio, and M. Farajtabar, “GSM-symbolic: Understanding the limitations of mathematical reasoning in large language models,” *arXiv preprint arXiv:2410.05229*, 2024. [Online]. Available: <https://arxiv.org/abs/2410.05229>
- [15] K. Huang, J. Guo, Z. Li, X. Ji, J. Ge, W. Li, Y. Guo, T. Cai, H. Yuan, R. Wang, Y. Wu, M. Yin, S. Tang, Y. Huang, C. Jin, X. Chen, C. Zhang, and M. Wang, “MATH-perturb: Benchmarking LLMs’ math reasoning abilities against hard perturbations,” *arXiv preprint arXiv:2502.06453*, 2025. [Online]. Available: <https://arxiv.org/abs/2502.06453>
- [16] G. Chatziveroglou, R. Yun, and M. Kelleher, “Exploring LLM reasoning through controlled prompt variations,” *arXiv preprint arXiv:2504.02111*, 2025. [Online]. Available: <https://arxiv.org/abs/2504.02111>
- [17] Z. Sun, G. Dai, I. Tsang, and H. Ye, “Numerical sensitivity and robustness: Exploring the flaws of mathematical reasoning in large language models,” *arXiv preprint arXiv:2511.08022*, 2025. [Online]. Available: <https://arxiv.org/abs/2511.08022>
- [18] Z. Chen, W. Du, W. Zhang, K. Liu, J. Liu, M. Zheng, J. Zhuo, S. Zhang, D. Lin, K. Chen, and F. Zhao, “T-Eval: Evaluating the tool utilization capability of large language models step by step,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024, pp. 9510–9529.
- [19] Google, “Introducing Gemini 3 Flash: Benchmarks, global availability,” 2025, [Online]. Available: <https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/>