EQUATION10K : TEACHER-STYLE ERROR DIAGNOSIS IN HANDWRITTEN ARITHMETIC SOLUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models (VLMs) are increasingly considered for educational assessment, yet existing mathematical benchmarks mostly ask models to solve a problem, recognize handwritten expressions, or locate a single error. A teacher must do more: read a multi-line handwritten solution, verify each transformation, identify the earliest misconception, distinguish a new mistake from downstream error propagation, and remain reliable across diverse arithmetic structures and handwriting styles. We introduce EQUATION10K , a procedurally generated benchmark for fine-grained diagnosis of handwritten step-by-step arithmetic. The benchmark contains 10,000 solution images spanning 10 problem families, five difficulty levels, 12 handwriting styles, and 4,272 operator-structure templates. Each solution is paired with executable step traces and labels for correctness, first-error position, and five diagnostic states: wrong value, wrong operator, incorrect order of operations, calculation error, and propagated error. We evaluate representative proprietary and open VLMs using step correctness, error-type accuracy, joint step accuracy, exact solution grading, and first-error localization. Our experiments reveal two consistent patterns: joint diagnostic accuracy decreases at higher structural difficulty levels, and performance improves when handwritten inputs are replaced by cleaner printed or oracle-text representations, while non-trivial errors remain even under oracle text. These results indicate that both visual transcription and step-level process verification remain challenging for current VLMs. EQUATION10K provides a scalable testbed for studying multimodal process supervision and reliable AI-assisted grading. We open source the project at https://anonymous.4open.science/r/Equation10K-61BE.

1 INTRODUCTION

Automated answer grading is an important research direction in educational AI, with the goal of reducing grading workload while improving the efficiency and consistency of assessment Karan & Dewan (2026). Recent large language models (LLMs) have shown promising performance on text-based assessment tasks that rely on semantic understanding and answer matching Chen et al. (2026); Yao et al. (2026). However, grading handwritten mathematical work poses additional challenges. A handwritten solution may combine numbers, operators, fractions, exponents, and spatially structured notation, while correctness depends not only on the final answer but also on the sequence of intermediate transformations Levine et al. (2027). Consequently, reliable grading requires both visual interpretation of the written mathematics and step-level reasoning about how one state is transformed into the next.

A mathematics teacher does not grade a worked solution by checking only its final answer. Consider the student response in Figure 1. The first two transformations are correct. The third displayed step introduces an operator error, while the final step is locally consistent with the already corrupted intermediate state and is therefore labeled as a propagated error. A useful grading system should distinguish the source of an error from its downstream consequences. This distinction matters for tutoring, feedback generation, and scalable assessment: treating every later incorrect state as an independent mistake is less informative than identifying the first erroneous transition and recognizing when subsequent steps merely propagate it.

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

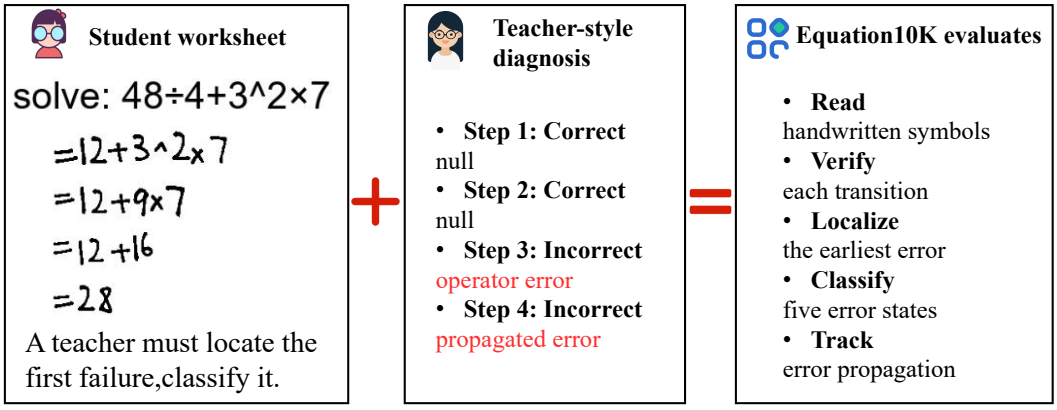

Figure 1: **Running example.** Teacher-style grading requires more than final-answer verification. A model must read handwriting, verify each transition, localize the first error, assign a diagnostic category, and track whether later incorrect lines merely propagate an earlier mistake.

Table 1: Comparison with representative benchmarks. "Propagation" denotes explicit distinction between a new error and a later step that only inherits an earlier wrong state.

Benchmark	Visual	Handwritten	Step-level	First error	Error type	Propagation	Procedural scale
MathVista (Lu et al., 2024)	✓	✗	✗	✗	✗	✗	✗
DynaMath (Zou et al., 2025)	✓	✗	✗	✗	✗	✗	✓
ProcessBench (Zheng et al., 2025)	✗	✗	✓	✓	✗	✗	✗
PRMBench (Song et al., 2025)	✗	✗	✓	–	✓	✗	✗
FERMAT (Nath et al., 2025)	✓	✓	✓	✓	✓	✗	✗
EQUATION10K	✓	✓	✓	✓	✓	✓	✓

Existing benchmarks cover different parts of this problem but do not jointly evaluate them. Visual mathematical reasoning benchmarks such as MathVista and DynaMath evaluate multimodal problem solving and robustness (Lu et al., 2024; Zou et al., 2025); MathCheck further argues that a benchmark should expose capability boundaries rather than collapse reasoning into a single score (Zhou et al., 2025). However, these benchmarks primarily evaluate *solving* mathematical problems. ProcessBench and PRMBench evaluate step-level verification, but operate on textual reasoning traces rather than handwritten student work (Zheng et al., 2025; Song et al., 2025). FERMAT directly studies handwritten mathematical grading and demonstrates that current VLMs still struggle with handwritten reasoning (Nath et al., 2025), but its core tasks emphasize detection, localization, and correction over a smaller collection of manually curated school-math solutions. The resulting gap is a benchmark that combines **visual handwriting**, **step-level verification**, **fine-grained error diagnosis**, **explicit error propagation**, and **procedurally controlled difficulty** at scale.

Table 1 summarizes this gap. Our design follows a broader trend in benchmark research: procedurally controllable generation for robustness (Zou et al., 2025), structured construction pipelines that preserve traceability (Luo et al., 2025), and diagnostic evaluation dimensions rather than a single end-task score (Zhu et al., 2024; Xie et al., 2025).

Research questions. We organize the paper around three questions. **RQ1:** How can we construct a scalable benchmark that preserves the causal structure of student mistakes while varying mathematical structure and handwriting? **RQ2:** How does diagnostic performance vary across models, structural difficulty levels, and arithmetic problem families? **RQ3:** To what extent is diagnostic performance affected by the input representation, and how much error remains when visual transcription is simplified or removed?

Design considerations. A useful benchmark for this setting should satisfy five properties. It should be (i) *step-faithful*, with an executable trace that certifies every transition; (ii) *diagnostic*, separating the source of a newly introduced error from its downstream propagation; (iii) *visually diverse*, so that success does not reduce to recognition of a single clean rendering style; (iv) *structurally broad*,

spanning multiple operators, parentheses, fractions, decimals, and exponents; and (v) *scalable and reproducible*, allowing new instances to be generated without manual rewriting.

Contributions. Our main contributions are:

- We introduce EQUATION10K , a 10K-example benchmark for teacher-style diagnosis of handwritten arithmetic solutions, covering 10 problem families, five difficulty levels, and 12 handwriting styles (Section 3).
- We develop an executable construction pipeline that enumerates expression structures, performs exact arithmetic with an AST-based solver, injects controlled student-like errors, and automatically labels downstream error propagation (Section 3.2).
- We define a multi-level evaluation protocol combining step correctness, diagnostic error classification, exact solution grading, and first-error localization, enabling analysis beyond final-answer accuracy (Section 4).
- We conduct systematic analyses across model families, structural difficulty levels, arithmetic problem families, and input representations. The results show that performance deteriorates at higher structural difficulty levels and improves substantially when visual transcription is simplified, while non-trivial diagnostic errors remain even under oracle-text input (Section 6).

2 RELATED WORK

Visual mathematical reasoning. MathVista combines diverse mathematical and visual tasks and shows a persistent human–model gap in multimodal reasoning (Lu et al., 2024). DynaMath uses programmatic generation to test whether VLM reasoning remains stable under controlled visual and numeric variants (Zou et al., 2025). MathCheck emphasizes behavior-level robustness across transformed tasks (Zhou et al., 2025), while Omni-MATH studies substantially harder mathematical reasoning in text (Gao et al., 2025). These benchmarks motivate our emphasis on controllable generation and capability-oriented analyses, but they do not model the pedagogical task of diagnosing a student's intermediate work.

Process verification. A parallel literature argues that final-answer correctness is insufficient for reliable reasoning. Process-based feedback exposes reasoning errors that outcome supervision can miss (Uesato et al., 2022), and PRM800K demonstrates the value of dense step-level supervision (Lightman et al., 2024). Math-Shepherd constructs process supervision automatically for verification and reinforcement learning (Wang et al., 2024). For evaluation, ProcessBench asks models to identify the earliest erroneous step in textual solutions (Zheng et al., 2025), while PRMBench measures fine-grained process-reward-model behavior across challenging error patterns (Song et al., 2025). EQUATION10K shares the emphasis on intermediate correctness but adds handwritten perception and an explicit PROPAGATION state, distinguishing a newly introduced error from a locally valid transformation applied to an already corrupted state.

Handwritten mathematics and educational assessment. Handwritten mathematical expression recognition has traditionally focused on transcription. CROHME standardizes online, offline, and bimodal recognition (Xie et al., 2023); Watch, Attend and Parse introduced end-to-end image-to-LaTeX recognition (Zhang et al., 2017); and later systems improve structural modeling and robustness with syntax-aware decoding, HME100K-scale data, and bidirectional attention (Yuan et al., 2022; Han et al., 2022). Recognition, however, is only the first stage of grading: a useful assessor must verify transformations and diagnose mistakes. FERMAT is the closest prior benchmark, evaluating detection, localization, and correction on handwritten student solutions (Nath et al., 2025). Equation10K is complementary: by restricting the domain to executable arithmetic, it scales to 10K examples while retaining exact provenance for every intermediate state, error source, and downstream propagation label.

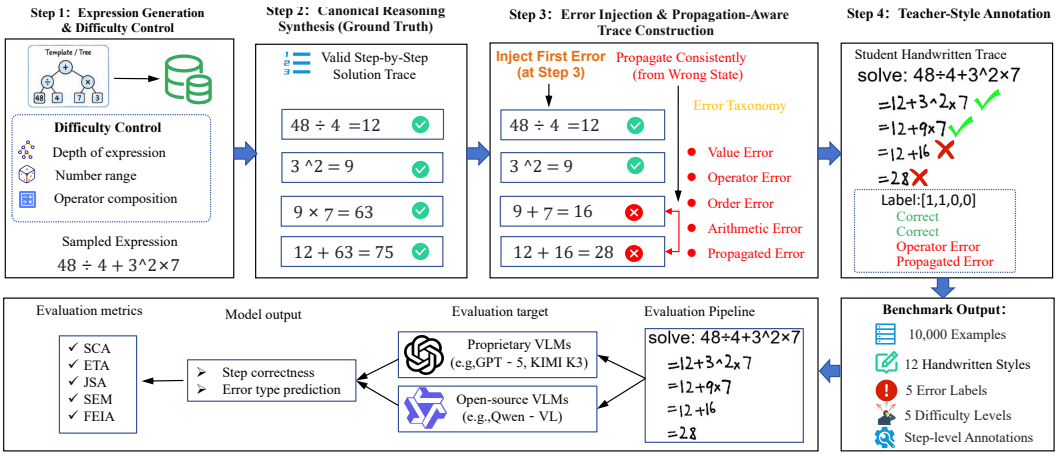

Figure 2: **A Teacher-Style Eror Diagnosis Benchmark for Handwritten Arithmetic Solutions.** Expression structure, numeric realization, exact AST execution, controlled error injection, and handwriting rendering are separated so that every benchmark label remains traceable to an executable state transition.

3 THE EQUATION10K BENCHMARK

3.1 TASK FORMULATION

Each sample contains an original arithmetic expression q, a rendered handwritten image I, and an ordered sequence of displayed student steps $s_1, \ldots, s_T$. For each step t, the benchmark provides a correctness label $c_t \in \{0, 1\}$ and, when $c_t = 0$, an error label e_t from

$$\mathcal{E} = \{\text{VALUE}, \text{OPERATOR}, \text{ORDER}, \text{CALCULATION}, \text{PROPAGATION}\}.$$

The model must return one structured prediction for every visible line. We additionally define the first-error index

$$f = \min\{t : c_t = 0\},$$

with $f = \varnothing$ for a completely correct solution. This makes the benchmark simultaneously a perception task, a local transition-verification task, a sequence diagnosis task, and a causal error-localization task.

The five states correspond to the implementation labels: *carelessly wrote down the wrong value*, *wrong operation symbol*, *incorrect parentheses or order of operations*, *calculation error*, and *pre-order error propagation*. We use the shorter names in the paper for readability while preserving the exact strings in the released evaluator.

3.2 CONSTRUCTION PIPELINE (RQ1)

Figure 2 summarizes the pipeline. Its central principle is to generate the *reasoning trace* before rendering the image, so the ground truth never depends on post-hoc visual annotation.

Structure enumeration. For expressions containing three to five numbers, we enumerate operator sequences over $\{+, -, \times, \div\}$ and recursively enumerate binary-tree parenthesizations. This produces 4,272 structural templates. Each template is assigned a difficulty score based on operator diversity, precedence changes, same-level operator mixing, operand count, and parenthesis depth. The score is discretized into five levels, ranging from straightforward same-operator computation to nested parentheses and multiple interacting rules.

Problem families and numeric realization. The benchmark contains 10 families: addition, subtraction, multiplication, division, mixed operations, parenthesized expressions, fractions, decimals, exponentiation, and random combinations. Numeric values are sampled from curated integer, decimal, and fraction pools. Every instantiated expression is converted into an exact symbolic form and rejected if it is syntactically invalid, divides by zero, or produces an extreme value.

Executable step traces. We parse each expression into an abstract syntax tree (AST). At each iteration, the solver selects the currently valid operation according to parentheses and operator precedence, evaluates exactly using rational arithmetic, replaces the corresponding node, and records the before/after expressions. This produces a deterministic gold trace for correct solutions.

Student-like error injection. At a selected step, the generator may inject one primary error: (1) copy or write a wrong value, (2) replace the operation symbol, (3) violate parentheses or operation order, or (4) produce an incorrect numerical result while using the intended operation. The generator verifies that the corrupted AST remains executable. Once a primary error occurs, subsequent locally valid transformations are labeled PROPAGATION. This implements the teacher-like distinction illustrated in Figure 1: a later step can be locally valid but globally incorrect because its input state is already wrong.

Handwriting rendering. The trace is rendered as a multi-line solution image using one of 12 handwriting fonts. Font size, line spacing, word spacing, and per-character spatial perturbations add visual variability. The original expression is shown as the problem statement, while only the displayed solution lines are graded. The benchmark evaluator sends no ground-truth text or metadata to the VLM.

3.3 COMPOSITION AND STATISTICS

The target release contains 10,000 images: 200 examples for every problem-family/difficulty cell. In our fixed-seed generation pass, the benchmark contains 37,048 displayed steps (3.70 per solution on average). Difficulty increases trace length from 2.70 steps at level 1 to 4.70 at level 5. Table 2 summarizes the benchmark.

rimary error categories are sampled only when they are structurally feasible. In particular, order-of-operations errors require a suitable competing operation or parenthesized structure and are therefore naturally rarer than value, operator, or calculation errors. We retain these structural constraints rather than forcing semantically artificial corruptions. Consequently, the diagnostic labels are not class-balanced, and aggregate error-type accuracy should be interpreted with this class distribution in mind.

Table 2: EQUATION10K statistics from the fixed-seed generator pass used in our experiments.

Statistic	Value
Solution images	10,000
Displayed solution steps	37,048
Problem families	10
Difficulty levels	5
Handwriting styles	12
Expression templates	4,272
Mean / median steps	3.70 / 4
Clean / error-containing solutions	5,000 / 5,000
Correct step labels	22,437
Primary error steps	5,000
Propagation steps	9,611

4 EVALUATION PROTOCOL

4.1 MODELS AND INFERENCE

We evaluate ten representative vision language models covering proprietary systems, large hosted models, and locally deployed open models. The evaluated systems are GPT-5.6-Sol [1], KIMI K3 [2], Claude-Opus-4.5 [3], Deepseek-v4-Flash-Vision-Exp [4], Qwen3.8-Max [5], Qwen3-VL-Plus [5], Qwen3.8-Flash-Next [5], Qwen3.8-27B Qwen Team (2026), Qwen3-VL-8B-Instruct Team (2025), and Qwen2.5-VL-7B-Instruct Bai et al. (2025). This selection spans a broad range of model scales and deployment settings, allowing us to compare frontier systems with smaller open alternatives under the same diagnostic task.

[1] https://developers.openai.com/api/docs/models/gpt-5.6-sol
[2] https://platform.kimi.ai
[3] https://www.anthropic.com/claude/opus
[4] https://api.deepseek.com
[5] https://qwen.ai/apiplatform

Table 3: **Main results on EQUATION10K .** Higher values indicate better performance for all metrics.

Model	SCA ↑	ETA ↑	JSA ↑	SEM ↑	FEIA ↑
GPT-5.6 sol	0.831	0.898	0.780	0.613	0.693
KIMI K3	0.823	0.871	0.769	0.606	0.681
Qwen3.8-Max	0.813	0.891	0.770	0.597	0.679
Claude-Opus-4.5	0.816	0.869	0.759	0.599	0.674
Deepseek-v4-flash-vision-exp	0.819	0.894	0.775	0.597	0.677
Qwen3-vl-plus	0.799	0.763	0.765	0.582	0.675
Qwen3.8-Flash-Next	0.761	0.748	0.759	0.578	0.662
Qwen/Qwen3.8-27B	0.649	0.552	0.579	0.192	0.486
Qwen3-VL-8B-Instruct	0.482	0.376	0.272	0.075	0.385
Qwen2.5-VL-7B-Instruct	0.476	0.223	0.147	0.001	0.239

4.2 EVALUATION METRICS

We report five complementary metrics already implemented by the evaluator.**Step Correctness Accuracy (SCA)** evaluates the binary correct/incorrect decision; **Error-Type Accuracy (ETA)** is computed on ground-truth error steps; **Joint Step Accuracy (JSA)** requires both correctness and the exact error type; **Sample Exact Match (SEM)** requires all steps in a solution to be jointly correct; and **First-Error Index Accuracy (FEIA)** measures whether the earliest error, including "no error," is correctly localized. We treat JSA and FEIA as the primary metrics because they most directly reflect teacher-style diagnosis. Detailed calculation formulas for each metrics are provided in Appendix B.

5 EXPERIMENTAL SETUP

We evaluate all models on the same EQUATION10K using an identical prompt and output format. Hosted models are accessed through their corresponding inference interfaces, while open models are evaluated locally using five NVIDIA RTX 3090 GPUs. All models use the same task instruction and structured output schema and deterministic decoding strategy.The prompts are provided in the appendix.

6 RESULTS AND ANALYSIS

6.1 OVERALL PERFORMANCE (RQ2)

Table 3 reports overall performance on EQUATION10K . GPT-5.6-Sol obtains an SCA of 0.831, an ETA of 0.898, a JSA of 0.780, an SEM of 0.613, and an FEIA of 0.693. Several other large hosted models achieve similar step-level performance, but none exceeds 0.700 FEIA or 0.620 SEM. Thus, relatively strong step-level accuracy does not translate directly into fully consistent grading of complete multi-step solutions.

A substantial gap emerges between step-level evaluation and complete-solution grading. For GPT-5.6-Sol, SCA is 0.831 whereas SEM is 0.613, a difference of 21.8 percentage points. This gap reflects the compounding effect of local mistakes: a sample is counted as correct under SEM only when every step is jointly classified correctly. JSA is also lower than SCA, reflecting the additional requirement that an incorrect step receive the correct diagnostic label rather than only a binary correctness decision.

First-error prediction remains challenging. The highest FEIA in Table 3 is 0.693, despite several models achieving substantially higher step-level correctness scores. FEIA is an end-to-end metric: for erroneous solutions, the model must identify the earliest incorrect step rather than a later consequence, while for clean solutions it must correctly predict that no error is present. The difference between FEIA and step-level accuracy therefore highlights the additional sequence-level consistency required by teacher-style diagnosis.

The gap between stronger frontier systems and smaller open models is also pronounced. Qwen3-VL-8B-Instruct reaches only 0.272 JSA and 0.075 SEM, while Qwen2.5-VL-7B-Instruct falls to

Table 4: **Joint step accuracy across difficulty levels on EQUATION10K .** Higher levels correspond to greater structural complexity in the generated arithmetic expressions.

Model	JSA↑				
	Level 1	Level 2	Level 3	Level 4	Level 5
GPT-5.6 sol	0.822	0.811	0.810	0.802	0.655
KIMI K3	0.808	0.787	0.787	0.783	0.680
Qwen3.8-Max	0.806	0.790	0.783	0.780	0.691
Claude-Opus-4.5	0.799	0.775	0.797	0.763	0.661
Deepseek-v4-flash-vision-exp	0.806	0.805	0.780	0.779	0.704
Qwen3-vl-plus	0.802	0.787	0.785	0.683	0.769
Qwen3.8-Flash-Next	0.797	0.792	0.789	0.781	0.636
Qwen/Qwen3.8-27B	0.618	0.615	0.592	0.596	0.473
Qwen3-VL-8B-Instruct	0.301	0.291	0.286	0.279	0.203
Qwen2.5-VL-7B-Instruct	0.179	0.158	0.157	0.151	0.089

0.147 JSA and nearly zero SEM. These results indicate that success on isolated visual arithmetic operations does not directly translate into reliable process level assessment. Accurate grading requires the model to maintain consistency across the full solution trace while simultaneously reasoning about correctness, error semantics, and causal propagation.

6.2 PERFORMANCE DEGRADES AT HIGHER STRUCTURAL DIFFICULTY (RQ2)

Table 4 reports JSA across the five benchmark difficulty levels. Every evaluated model obtains lower JSA at Level 5 than at Level 1, and the average across models decreases from 0.674 to 0.556. For several stronger systems, performance is comparatively stable over Levels 1–4 before dropping more sharply at Level 5. For example, GPT-5.6-Sol changes from 0.822 at Level 1 to 0.802 at Level 4 and then decreases to 0.655 at Level 5.

These results show that the benchmark becomes more challenging at higher structural difficulty levels. However, the difficulty score jointly reflects several factors, including operator diversity, precedence interactions, operand count, parenthesis depth, and longer solution traces. The observed degradation should therefore not be attributed to a single causal factor such as compositional reasoning alone. Instead, it indicates that combinations of increased structural and sequential complexity expose limitations that are less visible on simpler instances.

Difficulty also changes the relative ranking of models. GPT-5.6-Sol achieves the highest JSA at Level 1 and remains among the strongest systems through Level 4, yet Qwen3-vl-plus obtains the highest Level 5 score at 0.769. Deepseek-v4-flash-vision-exp also reaches 0.704 at Level 5, exceeding several models with stronger overall results. These ranking changes indicate that aggregate benchmark performance does not fully characterize robustness to compositional complexity. Models with similar overall accuracy can exhibit substantially different behavior once multiple arithmetic rules must be coordinated within the same solution.

The smaller open models show a different but equally clear limitation. Qwen3-VL-8B-Instruct declines from 0.301 at Level 1 to 0.203 at Level 5, while Qwen2.5-VL-7B-Instruct decreases from 0.179 to 0.089. Their performance is already low on structurally simpler examples and deteriorates further as difficulty increases. Taken together, these results show that compositional difficulty amplifies existing weaknesses in process diagnosis rather than merely introducing a uniform penalty across models.

6.3 PERFORMANCE ACROSS PROBLEM FAMILIES (RQ2)

Table 5 reports joint step accuracy across the ten problem families in EQUATION10K . In contrast to the pronounced degradation across difficulty levels, performance varies only moderately across arithmetic families. GPT-5.6-Sol ranges from 0.763 on fractions to 0.797 on addition. KIMI K3 ranges from 0.754 on decimals to 0.786 on exponentiation. Similar ranges are observed for the other strong models. This relative stability suggests that no single arithmetic family accounts for the overall difficulty of the benchmark.

Table 5: **Joint step accuracy (JSA) across problem families on EQUATION10K .** Results are reported separately for the ten arithmetic families used in benchmark construction.

Model	JSA↑									
	addition	subtraction	multiplication	division	mixed operations	parentheses	fraction	decimal	exponentiation	random combination
GPT-5.6 sol	0.797	0.790	0.771	0.781	0.774	0.772	0.763	0.793	0.765	0.793
KIMI K3	0.761	0.759	0.784	0.784	0.768	0.758	0.761	0.754	0.786	0.776
Qwen3.8-Max	0.786	0.766	0.770	0.751	0.762	0.788	0.764	0.779	0.758	0.775
Claude-Opus-4.5	0.756	0.739	0.774	0.752	0.741	0.742	0.779	0.778	0.754	0.774
Deepseek-v4-flash-vision-exp	0.782	0.779	0.787	0.773	0.788	0.766	0.752	0.776	0.771	0.777
Qwen3-vl-plus	0.772	0.770	0.746	0.764	0.769	0.751	0.772	0.777	0.760	0.771
Qwen3.8-Flash-Next	0.777	0.746	0.763	0.757	0.759	0.767	0.760	0.760	0.740	0.760
Qwen/Qwen3.8-27B	0.571	0.570	0.566	0.588	0.570	0.592	0.591	0.564	0.588	0.590
Qwen3-VL-8B-Instruct	0.258	0.269	0.262	0.283	0.263	0.277	0.280	0.279	0.269	0.280
Qwen2.5-VL-7B-Instruct	0.159	0.130	0.158	0.136	0.136	0.147	0.159	0.149	0.144	0.152

The strongest model also depends on the problem family. GPT-5.6-Sol performs best on addition, subtraction, decimals, and random combinations. KIMI K3 performs best on division and exponentiation. Deepseek-v4-flash-vision-exp performs best on multiplication and mixed operations, Qwen3.8-Max on parenthesized expressions, and Claude-Opus-4.5 on fractions. KIMI K3 obtains the strongest result on exponentiation. These differences indicate that models exhibit distinct arithmetic capability profiles even when their aggregate performance is similar.

More importantly, structurally richer problem families are not uniformly harder than elementary arithmetic categories. For example, GPT-5.6-Sol reaches 0.793 on random combinations, which is higher than its scores on multiplication, mixed operations, parentheses, fractions, and exponentiation. Across all evaluated models, the average performance difference between the strongest and weakest problem families is also relatively small. This finding contrasts with the much larger decline observed at the highest difficulty level.

Overall, variation across problem families is smaller than the degradation observed across structural difficulty levels. This suggests that family identity alone is not a sufficient proxy for benchmark difficulty: performance depends not only on whether an expression contains addition, division, fractions, or exponents, but also on the structural and sequential demands of the complete solution trace. We therefore analyze problem family and structural difficulty as complementary evaluation dimensions.

6.4 INPUT-REPRESENTATION ABLATION (RQ3)

To examine how input representation affects diagnostic performance, we evaluate two alternative representations of the same underlying solution traces. In the *Printed Image* condition, each trace is rendered using a clean typeset font while preserving its mathematical content and line structure. In the *Oracle Text* condition, the model receives a textual transcription of the problem and solution steps without access to correctness labels, diagnostic labels, or execution metadata. The printed-image comparison primarily reduces handwriting-specific visual variability, whereas the oracle-text condition removes visual transcription entirely and should be interpreted as an upper-bound representation rather than as a perfectly controlled isolation of a single perception factor.

Figure 3 shows that both handwriting perception and process reasoning contribute to the performance gap. Replacing handwriting with printed expressions improves JSA for every evaluated model. GPT-5.6-Sol increases from 0.780 to 0.832, Deepseek-v4-flash-vision-exp increases from 0.775 to 0.792, Qwen3.8 27B increases from 0.579 to 0.611, and Qwen2.5-VL-7B-Instruct increases from 0.157 to 0.197. These gains show that handwriting introduces a consistent visual recognition burden, although its magnitude varies considerably across models.

Providing oracle text produces a substantially larger improvement. GPT-5.6-Sol reaches 0.929, while Deepseek-v4-flash-vision-exp reaches 0.902. The gains are even larger for the smaller models. Qwen3.8 27B improves from 0.579 with handwritten input to 0.812 with oracle text, while Qwen2.5-VL-7B-Instruct rises from 0.157 to 0.589. The larger improvement from textual access indicates that difficulty cannot be attributed only to recognizing handwritten symbols. Recovering a reliable symbolic representation from visual solution traces remains an important part of the challenge.

Importantly, oracle text does not eliminate diagnostic errors. GPT-5.6-Sol obtains 0.929 JSA and Deepseek-v4-flash-vision-exp obtains 0.902 even when the solution trace is provided symbolically. Thus, explicit transcription is not sufficient for perfect teacher-style diagnosis: models must still verify each transformation and assign consistent step-level labels. Taken together, the ablation indicates that

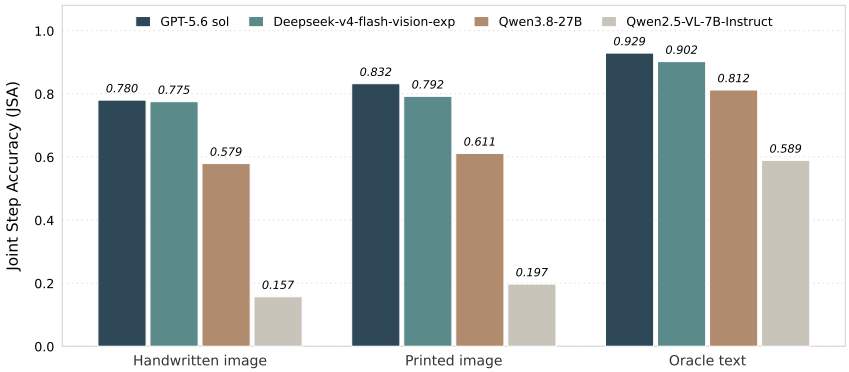

Figure 3: **Modality ablation on EQUATION10K .** JSA is evaluated using handwritten images, matched printed images, and oracle text representations of the same solution traces.

visual representation is one source of error, while residual errors under oracle text demonstrate that textual process verification and diagnostic consistency also remain imperfect.

The ablation therefore reveals two complementary bottlenecks. Handwritten input creates a perceptual challenge, while the remaining errors under oracle text expose limitations in mathematical process reasoning and diagnostic consistency. The relative contribution of these factors also depends strongly on model capability, with weaker models benefiting much more from oracle transcription while still remaining substantially below the strongest systems.

7 LIMITATIONS AND ETHICAL CONSIDERATIONS

EQUATION10K is intentionally controlled and therefore does not capture the full diversity of real classrooms. The current version focuses on arithmetic expressions rather than word problems, proofs, diagrams, or free-form derivations. Synthetic handwriting does not reproduce all phenomena in real student work, such as crossed-out symbols, crowded margins, arrows, diagrams, idiosyncratic notation, or low-quality scans. Error injection is also model-based: although the four primary errors are designed to resemble common mistakes, they are not a substitute for a taxonomy derived from large-scale authentic classroom data. Future versions should add teacher-validated real student traces and multilingual notation.

Educational deployment introduces additional risks. A benchmark score does not establish that a VLM is safe to grade students autonomously. False diagnosis can produce misleading feedback or unfair assessment, particularly when handwriting quality varies across students. We therefore frame EQUATION10K as an evaluation and research resource for human-in-the-loop systems, not as evidence for replacing teachers. Any public release of future real student work should require appropriate consent, de-identification, and institutional review.

8 CONCLUSION

We presented EQUATION10K , a benchmark for evaluating whether VLMs can grade handwritten arithmetic in a teacher-like, step-aware manner. Its key contribution is not simply more visual math questions, but an executable link between each handwritten line and the causal structure of the student's reasoning. By combining procedural expression generation, exact AST traces, controlled error injection, propagation labels, diverse handwriting, and multiple diagnostic metrics, EQUATION10K turns a simple arithmetic domain into a precise test of multimodal process supervision. The experiments identify difficulty scaling, order-of-operations reasoning, propagation tracking, and handwriting perception as distinct capability bottlenecks. We hope the benchmark will support more reliable multimodal critics and educational assistants that can explain *where* reasoning went wrong rather than merely state that an answer is wrong.

LLM USAGE DISCLOSURE

We used an AI language model solely for language editing, including improving sentence structure, grammar, clarity, and overall readability of the manuscript. The AI tool was not used to generate research ideas, conduct experiments, analyze results, develop proofs, or produce substantive scientific content. All final content was reviewed and verified by the authors, who take full responsibility for the accuracy and integrity of the manuscript.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *CoRR*, abs/2502.13923, 2025. doi: 10.48550/ARXIV.2502.13923. URL https://doi.org/10.48550/arXiv.2502.13923.

Lei Chen, Tengteng Cheng, Boyu Gao, Zitao Liu, and Weiqi Luo. Learning from scoring disagreements: Contrastive error mining for efficient and robust llm-based assessment. In Sven Koenig, Chad Jenkins, and Matthew E. Taylor (eds.), *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, pp. 30243–30251. AAAI Press, 2026. doi: 10.1609/AAAI.V40I36.40275. URL https://doi.org/10.1609/aaai.v40i36.40275.

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

Xiaoxiang Han, Qiaohong Liu, Ziqi Han, Yuanjie Lin, and Naiyue Xu. Handwritten mathematical expression recognition via gcattention-based encoder and bidirectional mutual learning transformer. In *Pattern Recognition and Computer Vision: 5th Chinese Conference, PRCV 2022, Shenzhen, China, November 4–7, 2022, Proceedings, Part III*, pp. 282–294, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-18912-8. doi: 10.1007/978-3-031-18913-5_22. URL https://doi.org/10.1007/978-3-031-18913-5_22.

Vishaal K. Vincent Karan and M. Ali Akber Dewan. A review of automated short answer grading systems. In Robert A. Sottilare and Jessica Schwarz (eds.), *Adaptive Instructional Systems*, pp. 190–200, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-30015-7.

Jacob Levine, Miguel Aenlle, Craig Zilles, Matthew West, and Mariana Silva. Automated grading of handwritten mathematics using vision-capable llms. In Emmanuel G. Blanchard, Guanliang Chen, Min Chi, and Seiji Isotani (eds.), *Artificial Intelligence in Education*, pp. 627–636, Cham, 2027. Springer Nature Switzerland. ISBN 978-3-032-29755-6.

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

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=KUNzEQMWU7.

Tianqi Luo, Chuhan Huang, Leixian Shen, Boyan Li, Shuyu Shen, Wei Zeng, Nan Tang, and Yuyu Luo. nvbench 2.0: A benchmark for natural language to visualization under ambiguity. *CoRR*,

abs/2503.12880, 2025. doi: 10.48550/ARXIV.2503.12880. URL https://doi.org/10.48550/arXiv.2503.12880.

Oikantik Nath, Hanani Bathina, Mohammed Safi Ur Rahman Khan, and Mitesh M Khapra. Can vision-language models evaluate handwritten math? In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14784–14814, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.720. URL https://aclanthology.org/2025.acl-long.720/.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. PRMBench: A fine-grained and challenging benchmark for process-level reward models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 25299–25346, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1230. URL https://aclanthology.org/2025.acl-long.1230/.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

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 Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Yejing Xie, Harold Mouchère, Foteini Simistira Liwicki, Sumit Rakesh, Rajkumar Saini, Masaki Nakagawa, Cuong Tuan Nguyen, and Thanh-Nghia Truong. Icdar 2023 crohme: Competition on recognition of handwritten mathematical expressions. In *Document Analysis and Recognition - ICDAR 2023: 17th International Conference, San José, CA, USA, August 21–26, 2023, Proceedings, Part II*, pp. 553–565, Berlin, Heidelberg, 2023. Springer-Verlag. ISBN 978-3-031-41678-1. doi: 10.1007/978-3-031-41679-8_33. URL https://doi.org/10.1007/978-3-031-41679-8_33.

Yupeng Xie, Zhiyang Zhang, Yifan Wu, Sirong Lu, Jiayi Zhang, Zhaoyang Yu, Jinlin Wang, Sirui Hong, Bang Liu, Chenglin Wu, and Yuyu Luo. Visjudge-bench: Aesthetics and quality assessment of visualizations. *CoRR*, abs/2510.22373, 2025. doi: 10.48550/ARXIV.2510.22373. URL https://doi.org/10.48550/arXiv.2510.22373.

Xuechun Yao, Tianqi Liu, Chongkai Liu, and Hui Li. Application of knowledge enhancement bert model in automatic grading of student answers. In *Proceedings of the 2026 7th International Conference on Computer Information and Big Data Applications*, CIBDA '26, pp. 1693–1697, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722578. doi: 10.1145/3813822.3814095. URL https://doi.org/10.1145/3813822.3814095.

Ye Yuan, Xiao Liu, Wondimu Dikubab, Hui Liu, Zhilong Ji, Zhongqin Wu, and Xiang Bai. Syntax-aware network for handwritten mathematical expression recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4543–4552, 2022. doi: 10.1109/CVPR52688.2022.00451.

Jianshu Zhang, Jun Du, Shiliang Zhang, Dan Liu, Yulong Hu, Jinshui Hu, Si Wei, and Lirong Dai. Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition. *Pattern Recognition*, 71:196–206, 2017. ISSN 0031-3203.

doi: https://doi.org/10.1016/j.patcog.2017.06.017. URL https://www.sciencedirect.com/science/article/pii/S0031320317302376.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. ProcessBench: Identifying process errors in mathematical reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.50. URL https://aclanthology.org/2025.acl-long.50/.

Zihao Zhou, Shudong Liu, Maizhen Ning, Wei Liu, Jindong Wang, Derek F. Wong, Xiaowei Huang, Qiufeng Wang, and Kaizhu Huang. Is your model really A good math reasoner? evaluating mathematical reasoning with checklist. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=nDvgHIBRxQ.

Yizhang Zhu, Shiyin Du, Boyan Li, Yuyu Luo, and Nan Tang. Are large language models good statisticians? In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/729786203d330da046dd8091c2d92a66-Abstract-Datasets_and_Benchmarks_Track.html.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=VOAMTA8jKu.

APPENDIX CONTENTS

A Benchmark Error Taxonomy **14**

B Evaluation Metrics **14**

C Benchmark Samples **15**

D Visualization of 12 handwriting styles **16**

E Evaluation Prompt **17**

F Step Correctness Accuracy by Problem Family **17**

G Step Correctness Accuracy by Difficulty **17**

A BENCHMARK ERROR TAXONOMY

EQUATION10K distinguishes five diagnostic states for incorrect solution steps. The first four states identify the source of a newly introduced error, while PROPAGATION denotes a locally valid transformation that operates on an already corrupted state. This distinction allows the benchmark to separate the origin of a mistake from its downstream consequences.

Table 6 summarizes the paper terminology, the exact evaluator strings, and the corresponding definitions. We preserve the evaluator strings in the released annotations to maintain consistency with the benchmark generation and scoring pipeline.

Table 6: Step-level diagnostic states used by EQUATION10K .

Paper name	Evaluator string	Definition
VALUE	Carelessly wrote down the wrong value.	A number/value is copied, substituted, or written incorrectly.
OPERATOR	Wrong operation symbol	The operator itself changes, e.g., $+$ becomes $-$ or $\times$ becomes $\div$.
ORDER	Incorrect parentheses or order of operations	The student violates parentheses, precedence, or valid evaluation order.
CALCULATION	Calculation error	The intended operation is correct, but its arithmetic result is wrong.
PROPAGATION	Pre-order error propagation	The line is locally valid but continues a previously corrupted state.

B EVALUATION METRICS

We report five metrics for step-level error detection and classification. Let N be the number of samples, T_i the number of steps in sample i, and $T = \sum_{i=1}^{N} T_i$. For step j in sample i, y_{ij} and $\hat{y}_{ij}$ denote the ground-truth and predicted correctness labels, while e_{ij} and $\hat{e}_{ij}$ denote the corresponding error types.

Step Correctness Accuracy (SCA). SCA measures whether each step is correctly classified as correct or erroneous:

$$\mathrm{SCA} = \frac{1}{T} \sum_{i=1}^{N} \sum_{j=1}^{T_i} \mathbb{I}[\hat{y}_{ij} = y_{ij}]. \tag{1}$$

Error Type Accuracy (ETA). ETA measures error-type classification accuracy only on ground-truth erroneous steps:

$$\mathrm{ETA} = \frac{\sum_{i,j:y_{ij}=0} \mathbb{I}[\hat{e}_{ij} = e_{ij}]}{\sum_{i,j} \mathbb{I}[y_{ij} = 0]}. \tag{2}$$

Joint Step Accuracy (JSA). JSA requires both the correctness label and error type to be correct:

$$\mathrm{JSA} = \frac{1}{T} \sum_{i=1}^{N} \sum_{j=1}^{T_i} \mathbb{I}\left[\hat{y}_{ij} = y_{ij} \wedge \hat{e}_{ij} = e_{ij}\right], \tag{3}$$

where the error type of a correct step is defined as $\varnothing$.

First Error Index Accuracy (FEIA). Let f_i and $\hat{f}_i$ denote the first erroneous step in the ground truth and prediction, respectively. FEIA is

$$\mathrm{FEIA} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}[\hat{f}_i = f_i]. \tag{4}$$

If no error exists, the first-error index is defined as $\varnothing$.

Sample Exact Match (SEM). SEM measures the proportion of samples for which all step-level correctness labels and error types are exactly correct:

$$\mathrm{SEM} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[\forall j,\ \hat{y}_{ij} = y_{ij} \wedge \hat{e}_{ij} = e_{ij}\right]. \tag{5}$$

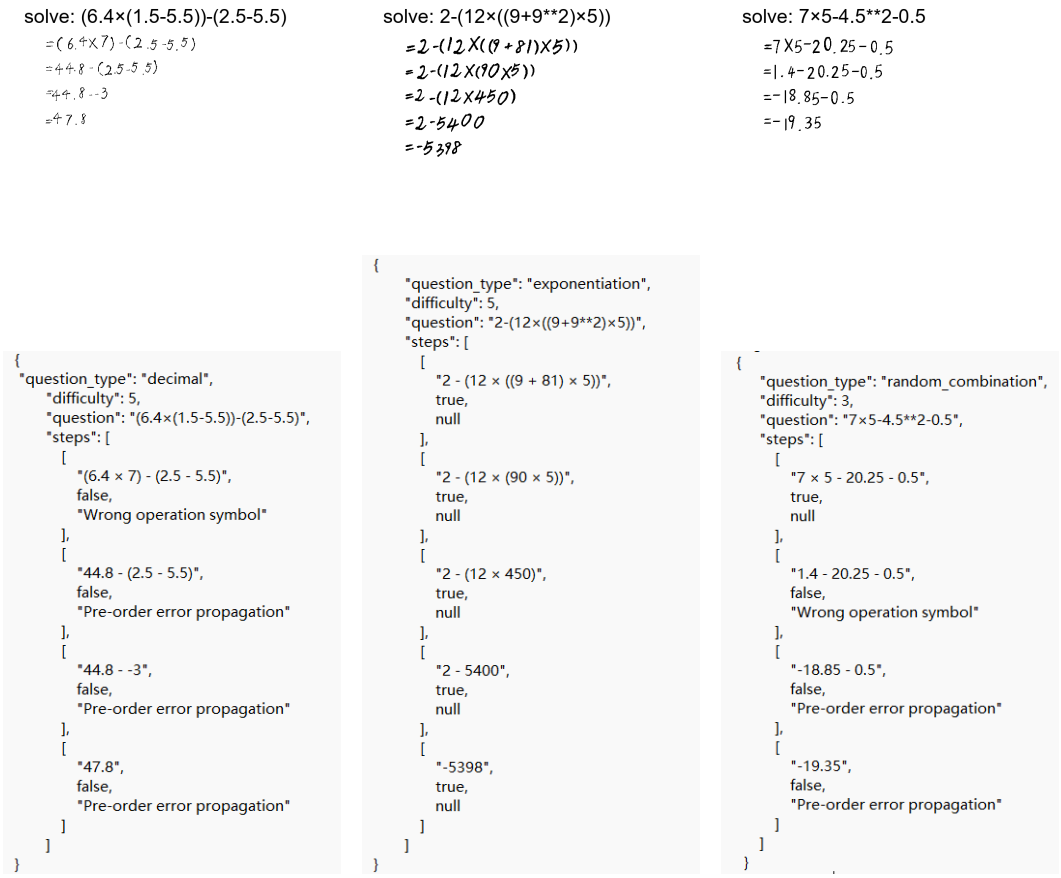

Figure 4: sample of EQUATION10K

SCA evaluates binary error detection, ETA evaluates error-type classification, JSA combines both at the step level, FEIA measures first-error localization, and SEM evaluates exact agreement at the sample level. “‘

C BENCHMARK SAMPLES

Figure 4 presents representative examples from EQUATION10K together with their structured annotations. Each example contains a handwritten solution image and the corresponding step level labels generated from the executable trace. The annotations record the problem family, difficulty level, original expression, displayed solution steps, correctness of each step, and diagnostic state when an error is present.

The examples illustrate how EQUATION10K distinguishes newly introduced mistakes from their downstream consequences. Panel **a** contains an incorrect operation followed by steps labeled as PROPAGATION. Panel **b** shows a fully correct solution in which every transformation is validated by the executable solver. Panel **c** provides another example in which an early operator error corrupts the intermediate state and subsequent locally valid transformations are labeled as PROPAGATION. These examples demonstrate that the benchmark preserves both the visual student solution and the causal structure of the underlying reasoning process.

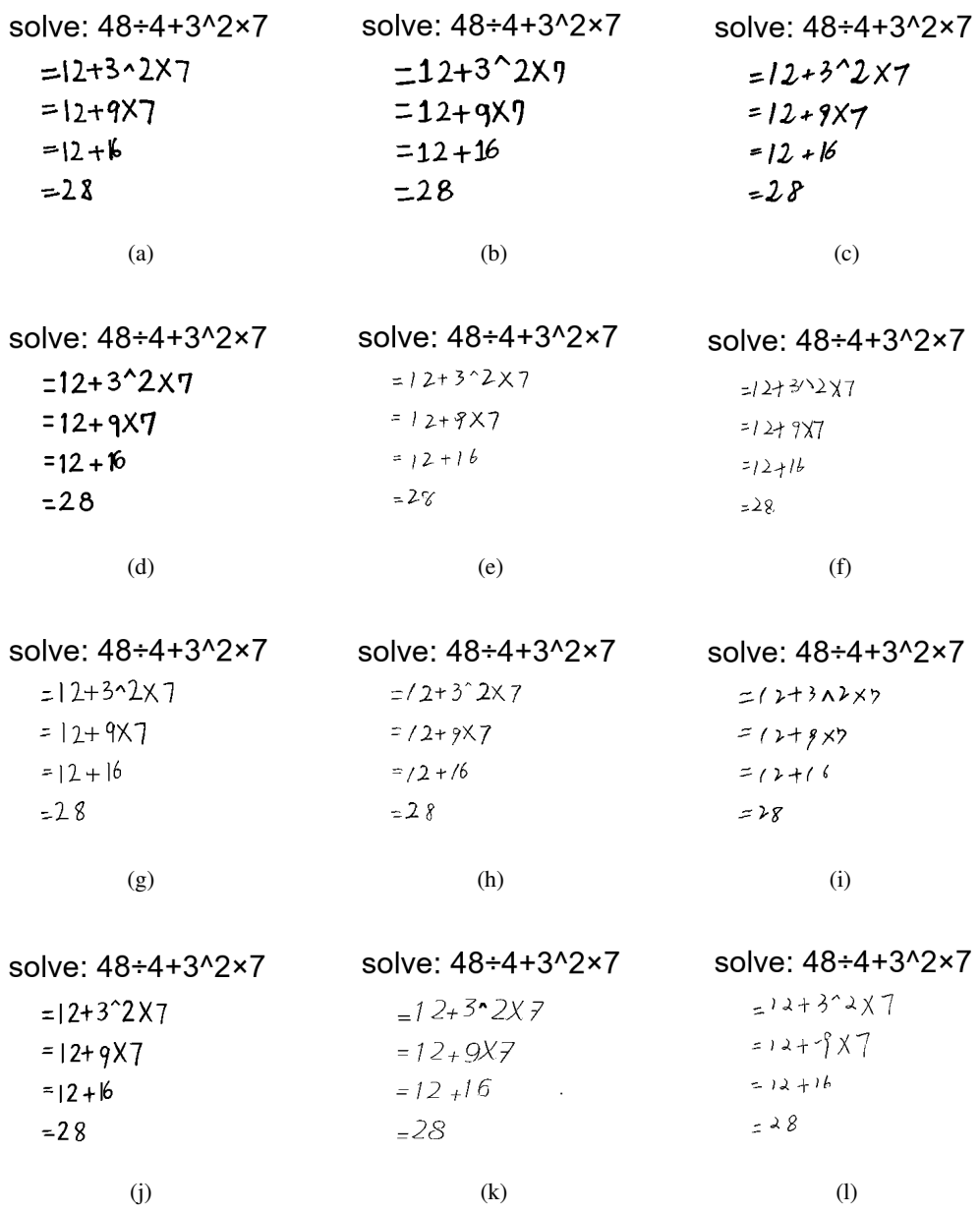

Figure 5: **Visualization of the 12 handwriting styles in EQUATION10K .** The same problem and solution trace are rendered using different handwriting styles while the mathematical content and annotations remain unchanged.

D VISUALIZATION OF 12 HANDWRITING STYLES

EQUATION10K includes 12 handwriting styles to introduce visual diversity while preserving the underlying mathematical content. Figure 5 visualizes the same problem and solution trace rendered with all 12 styles. The variation in character shape, spacing, stroke appearance, and alignment creates different visual conditions without changing the reasoning process or ground truth labels.

E Evaluation Prompt

We use the same evaluation prompt for all models. Figure 6 presents the complete instruction used during inference. The prompt requires the model to identify all visible solution steps, evaluate each step independently, assign a diagnostic state to every incorrect step, and return predictions in a fixed JSON format. No ground truth transcription, executable trace, correctness label, or error annotation is provided to the model.

F Step Correctness Accuracy by Problem Family

Figure 7 reports step correctness accuracy across the ten problem families. The strongest models remain relatively stable across categories. GPT-5.6-Sol ranges from 0.821 to 0.844, while Qwen3.8 Max ranges from 0.806 to 0.820. No single problem family consistently produces the lowest SCA across models. This result is consistent with the main analysis and suggests that variation in arithmetic family contributes less to performance degradation than structural difficulty.

G Step Correctness Accuracy by Difficulty

Figure 8 reports step correctness accuracy across difficulty levels. In contrast to JSA, SCA remains relatively stable as difficulty increases. GPT-5.6-Sol ranges from 0.812 to 0.844 across all five levels, while the other strong models show similarly modest variation. No consistent decline appears from Level 1 to Level 5.

This contrast suggests that structural complexity affects fine grained diagnosis more strongly than binary correctness recognition. Models can often determine whether a step is correct even on difficult problems, yet assigning the appropriate diagnostic state becomes substantially harder as the reasoning structure grows more complex.

You are grading a mathematical step-by-step solution shown in an image.
First identify the original problem and all visible solution steps by yourself. Then grade every solution step you can identify, from top to bottom.
For each incorrect step, use exactly one of these error labels:
- ”Carelessly wrote down the wrong value.”
- ”Wrong operation symbol”
- ”Incorrect parentheses or order of operations”
- ”Calculation error”
- ”Pre-order error propagation”
Definitions: - ”Carelessly wrote down the wrong value.”: a value, number, or symbol was copied, transcribed, substituted, or written incorrectly.
- ”Wrong operation symbol”: an operation sign/operator was changed or written incorrectly.
- ”Incorrect parentheses or order of operations”: parentheses, precedence, or operation order is incorrect.
- ”Calculation error”: the intended operation is appropriate, but the arithmetic result is wrong.
- ”Pre-order error propagation”: an earlier displayed step already introduced an error, and the current transformation is locally valid but continues that previous error.
Rules: - The original problem statement is NOT a solution step.
- Determine the number of solution steps only from the image.
- Do not assume a fixed number of steps.
- Do not omit a step merely because it looks incorrect.
- If a current step introduces a new independent error, use the concrete error type instead of ”Pre-order error propagation”.
- A correct step must have ”error_type”: null.
- Preserve each recognized step’s mathematical text in ”step_text”.
- Return steps in the same top-to-bottom order as the image.
- Return JSON only, without Markdown fences or explanations.
Required JSON structure: { ”steps”: [{ ”step_text”: ”the mathematical expression shown in this step”, ”is_correct”: true, ”error_type”: null }] }

(a) using images as input.

You are an evaluator of mathematical solution steps. Your task is to judge whether the CURRENT solution step is correct given the problem and any preceding textual solution steps that are provided.
For an incorrect current step, use exactly one of these error labels:
- ”Carelessly wrote down the wrong value.”
- ”Wrong operation symbol”
- ”Incorrect parentheses or order of operations”
- ”Calculation error”
- ”Pre-order error propagation”
Definitions: - ”Carelessly wrote down the wrong value.”: a value, number, or symbol was copied, transcribed, substituted, or written incorrectly.
- ”Wrong operation symbol”: an operation sign/operator was changed or written incorrectly.
- ”Incorrect parentheses or order of operations”: parentheses, precedence, or operation order is incorrect.
- ”Calculation error”: the intended operation is appropriate, but the arithmetic result is wrong.
- ”Pre-order error propagation”: an earlier displayed step already introduced an error, and the current transformation is locally valid but continues that previous error.
Rules:
- Grade ONLY the current step requested by the user.
- Do not invent or infer any hidden correctness/error labels.
- If the current step introduces a new independent error, use the concrete error type rather than ”Preorder error propagation”.
- A correct step must have ”error_type”: null.
- Preserve the current step text exactly in ”step_text”.
- Return JSON only, without Markdown fences or explanations.
Required JSON structure: { ”step_text”: ”the current mathematical expression”, ”is_correct”: true, ”error_type”: null }

(b) using text as input.

Figure 6: **Evaluation prompt used for EQUATIONS10K.** All evaluated models receive the same grading instruction and structured output specification.

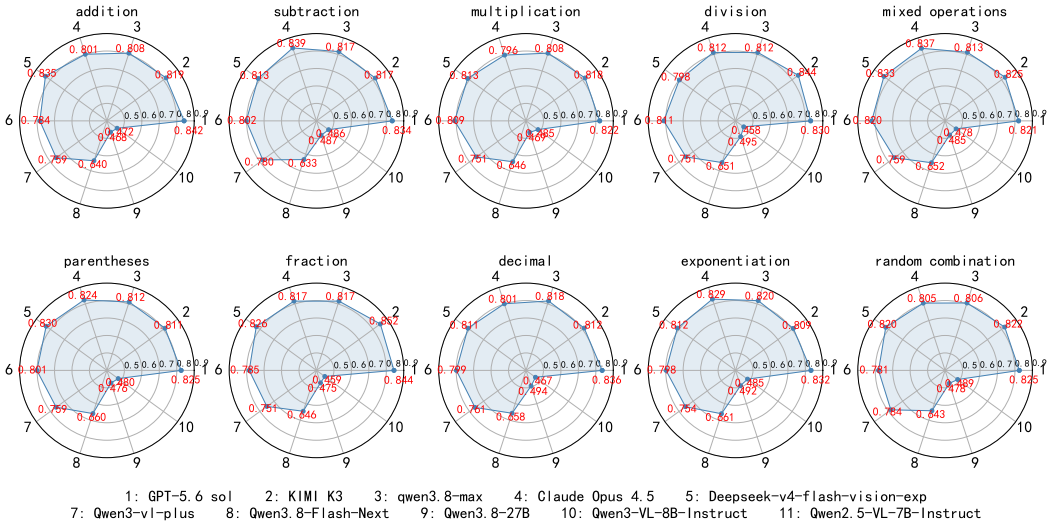

Figure 7: **Step correctness accuracy across problem families.** Results show SCA for each of the ten arithmetic families in EQUATION10K .

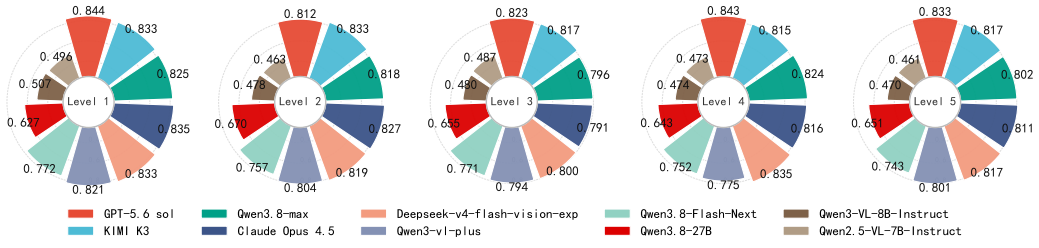

Figure 8: **Step correctness accuracy across difficulty levels.** SCA remains comparatively stable across the five structural difficulty levels.