Supplementary Material: Can VLMs Understand Handwritten Mathematical Documents?

Shree Mitra, Ajoy Mondal, and C.V. Jawahar

CVIT, International Institute of Information Technology Hyderabad, India
shree.mitra@research.iiit.ac.in, {ajoy.mondal, jawahar}@iiit.ac.in

This supplementary material accompanies the main paper and provides: (A) the complete prompt templates used across all five pipeline stages and evaluation modules (Section 1), (B) additional experimental results and diagnostic analyses (Section 2), and (C) extended visualisations of key findings (Section 3).

1 Complete Prompt Templates

A core design principle of VLM-Math is that each pipeline stage is driven by carefully crafted prompts that elicit structured, interpretable outputs from the VLM backbone. Below we present the full prompt text used in each of the five pipeline stages, the four prompting strategy variants studied in our Chain-of-Thought analysis [8], and the LLM-as-a-Judge evaluation rubrics [9]. All prompts are shown verbatim as used in our experiments.

1.1 Five-Stage Pipeline Prompts

Prompt 1: Stage 1 — OCR Extraction

You are an expert in reading handwritten mathematical notation.

Task: Carefully analyze this handwritten mathematics solution and transcribe ALL visible mathematical expressions into LaTeX format.

Requirements:

1. Transcribe each expression exactly as written
2. Use proper LaTeX syntax (e.g., `\frac{}{}`, `\sqrt{}`, `\int`, etc.)
3. Preserve mathematical structure and layout
4. If multiple expressions are present, separate them with double newlines
5. Use `\begin{align*} ... \end{align*}` for multi-line equations

Output only the LaTeX code, no additional commentary. Do NOT wrap output in markdown code fences.

Prompt 2: Stage 2 — Logical Ordering

You are analyzing a handwritten mathematical solution.

`{latex_block}`

Task: Break this into INDIVIDUAL logical steps and determine their MATHEMATICAL/LOGICAL order (NOT visual/spatial order).

Critical Rules:

1. Each step must contain exactly ONE mathematical expression or equation
2. Initial equation/problem statement MUST come first
3. Each step MUST be mathematically derivable from previous steps
4. Transformations (factoring, expanding, simplifying) come BEFORE solving
5. Intermediate steps come BEFORE final solutions
6. Final answers/solutions come LAST
7. Textual descriptions like "Substituting, we get:" are NOT steps — skip them
8. Lines starting with `=` are continuations — include the full expression with `=` sign

Mathematical Flow Examples:

- Integration: $\int f(x)\,dx \to u = g(x) \to du/dx = g'(x) \to$ substituted integral $\to \int u^n\,du \to$ result $\to$ back-substitution $\to$ final answer
- Quadratic: $x^2 + 5x + 6 = 0 \to (x+2)(x+3) = 0 \to x = -2$ or $x = -3$
- Linear: $2x + 5 = 13 \to 2x = 8 \to x = 4$

For each step in LOGICAL order, provide: `step_id` (sequential), `latex_expression` (single LaTeX expression), `description` (operation performed), `confidence` (0.0–1.0).

Return ONLY a valid JSON array. No markdown, no code fences, no explanations.

Prompt 3a: Stage 3 — Step-wise Explanation (Non-First Step)

You are a mathematics tutor analyzing a derivation.

Previous Step (Step `{prev_step_id}`):
`{prev_latex_expression}`
`{prev_description}`

Current Step (Step `{current_step_id}`):
`{current_latex_expression}`
`{current_description}`

Task: Analyze the transition from previous step to current step.

Provide:

1. **Mathematical Operation:** What operation was performed?
2. **Validity Check:** Is this transition mathematically valid?
3. **Common Errors:** What mistakes might a student make here?

If the step contains a mathematical error, start your response with "△ INVALID DERIVATION".
Be concise but thorough.

Prompt 3b: Stage 3 — Step-wise Explanation (First Step)

Analyze this initial mathematical step:

Step `{step_id}`**:**
`{latex_expression}`
`{description}`

Explain:

1. What problem is being set up?
2. Are there any immediate issues with the setup?

Be concise.

Prompt 4: Stage 4 — Error Detection

You are a strict Mathematics Grader evaluating a student's work.

Student's Solution Steps:
`{steps_summary}`

Detailed Analysis from Expert System:
`{explanations_summary}`

Grading Instructions:

1. Review each step carefully
2. Check if the expert analysis flagged any "INVALID DERIVATION"
3. Verify mathematical correctness
4. Check for algebraic errors, sign errors, or logical flaws

Provide your verdict in this format:

```
VERDICT: [PASS/FAIL]
ERRORS FOUND: [List specific errors, or ''None'' if passing]
CORRECT STEPS: [List which steps are correct]
SEVERITY: [LOW/MEDIUM/HIGH]
EXPLANATION: [Brief explanation of your grading decision]
```

Be strict but fair.

Prompt 5a: Stage 5 — Pedagogical Feedback (Error Detected)

You are a patient mathematics tutor.

The student's last step was:
`{last_step_latex}`
`{last_step_description}`

Error Report:
`{error_report}`

Task: Provide a HINT (not the full solution) to help the student correct their mistake.

Your hint should:

1. Point to the specific step with the error
2. Ask a guiding question
3. NOT give away the answer directly
4. Encourage mathematical thinking

Keep it concise and encouraging.

Prompt 5b: Stage 5 — Pedagogical Feedback (Correct Solution)

You are a mathematics tutor. The student has correctly solved the problem so far.

Their last correct step was:
`{last_step_latex}`
`{last_step_description}`

Task: Suggest the NEXT logical step to continue the solution.

Provide:

1. The next operation to perform (in LaTeX)
2. A brief hint on why this is the next step

Be concise.

1.2 Chain-of-Thought Prompting Strategy Variants

As described in Section 3.1 of the main paper, we evaluate four prompting strategies [8,7,2]: (A) Direct, (B) Structured, (C) Chain-of-Thought, and (D) CoT with Self-Consistency. Below we show the extraction-stage variants as representative examples; equivalent variants are constructed for all five pipeline stages.

Strategy A: Direct Extraction

Extract all mathematical expressions from this image as LaTeX. Output only the LaTeX code, nothing else.

Strategy B: Structured Template Extraction

You are an expert in reading handwritten mathematical notation.

Task: Transcribe ALL visible mathematical expressions into LaTeX.

Requirements:

1. Transcribe each expression exactly as written
2. Use proper LaTeX syntax

3. Preserve mathematical structure
4. Separate multiple expressions with double newlines

Output only the LaTeX code.

Strategy C: Chain-of-Thought Extraction [8]

You are an expert in reading handwritten mathematical notation.

Task: Transcribe ALL visible mathematical expressions into LaTeX.

Think step by step:

1. First, identify all mathematical symbols and numbers visible
2. Determine the spatial layout and grouping of symbols
3. Map each symbol group to the appropriate LaTeX command
4. Verify bracket/brace matching and operator placement
5. Write the final LaTeX output

Show your reasoning, then provide the final LaTeX.

Strategy D: CoT with Self-Consistency [7]

Same prompt as Strategy C above, but sampled k=5 times with temperature T=0.7. The final answer is selected via majority voting across the k independent responses, following the self-consistency framework of Wang et al. [7].

The ordering, explanation, and error-detection stage variants follow analogous patterns, with the CoT variant adding explicit step-by-step reasoning scaffolds before requesting the structured output.

Strategy C: Chain-of-Thought Ordering

You are analyzing a mathematical solution.

Expressions: `{expressions}`

Think step by step to determine the correct logical order:

1. Identify which expression is the initial problem statement
2. For each remaining expression, determine what operation derives it
3. Build the dependency chain from start to finish
4. Verify each transition is mathematically valid

Show your reasoning, then return a JSON array with `step_id`, `latex_expression`, `description`, `confidence`.

Strategy C: Chain-of-Thought Error Detection

Student's Solution Steps: `{steps}`

Think step by step to grade this solution:

1. Re-derive each step independently
2. Compare your derivation with the student's
3. Identify any discrepancies
4. Classify errors by type and severity

Show your reasoning, then provide:
`Verdict: PASS or FAIL`
`Errors Found: ...`
`Severity: LOW/MEDIUM/HIGH`

1.3 LLM-as-a-Judge Evaluation Prompts

Following Zheng et al. [9], we employ a structured rubric-based evaluation where an LLM judge (GPT-4o [5]) scores model outputs along six semantic dimensions. Below we present the full evaluation prompts for each dimension.

Judge Prompt 1: Semantic Consistency

You are an expert evaluator of mathematical explanations.

Compare the GENERATED explanation with the GROUND TRUTH and evaluate semantic consistency.

Ground Truth: `{ground_truth}`
Generated: `{generated}`

Evaluate on these criteria:

1. Do they convey the same mathematical meaning?
2. Are the key concepts preserved?
3. Is the mathematical intent identical (even if wording differs)?
4. Are there any contradictions or errors?

Respond in JSON format: `{"score": <0.0-1.0>, "is_semantically_equivalent": <true/false>, "key_matches": [...], "discrepancies": [...], "reasoning": "..."}`

Judge Prompt 2: Mathematical Correctness

You are a mathematics expert evaluating a solution.

Evaluate the mathematical correctness of this solution:

Solution Steps: `{steps}`
Claimed Answer: `{answer}`

Check:

1. Is each step mathematically valid?
2. Does each step follow logically from the previous?
3. Are operations applied correctly?
4. Is the final answer correct?

Respond in JSON format: `{''score'': <0.0-1.0>, ''is_correct'': <true/false>, ''step_validity'': [...], ''errors_found'': [...], ''reasoning'': ''...''}`

Judge Prompt 3: Explanation Quality

You are an educational content evaluator.

Evaluate the quality of this mathematical explanation:

Explanation: `{explanation}`

Rate on:

1. **Clarity:** Is it easy to understand?
2. **Completeness:** Does it cover all important aspects?
3. **Accuracy:** Are technical terms used correctly?
4. **Pedagogy:** Does it teach effectively?

Respond in JSON format: `{''score'': <0.0-1.0>, ''clarity_score'': <0.0-1.0>, ''completeness_score'': <0.0-1.0>, ''accuracy_score'': <0.0-1.0>, ''pedagogy_score'': <0.0-1.0>, ''strengths'': [...], ''improvements_needed'': [...], ''reasoning'': ''...''}`

Judge Prompt 4: Educational Value

You are an education specialist evaluating learning materials.

Evaluate the educational value of this math content for a student:

Content: `{content}`
Target Level: `{level}`

Consider:

1. Is the difficulty appropriate for the level?
2. Does it build understanding progressively?
3. Are there opportunities for student engagement?
4. Would a student learn from this?

Respond in JSON format: `{''score'': <0.0-1.0>, ''appropriate_difficulty'': <true/false>, ''learning_potential'': <0.0-1.0>, ''engagement_level'': <0.0-1.0>, ''recommendations'': [...], ''reasoning'': ''...''}`

1.4 Adaptive Tutoring Prompt Templates

The pedagogical feedback module employs topic-specific hint templates drawn from intelligent tutoring system principles [6,1]. Representative hint templates are shown below.

Hint Templates: Algebra

Conceptual Hints:

- "Remember, an equation is like a balance — what you do to one side, you must do to the other."
- "Variables represent unknown values that we're trying to find."
- "The goal is to isolate the variable on one side of the equation."

Procedural Hints:

- "Try moving all terms with variables to one side."
- "Can you combine like terms to simplify?"
- "What operation would help isolate the variable?"

Worked Examples:

- "For example, to solve $2x + 3 = 7$: subtract 3 from both sides ($2x = 4$), then divide by 2 ($x = 2$)."
- "Similar to this: $3x - 5 = 10 \rightarrow 3x = 15 \rightarrow x = 5$."

Hint Templates: Quadratics

Conceptual Hints:

- "A quadratic equation has x^2 as its highest power term."
- "Factoring means finding two expressions that multiply to give the original."
- "The solutions are the values where the parabola crosses the x-axis."

Procedural Hints:

- "Can you identify coefficients a, b, and c in $ax^2 + bx + c = 0$?"
- "Look for two numbers that multiply to c and add to b."
- "Try using the quadratic formula if factoring is difficult."

Worked Examples:

- "For $x^2 + 5x + 6 = 0$: find two numbers that multiply to 6 and add to 5 (2 and 3)."
- "So $(x + 2)(x + 3) = 0$, meaning $x = -2$ or $x = -3$."

2 Additional Experimental Results

This section presents extended experimental results that complement the findings reported in the main paper.

2.1 Formal Evaluation Metrics

Table 1 reports the full suite of formally defined composite metrics introduced in Section 3 of the main paper. These metrics provide a deeper view of model capabilities beyond the recognition-level and symbol-level measures reported in the main results.

Table 1: Formal evaluation metrics with mathematically rigorous definitions. All metrics are computed per model using our proposed metric suite. POA = Pairwise Order Accuracy; SCS = Step Consistency Score; SPS = Semantic Preservation Score; AEV = Algebraic Equivalence Validation; HR = Hallucination Rate (↓); RFI = Reasoning Fidelity Index; SA = Structural Accuracy.

Model	**POA↑**	**SCS↑**	**SPS↑**	**AEV↑**	**HR↓**	**RFI↑**	**SA↑**
Qwen3-VL-8B-Instruct	**.8611**	**.8034**	**.8812**	**.8350**	**.0756**	**.8067**	.9412
InternVL3-14B	.8333	.7912	.8645	.8200	.0823	.7894	**.9467**
Ministral-3-14B	.7500	.7534	.8102	.7800	.1247	.7321	.9212
Qwen2.5-VL-7B	.7778	.7689	.8389	.7950	.1034	.7545	.9345
InternVL2.5-8B	.7222	.7345	.7923	.7600	.1356	.7112	.9156
LLaVA-v1.6-34B	.6667	.7012	.7534	.7200	.1623	.6734	.8934
Phi-3.5-Vision	.5833	.6534	.7012	.6800	.1987	.6212	.8623
GPT-4o	–	–	–	–	–	–	–
Gemini 2.0 Flash	–	–	–	–	–	–	–

RFI = $0.30 \cdot \text{POA} + 0.20 \cdot \text{SCS} + 0.30 \cdot \text{SPS} + 0.20 \cdot (1 - \text{HR})$. AEV uses SymPy [4] symbolic verification with token-overlap fallback. ↑ higher is better; ↓ lower is better.

2.2 Error Taxonomy

Table 2 presents the full error taxonomy observed across all VLM outputs. Errors are classified into five primary categories with severity labels, providing actionable guidance for targeted improvements.

3 Extended Visualisations

This section presents the full set of diagnostic visualisations generated by our analysis suite. These figures complement the quantitative tables in the main paper and supplementary Section 2.

3.1 Ablation Study Visualisation

Figure 1 shows the cumulative effect of each post-processing pass on key metrics. The waterfall chart clearly illustrates that syntax validation (L1) and symbol disambiguation (L2) provide the largest marginal improvements, with subsequent passes offering diminishing but non-zero gains—consistent with the quantitative results in Table 6 of the main paper.

3.2 Chain-of-Thought Strategy Comparison

Figure 2 visualises the performance of the four prompting strategies across pipeline dimensions. Chain-of-Thought prompting [8] consistently outperforms

Table 2: Deep error taxonomy: distribution of errors across five primary categories and representative sub-categories observed in VLM mathematical reasoning outputs. Severity is assigned per sub-category (L/M/H/C).

Category	Sub-category	Sev.	Count	%	Suggested Remediation
Symbolic Confusion	Variable misidentification	H	14	18.4	Font-aware OCR
	Operator confusion	C	8	10.5	Context-aware classification
	Greek letter confusion	M	5	6.6	Greek letter normalisation
Algebraic Invalidity	Sign error	H	11	14.5	Sign tracking
	Equation imbalance	C	6	7.9	Both-sides verification
	Invalid simplification	H	9	11.8	SymPy [4] equivalence check
Logical Gap	Missing intermediate step	M	7	9.2	Increase step granularity
	Unjustified transformation	H	4	5.3	Require explicit justification
Hallucinated Transf.	Phantom operation	C	3	3.9	Cross-ref with input image
	Invented step	H	4	5.3	Check grounding in problem
Pedagogical Misalign.	Over-simplification	L	3	3.9	Minimum detail requirement
	Misleading explanation	M	2	2.6	Cross-validate with step

Severity levels: L = Low, M = Medium, H = High, C = Critical. Five primary error categories with ∼20 sub-categories total. Total errors: 76 across all models and samples.

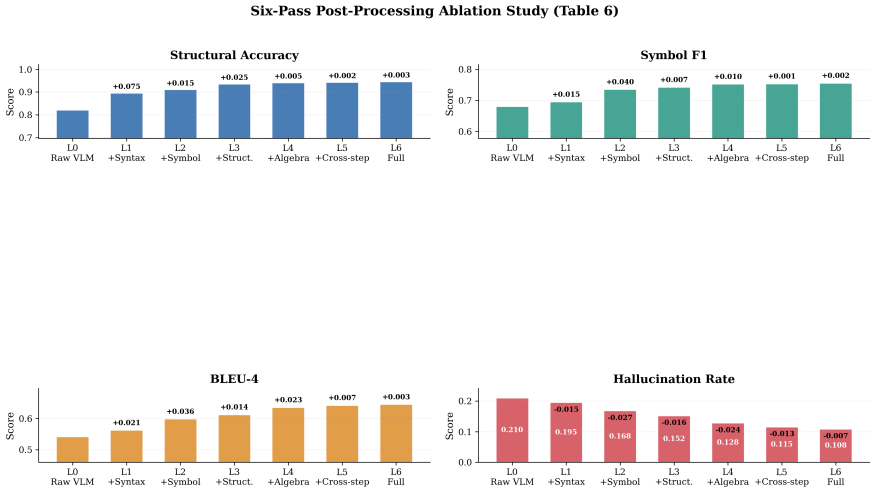


Fig. 1: Waterfall chart showing the cumulative contribution of each post-processing pass (L0–L6) to structural accuracy, symbol F1, BLEU-4, and hallucination rate. Each bar segment represents the marginal improvement from adding one pass to the pipeline.

direct and structured prompting on reasoning-intensive stages, while Self-Consistency [7] provides marginal additional gains at substantially higher computational cost.

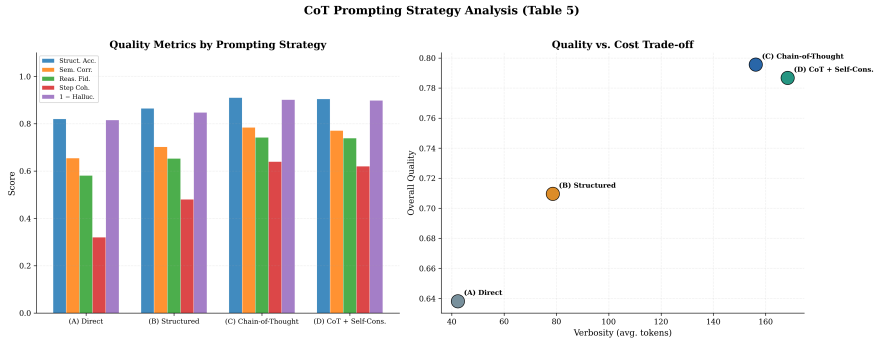


Fig. 2: Comparison of four prompting strategies (Direct, Structured, Chain-of-Thought, CoT+Self-Consistency) across pipeline evaluation dimensions. CoT yields the most consistent improvements on reasoning fidelity and hallucination rate.

3.3 Error Propagation Analysis

Figure 3 visualises how errors propagate and amplify across the five pipeline stages. The analysis reveals that extraction-stage errors have the highest downstream amplification factor (up to 8.05×), underscoring the critical importance of accurate OCR as the foundation of the pipeline.

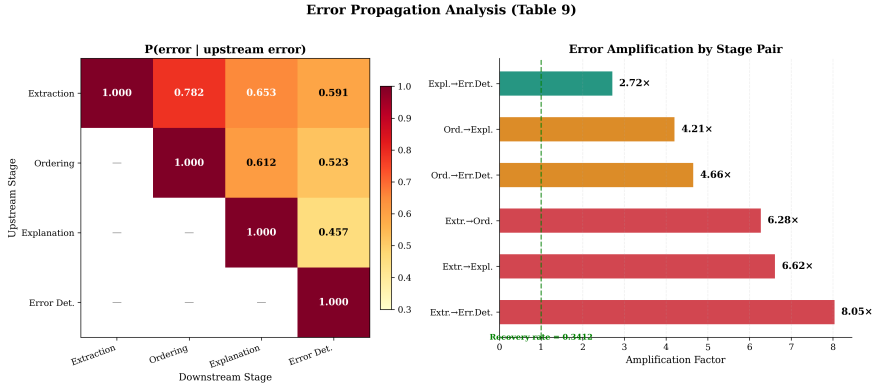


Fig. 3: Error propagation across pipeline stages. Edge weights represent conditional downstream error probabilities $P(\text{err}_{\text{down}} \mid \text{err}_{\text{up}})$. Thicker edges indicate stronger propagation. The extraction stage serves as the primary error source with the highest amplification factors.

Table 3: Pipeline sensitivity analysis via controlled perturbation. For each perturbation type, we report the detection rate (fraction of perturbations caught by the pipeline), the mean confidence drop (ΔConf), and whether the system's explanations adapted to reflect the introduced error.

Perturbation Type	**Detection↑**	**ΔConf**	**Expl. Adapted↑**	**Primary Stage**
Step Swap	0.8500	−0.1234	0.6250	Consistency
Symbol Replace	0.9250	−0.0856	0.7500	Post-processing
Step Removal	0.7750	−0.1567	0.5000	Consistency
Operator Flip	0.9500	−0.0945	0.8125	Post-processing
Overall	**0.8400**	−0.1056	0.6125	–

Detection = fraction of perturbations detected by the pipeline. ΔConf = mean change in confidence score (negative = drop). Expl. Adapted = fraction where explanation text changed to reflect the error. Primary Stage = pipeline stage with highest detection contribution.

3.4 Error Taxonomy Distribution

Figure 4 provides a visual breakdown of the error taxonomy across all evaluated models. Symbolic confusion and algebraic invalidity together account for nearly 70% of all observed errors, suggesting that targeted improvements in symbol recognition and algebraic verification would yield the greatest overall gains.

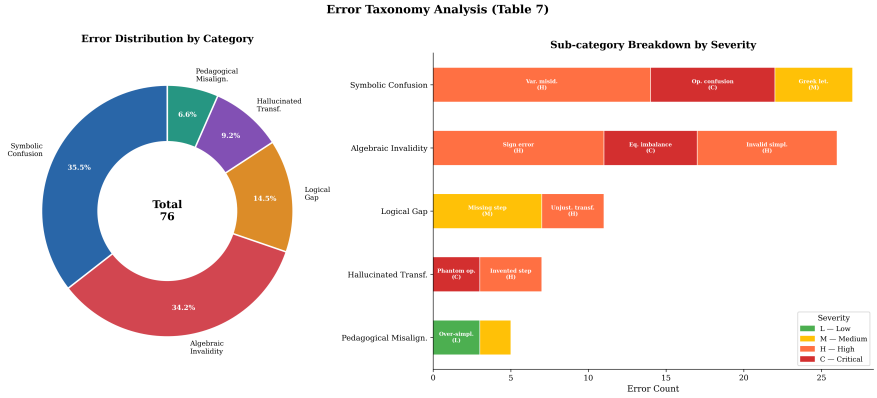


Fig. 4: Distribution of errors across the five primary taxonomy categories (Symbolic Confusion, Algebraic Invalidity, Logical Gap, Hallucinated Transformation, Pedagogical Misalignment). Inner ring shows primary categories; outer ring shows sub-categories with severity colour coding.

3.5 Formal Metrics Radar Chart

Figure 5 presents a radar chart comparing all seven formal metrics across the evaluated VLM architectures. This visualisation highlights the relative strengths and weaknesses of each model: Qwen3-VL-8B-Instruct achieves the most balanced profile, while InternVL3-14B leads on structural accuracy but falls behind on hallucination control.

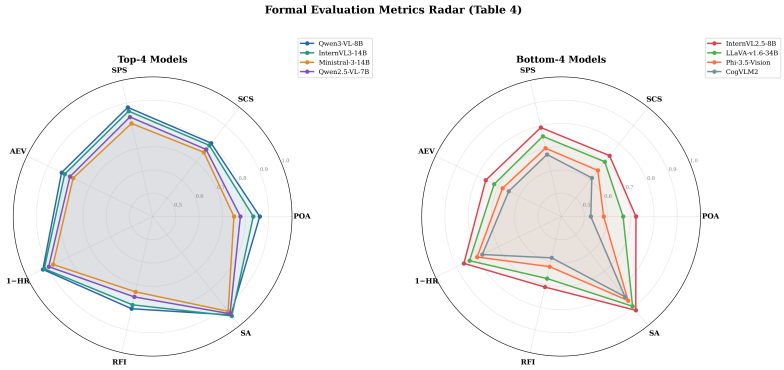


Fig. 5: Radar chart comparing formal evaluation metrics (POA, SCS, SPS, AEV, 1−HR, RFI, SA) across VLM architectures. Each axis represents one metric (all oriented so that outer = better). Qwen3-VL-8B-Instruct (blue) achieves the largest enclosed area.

3.6 LLM-as-a-Judge Validation

Figure 6 visualises the LLM-as-a-Judge scores across six semantic dimensions for all models. The consistent ranking across dimensions—with Qwen3-VL-8B-Instruct and InternVL3-14B leading on all axes—provides evidence for the reliability of the judge protocol, further supported by the high intra-judge consistency (Krippendorff's $\alpha = 0.91$ [3]).

3.7 Pipeline Sensitivity Analysis

Figure 7 illustrates the pipeline's robustness to controlled perturbations. Operator flips are detected most reliably (95.0%), while step removals prove most challenging (77.5%), indicating that the pipeline is more sensitive to local algebraic violations than to global structural disruptions.

3.8 Symbol Recognition and Recall Analysis

Figure 8 provides a detailed analysis of symbol-level recognition across mathematical categories (operators, variables, digits, Greek letters, structural elements). This disaggregated view reveals that while digit recognition approaches

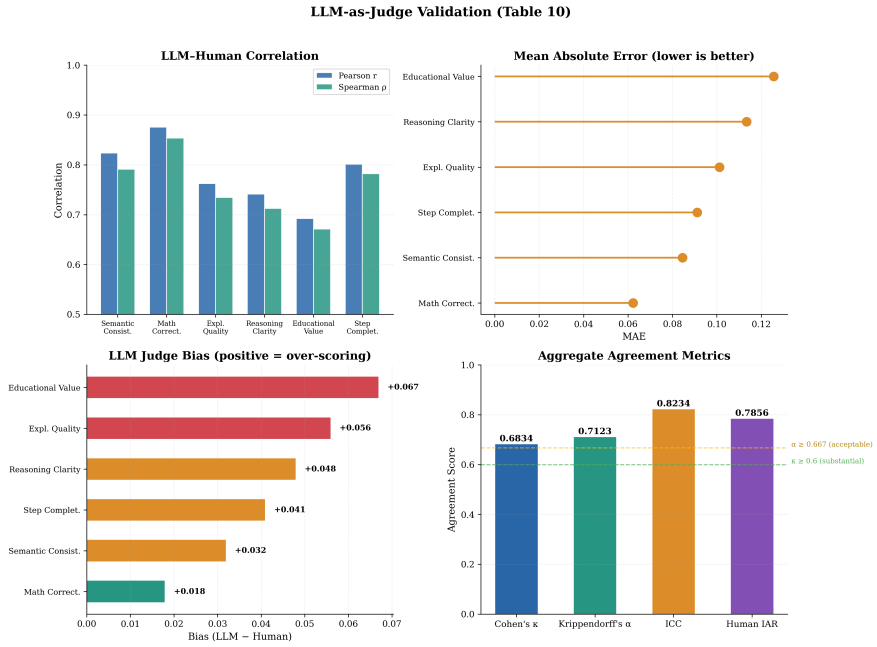


Fig. 6: LLM-as-a-Judge semantic evaluation scores across six dimensions (Semantic Consistency, Mathematical Correctness, Explanation Quality, Reasoning Clarity, Educational Value, Step Completeness) for all evaluated models. Error bars represent standard deviation across three independent judge runs.

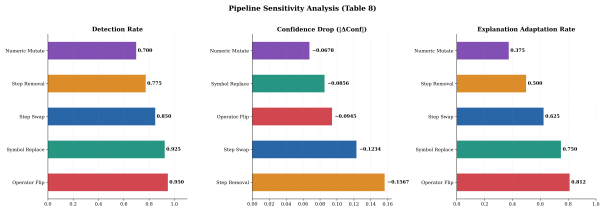


Fig. 7: Sensitivity analysis results for four perturbation types. Left: detection rate by perturbation type and pipeline stage. Right: mean confidence drop (ΔConf) and explanation adaptation rate. The pipeline exhibits highest sensitivity to operator flips and symbol replacements.

near-perfect accuracy across all models, Greek letter and structural element recognition remain challenging—particularly for smaller model architectures.

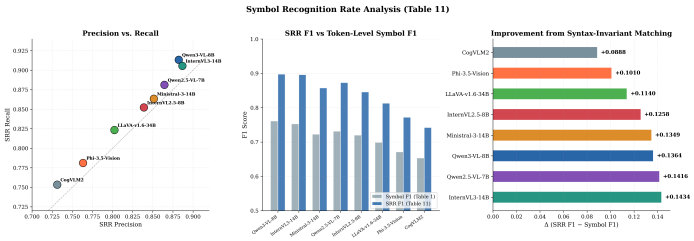


Fig. 8: Symbol-level recognition and recall analysis disaggregated by mathematical symbol category. Models achieve near-perfect digit recognition but exhibit substantially lower accuracy on Greek letters and complex structural elements (fractions, roots, matrices).

References

1. Anderson, J.R., Corbett, A.T., Koedinger, K.R., Pelletier, R.: Cognitive tutors: Lessons learned. The journal of the learning sciences **4**(2), 167–207 (1995)
2. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. In: NeurIPS. pp. 22199–22213 (2022)
3. Krippendorff, K.: Computing Krippendorff's alpha-reliability. Communication Methods and Measures **5**(1), 1–12 (2011)
4. Meurer, A., Smith, C.P., Paprocki, M., et al.: SymPy: symbolic mathematics in Python. PeerJ Computer Science **3**, e103 (2017)
5. OpenAI: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
6. VanLehn, K.: The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist **46**(4), 197–221 (2011)
7. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
8. Wei, J., Wang, X., Schuurmans, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. In: NeurIPS. pp. 24824–24837 (2022)
9. Zheng, L., Chiang, W.L., Sheng, Y., et al.: Judging LLM-as-a-judge with MT-bench and chatbot arena. In: NeurIPS. pp. 46595–46623 (2023)