MathStepGrader: Step-level Visual-Logical Process Supervision for Handwritten Mathematical Verification

Anonymous ACL submission

Abstract

Automated verification of handwritten mathematical solutions requires both faithful recognition of handwritten content and step-wise logical verification. Existing methods typically address this task either through a two-stage pipeline or an end-to-end framework based on multimodal large language models (MLLMs). The former relies on separate models for handwritten content recognition and content verification; however, this separation may introduce error propagation and make it difficult to determine whether an error stems from inaccurate recognition or from an actual logical mistake. The latter, in contrast, often emphasizes visual understanding while offering limited process-level logical verification. In this paper, we address these challenges by jointly enhancing the visual understanding and logical reasoning abilities of MLLMs. We propose MathStepGrader, a step-level process supervision framework for handwritten mathematical verification. MathStepGrader decomposes raw handwritten solutions into ordered step-level visual units and introduces a dual-reward objective that jointly optimizes recognition fidelity and logic reasoning ability. It further incorporates multi-agent feedback for arithmetic, symbolic, and textual verification. Experiments on the FERMAT benchmark demonstrate consistent improvements across multiple open-weight multimodal backbones and various training paradigms. Compared with direct step-level prompting, SFT+MathStepGrader achieves gains of up to 7.6 absolute percentage points in final accuracy. The results demonstrate the effectiveness of the proposed method in both handwritten recognition and logical verification, highlighting its potential value for educational applications such as math research.

1 Introduction

Automated verification of handwritten mathematical solutions aims to determine whether a student's derivation is logically valid from the handwritten solution itself. Unlike reference-based grading or answer-only assessment, this task usually does not assume access to a gold derivation or step-by-step reference solution at inference time. It can therefore be viewed as a multimodal mistake-finding task: the system must faithfully recover a visually perceived derivation process and identify whether, and where, the derivation first becomes invalid. This makes the task especially challenging due to three factors: ambiguous handwritten symbols, irregular two-dimensional layouts, and the need for step-level logical reasoning over noisy visual evidence (Mahdavi et al., 2019; Truong et al., 2024).

Existing methods typically address this task through either a conventional recognition-then-verification pipeline or an end-to-end MLLM-based framework. Recognition-then-verification pipelines first transcribe handwritten mathematical content and then assess the recognized solution with a separate reasoning module (Levine et al., 2026). While explicit, this separation may cause recognition errors to propagate to the verification stage, making it difficult to determine whether an incorrect judgment comes from imperfect transcription or from an actual logical mistake. End-to-end MLLM-based approaches directly evaluate handwritten solution images with vision-language models. Although these methods benefit from recent advances in visual understanding and mathematical reasoning, they often emphasize whole-image understanding while providing limited process-level logical verification.

Figure 1 compares the above two paradigms with MathStepGrader. Different from recognition-then-verification pipelines and end-to-end MLLM verification, MathStepGrader preserves local visual evidence through ordered step-level visual units and performs context-aware verification over the derivation process.

To address these issues, we propose MathStepGrader, a step-level visual-logical process super-

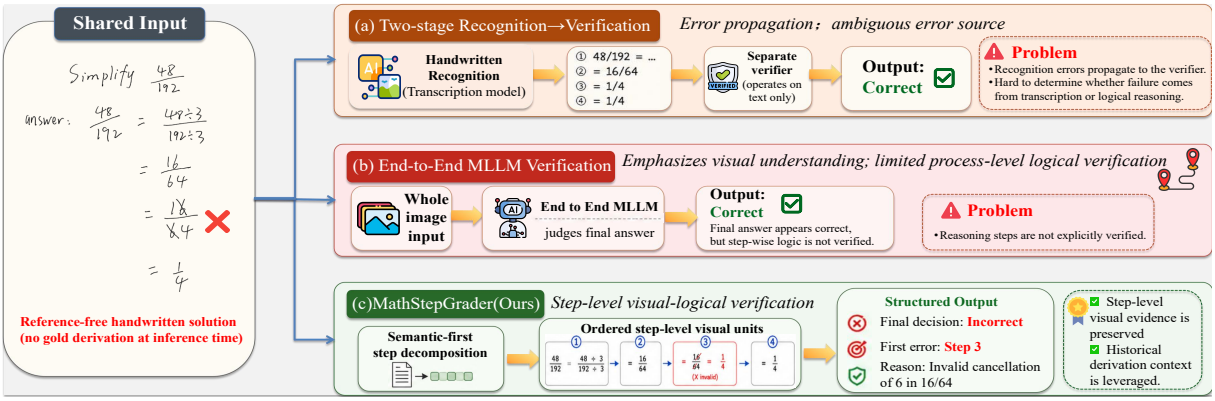

Figure 1: Comparison of MathStepGrader with two-stage and end-to-end paradigms for handwritten mathematical verification.

vision framework for handwritten mathematical verification. Its core idea is to preserve local visual evidence and verify each derivation step under its historical derivation context rather than directly predicting a solution-level label. Specifically, MathStepGrader first converts raw handwritten solutions into ordered step-level visual units, reducing page-level visual clutter and aligning visual evidence with reasoning states. It then performs context-aware step-wise verification by conditioning each step on previous equations and intermediate results. Finally, it introduces a dual-reward objective that jointly optimizes recognition fidelity and logic-based verification.

We evaluate MathStepGrader on the FERMAT benchmark across multiple open-weight multimodal backbones and compare full-image end-to-end verification, direct step-level input, SFT, reward-only MathStepGrader, and SFT+MathStepGrader. Results show that step-level decomposition is the primary source of improvement by reducing page-level visual clutter, while dual-reward optimization provides consistent refinement once the input representation is fixed. For example, on Qwen2.5-VL-7B, MathStepGrader improves accuracy from 0.708 to 0.762, and SFT+MathStepGrader reaches 0.780. Diagnostic analyses on joint accuracy and first-error localization show that the framework improves both final correctness prediction and process-level faithfulness.

In summary, our contributions are fourfold. First, we formulate handwritten mathematical solution assessment as a reference-free visual-logical verification problem for MLLMs. We characterize it as a multimodal mistake-finding task, where the model must recover the derivation process from handwritten images, recognize value-sensitive mathematical content, and identify logical mistakes under historical derivation context without relying on a reference solution. This formulation connects process-level verification with educational handwritten-math assessment. Second, we propose MathStepGrader, a step-level process supervision framework that constructs ordered step-level visual units and performs context-aware step-wise verification, aligning local visual evidence with reasoning states. Third, we design a dual-reward optimization mechanism that jointly improves value-sensitive recognition and logic-based verification; the logic reward further integrates step-level correctness agreement, multi-agent feedback, and first-error anchoring for local derivation checking and error localization. Fourth, we construct a processed step-level version of the FERMAT benchmark with ordered step crops, step indices, metadata, and train/test splits generated by our automatic preprocessing pipeline. Experiments across multiple open-weight MLLM backbones show consistent gains in final accuracy, recognition-grounded joint accuracy, and first-error localization. We will release the code and processed data to facilitate reproducible research on visual-logical handwritten mathematical verification.

2 Related Work

Textual Step-level Verification and Mistake Finding. Step-level verification has been widely studied in text-based mathematical reasoning. Chain-of-thought and self-consistency methods improve reasoning by generating intermediate rationales and selecting answers supported by consistent reasoning paths (Wei et al., 2023; Wang et al., 2023). Beyond answer selection, recent work stud-

ies whether LLMs can check their own or other models' reasoning traces, such as SelfCheck and related self-verification methods (Miao et al., 2023). Mistake-finding approaches further aim to identify the erroneous step in a multi-step solution, providing more fine-grained supervision than final-answer correctness alone (Jiang et al., 2025; Zhang et al., 2025c). These works are closely related to our setting because they treat mathematical reasoning as a process-level verification problem. However, they usually assume clean textual reasoning traces, whereas our task requires recovering and verifying the reasoning process from handwritten mathematical images without access to a reference derivation at inference time.

Process Supervision and Multimodal Verification. Process supervision improves mathematical reasoning by evaluating intermediate steps rather than only final answers (Lightman et al., 2023; Wang et al., 2024; Luo et al., 2024). Recent multimodal process-reward and visual-math studies extend this idea to multimodal mathematical reasoning and evaluation (Lu et al., 2024a; Xiao et al., 2024; Du et al., 2025; Zhang et al., 2025b). Related visual-reasoning and critic-based studies further examine whether MLLMs can identify reasoning errors from multimodal inputs (Zhang et al., 2025a). Tool-augmented and multi-agent methods provide verification signals through calculation, symbolic checking, and collaborative reasoning (Qu et al., 2025; Chen et al., 2024, 2025). These studies demonstrate the value of process-level feedback, but most of them focus on textual reasoning traces or general visual-math inputs. In contrast, handwritten mathematical verification requires both value-sensitive recognition of handwritten symbols and step-wise logical verification over visually grounded derivations.

Handwritten Mathematical Verification and Grading. Handwritten mathematical verification is related to handwritten mathematical expression recognition (HMER), which converts handwritten formulas into textual or symbolic representations. Prior benchmarks and neural methods have advanced recognition of two-dimensional mathematical structures (Mahdavi et al., 2019; Truong et al., 2024; Jianhua et al., 2025), yet OCR studies show that text-rich visual inputs remain challenging for current MLLMs (Liu et al., 2024). Recent work further shows that VLMs may over-correct handwritten mathematical solutions, producing plausible but unfaithful transcriptions that hide student errors (Seong et al., 2026). Beyond recognition, educational assessment systems and benchmarks evaluate VLMs on handwritten solutions and rubric-based mathematical grading (Nath et al., 2025; Yu et al., 2026; Khrulev, 2025b,a; Nguyen et al., 2025; Levine et al., 2026). These works target a similar educational application, but they often emphasize solution-level assessment or rubric alignment. MathStepGrader instead formulates handwritten mathematical assessment as reference-free visual-logical verification, combining step-level visual units, recognition-oriented rewards, and process-level mistake localization.

3 Methodology

3.1 System Overview

Figure 2 illustrates the overall architecture of MathStepGrader. Given a raw handwritten solution, the framework first constructs an ordered sequence of step-level visual units. The policy model then generates a structured grading trajectory conditioned on historical derivation context, and the trajectory is scored by a dual-reward function for GRPO-based optimization. The following sections describe step-level visual construction, dual-reward modeling, and policy optimization in detail.

3.2 Step-level Visual Unit Construction

Handwritten mathematical solutions are often full-page images with irregular layouts, writing drift, marginal notes, erasures, and mixed textual-symbolic expressions. Direct full-page grading introduces visual clutter and makes it difficult to align local visual evidence with the corresponding reasoning step. This motivates decomposing the original image into step-level visual units, where each local region is associated with a specific derivation state. However, physical segmentation methods based on pixel-level continuity (Louloudis et al., 2008) may not match the logical structure of mathematical derivations. MathStepGrader therefore adopts a semantic-first preprocessing pipeline that infers the solution structure before grounding each step back into image space.

The semantic-first pipeline consists of three agents. The Sensory Enhancement Agent (SEA) normalizes the raw image and enhances faint strokes. The Semantic-Spatial Perception Agent (SSPA) groups text fragments and visual regions into logical steps using semantic and spatial cues.

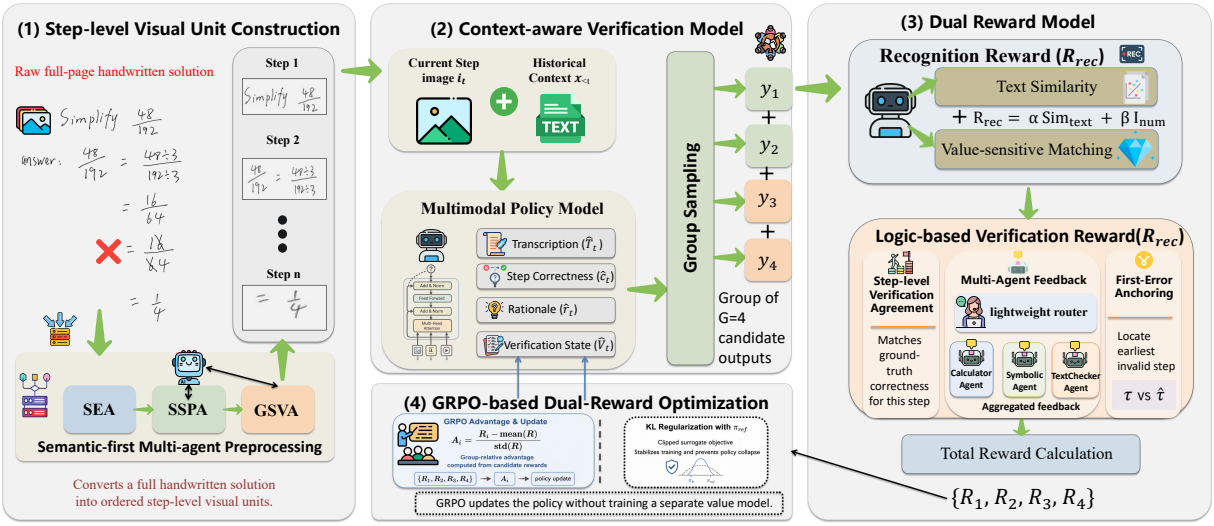

Figure 2: Overview of MathStepGrader. The left module summarizes the semantic-first preprocessing pipeline; detailed SEA–SSPA–GSVA operations are shown in Appendix A.

The Geometric Stitching and Validation Agent (GSVA) stitches co-indexed regions into complete step-level crops and validates whether each crop preserves the expected step content. Together, these agents convert an unconstrained handwritten page into an ordered sequence of logically coherent and visually complete step-level units. Detailed implementation is provided in Appendix A.

Preprocessing only constructs step-level visual units and never accesses correctness labels; all backbones use the same automatic crops. After preprocessing, each solution is represented as an ordered crop sequence $S = \{i_1, i_2, \ldots, i_n\}$, where each crop i_t corresponds to a logically atomic reasoning step and provides local visual evidence for context-aware verification.

3.3 Context-aware Verification Model

Given the ordered step-level visual units, MathStepGrader uses a multimodal policy model π_θ for context-aware verification. For each step t, the model receives the current crop i_t and historical derivation context $X_{<t}$, including previous transcriptions, intermediate equations, and accumulated verification states. We denote the input as $\mathcal{I}_t = (i_t, X_{<t})$, which allows the model to verify the current step with reference to preceding derivation states rather than treating each crop independently.

Conditioned on $\mathcal{I}_t$, the policy model generates four structured fields: the step transcription $\hat{T}_t$, step correctness prediction $\hat{c}_t$, rationale $\hat{r}_t$, and verification state $\hat{V}_t$, which respectively support recognition reward computation, step-level validity checking, rationale-consistency feedback, and trajectory-level tracking of the final decision and predicted first-error step, forming a verification trajectory y.

3.4 Dual Reward Modeling

Given the context-aware inputs, the policy model samples structured verification trajectories. Each trajectory y contains step transcriptions, step-level correctness predictions, rationales, verification states, the final decision, and the predicted first-error step. MathStepGrader uses a practical dual reward to separate recognition errors from step-wise logical judgment errors.

Recognition reward. The recognition reward $R_{\text{rec}}(y)$ evaluates whether the model faithfully transcribes the handwritten content in each step, with particular emphasis on value-sensitive elements such as digits, signs, operators, variables, coefficients, and exponents. For a trajectory with n predicted steps, we define

$$R_{\text{rec}}(y) = \frac{1}{n} \sum_{t=1}^{n} \left[\alpha \text{Sim}_{\text{text}}(\hat{y}_t, y_t^*) + \beta I_{\text{num}}(\hat{y}_t, y_t^*) \right]. \quad (1)$$

Here, $\hat{y}_t$ and y_t^* denote the predicted and reference transcriptions of step t. Sim_{text} measures normalized textual similarity, while I_{num} measures whether mathematically sensitive elements are preserved. We use $\beta > \alpha$ because value-level errors can change the validity of an entire derivation step even when the surrounding text remains similar.

Logic-based verification reward. The logic-based reward $R_{\text{logic}}(y)$ evaluates whether the model correctly verifies the derivation process

rather than only predicting the final answer. It consists of three normalized components: step-level grading agreement R_{step}, multi-agent verifier feedback R_{agent}, and first-error anchoring R_{FEA}. These components are normalized to $[0, 1]$ and averaged to obtain the final logic reward.

Step-level verification agreement. For each predicted step, R_{step} measures whether the model's step-level correctness prediction matches the reference label. Let $g_t^* \in \{0, 1\}$ denote the reference correctness label for step t, where 1 indicates a correct step and 0 indicates an incorrect step. Let $\hat{g}_t$ denote the model prediction. The step-level agreement score is defined as $r_t^{\text{step}} = 1$ if $\hat{g}_t = g_t^*$, and $r_t^{\text{step}} = 0$ otherwise. R_{step} is obtained by averaging r_t^{step} over all predicted steps.

Multi-agent verifier feedback. To provide external logical evidence, MathStepGrader uses three specialized verifier agents as reward scorers: a Calculator Agent for arithmetic and numerical equivalence, a Symbolic Agent for algebraic transformations, and a TextChecker Agent for textual consistency and label-rationale agreement. For each predicted step, a lightweight router selects applicable agents according to the recognized step content and the model rationale. Arithmetic operations are routed to the Calculator Agent, algebraic transformations to the Symbolic Agent, and textual explanations or consistency checks to the TextChecker Agent. A step may be routed to multiple agents, and their normalized scores are averaged to obtain r_t^{agent}. R_{agent} is then computed by averaging r_t^{agent} over all predicted steps. The verifier agents are used only for reward scoring; gradients are not propagated through them.

First-error anchoring. First-error anchoring provides a trajectory-level signal for locating where the derivation first becomes invalid. We compare the reference first-error step τ with the predicted one $\hat{\tau}$, using a special no-error index for fully correct solutions. $R_{\text{FEA}}(y)$ is set to 1 if $\hat{\tau} = \tau$, and 0 otherwise. This component encourages the model to identify the causal error in the derivation instead of only matching the final correctness label.

Total reward. For each sampled verification trajectory y_i, the final reward combines recognition fidelity and logic-based verification:

$$R(y_i) = \lambda_{\text{rec}}(s) R_{\text{rec}}(y_i) + \lambda_{\text{logic}}(s) R_{\text{logic}}(y_i), \tag{2}$$

where s denotes training progress and $\lambda_{\text{rec}}(s) + \lambda_{\text{logic}}(s) = 1$. Since both reward terms are normalized to $[0, 1]$, the final reward also lies in $[0, 1]$. We use a perception-to-logic curriculum: early training assigns a larger weight to R_{rec} to stabilize value-sensitive transcription, while later training increases R_{logic} to strengthen step-wise verification, verifier-consistent reasoning, and first-error localization.

3.5 GRPO-based Dual-Reward Optimization

We optimize MathStepGrader with Group Relative Policy Optimization (GRPO). Given a training instance x, the policy model samples a group of G candidate verification trajectories $\{y_1, \ldots, y_G\}$. Each candidate is parsed into structured step-level fields and scored online by the dual-reward function, yielding $\{R_1, \ldots, R_G\}$.

GRPO updates the policy according to relative rewards within the sampled group, using the group mean reward as a baseline instead of training an additional value model. Specifically, the advantage of a candidate is computed from its reward relative to the group reward distribution. This is suitable for our setting because the reward combines heterogeneous signals, including recognition fidelity, step-wise correctness, verifier feedback, and first-error anchoring. We further apply KL regularization to keep the updated policy close to a frozen reference model. In the reward-only MathStepGrader setting, the reference model is the original backbone; in the SFT+MathStepGrader setting, it is the supervised warm-up checkpoint. Thus, MathStepGrader performs reward-based policy optimization over sampled verification trajectories rather than reward-weighted supervised learning. Additional optimization details are provided in Appendix C.

4 Experiments

We evaluate MathStepGrader on automated handwritten mathematical verification, examining the effects of step-level visual decomposition, dual-reward optimization, and process-level feedback on final accuracy, recognition-grounded verification, and first-error localization.

4.1 Experimental Setup

Dataset. We evaluate on FERMAT, which contains 2,244 handwritten mathematical solutions across arithmetic, algebra, geometry, trigonometry, and calculus. We split the data by root question into an 80/20 train-test split to avoid question-level leakage, using the test set only for final reporting.

Table 1: Comprehensive results on the FERMAT benchmark. Bold values indicate the best image-based setting in each row. Green values in the MSG column indicate improvements of reward-only MathStepGrader over SFT.

Model Name	Final Acc.						Joint Acc.	
	Oracle Text	Raw E2E	Direct	SFT	MSG	SFT+MSG	Raw E2E	MSG
Qwen2.5-VL-7B	0.891	0.486	0.708	0.742	0.762 (+2.0)	**0.780**	0.354	0.512
Qwen2.5-VL-32B	0.915	0.523	0.735	0.764	0.782 (+1.8)	**0.795**	0.423	0.575
Qwen2.5-VL-72B	0.940	0.548	0.755	0.777	0.791 (+1.4)	**0.815**	0.450	0.610
Qwen3-VL-8B	0.900	0.501	0.748	0.771	0.786 (+1.5)	**0.804**	0.363	0.530
Qwen3-VL-32B	0.929	0.555	0.764	0.788	0.802 (+1.4)	**0.829**	0.450	0.624
Qwen3-VL-235B-A22B	0.951	0.624	0.795	0.826	0.844 (+1.8)	**0.855**	0.492	0.688
DeepSeek-VL2-Tiny	0.846	0.443	0.670	0.702	0.715 (+1.3)	**0.728**	0.327	0.499
DeepSeek-VL2-Small	0.869	0.459	0.704	0.731	0.748 (+1.7)	**0.757**	0.341	0.508
DeepSeek-VL2	0.889	0.481	0.719	0.746	**0.766** (+2.0)	0.762	0.361	0.532
Llama-3.2-11B-Vision	0.880	0.488	0.726	0.766	0.786 (+2.0)	**0.793**	0.381	0.548
Llama-3.2-90B-Vision	0.931	0.555	0.757	0.784	0.813 (+2.9)	**0.833**	0.490	0.655
Gemma-3-4B-it	0.853	0.468	0.704	0.731	0.748 (+1.7)	**0.759**	0.345	0.508
Ovis2-34B	0.906	0.530	0.726	0.753	**0.775** (+2.2)	0.771	0.428	0.604
InternVL3-78B	0.915	0.570	0.751	0.782	0.802 (+2.0)	**0.813**	0.470	0.644

Backbone Models. We evaluate MathStepGrader on diverse open-weight multimodal backbones, including Qwen2.5-VL (Bai et al., 2025b), Qwen3-VL (Bai et al., 2025a), DeepSeek-VL2 (Wu et al., 2024), Llama-Vision, Gemma-3 (Team et al., 2025), Ovis2 (Lu et al., 2024b), and InternVL3 (Zhu et al., 2025), covering different architectures and scales.

Compared Settings. We compare six settings. *Raw E2E* directly feeds the full handwritten image to the model. *Direct* uses automatic step-level crops without training. *SFT* fine-tunes the backbone with supervised step-level annotations. *MSG* applies reward-only MathStepGrader with dual-reward optimization, while *SFT+MSG* further applies the same optimization after supervised warm-up. *Oracle Text* uses ground-truth step transcriptions to isolate logical verification from recognition errors. Handwritten mathematical solutions are text-rich visual documents with symbolic layouts. Since OCR-based document pipelines may propagate upstream perception errors to downstream reasoning (Livathinos et al., 2025), and recent multimodal document models reduce reliance on such pipelines (Liao et al., 2025), we focus on end-to-end multimodal verification rather than standalone recognition-plus-checking systems.

Evaluation Metrics. We report final accuracy and joint accuracy. Final accuracy measures whether the predicted solution-level label matches the human annotation. For image-based settings, joint accuracy requires both a correct final decision and normalized step-level recognition matching. Normalization ignores harmless formatting differences but requires exact preservation of value-sensitive elements, including digits, signs, operators, variables, exponents, and equation transformations; any mismatch in these elements is counted as a recognition failure.

Implementation Details. For all open-weight backbones, Direct, SFT, MSG, and SFT+MSG use the same input format and structured output schema. All step-level inputs are fully automatic agent-only crops generated by the SEA–SSPA–GSVA pipeline without manual boundary correction. During reward-based optimization, multiple grading trajectories are sampled for each instance and scored by the dual-reward function in Section 3.4. Hyper-parameters, reward weights, training schedules, and decoding configurations are fixed before final evaluation to avoid test-set tuning. Additional implementation and run-stability details are provided in Appendix C.

4.2 Main Results

Table 1 reports the main results on FERMAT. All step-level inputs are automatic agent-only crops unless otherwise specified; human-verified crops are reported only as an upper-bound reference in Appendix B.

Overall, MathStepGrader improves final grading accuracy across model families and scales. The comparison between Raw E2E and Direct shows that the largest gains come from replacing full-page

Table 2: Component ablation under the reward-only MathStepGrader setting. Blue values mark the largest drop. Model names are abbreviated due to space limits.

Model	MSG	–Rec.	–Logic	–Ver.	–FEA	Δ
Qwen2.5-7B	**0.762**	0.735	0.746	0.755	0.753	-2.7
Qwen2.5-32B	**0.782**	0.748	0.766	0.773	0.777	-3.3
Qwen2.5-72B	**0.791**	0.766	0.762	0.784	0.782	-2.9
Qwen3-8B	**0.786**	0.755	0.773	0.782	0.780	-3.1
Qwen3-32B	**0.802**	0.775	0.784	0.793	0.797	-2.7
Qwen3-235B	**0.844**	0.808	0.829	0.837	0.835	-3.6
DSVL2-Tiny	**0.715**	0.693	0.702	0.708	0.710	-2.2
DSVL2-Small	**0.748**	0.719	0.737	0.744	0.739	-2.9
DSVL2	**0.766**	0.742	0.751	0.759	0.762	-2.4
Llama-11B	**0.786**	0.759	0.775	0.780	0.777	-2.7
Llama-90B	**0.813**	0.782	0.797	0.806	0.802	-3.1
Gemma-4B	**0.748**	0.722	0.735	0.744	0.742	-2.7
Ovis2-34B	**0.775**	0.753	0.748	0.766	0.771	-2.7
InternVL3-78B	**0.802**	0.768	0.786	0.795	0.791	-3.3

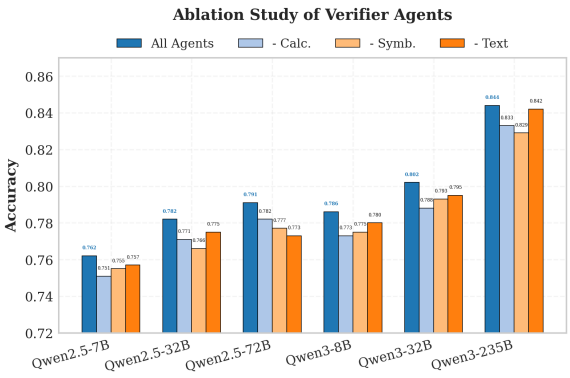

Figure 3: Ablation study of specialized verifier agents on Qwen backbones.

images with ordered step-level visual inputs. Without additional training, Direct improves Qwen2.5-VL-7B from 0.486 to 0.708 and Qwen3-VL-32B from 0.555 to 0.764. This confirms that page-level perception and layout organization are major bottlenecks for handwritten mathematical grading, and that semantic step-level decomposition provides the input representation needed for subsequent process supervision.

After fixing the step-level input representation, training-based settings provide further improvements. SFT improves over Direct by learning the structured step-wise grading format from supervised annotations. More importantly, reward-only MSG improves over SFT on all evaluated backbones, showing that dual-reward process optimization adds useful refinement beyond static supervised learning. For example, Qwen2.5-VL-7B improves from 0.742 under SFT to 0.762 under MSG, and Llama-3.2-90B-Vision improves from 0.784 to 0.813. In most cases, SFT+MSG achieves the best image-based performance, indicating that supervised warm-up and reward-based optimization are generally complementary; for instance, Qwen3-VL-32B reaches 0.829 and Qwen3-VL-235B-A22B reaches 0.855 under SFT+MSG.

The Oracle Text results reveal a clear perception-to-grading gap. When ground-truth step transcriptions are provided, all backbones achieve higher final accuracy than under Raw E2E. For example, Qwen2.5-VL-7B obtains 0.891 with Oracle Text but only 0.486 with Raw E2E. This gap indicates that current MLLMs have stronger logical grading ability when handwriting recognition and layout ambiguity are removed, while end-to-end image-based grading remains limited by perception errors.

Joint accuracy further evaluates whether the final grading decision is grounded in faithfully recognized solution content. Across backbones with available joint evaluation, MSG consistently outperforms Raw E2E. For example, Qwen2.5-VL-32B improves from 0.423 to 0.575, and Qwen3-VL-235B-A22B improves from 0.492 to 0.688. These results show that MathStepGrader improves not only the final correctness label but also recognition-grounded grading faithfulness.

Additional first-error localization results are provided in Appendix D, showing that MathStepGrader improves exact localization of the earliest invalid step over SFT.

4.3 Ablation Study

We conduct ablation studies under the reward-only MathStepGrader setting after fixing the step-level input representation. The full reward-only model is denoted as *MSG*. We consider two levels of ablation: reward-level ablations remove either the recognition reward or the entire logic-based reward, while logic-component ablations remove verifier feedback or first-error anchoring with the remaining logic reward unchanged.

Table 2 reports the component-level ablation results. For compactness, model names in Table 2 are abbreviated, following the same order as Table 1. Removing either the recognition reward or the logic-based reward degrades performance in most settings, showing that faithful transcription and step-wise verification provide complementary supervision. The drop is often larger without the recognition reward, suggesting that reliable logical grading depends on preserving value-sensitive

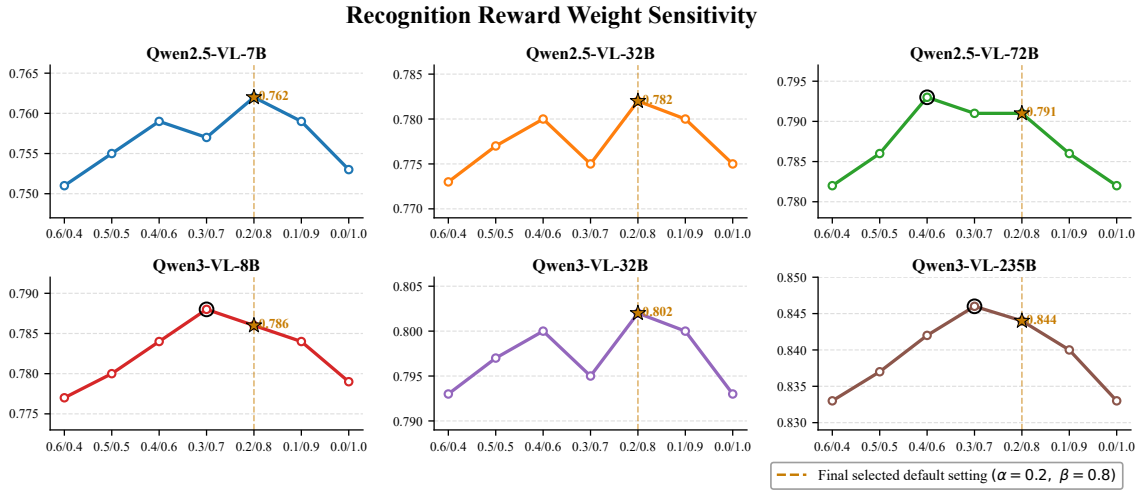

Figure 4: Sensitivity of recognition reward weights under the reward-only MathStepGrader setting on six Qwen backbones.

handwritten content such as digits, operators, variables, and exponents.

Logic-component ablations show that verifier feedback and first-error anchoring provide useful process-level signals. Removing verifier feedback eliminates arithmetic, symbolic, and textual consistency checks, while removing first-error anchoring weakens error localization. Although the drops are moderate and a few backbones show small fluctuations, the full MathStepGrader objective generally achieves the best or near-best performance.

We further analyze verifier agents on the six Qwen backbones in Figure 3. Unlike *w/o Verifier*, this experiment keeps the multi-agent framework and removes one verifier at a time. Removing individual verifiers generally weakens performance, with Calculator and Symbolic agents showing larger effects than TextChecker, suggesting that numerical and algebraic checks are especially important for step-level mathematical verification.

4.4 Hyper-parameter Analysis

We analyze the sensitivity of MathStepGrader to the internal weights of the recognition reward under the reward-only setting. Figure 4 reports results on six Qwen backbones with different (α, β) configurations. Overall, moderately emphasizing value-sensitive preservation yields the best or near-best performance. The default setting $(\alpha = 0.2, \beta = 0.8)$ performs consistently well, suggesting that handwritten mathematical verification benefits from prioritizing digits, signs, operators, variables, and exponents.

Both extremes are less effective. Excessively large β may underweight broader semantic and structural context, while overly large α makes the reward more tolerant of value-sensitive mistakes because global textual similarity may remain high despite an incorrect number or operator. These results suggest that recognition supervision should balance global transcription consistency with mathematical fidelity. Additional reward-balance results between R_{rec} and R_{logic} are provided in Appendix E.

5 Conclusion

We present MathStepGrader, a step-level process supervision framework for reference-free handwritten mathematical verification. Instead of directly predicting a solution-level label from a full handwritten page, MathStepGrader constructs ordered step-level visual units and performs context-aware visual-logical verification under historical derivation context. Its dual-reward objective combines value-sensitive recognition fidelity with logic-based verification, supported by multi-agent feedback and first-error anchoring. Experiments on FERMAT across diverse open-weight multimodal backbones show that semantic step-level decomposition substantially improves over full-image verification, while dual-reward optimization provides consistent additional gains. Further analyses on joint accuracy, ablations, hyper-parameters, and first-error localization demonstrate that reliable handwritten mathematical verification requires both faithful visual recognition and process-level logical reasoning. Together, these results suggest that step-level visual-logical process supervision is a promising direction for building more faithful and interpretable handwritten mathematical verification systems with educational grading applications.

Limitations

Although MathStepGrader improves automated handwritten mathematical grading through step-level process supervision, several limitations remain. First, the framework depends on the quality of step-level visual units. Inaccurate crop boundaries, missing fragments, ambiguous symbols, incomplete strokes, or noisy handwriting may affect step recognition and introduce errors into subsequent logic-based verification. Second, the current multi-agent feedback module relies on predefined verifier roles, including arithmetic, symbolic, and textual consistency checking. While these verifiers cover common mathematical reasoning patterns, they may not fully capture complex proof strategies, highly diagram-dependent reasoning, or domain-specific mathematical conventions. Third, our experiments are conducted mainly on the FERMAT benchmark. Although FERMAT contains diverse handwritten solutions, further evaluation on broader datasets, different handwriting styles, and varied educational scenarios is needed to assess generalization. Future work may explore more robust step segmentation, adaptive verifier selection, and broader cross-domain evaluation.

References

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025b. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Pei Chen, Boran Han, and Shuai Zhang. 2024. Comm: Collaborative multi-agent, multi-reasoning-path prompting for complex problem solving. *Preprint*, arXiv:2404.17729.

Shuaihang Chen, Yuanxing Liu, Wei Han, Weinan Zhang, and Ting Liu. 2025. A survey on llm-based multi-agent system: Recent advances and new frontiers in application. *Preprint*, arXiv:2412.17481.

Lingxiao Du, Fanqing Meng, Zongkai Liu, Zhixiang Zhou, Ping Luo, Qiaosheng Zhang, and Wenqi Shao. 2025. Mm-prm: Enhancing multimodal mathematical reasoning with scalable step-level supervision. *Preprint*, arXiv:2505.13427.

Zhuoxuan Jiang, Haoyuan Peng, Shanshan Feng, Fan Li, and Dongsheng Li. 2025. Llms can find mathematical reasoning mistakes by pedagogical chain-of-thought. *Preprint*, arXiv:2405.06705.

Zhu Jianhua, Gao Liangcai, Zhao Wenqi, Peng Shuai, Hu Pengfei, and Du Jun. 2025. Offline recognition of handwritten mathematical expressions: a survey. *Journal of Image and Graphics*, 30(11):3506–3523.

Ruslan Khrulev. 2025a. Check-mat: Checking handwritten mathematical answers for the russian unified state exam. *Preprint*, arXiv:2507.22958.

Ruslan Khrulev. 2025b. CHECK-MAT: Probing the mathematical reasoning and rubric-alignment of vision-language models on handwritten solutions. In *Proceedings of The 3rd Workshop on Mathematical Natural Language Processing (MathNLP 2025)*, pages 77–94, Suzhou, China. Association for Computational Linguistics.

Jacob Levine, Miguel Aenlle, Craig Zilles, Matthew West, and Mariana Silva. 2026. Automated grading of handwritten mathematics using vision-capable llms. *Preprint*, arXiv:2605.19043.

Wenhui Liao, Jiapeng Wang, Hongliang Li, Chengyu Wang, Jun Huang, and Lianwen Jin. 2025. Doclayllm: An efficient multi-modal extension of large language models for text-rich document understanding. *Preprint*, arXiv:2408.15045.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12).

Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. 2025. Docling: An efficient open-source toolkit for ai-driven document conversion. *Preprint*, arXiv:2501.17887.

Georgios Louloudis, Basilios Gatos, Ioannis Pratikakis, and Constantin Halatsis. 2008. Text line detection in handwritten documents. *Pattern recognition*, 41(12):3758–3772.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024a. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*.

Shiyin Lu, Yang Li, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Han-Jia Ye. 2024b. Ovis: Structural embedding alignment for multimodal large language model. *Preprint*, arXiv:2405.20797.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

Mahshad Mahdavi, Richard Zanibbi, Harold Mouchere, Christian Viard-Gaudin, and Utpal Garain. 2019. Icdar 2019 crohme + tfd: Competition on recognition of handwritten mathematical expressions and typeset formula detection. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1533–1538.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. *Preprint*, arXiv:2308.00436.

Oikantik Nath, Hanani Bathina, Mohammed Safi Ur Rahman Khan, and Mitesh M Khapra. 2025. Can vision-language models evaluate handwritten math? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14784–14814, Vienna, Austria. Association for Computational Linguistics.

Thu Phuong Nguyen, Duc M. Nguyen, Hyotaek Jeon, Hyunwook Lee, Hyunmin Song, Sungahn Ko, and Taehwan Kim. 2025. VEHME: A vision-language model for evaluating handwritten mathematics expressions. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31793–31813, Suzhou, China. Association for Computational Linguistics.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-rong Wen. 2025. Tool learning with large language models: a survey. *Frontiers of Computer Science*, 19(8).

Jin Seong, Wencke Liermann, Minho Kim, Jong hun Shin, and Soojong Lim. 2026. When vlms 'fix' students: Identifying and penalizing over-correction in the evaluation of multi-line handwritten math ocr. *Preprint*, arXiv:2604.22774.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Thanh-Nghia Truong, Cuong Tuan Nguyen, Richard Zanibbi, Harold Mouchère, and Masaki Nakagawa. 2024. A survey on handwritten mathematical expression recognition: The rise of encoder-decoder and gnn models. *Pattern Recognition*, 153:110531.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, Zhenda Xie, Yu Wu, Kai Hu, Jiawei Wang, Yaofeng Sun, Yukun Li, Yishi Piao, Kang Guan, Aixin Liu, and 8 others. 2024. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *Preprint*, arXiv:2412.10302.

Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. 2024. Logicvista: Multimodal llm logical reasoning benchmark in visual contexts. *Preprint*, arXiv:2407.04973.

Zhiqi Yu, Xingping Liu, Haobin Mao, Mingshuo Liu, Long Chen, Jack Xin, and Yifeng Yu. 2026. Evaluating ai grading on real-world handwritten college mathematics: A large-scale study toward a benchmark. *Preprint*, arXiv:2603.00895.

Di Zhang, Junxian Li, Jingdi Lei, Xunzhi Wang, Yujie Liu, Zonglin Yang, Jiatong Li, Weida Wang, Suorong Yang, Jianbo Wu, Peng Ye, Wanli Ouyang, and Dongzhan Zhou. 2025a. Critic-v: Vlm critics help catch vlm errors in multimodal reasoning. *Preprint*, arXiv:2411.18203.

Jianghangfan Zhang, Yibo Yan, Kening Zheng, Xin Zou, Song Dai, and Xuming Hu. 2025b. Gm-prm: A generative multimodal process reward model for multimodal mathematical reasoning. *Preprint*, arXiv:2508.04088.

Tianyang Zhang, Zhuoxuan Jiang, Haotian Zhang, Lin Lin, and Shaohua Zhang. 2025c. Mathmistake checker: A comprehensive demonstration for step-by-step math problem mistake finding by prompt-guided llms. *Preprint*, arXiv:2503.04291.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, and 32 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *Preprint*, arXiv:2504.10479.

A Preprocessing Details

This appendix details the semantic-first multi-agent preprocessing pipeline introduced in Section 3.2. Given a raw handwritten solution image I, the pipeline constructs an ordered sequence of step-level crops $S = \{i_1, \ldots, i_n\}$, where each crop corresponds to a logically atomic reasoning step. As shown in Figure 5, the pipeline consists of three collaborative agents: the Sensory Enhancement Agent (SEA), the Semantic-Spatial Perception Agent (SSPA), and the Geometric Stitching and Validation Agent (GSVA).

A.1 Sensory Enhancement Agent

The Sensory Enhancement Agent (SEA) standardizes raw handwritten images before semantic-spatial parsing. In our dataset, most images are complete and globally upright, so the main difficulty does not come from page-level rotation or perspective distortion. Instead, student solutions often contain local writing drift, irregular line trajectories, marginal notes, background artifacts, and varying stroke contrast. SEA is therefore designed to improve the visual readability of handwritten mathematical content while preserving the original spatial structure for step-level grouping.

SEA first resizes each input image to a unified resolution range while preserving its aspect ratio. Given an input image I with height H and width W, we compute the resizing scale as:

$$s = \min\left(1, \frac{L}{\max(H, W)}\right), \tag{3}$$

where L denotes the maximum allowed length of the longer side. The resized image is obtained as $I_{\text{raw}} = \text{Resize}(I, sH, sW)$. This operation reduces computational overhead and avoids excessive visual tokens while preserving the geometry of fractions, exponents, radicals, and integral signs.

SEA then enhances stroke visibility through contrast adjustment and edge sharpening. These operations improve the separation between handwriting strokes and background noise caused by ruled lines, shadows, faint ink, or local over-writing. To reduce irrelevant visual content, SEA further estimates a lightweight handwritten foreground mask and suppresses peripheral artifacts only when they are spatially detached from the main solution region. This suppression is applied conservatively to avoid removing layout cues that may be useful for semantic-spatial step grouping.

A.2 Semantic-Spatial Perception Agent

The Semantic-Spatial Perception Agent (SSPA) performs semantic-spatial grouping over the standardized image produced by SEA. Its goal is to determine which visual fragments belong to the same logical reasoning step. Unlike physical line segmentation, SSPA jointly considers textual content, mathematical structure, and spatial layout cues. This is important for handwritten mathematical solutions, where a single reasoning step may span multiple physical lines, while visually adjacent fragments may correspond to different logical steps.

Given the enhanced image I_{enh}, SSPA extracts candidate text fragments, their spatial locations, and their logical step assignments:

$$\mathcal{B} = \{(t_k, b_k, z_k)\}_{k=1}^{K}, \tag{4}$$

where t_k denotes the recognized text fragment, $b_k = [y_{\min}, x_{\min}, y_{\max}, x_{\max}]$ denotes its bounding box in image coordinates, and $z_k \in \{1, \ldots, n\}$ denotes the predicted logical step identifier. Fragments with the same z_k are treated as belonging to the same reasoning step and are later stitched by GSVA into a step-level crop.

SSPA uses both semantic and spatial evidence. Semantic evidence includes logical anchors such as "Solution", "From Eq. (1)", "Therefore", and "In conclusion", which often indicate transitions between reasoning units. Spatial evidence includes indentation, line breaks, local spacing, alignment patterns, and trailing whitespace. Combining these cues allows SSPA to group fragments according to derivation logic rather than physical layout alone.

In our implementation, SSPA is instantiated with Qwen3.5-Plus, an instruction-following multimodal model with text recognition, coordinate grounding, and structured output capabilities. Given the input image and a predefined JSON schema, the model outputs recognized fragments, bounding boxes, and logical step identifiers. The SSPA prompt uses only the input image and the output schema; it does not access solution-level labels, step-level correctness annotations, or test-set information. Figure 6 shows the prompt template, segmentation rules, and structured JSON output format used by SSPA.

Figure 7 illustrates an example parsing result. The left panel presents the structured JSON output, while the right panel visualizes how detected fragments are grouped into logical steps on the orig-

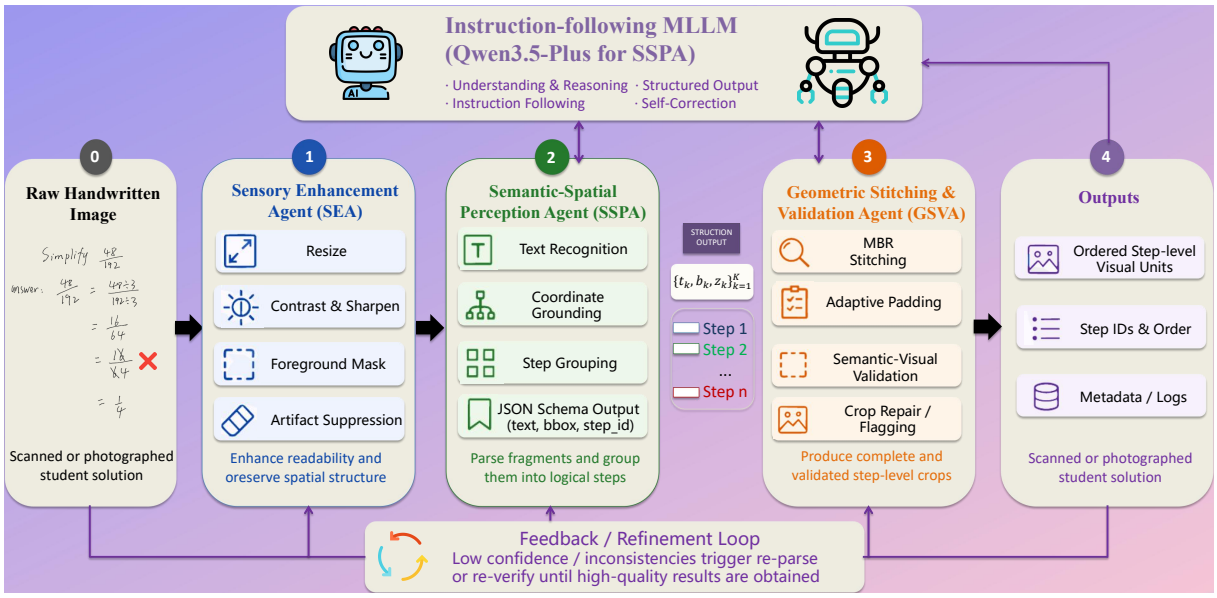

Figure 5: Overview of the semantic-first multi-agent pre-processing pipeline. The system normalizes raw handwritten solution images, identifies semantic-spatial step units, and converts them into verified step-level crops.

inal handwritten solution. The example shows that SSPA can assign physically separated fragments to the same `step_id` when they jointly express one intermediate computation, and can separate visually adjacent fragments when they correspond to different reasoning units. This semantic-first behavior differs from purely physical segmentation, which may split a multi-line reasoning step or merge unrelated fragments.

The parsed output $\mathcal{B}$ is not used as the final training instance directly. Instead, it serves as the semantic-spatial input for GSVA, which performs geometric stitching and validation in the next stage. Formally, SSPA induces a semantic-spatial parsing function $g_\theta : I_{\text{enh}} \to \mathcal{B}$, where g_θ denotes the parser instantiated by Qwen3.5-Plus.

A.3 Geometric Stitching & Validation Agent

The Geometric Stitching & Validation Agent (GSVA) finalizes the step-level image crops based on the semantic-spatial fragments produced by SSPA. Given the parsed fragment set $\mathcal{B} = \{(t_k, b_k, z_k)\}_{k=1}^{K}$, GSVA has two main functions: adaptive geometric stitching and semantic-visual consistency validation.

(1) Adaptive geometric stitching. For all fragments assigned to the same logical step identifier $z_k = t$, GSVA first maps their normalized bounding boxes back to the original image coordinate system and computes a global minimum bounding rectangle (MBR). To avoid truncating extended mathematical notations, such as fractions, radicals, integral signs, superscripts, and descenders, GSVA further applies an adaptive padding term p_t according to the spatial distribution of the grouped fragments. The final crop for step t is formulated as:

$$i_t = \text{Crop}\Big(I_{\text{orig}}, \text{MBR}(\{\phi(b_k) \mid z_k = t\}) \oplus p_t\Big), \tag{5}$$

where I_{orig} denotes the original-resolution image, $\phi(\cdot)$ maps the normalized SSPA coordinates back to the original image space, and $\oplus p_t$ denotes padding expansion around the MBR. This operation converts semantically grouped fragments into a complete physical crop while preserving fine-grained handwritten mathematical details.

(2) Semantic-visual consistency validation. After geometric stitching, GSVA performs semantic-visual consistency validation before the crop is accepted as a step-level visual unit. The key idea is to use the SSPA output as an expected semantic reference: for step t, the recognized fragments $\{t_k \mid z_k = t\}$ describe the mathematical content that should appear in the final crop i_t. GSVA then re-inspects the cropped image with a multimodal checker and verifies whether the visual content of i_t matches this expected step-level text. The checker uses only the crop and the SSPA-produced semantic reference, without accessing solution-level labels, step-level correctness labels, or test-set information.

This validation focuses on three common crop-level failure cases:

Role: You are an expert in document image analysis and logical structure parsing.

Task: Analyze the provided handwritten mathematical solution image. Extract all handwritten text fragments, identify their bounding boxes, and group them into reasoning steps according to mathematical derivation logic and visual layout.

Rules for Step Segmentation:
1. Semantic logic: Pay attention to logical connectors such as “Therefore”, “Thus”, “Hence”, “In conclusion”, or explicit numbering, which often indicate the beginning of a new reasoning step.
2. Visual layout cues: Pay close attention to line breaks and blank spaces. If a sentence has ended and the following sentence starts on a new line with clear spacing, it should be treated as a new logical step and assigned a new step_id.
3. Continuity: A long sentence or formula that spans multiple physical lines should share the same step_id.
4. Problem statements should be extracted as a separate fragment.

Output Format:
Return only a JSON array. Do not include Markdown, explanations, prefixes, or suffixes. Each JSON object represents a text line or short phrase and contains:
- step_id: an integer starting from 1. Text blocks belonging to the same logical derivation step should share the same step_id.
- text: the extracted handwritten text.
- bbox: [ymin, xmin, ymax, xmax], the relative bounding box coordinates in the image, normalized to the range 0–1000.

Example Output:
```
[
  {"step_id": 1, "text": "Therefore, the third side has to be less than", "bbox": [150, 100, 200, 900]},
  {"step_id": 1, "text": "the sum of the two sides. The third side is thus,", "bbox": [220, 80, 270, 920]},
  {"step_id": 1, "text": "less than 8 + 6 = 14 cm.", "bbox": [290, 80, 340, 500]},
  {"step_id": 2, "text": "The side cannot be less than the difference of", "bbox": [380, 120, 430, 900]}
]
```

Figure 6: Prompt template and structured output schema used by SSPA.

- **Boundary truncation or overflow:** the crop cuts off important mathematical symbols, such as fractions, superscripts, radicals, or equation endings, or includes content from neighboring steps;
- **Non-semantic artifacts:** the crop contains detached marks, marginal notes, background noise, or irrelevant strokes that do not belong to the current reasoning step;
- **Incomplete logical content:** the crop misses one or more fragments required to express the complete step, causing the visual crop to be inconsistent with the expected semantic content.

When an issue is detected, GSVA first attempts to repair the crop by expanding the bounding region or refining the merged box according to the missing or overflowed fragments. If the inconsistency cannot be resolved automatically, the sample is flagged for quality inspection. Unless otherwise specified, the main experiments use the automatically produced agent-only crops without manual boundary correction; human-verified crops are used only as an upper-bound reference in the appendix. This validation stage reduces boundary errors and missing-content artifacts, ensuring that the final step-level crops are visually complete and semantically aligned with their corresponding logical reasoning steps.

B Annotation Efficiency and Crop Quality

This appendix analyzes the efficiency of the semantic-first preprocessing pipeline and the effect of crop quality on downstream grading. Since MathStepGrader uses step-level visual units as its default input, we compare pure manual annotation, automatic agent-only crops, and human-verified crops on the FERMAT dataset, which contains 2,244 handwritten solution images.

All main experiments use agent-only crops generated by the automatic SEA–SSPA–GSVA pipeline without manual boundary correction. Human-verified crops are included only as an upper-bound reference to quantify the remaining gap caused by crop quality.

As shown in Table 3, the SEA–SSPA–GSVA pipeline substantially reduces annotation cost. The average processing time per image decreases from

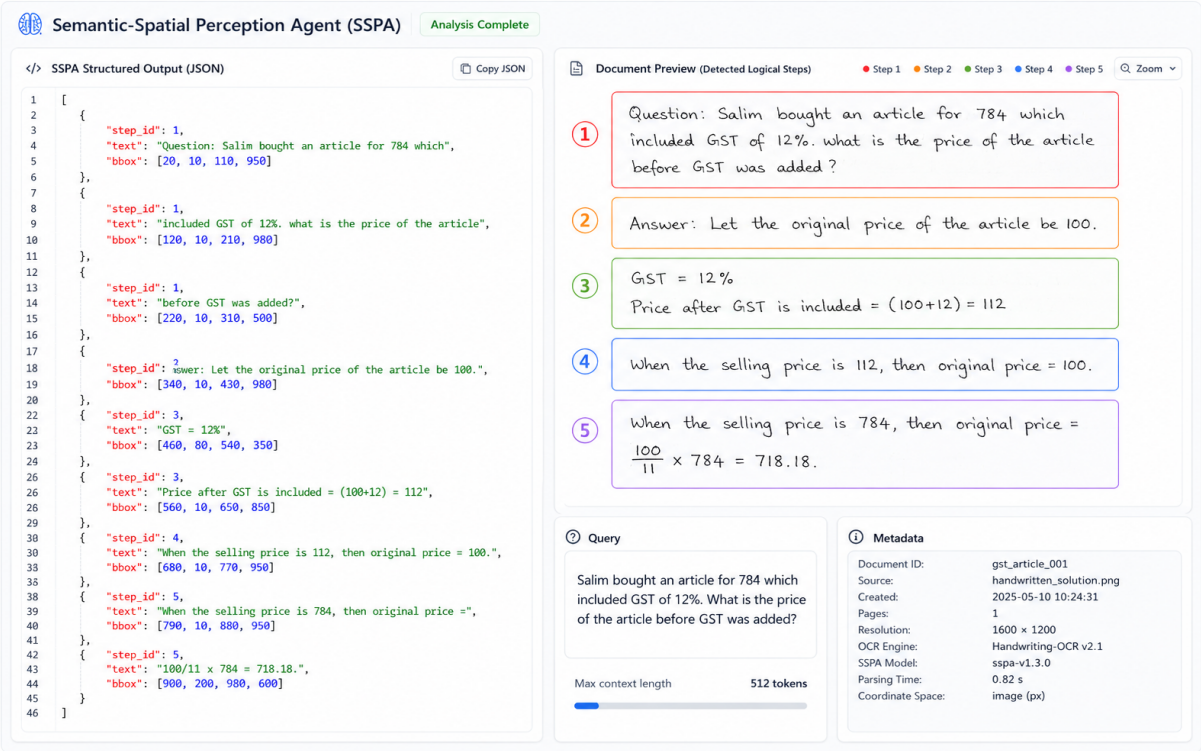

Figure 7: Example output of the Semantic-Spatial Perception Agent (SSPA).

Table 3: Annotation efficiency of the semantic-first preprocessing pipeline on the FERMAT dataset.

Metric	Manual	Agent-only
Total samples	2,244	2,244
Avg. time per image	120s	**18s**
Crop acceptance rate	99.5%	94.2%
Total annotation time	≈74h	**≈11h**

120 seconds to 18 seconds, reducing the total annotation time from approximately 74 hours to 11 hours. Although fully manual verification can achieve a slightly higher crop acceptance rate, the automatically generated agent-only crops already reach 94.2% acceptance, providing a reliable and scalable input source for downstream training and evaluation.

We further evaluate how different input types affect downstream grading performance. As shown in Table 4, full-image input performs the worst because the model must jointly handle page-level perception, layout organization, and logical grading. In contrast, agent-only crops already provide substantial gains by decomposing the full handwritten page into ordered step-level visual units. Human-verified crops further improve performance, but the gain is relatively small compared with the improvement from full-image input to agent-only crops. Considering the much higher manual cost required by human verification, we use agent-only crops as the default input in the main experiments, while reporting human-verified results here as an upper-bound reference.

Overall, these results show that the major improvement comes from semantic step-level decomposition rather than manual correction alone. The agent-only pipeline removes redundant whitespace, preserves coherent reasoning units, and reduces non-informative visual tokens, enabling the vision encoder to focus on handwritten mathematical content. In addition, the processed crops reduce the storage footprint from 4.61 GB to 900 MB, improving data density and I/O efficiency for large-scale multimodal training.

C Implementation Details

This appendix provides additional details about data formatting, supervised fine-tuning, reward-based optimization, inference, computational resources, reward scheduling, and implementation choices for large-scale backbones.

Preprocessing model. For preprocessing, SSPA is instantiated with a strong instruction-following MLLM to obtain reliable semantic-spatial step

Table 4: Effect of full-image input and step-level crop quality on downstream grading. Agent-only crops are produced by the SEA–SSPA–GSVA pipeline without manual correction, while human-verified crops additionally require manual boundary checking.

Input Type	Setting	Qwen2.5-VL			Qwen3-VL			DeepSeek-VL2		
		7B	32B	72B	8B	32B	235B	Tiny	Small	Base
Full image	Direct	0.486	0.523	0.548	0.501	0.555	0.624	0.443	0.459	0.481
	SFT	0.552	0.595	0.624	0.568	0.628	0.706	0.508	0.526	0.552
Agent-only crops	Direct	0.708	0.735	0.755	0.748	0.764	0.795	0.670	0.704	0.719
	SFT	0.742	0.764	0.777	0.771	0.788	0.826	0.702	0.731	0.746
Human-verified crops	Direct	0.724	0.751	0.762	0.771	0.780	0.811	0.688	0.719	0.735
	SFT	**0.755**	**0.777**	**0.784**	**0.791**	**0.802**	**0.820**	**0.717**	**0.744**	**0.760**

grouping. This model is used only to construct step-level visual units and does not access solution-level correctness labels or grading decisions. All downstream backbones are evaluated on the same automatically generated crops, so the reported comparisons reflect differences in visual-logical verification under a fixed input transformation. The quality of the upstream parser remains an important dependency, which we discuss in the limitations.

Computational resources and training framework. All trainable open-weight experiments are conducted on a server with 4 NVIDIA A100 GPUs. We use LoRA-based parameter-efficient fine-tuning for all trainable backbones to reduce memory consumption while preserving the pre-trained multimodal model. Depending on model compatibility, supervised fine-tuning is implemented with LLaMA-Factory, ms-swift, or Hugging Face PEFT. Reward-based optimization is implemented with GRPO-compatible training backends. Across backbones and training frameworks, we keep the same input schema, output format, reward computation, and evaluation protocol. We use bfloat16 mixed precision when supported and enable gradient checkpointing for large models. For very large backbones whose full reward optimization is computationally expensive, we keep the visual encoder frozen and update only LoRA adapters on the language-model projection and attention/MLP layers. Inference is performed with batched generation under the same decoding configuration within each backbone family.

Input construction. For Raw E2E, the model receives the original full-page handwritten solution image. For Direct, SFT, MSG, and SFT+MSG, the model receives an ordered sequence of step-level crops generated by the automatic SEA–SSPA–GSVA preprocessing pipeline. These are agent-only crops and do not involve manual boundary correction in the main experiments. Step crops are sorted according to their predicted logical order and paired with the same task instruction across all step-level settings. This ensures that performance differences among Direct, SFT, MSG, and SFT+MSG mainly reflect adaptation and reward optimization rather than changes in input formatting.

Output schema. All step-level settings use the same structured output schema. The model is required to produce: (1) a transcription for each step, (2) a step-level correctness label, (3) a short rationale for the step judgment, (4) the final solution-level correctness decision, and (5) the predicted first erroneous step. The output is parsed into structured fields before reward computation and evaluation. Invalid JSON outputs or missing required fields are treated as malformed trajectories and receive a low format score during reward-based optimization. During evaluation, final accuracy is computed from the predicted solution-level correctness label. Joint accuracy is computed only when both the recognized solution content and the final decision match the reference.

Supervised fine-tuning. For SFT, we fine-tune the backbone models on the training split using structured step-level annotations. The SFT target follows the same output schema used during inference. Unless otherwise specified, LoRA adapters are applied to the major projection layers of the language model, including query, key, value, output, gate, up, and down projection layers when supported by the backbone. We use rank $r = 8$,

scaling factor $\alpha_{\text{LoRA}} = 32$, and LoRA dropout of 0.05. Optimization is performed with AdamW, cosine learning-rate decay, and a short warm-up stage. The same SFT recipe is used for the SFT setting and for the supervised warm-up stage in SFT+MathStepGrader. For large-scale backbones, we reduce the per-device batch size and use gradient accumulation to keep the effective batch size comparable across models.

GRPO-based reward optimization. For the reward-only MSG setting, the policy is initialized from the original backbone. For SFT+MathStepGrader, the policy is initialized from the corresponding SFT checkpoint. For each training instance, the policy samples a group of $G = 4$ structured trajectories. Each trajectory is parsed into step-wise predictions and scored by the dual-reward function described in Section 3.4. GRPO updates the policy using relative rewards within the sampled group, with the group mean reward serving as the baseline. This avoids training an additional value model and is suitable for our setting, where the reward combines recognition fidelity, step-level correctness, verifier feedback, and first-error anchoring. We apply KL regularization against a frozen reference policy: the original backbone is used as the reference for MSG, while the SFT checkpoint is used as the reference for SFT+MathStepGrader.

Training hyper-parameters. Unless otherwise specified, SFT uses AdamW with a learning rate of 2×10^{-5}, cosine decay, a warm-up ratio of 0.03, and an effective batch size of 128. We train for 3 epochs and select the final checkpoint under the fixed training budget. For GRPO-based reward optimization, we use a learning rate of 1×10^{-6}, an effective batch size of 64, group size $G = 4$, and a KL coefficient of 0.04 against the frozen reference policy. The maximum generation length is set to 1024 tokens for all structured outputs. The group size $G = 4$ is chosen as a trade-off between GPU memory constraints, long multimodal context length, and the need for stable within-group reward comparison on 4 A100 GPUs.

Reward computation and scheduling. The recognition reward uses the default setting ($\alpha = 0.2, \beta = 0.8$), which emphasizes value-sensitive transcription while retaining a global textual similarity term. The logic reward combines step-level correctness agreement, multi-agent verifier feedback, and first-error anchoring. All reward components are normalized before aggregation. To reduce early reward noise, we use a perception-to-logic reward schedule. At the beginning of reward optimization, a larger weight is assigned to R_{rec} to stabilize handwritten recognition and value-sensitive transcription. As training proceeds, the weight of R_{logic} is gradually increased to strengthen step-wise logical verification and first-error localization. Formally, the final reward is

$$R(y) = \lambda_{\text{rec}}(s) R_{\text{rec}}(y) + \lambda_{\text{logic}}(s) R_{\text{logic}}(y),$$

where s denotes training progress and $\lambda_{\text{rec}}(s) + \lambda_{\text{logic}}(s) = 1$. In our default schedule, the reward balance is updated from a recognition-oriented stage to a more logic-oriented stage during training. The sensitivity of this schedule is analyzed in Appendix E.

Table 5: First-error localization accuracy on Qwen backbones. Δ reports the improvement of MSG over SFT in percentage points.

Backbone	Direct	SFT	MSG	Δ
Qwen2.5-VL-7B	0.474	0.514	0.566	+5.2
Qwen2.5-VL-32B	0.503	0.552	0.604	+5.2
Qwen2.5-VL-72B	0.532	0.581	0.626	+4.5
Qwen3-VL-8B	0.492	0.539	0.590	+5.1
Qwen3-VL-32B	0.548	0.597	0.653	+5.6
Qwen3-VL-235B-A22B	0.597	0.641	0.688	+4.7

Verifier agents. The verifier module contains three specialized agents. The Calculator Agent checks arithmetic consistency and numerical equivalence. The Symbolic Agent checks algebraic transformations and expression equivalence. The TextChecker Agent checks textual mathematical statements and local consistency between the transcription, rationale, and step judgment. A lightweight router selects applicable agents according to the recognized step content and the model rationale. Arithmetic expressions are routed to the Calculator Agent, algebraic transformations to the Symbolic Agent, and textual explanations or label-rationale consistency checks to the TextChecker Agent. A step may be routed to multiple agents, and their normalized scores are averaged as verifier feedback. These verifier agents are used only for scoring sampled trajectories; gradients are not propagated through them.

Model selection and test usage. To avoid test-set tuning, hyper-parameters, reward weights, training

Table 6: Effect of reward balance under the reward-only MathStepGrader setting on Qwen backbones. Fixed settings use constant $(\lambda_{\text{rec}}, \lambda_{\text{logic}})$, while dynamic updating adjusts the reward balance according to the training stage. Bold values indicate the best result in each column.

Setting	λ_{rec}	λ_{logic}	Qwen2.5-VL-7B	Qwen2.5-VL-32B	Qwen2.5-VL-72B	Qwen3-VL-8B	Qwen3-VL-32B	Qwen3-VL-235B
Rec-heavy	0.7	0.3	0.757	0.777	0.786	0.780	0.797	0.844
Slight Rec	0.6	0.4	0.759	0.780	0.788	0.784	0.800	**0.862**
Balanced	0.5	0.5	0.755	0.775	0.784	0.782	0.802	0.837
Slight Logic	0.4	0.6	0.751	0.773	0.788	0.784	0.795	0.846
Logic-heavy	0.3	0.7	0.746	0.768	0.777	0.775	0.791	0.822
Dynamic update	–	–	**0.762**	**0.782**	**0.791**	**0.786**	**0.802**	0.844

schedules, and decoding configurations are fixed before final test evaluation and kept unchanged across comparable settings. We do not select checkpoints according to test performance. Instead, each training-based setting uses the checkpoint obtained under the same fixed training budget. The test split is used only for final reporting. We additionally repeat representative runs with different random seeds to assess the stability of the reported trends.

Joint accuracy computation. Before computing joint accuracy, predicted and reference step transcriptions are normalized by removing whitespace, standardizing mathematical symbols, and ignoring harmless punctuation or wording differences. We then compare the solution content at the step level. A recognition match requires preservation of mathematically relevant elements, including numerical values, signs, operators, variables, exponents, and equation transformations. For algebraic expressions that can be parsed by the symbolic checker, we additionally allow symbolically equivalent forms to match even when their surface strings differ. If symbolic parsing fails, we fall back to the normalized value-sensitive matching rule. The final joint decision is correct only when all step-level recognition checks pass and the predicted final correctness label matches the human annotation.

D First-error Localization Analysis

We further evaluate whether MathStepGrader can identify the causal point where a derivation first becomes invalid. This analysis complements final grading accuracy, because a model may predict the correct solution-level label without correctly locating the earliest erroneous step. We report First-error Localization Accuracy, where a prediction is counted as correct only when the predicted first erroneous step exactly matches the reference. For fully correct solutions, both the reference and prediction use a special no-error index.

As shown in Table 5, MathStepGrader consistently improves first-error localization over both Direct and SFT across Qwen backbones. The improvement over SFT is consistently around five percentage points, suggesting that dual-reward process supervision helps the model locate the earliest causal error rather than only predicting the final solution-level label. This result supports the role of first-error anchoring as a trajectory-level signal for process-faithful grading.

E Reward Balance Analysis

We further analyze the balance between recognition and logic-based grading rewards on six Qwen backbones under the reward-only MathStepGrader setting. We vary $(\lambda_{\text{rec}}, \lambda_{\text{logic}})$, which control the relative contributions of R_{rec} and R_{logic}. Fixed configurations use constant reward weights throughout training, while dynamic updating gradually shifts the optimization focus from recognition fidelity to logic-based verification.

As shown in Table 6, recognition-oriented fixed configurations generally outperform logic-heavy settings. This is because logic-based grading is applied to recognized step representations: if the model misreads numerical values, operators, variables, or symbolic expressions, the logic reward may be computed over a corrupted reasoning state. Therefore, emphasizing recognition fidelity helps stabilize image-to-text grounding and provides a more reliable basis for subsequent verification.

However, an overly recognition-heavy setting is not optimal either. When the model focuses too much on transcription fidelity, it may preserve local mathematical symbols but still fail to judge whether the transformation between steps is logically valid. Conversely, logic-heavy settings may introduce noisy feedback before handwritten recognition has been sufficiently stabilized. These observations suggest that the relative importance of recognition and logic rewards changes during training.

Motivated by this trend, MathStepGrader adopts dynamic reward updating. Training starts with a stronger emphasis on R_{rec} to improve value-sensitive transcription, and then gradually increases the contribution of R_{logic} to strengthen step-wise verification and first-error localization. The dynamic schedule achieves the best or near-best performance across Qwen backbones, indicating that handwritten mathematical grading benefits from a curriculum that moves from perception stabilization to logical verification.