Binned Relative Policy Optimization towards Efficient Thinking of VLMs

Anonymous ACL submission

Abstract

Recent VLMs improve multimodal reasoning by generating explicit chains of thought, but often overthink with redundant reasoning. Although conventional length penalties mitigate this redundancy, they rely on coarse global rewards that fail to discern intermediate process quality or adapt to task complexity. To address this dilemma, we propose Binned Relative Policy Optimization (BRPO), a training-efficient framework that enables non-trivial process-level evaluation without introducing costly process reward models. Specifically, BRPO leverages high entropy lexical markers to segment reasoning trajectories into discrete intermediate prefixes, evaluating their problem-solving potential through forced answer completions. Building on this, we introduce Bin Normalization along with a length-aware adjustment to map prefixes into process bins, dynamically calibrating rewards and regulating length constraints based on reasoning stages and visual complexity. Experiments on six multimodal math benchmarks show that our method improves both accuracy and efficiency: on Qwen2.5-VL-7B, it raises average accuracy from 47.8% to 53.6% while reducing the average response length by 39.0%.

1 Introduction

Recent advancements in Vision-Language Models (VLMs) have made substantial progress in multimodal reasoning. By leveraging algorithms like Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to incentivize explicit chains of thought, VLMs can solve highly complex visual problems (Wei et al., 2023; Zhang et al., 2024b; Deng et al., 2025; Wang et al., 2025a). However, this ability also brings a tendency to overthink: even when the visual evidence is simple or already sufficient, VLMs may continue producing lengthy reasoning chains. Such redundant reasoning not only increases inference cost but also introduces spurious premises that harm final answers (Zhou et al., 2026; Cuesta-Ramirez et al., 2025; Sui et al., 2025).

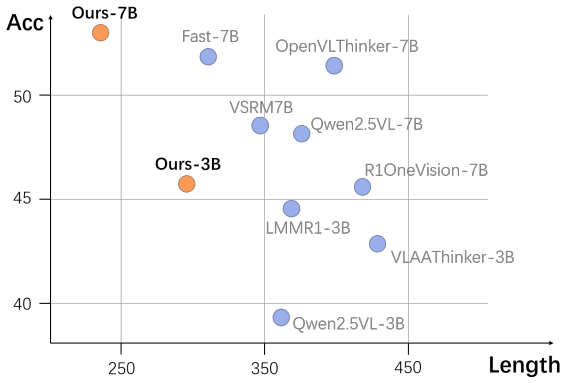

Figure 1: Comparison of average accuracy and response length across six benchmarks.

A straightforward way to discourage overthinking is to add length penalties during reinforcement learning (Shen et al., 2026; Ling et al., 2025; Xiang et al., 2025). However, because mainstream algorithms like GRPO rely exclusively on outcome-level rewards to judge the final answer, existing formulations can only apply these penalties globally across the entire reasoning trajectory. This setup makes it exceptionally challenging to discern which segments of a Chain of Thought (CoT) provide genuine intellectual value and which are merely redundant.

In this paper, we address the overthinking issue from the perspective of process-level rewarding. Without introducing costly process reward models, we propose Binned Relative Policy Optimization (BRPO), which enables process-level evaluation through a lightweight prefix splitting mechanism. Grounded in the insight that high entropy lexical markers often signify uncertainty, deliberation, or strategic shifts, we utilize a predefined set of pivot phrases (e.g., wait, but, however) as demarcation points to segment the generated reasoning text into

distinct intermediate prefixes. To assess the problem solving potential of each prefix, we compel the model to truncate its ongoing reasoning to directly produce final answers, subsequently verifying the correctness of their outcomes.

At the core of our proposed BRPO is the Bin Normalization technique. Standard GRPO computes vanilla Group Normalization across multiple full length outputs sampled from the same prompt. However, directly applying such global normalization to process-level prefix rewards is inherently flawed. The underlying value of an intermediate reasoning prefix is tightly coupled with its temporal stage: a moderate reward early in the trajectory may signal a promising reasoning direction, whereas the same reward near the sequence end implies stagnation. To rectify this, we introduce the prefix ratio, defined as the length of the current prefix relative to the total response length, as a topology-agnostic proxy for the reasoning stage. BRPO then maps each prefix to its corresponding progress bin based on its prefix ratio, and subsequently normalizes rewards independently within each localized bin. The bin boundaries are initialized uniformly and dynamically adjusted using batch statistics to counteract uneven prefix distributions.

To promote efficiency without inducing premature shortcuts, we further introduce a length-aware reward mechanism. Rather than enforcing a static, uniform penalty, we dynamically regularize each prefix contingent on its bin-conditioned performance. Specifically, brevity is credited only when a prefix outperforms the specific bin average baseline, whereas excessively long reasoning is penalized if its performance falls below the average. This adjustment is adaptively scaled by the prefix ratio, imposing mild length pressure during early visual grounding but accelerating the constraint toward the sequence end, where protracted generation typically stems from redundant verification or delayed termination. Moreover, a visual-complexity coefficient is added to modulate the compression intensity, enforcing concise deduction on straightforward tasks while safeguarding an ample reasoning budget for intricate scenes.

We validate our method on six prominent multimodal mathematical reasoning benchmarks: MathVision (Wang et al., 2024a), MathVerse (Zhang et al., 2024a), MathVista (Lu et al., 2024), DynaMath (Zou et al., 2025), MM-Math (Sun et al., 2024), and WeMath (Qiao et al., 2024). Built upon Qwen2.5-VL-7B (Bai et al., 2025), our method improves the average accuracy from 47.8% to 53.6% while achieving a 39.0% reduction in average response length. Outperforming both general reasoning methods and competitive adaptive/efficient thinking baselines, our model delivers superior accuracy with remarkable conciseness (Figure 1). These outcomes underscore that our method concurrently optimizes reasoning quality and computational efficiency, rather than settling for a simplistic trade-off via response shortening.

2 Related Work

Process Rewards. Process supervision improves multi-step reasoning by providing feedback to intermediate steps rather than only final answers (Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2024b). Recent works scale process rewards with automated verification, Monte Carlo rollouts, and learned PRMs (Setlur et al., 2024; Wang et al., 2025c; Chen et al., 2026). However, most methods treat step scores as directly comparable scalar rewards. We instead calibrate process rewards by reasoning progress, comparing each prefix only with prefixes at a similar stage.

VLM Reasoning and Overthinking. Chain-of-thought reasoning improves the reasoning ability of LLMs (Wei et al., 2023; Kojima et al., 2023) and has been extended to VLMs for multimodal mathematical and visual reasoning (Zhang et al., 2024b; Peng et al., 2025; Wang et al., 2025a; Huang et al., 2026). Recent methods encourage longer reasoning through RL, self-reflection, or tool use (Deng et al., 2025; Wu et al., 2026). While beneficial for hard problems, long reasoning can cause overthinking, redundant verification, and visual hallucination (Zheng et al., 2024; Cuesta-Ramirez et al., 2025). Our work focuses on reducing redundant reasoning while preserving useful visual reasoning.

Length Penalties and Adaptive Thinking. Length penalties and adaptive thinking methods aim to reduce unnecessary reasoning by controlling token budgets or adjusting thinking length according to difficulty (Shen et al., 2026; Xiao and Gan, 2025; Han et al., 2025; Wen et al., 2025). These methods show that reasoning length should vary across problems. However, outcome-level length penalties are coarse-grained and cannot identify where redundancy occurs. We introduce a process-level, quality-gated length adjustment that rewards concise prefixes only when they outperform a bin-

conditioned baseline, with strength modulated by prefix ratio and visual complexity.

3 Binned Relative Policy Optimization

3.1 Framework

Given a visual reasoning problem $x = (I, q)$, where I is the image and q is the question, a policy model π_θ generates multiple reasoning responses, denoted as $\mathcal{Y} = \{y^a\}_{a=1}^N$. Each response is a token sequence $y^a = (y_1^a, \ldots, y_{T^a}^a)$, where T^a is the response length. Our goal is to train the model to produce responses that are both accurate and efficient. Instead of only assigning an outcome-level reward to the final answer, we construct process-level rewards for intermediate prefixes of the response and use them to guide policy optimization.

The overview of our method is shown in Figure 2. It consists of three steps. First, we extract intermediate reasoning prefixes and assess their problem-solving potential by forcing the model to complete final answers from each prefix and verifying the completed answers. Second, we calibrate the raw prefix-level rewards by comparing each prefix only with prefixes at a similar reasoning stage. Third, we add a quality-gated length adjustment to encourage concise reasoning without rewarding premature shortcuts. The calibrated process rewards are then propagated to token-level advantages and used to optimize the policy model.

3.2 Prefix Thinking Verification

For a generated response $y^a = (y_1^a, \ldots, y_{T^a}^a)$ to problem q, we first identify a set of intermediate reasoning prefixes. Following prior observations that high entropy lexical markers often correspond to uncertainty, deliberation, or transitions in reasoning (Wang et al., 2025b; Yue et al., 2025), we construct a predefined set of high entropy words and phrases, including *wait*, *but*, and *however*, among others. These markers indicate points where the model may revise, branch, or consolidate its reasoning, and thus serve as natural boundaries for process-level evaluation.

Concretely, we scan the generated reasoning text for occurrences of these high entropy markers and use their positions as candidate prefix boundaries. Let $\{t_i^a\}_{i=1}^{m_a}$ denote the resulting prefix positions for response y^a, ordered from left to right such that $1 \le t_1^a < \cdots < t_{m_a}^a \le T^a$, where i indexes the prefix, and m_a is the number of extracted prefixes. Each prefix $y_{\le t_i^a}^a = (y_1^a, \ldots, y_{t_i^a}^a)$ is treated as a partial reasoning state. To evaluate whether this partial state can support solving the problem, we perform prefix-conditioned answer verification. Specifically, we append a stop-thinking suffix, e.g., `</think>`, after the prefix to prompt the model to terminate reasoning and produce final answers. For each prefix, we sample multiple answers and verify whether they match the ground truth answer. The raw prefix-level reward r_i^a is defined as the answer accuracy among these sampled answers:

$$r_i^a = \frac{1}{n} \sum_{k=1}^{n} \mathbb{I}\left(o_{i,k}^a = o^*\right), \tag{1}$$

where $o_{i,k}^a$ is the k-th sampled answer from prefix $y_{\le t_i^a}^a$, o^* is the ground-truth answer, and n is the number of sampled answers.

Intuitively, r_i^a measures whether the current prefix has accumulated sufficient and useful reasoning evidence to recover the correct answer. However, raw prefix-level rewards are not directly comparable across different responses. A prefix near the end of a response is usually closer to the answer and may receive a higher score even if the additional reasoning is redundant. We therefore calibrate prefix-level rewards according to reasoning progress.

3.3 Adaptive Progress Bin Partition

We use the relative position of a prefix as a proxy for its reasoning stage. For prefix $y_{\le t_i^a}^a$, we define its prefix ratio as

$$\rho_i^a = \frac{t_i^a}{T^a}, \tag{2}$$

where t_i^a is the prefix position and T^a is the length of the full response y^a. To compare prefixes at similar progress, we maintain K progress bins over $[0, 1]$, with bin boundaries $\{b_k\}_{k=0}^K$. The boundaries are initialized uniformly as $b_k = k/K$. Each prefix $y_{\le t_i}$ is assigned to the bin whose interval contains its prefix ratio ρ_i, so prefixes with similar relative positions are compared within the same progress bin. Since observed prefix ratios may be unevenly distributed during training, we update the bin boundaries using prefix ratio statistics from the current batch:

$$b_k \leftarrow \lambda b_k + (1 - \lambda)\,\mathrm{Quantile}\left(\mathcal{H}_\rho, \frac{k}{K}\right), \tag{3}$$

where $k = 1, \ldots, K-1$, $\mathcal{H}_\rho$ denotes the set of prefix ratios computed from all prefixes extracted

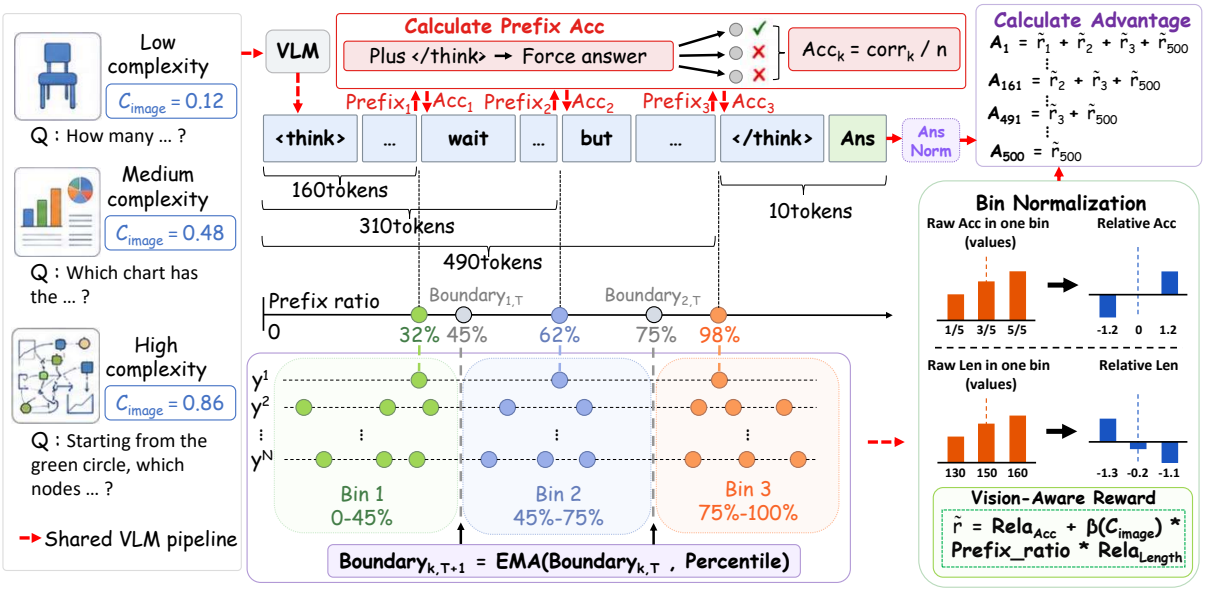

Figure 2: Overview of our method framework. We identify intermediate reasoning prefixes using high entropy markers, estimate prefix-level accuracy by forcing the model to answer from each prefix and verifying the sampled answers, and assign each prefix to an adaptive progress bin according to its prefix ratio. Within each bin, raw prefix accuracy and prefix length are normalized into relative signals. The final process reward combines bin-conditioned relative accuracy with a quality-gated, vision-aware length adjustment, and sparse rewards are accumulated into token-level advantages for policy optimization.

from all responses to all problems in the current batch, $\mathrm{Quantile}(\mathcal{H}_\rho, k/K)$ is the empirical k/K-quantile of this set, and λ is an exponential moving average coefficient. This Exponential Moving Average (EMA) update keeps bins aligned with the actual distribution of reasoning steps while avoiding abrupt boundary changes.

3.4 Binned Relative Accuracy Reward

We denote the collection of the raw rewards of all prefixes that belong to the same progress bin as:

$$\mathcal{R}_j = \{r_i^a \mid B(\rho_i^a) = j, 1 \le i \le m_a, 1 \le a \le N\}, \tag{4}$$

where $B(\cdot)$ denotes the bin assignment function, which maps a prefix ratio to the progress bin containing it. We then compute the mean μ_j and standard deviation σ_j of the rewards in $\mathcal{R}_j$. For all prefixes in the j-th bin, the binned relative reward is computed as

$$\hat{r}_i^a = \frac{r_i^a - \mu_j}{\sigma_j + \epsilon}, \tag{5}$$

where ϵ is a small constant for numerical stability.

This Bin Normalization changes the meaning of the reward. A prefix is no longer judged by whether its raw accuracy is high in absolute value, which would naturally favor later prefixes. Instead, it is judged by whether it performs better than other prefixes solving the same problem at a similar reasoning stage. Thus, progress binning helps identify genuinely useful intermediate steps rather than assigning positive credit mainly to prefixes that are simply closer to the final answer.

3.5 Binned Relative Length Reward

Bin-conditioned calibration improves the comparability of process rewards, but it does not explicitly encourage efficiency. We therefore introduce a length-aware adjustment that favors concise reasoning only when it is supported by reliable process quality.

Similarly, for each prefix $y^a_{\le t_i^a}$, we normalize its absolute length $\ell_i^a = t_i^a$ using comparable prefixes in the same progress bin. Let μ_j^ℓ and σ_j^ℓ denote the mean and standard deviation of prefix lengths in the j-th progress bin. We define

$$\hat{l}_i^a = \frac{\mu_j^\ell - \ell_i^a}{\sigma_j^\ell + \epsilon}. \tag{6}$$

Formally, a positive $\hat{l}_i^a$ indicates that the prefix exhibits superior brevity relative to its peers within the same bin, whereas a negative value reflects excessive length.

However, directly rewarding shorter prefixes may favor premature stopping rather than efficient reasoning. We therefore gate the length signal by the quality-conditioned reward:

$$g_i^a = \begin{cases} \max(\hat{l}_i^a, 0), & \hat{r}_i^a \geq 0, \\ \min(\hat{l}_i^a, 0), & \hat{r}_i^a < 0. \end{cases} \tag{7}$$

Thus, brevity is rewarded if a prefix surpasses the average performance baseline while remaining shorter than the mean length of its peer prefixes within the same bin. Conversely, excessive length is penalized when a prefix both underperforms the baseline and exceeds the average bin length. This safeguard ensures the model is not incentivized for superficial brevity at the expense of reliability, while simultaneously preventing it from expanding trailing, erroneous reasoning trajectories.

We further scale the length signal by the prefix ratio ρ_i^a. Early prefixes often correspond to visual grounding and exploratory reasoning, where additional tokens may be necessary. Late prefixes are more likely to contain redundant verification or delayed termination. Multiplying by $\rho_i^a = \frac{t_i^a}{T^a}$ therefore weakens length pressure early in the response and strengthens it near the end.

Finally, we modulate the strength of the length adjustment using a visual-complexity coefficient. Specifically, we use Gemini-3.1-Pro-Preview to assess the visual complexity of each example conditioned on both the image and the question, directly producing a score $c \in [0, 1]$. We define

$$\beta(c) = \beta_{\max} + (\beta_{\min} - \beta_{\max})c, \tag{8}$$

where $\beta_{\max} > \beta_{\min}$. Simple images receive stronger compression pressure, while complex images retain a larger reasoning budget.

The final calibrated process reward is

$$\tilde{r}_i^a = \text{clip}\left(\hat{r}_i^a + \beta(c)\rho_i^a g_i^a,\ -r_{\max}, r_{\max}\right). \tag{9}$$

3.6 Token-Level Advantage and Training

The calibrated process rewards are defined for intermediate prefixes, while policy optimization is performed at the token level. We therefore first place each reward back onto the response token sequence. For a prefix $y^a_{\leq t_i^a}$ ending at position t_i^a, its calibrated process reward $\tilde{r}_i^a$ is assigned to the boundary token $y^a_{t_i^a}$, i.e., the last token of the prefix that triggers the corresponding prefix verification. The final answer reward is first normalized within the GRPO group for the same problem, yielding $\tilde{r}^a_{T^a}$, and is then assigned to the last valid response token $y^a_{T^a}$.

Following DeepSeekMath (Shao et al., 2024), we convert these sparse token-level rewards into token-level advantages using a return-to-go style accumulation. For each token position u in response y^a, we compute its token-level advantage by accumulating all future process rewards that occur after this token, together with the normalized final answer reward. We denote the last process-reward position in response y^a as $t^a_{m_a}$, and define the index of the first prefix boundary not earlier than token position u as:

$$s_a(u) = \min\{i : t_i^a \geq u\}. \tag{10}$$

The advantage is defined as

$$A_u^a = \begin{cases} \displaystyle\sum_{i=s_a(u)}^{m_a} \tilde{r}_i^a + \tilde{r}^a_{T^a}, & u \leq t^a_{m_a}, \\ \tilde{r}^a_{T^a}, & u > t^a_{m_a}. \end{cases} \tag{11}$$

This accumulation assigns each token the sum of all future process and outcome rewards that occur after it. Intuitively, a token should receive credit for later verified reasoning states and the final answer that it helps produce, but not for rewards that have already occurred before the token is generated. Compared with assigning rewards only at the verified boundary tokens, return-to-go accumulation allows sparse process feedback to supervise all preceding tokens that contribute to useful intermediate reasoning states.

By combining bin-conditioned process rewards, quality-gated efficiency adjustment, and return-to-go advantage accumulation, our method encourages VLMs to preserve useful visual reasoning while reducing redundant reasoning.

4 Experiments

4.1 Setup

Training Setup. We train our model on ViRL39K from VL-Rethinker (Wang et al., 2025a). Before reinforcement learning, we clean the training set by removing duplicated problems and filtering examples with extreme difficulty. We directly perform reinforcement learning on Qwen2.5-VL series models without an additional supervised fine-tuning stage.

Evaluation Benchmarks. We evaluate the trained models on six widely used multimodal

Method	MathVision	MathVerse	MathVista	DynaMath	MM-Math	WeMath	Avg. Acc.	Avg. Len.
	Acc. / Len.	Acc. / Len.	Acc. / Len.	Acc. / Len.	Acc. / Len.	Acc. / Len.		
Closed-Source Model								
GPT-4o	29.2 / –	50.0 / –	63.8 / –	62.7 / –	31.8 / –	69.0 / –	51.1	–
Gemini2.5Pro	73.3 / –	60.3 / –	84.6 / –	68.8 / –	55.2 / –	78.0 / –	70.0	–
QwenVLMax	26.4 / –	47.3 / –	73.2 / –	62.3 / –	43.4 / –	45.6 / –	49.7	–
Base Qwen2.5-VL-3B: General Reasoning Methods								
Base	21.9 / 469.5	37.2 / 352.6	62.5 / 200.9	48.2 / 270.7	28.4 / 623.9	39.4 / 270.1	39.6	364.6
LMM-R1	25.1 / 470.8	41.2 / 333.7	64.2 / 180.5	53.4 / 252.6	33.6 / 701.6	49.9 / **260.2**	44.6	366.6
VLAA	24.1 / 581.7	42.2 / 424.5	64.0 / 167.3	52.1 / 227.7	30.4 / 668.4	47.4 / 468.5	43.4	423.0
Ours-3B	**27.0 / 366.2**	**43.5 / 304.6**	**65.4 / 158.4**	**54.5 / 221.4**	**35.0 / 459.1**	**56.6** / 264.8	**47.0**	**295.7**
Base Qwen2.5-VL-7B: General Reasoning Methods								
Base	26.3 / 525.1	49.0 / 307.7	68.4 / 203.4	52.8 / 215.9	34.9 / 654.8	55.4 / 311.0	47.8	369.7
MM-R1	31.0 / 1245.5	51.2 / 783.9	71.3 / 649.4	57.1 / 562.2	34.4 / 1303.9	60.2 / 728.8	50.9	879.0
VisionR1	– / –	**52.4** / 979.9	73.4 / 507.3	59.0 / 725.1	36.1 / 1347.8	– / –	–	–
R1-OneVision	30.7 / 464.0	46.2 / 411.8	65.9 / 256.6	55.2 / 369.2	30.1 / 611.9	52.8 / 308.6	46.8	403.0
OpenVLThinker	30.6 / 439.0	49.7 / 362.9	72.5 / 205.9	58.9 / 367.5	37.8 / 546.6	60.0 / 341.2	51.6	377.2
Base Qwen2.5-VL-7B: Adaptive and Efficient Reasoning Methods								
VSRM-7B	27.7 / 405.3	49.7 / 310.7	69.0 / 194.6	54.9 / 348.6	35.9 / 494.8	55.6 / 296.9	48.8	341.1
DAST-7B	27.0 / 481.1	48.5 / 280.1	72.1 / 158.0	57.8 / 328.1	37.6 / 644.8	61.5 / 198.9	50.8	348.5
FAST-7B	31.0 / 448.1	50.8 / 309.1	72.8 / **134.9**	58.6 / **153.9**	40.1 / 570.6	62.1 / 248.2	52.6	310.8
Ours-7B	**32.2 / 266.7**	51.2 / **204.7**	**73.9** / 147.7	**59.9** / 203.4	**40.2 / 357.2**	**64.1 / 173.2**	**53.6**	**225.5**

Table 1: Comparison with existing VLM reasoning, adaptive and efficient reasoning methods on multimodal mathematical reasoning benchmarks. Each benchmark cell reports Acc. / Len. Avg. Acc. and Avg. Len. are computed over all six benchmarks when all results are available. Higher Acc. and lower Len. are better. Response lengths are not reported for closed-source models. The best results are shown in bold, and the second-best results are underlined.

mathematical reasoning benchmarks: MathVision, MathVerse, MathVista, MM-Math, DynaMath, and WeMath. These benchmarks cover a broad range of visual reasoning scenarios, including diagram understanding, geometry, chart reasoning, visual mathematical problem solving, and dynamically composed multimodal questions.

Baselines. We compare our method with a range of general reasoning models, including LMM-R1 (Peng et al., 2025), VLAA-Thinker-3B (Chen et al., 2025), MM-R1 (Liang et al., 2025), VisionR1 (Huang et al., 2026), R1-OneVision (Yang et al., 2025), and OpenVLThinker (Deng et al., 2025), as well as adaptive and efficient reasoning methods such as VSRM (Yue et al., 2025), DAST (Shen et al., 2026), and FAST (Xiao and Gan, 2025). For all models, we report answer accuracy and the average generated response length.

4.2 Main Results

As shown in Table 1, our method consistently improves the accuracy-efficiency trade-off across both model scales. Ours-3B achieves the best accuracy on all six benchmarks among Qwen2.5-VL-3B-based methods. Compared with the base model, it improves the average accuracy by 7.4 points while reducing the average response length by 68.9 tokens, corresponding to an 18.9% reduction. This shows that our method does not merely shorten responses, but also improves reasoning correctness. The improvement is more pronounced in the 7B setting. Compared with the base model, Ours 7B model improves the average accuracy by 5.8 points and reduces the average response length by 144.2 tokens, corresponding to a 39.0% reduction. Compared with the strongest 7B general reasoning baseline with complete results, OpenVLThinker, Ours-7B improves the average accuracy by 2.0 points and reduces the average response length by 151.7 tokens, corresponding to a 40.2% reduction. Compared with the strongest adaptive and efficient reasoning baseline, FAST-7B, Ours-7B improves the average accuracy by 1.0 point and reduces the average response length by 85.3 tokens, corresponding to a 27.4% reduction. By calibrating process rewards according to reasoning progress and applying a quality-gated length adjustment, our method effectively balances correctness and efficiency.

We also compare with closed-source models to contextualize the remaining gap. Ours-7B achieves an average accuracy of 53.6%, outperforming GPT-4o and QwenVLMax in average accuracy,

Model	MathVision	MathVerse	MathVista	Acc.	Len.
Qwen-2.5-VL-7B	26.3	49.0	68.4	47.9	377.5
Naive GRPO	29.0	50.8	72.4	50.7	353.3
w/o BinNorm	28.6	49.5	69.3	49.1	327.0
w/o Len Reward	32.1	**53.0**	73.4	**52.8**	413.1
w/o prefix ratio					
Fix 0.1	31.1	51.3	72.3	51.5	398.7
Fix 0.5	30.5	50.4	70.3	50.4	356.8
Fix 1.0	28.7	48.2	68.5	48.5	267.4
Ours	**32.2**	51.2	**73.9**	52.4	**206.4**

Table 2: Ablation results on multimodal math benchmarks. Acc. denotes the average accuracy over MathVision, MathVerse, and MathVista. Len. denotes the average generated response length across these three benchmarks.

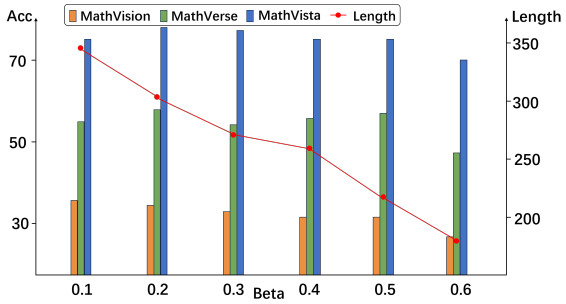

Figure 3: Effect of the length-reward coefficient β. Larger β imposes stronger efficiency pressure. Moderate values achieve a better accuracy-efficiency trade-off, while too small β fails to reduce redundant reasoning and too large β over-compresses useful reasoning.

Mapping	MathVision	MathVerse	MathVista	Acc.	Len.
Concave	**32.8**	50.4	73.3	52.2	242.1
Convex	32.4	**51.9**	72.6	52.3	223.7
Linear	32.2	51.2	**73.9**	**52.4**	**206.4**

Table 3: Effect of different mappings from visual complexity to the length-reward coefficient β.

while still lagging behind Gemini2.5Pro. **Notably, our model is trained purely with reinforcement learning, without distilling reasoning traces or answers from closed-source models.** These results suggest that bin-conditioned process rewards can narrow the gap to stronger VLMs while maintaining a substantially more efficient reasoning process.

4.3 Ablation Study

The ablations are shown in Table 2. Removing bin normalization, i.e., normalizing all prefix-level rewards together without considering prefix progress, leads to clear performance degradation across all benchmarks. This confirms that raw prefix-level rewards are not directly comparable across reasoning stages, and that comparing prefixes within similar progress bins is important for identifying genuinely useful intermediate steps. We next remove the length reward while keeping progress-binned reward normalization. This variant achieves comparable accuracy to our full method, but its average response length increases substantially from 206.4 to 413.1 tokens. This result shows that our length-aware reward effectively removes redundant reasoning while preserving the useful reasoning behavior. Finally, we study the role of the prefix ratio scaling factor in the length reward. We replace the adaptive prefix ratio with fixed constants 0.1, 0.5, and 1.0. When the factor is fixed to 0.1, the length pressure remains weak throughout the response, resulting in insufficient suppression of late-stage redundancy and much longer outputs. When it is fixed to 1.0, the length pressure becomes too strong even during early visual grounding and exploration, leading to accuracy drops on all three benchmarks.

4.4 Analysis

Effect of Complexity-Aware Length Control. Since our main method uses a visual-complexity-dependent coefficient $\beta(c)$, we first isolate the effect of compression strength by replacing it with fixed constants. Figure 3 shows that larger β generally reduces response length, but accuracy is not monotonic: weak length pressure fails to suppress redundant reasoning, whereas overly strong pressure over-compresses useful reasoning. We therefore use an adaptive range $[0.2, 0.5]$. We further compare concave, convex, and linear mappings from visual complexity to $\beta(c)$ in Table 3. Since all mappings perform similarly, we adopt the linear mapping for its simplicity and stable trade-off, applying stronger compression to simple examples while preserving more reasoning budget for complex ones.

Training Dynamics of Response Length. Figure 4 shows that the average generated length steadily decreases during training, indicating that the model learns to suppress redundant reasoning while keeping necessary reasoning. The examples further compare Global Normalization and Bin Normalization. Global normalization compares steps across all reasoning stages, which tends to assign lower rewards to early shorter prefixes because their raw accuracy is naturally lower. In contrast, Bin Normalization compares each step with

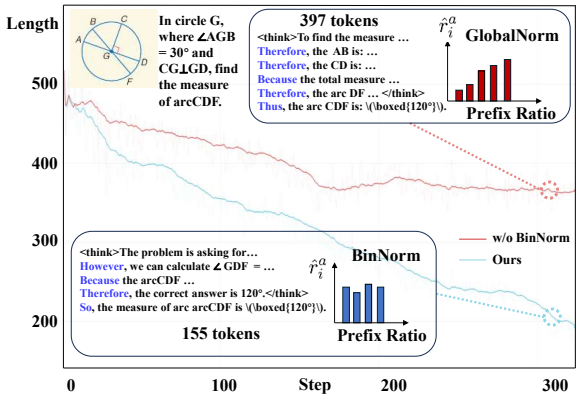

Figure 4: Training length dynamics and reward calibration. The average response length decreases during training, while Bin Normalization better reflects local step quality than Global Normalization.

Strategy	MathVision	MathVerse	MathVista	Acc.	Len.
Freeze	**32.8**	50.6	72.4	51.9	286.7
Divide Equally	30.3	50.5	71.3	50.7	283.2
EMA	32.2	**51.2**	**73.9**	**52.4**	**206.4**

Table 4: Effect of different bin boundary update strategies. Freeze keeps uniform bin boundaries; Divide Equally chooses boundaries so that each bin contains a similar number of prefixes; EMA denotes our moving-average percentile update.

prefixes at similar reasoning progress, making the reward better reflect local step quality. This helps preserve useful early reasoning while discouraging unnecessary late-stage verification.

Effect of Bin Boundary Updating. We compare three bin boundary strategies: our proposed EMA, which updates each boundary with 0.9 times the old boundary and 0.1 times the empirical percentile; Freeze, which keeps eight uniformly divided bins; and Divide Equally, which adjusts boundaries so that each bin contains a similar number of detected prefixes. As shown in Table 4, EMA achieves the best overall trade-off, obtaining the highest average accuracy and the shortest average response length among the three strategies. This suggests that EMA better balances statistical stability and stage consistency: it uses empirical percentiles as adaptive targets to avoid having most detected prefixes concentrated in a few bins, but updates the boundaries gradually rather than enforcing equal-count bins. In this way, EMA provides more stable normalization while preserving the coarse semantics of reasoning progress.

Number	MathVision	MathVerse	MathVista	Acc.	Len.
2 Bins	28.1	49.8	69.8	49.2	336.9
4 Bins	29.3	50.5	70.2	50.0	320.1
6 Bins	31.2	**51.5**	72.4	51.7	230.6
8 Bins	**32.2**	51.2	**73.9**	**52.4**	**206.4**
10 Bins	30.9	50.8	73.0	51.6	232.4

Table 5: Effect of the number of initial progress bins.

Effect of the Number of Initial Progress Bins. We further study the effect of the number of initial progress bins. As shown in Table 5, using too few bins leads to worse performance and longer responses. With only 2 or 4 bins, prefixes from substantially different reasoning stages are grouped together, making reward normalization less precise.

Increasing the number of bins improves both accuracy and efficiency. Moving from 4 to 6 bins substantially reduces the average response length, and 8 bins achieves the best average accuracy and the shortest responses. This suggests that a finer partition of reasoning progress provides more comparable prefix groups and more reliable relative reward estimation.

However, using too many bins does not bring further gains. With 10 bins, the accuracy drops compared with 8 bins, likely because each bin contains fewer comparable prefixes and the reward statistics become less stable. We therefore use 8 bins as the default setting.

5 Conclusion

We present a bin-conditioned process reward framework for efficient multimodal reasoning. Instead of relying on coarse outcome-level length penalties or directly optimizing uncalibrated step rewards, our method compares intermediate prefixes at similar reasoning progress and converts raw prefix-level feedback into stage-aware relative rewards. We further introduce a quality-gated length adjustment, modulated by prefix ratio and visual complexity, to reduce redundant reasoning without encouraging premature shortcuts. Experiments on six multimodal mathematical reasoning benchmarks show that our method improves the accuracy-efficiency trade-off, achieving higher accuracy with substantially shorter responses than strong VLM reasoning and adaptive thinking baselines. These results suggest that calibrating raw prefix-level rewards by reasoning progress is an effective way to preserve useful visual reasoning while mitigating overthinking.

Limitations

Despite the effectiveness of our method, several limitations remain. First, due to computational constraints, our experiments are conducted mainly on relatively small VLMs, i.e., Qwen2.5-VL-3B and Qwen2.5-VL-7B. With more computational resources, we plan to scale our method to larger VLM backbones in future work. Second, our vision-aware length adjustment uses an external closed-source model to estimate image complexity. Future work may replace it with a lightweight open-source model to reduce the cost of complexity estimation.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. 2025. Sft or rl? an early investigation into training r1-like reasoning large vision-language models. *Preprint*, arXiv:2504.11468.

Xinquan Chen, Chongying Yue, Bangwei Liu, Xuhong Wang, Yingchun Wang, and Chaochao Lu. 2026. Vr-prm: Process reward modeling via visual reasoning. *Preprint*, arXiv:2508.03556.

Jhouben Cuesta-Ramirez, Samuel Beaussant, and Mehdi Mounsif. 2025. Large reasoning models are not thinking straight: on the unreliability of thinking trajectories. *Preprint*, arXiv:2507.00711.

Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. 2025. Openvlthinker: Complex vision-language reasoning via iterative sft-rl cycles. *Preprint*, arXiv:2503.17352.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. 2025. Token-budget-aware llm reasoning. *Preprint*, arXiv:2412.18547.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. 2026. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *Preprint*, arXiv:2503.06749.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Qian Liang, Yujia Wu, Kuncheng Li, Jiwei Wei, Shiyuan He, Jinyu Guo, and Ning Xie. 2025. Mm-r1: Unleashing the power of unified multimodal large language models for personalized image generation. *Preprint*, arXiv:2508.11433.

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.

Zehui Ling, Deshu Chen, Hongwei Zhang, Yifeng Jiao, Xin Guo, and Yuan Cheng. 2025. Fast on the easy, deep on the hard: Efficient reasoning via powered length penalty. *Preprint*, arXiv:2506.10446.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2024. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *Preprint*, arXiv:2310.02255.

Yingzhe Peng, Gongrui Zhang, Miaosen Zhang, Zhiyuan You, Jie Liu, Qipeng Zhu, Kai Yang, Xingzhong Xu, Xin Geng, and Xu Yang. 2025. Lmm-r1: Empowering 3b lmms with strong reasoning abilities through two-stage rule-based rl. *Preprint*, arXiv:2503.07536.

Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Zhuoma GongQue, Shanglin Lei, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Yifan Zhang, Xiao Zong, Yida Xu, Muxi Diao, Zhimin Bao, Chen Li, and Honggang Zhang. 2024. We-math: Does your large multimodal model achieve human-like mathematical reasoning? *Preprint*, arXiv:2407.01284.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2024. Rewarding progress: Scaling automated process verifiers for llm reasoning. *Preprint*, arXiv:2410.08146.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. 2026. Dast: Difficulty-adaptive slow-thinking for large reasoning models. *Preprint*, arXiv:2503.04472.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Preprint*, arXiv:2503.16419.

Kai Sun, Yushi Bai, Ji Qi, Lei Hou, and Juanzi Li. 2024. Mm-math: Advancing multimodal math evaluation with process evaluation and fine-grained classification. *Preprint*, arXiv:2404.05091.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback. *Preprint*, arXiv:2211.14275.

Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhu Chen. 2025a. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *Preprint*, arXiv:2504.08837.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. 2024a. Measuring multimodal mathematical reasoning with math-vision dataset. *Preprint*, arXiv:2402.14804.

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

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025b. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *Preprint*, arXiv:2506.01939.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, Lewei Lu, Haodong Duan, Yu Qiao, Jifeng Dai, and Wenhai Wang. 2025c. Visualprm: An effective process reward model for multimodal reasoning. *Preprint*, arXiv:2503.10291.

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.

Hao Wen, Xinrui Wu, Yi Sun, Feifei Zhang, Liye Chen, Jie Wang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. 2025. Budgetthinker: Empowering budget-aware llm reasoning with control tokens. *Preprint*, arXiv:2508.17196.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. 2026. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. *Preprint*, arXiv:2505.19255.

Violet Xiang, Chase Blagden, Rafael Rafailov, Nathan Lile, Sang Truong, Chelsea Finn, and Nick Haber. 2025. Just enough thinking: Efficient reasoning with adaptive length penalties reinforcement learning. *Preprint*, arXiv:2506.05256.

Wenyi Xiao and Leilei Gan. 2025. Fast-slow thinking grpo for large vision-language model reasoning. *Preprint*, arXiv:2504.18458.

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, Bo Zhang, and Wei Chen. 2025. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *Preprint*, arXiv:2503.10615.

Chuhuai Yue, Chengqi Dong, Yinan Gao, Hang He, Jiajun Chai, Guojun Yin, and Wei Lin. 2025. Promoting efficient reasoning with verifiable stepwise reward. *Preprint*, arXiv:2508.10293.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. 2024a. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? *Preprint*, arXiv:2403.14624.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. 2024b. Multimodal chain-of-thought reasoning in language models. *Preprint*, arXiv:2302.00923.

Haojie Zheng, Tianyang Xu, Hanchi Sun, Shu Pu, Ruoxi Chen, and Lichao Sun. 2024. Thinking before looking: Improving multimodal llm reasoning via mitigating visual hallucination. *Preprint*, arXiv:2411.12591.

Shu Zhou, Rui Ling, Junan Chen, Xin Wang, Tao Fan, and Hao Wang. 2026. When more thinking hurts: Overthinking in llm test-time compute scaling. *Preprint*, arXiv:2604.10739.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. 2025. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. *Preprint*, arXiv:2411.00836.

A Implementation Details

We implement our method on the Qwen2.5-VL series, including Qwen2.5-VL-3B and Qwen2.5-VL-7B. Unless otherwise specified, both model sizes use the same training and reward-construction hyperparameters. We train the policy with a clipped policy-gradient objective. For each training problem, we sample $N = 5$ full reasoning responses from the current policy. The rollout temperature is set to 0.6, top-p to 0.95, and the maximum response length to 7424 tokens. The maximum prompt length is 32768, and overlong prompts are filtered with left truncation.

For prefix verification, each detected prefix is appended with `</think>` to force answer generation. We sample $n = 5$ answer completions for each prefix and use the fraction of verified correct answers as the raw prefix accuracy. Prefix-verification completions use the same temperature and top-p as the main rollout, with a maximum generation length of 768 tokens.

We use $K = 8$ progress bins initialized uniformly over $[0, 1]$. During training, bin boundaries are updated with an exponential moving average toward empirical prefix ratio quantiles from the current batch, with EMA coefficient $\lambda = 0.9$. The prefix ratio history buffer contains up to 4096 observed prefix ratios. We set the numerical stability constant to $\epsilon = 10^{-6}$, clip the calibrated process reward to $[-3, 3]$, and clip the normalized length signal to $[-1, 1]$. For the vision-aware length coefficient, we use $\beta_{\min} = 0.2$ and $\beta_{\max} = 0.5$.

For optimization, we use a training batch size of 256 prompts, with 5 responses sampled per prompt. The policy learning rate is 1×10^{-6}, weight decay is 0.01, and the policy is updated for one epoch per batch. We use a mini-batch size of 128 and a per-GPU micro-batch size of 8. The clipping range is 0.2, the loss is aggregated by token mean, and the entropy coefficient is set to 0. We include a reference-model KL loss with coefficient 0.01 and use the low-variance KL form. Training runs for 3 epochs on 8 GPUs with FSDP, parameter and optimizer offloading, gradient checkpointing, and vLLM rollout generation.

B Visual complexity assessment prompt

We compute visual-complexity scores offline using a strong multimodal model. The scoring prompt, shown in Figure 5, explicitly separates figure-reading burden from mathematical difficulty, and requires a scalar output in $[0, 1]$.

C Case Study

Figures 6 and 7 provide qualitative examples of the generated reasoning traces. In Figure 6, all models answer the hexagon-folding problem correctly. However, Base-7B and VisionR1 produce longer responses with redundant geometric descriptions, using 267 and 312 tokens, respectively. In contrast, Ours-7B directly captures the key symmetry relation and reaches the same correct answer with only 100 tokens. This shows that our method can reduce unnecessary reasoning while preserving the essential visual evidence.

Figure 7 further shows that shorter reasoning alone does not guarantee correctness. Base-7B generates a long response with both redundant verification and incorrect area estimates, leading to a wrong answer. FAST-7B shortens the response compared with the base model, but still introduces an incorrect area assignment and predicts the wrong segment. By contrast, Ours-7B keeps the reasoning concise, focuses on the correct area relationship induced by the diagonal partition, and produces the correct answer with fewer tokens. These examples suggest that our method does not simply truncate reasoning, but encourages the model to retain useful visual reasoning while suppressing redundant or misleading intermediate steps.

D Potential Risks

This work aims to improve the efficiency of multimodal reasoning by reducing redundant reasoning. A potential risk is that excessive compression may remove necessary intermediate reasoning, especially for visually complex examples. To mitigate this, our method gates the length adjustment by process quality and modulates its strength using visual complexity. Another risk is that more concise reasoning may make model errors harder to diagnose. Therefore, the method should be used with caution in high-stakes applications, and final answers should still be verified by external tools or human experts when reliability is critical.

E Artifacts

We use publicly available pretrained VLMs, training data, evaluation benchmarks, and evaluation tools. Specifically, our base models are Qwen2.5-VL-3B and Qwen2.5-VL-7B. The training data is derived from ViRL39K from VL-Rethinker.

The prompt for evaluating the visual complexity of mathematical problem figures.

You are an assistant for evaluating the visual complexity of mathematical problem figures. You will be given a math problem statement and its accompanying figure.

Your task is to evaluate only the overall complexity of the figure, considering how much visual information must be read and understood.
You may use the problem statement to judge the figure-reading requirement, but you must not solve the problem, judge correctness, infer the answer, or output any answer choice.
The complexity score should consider:
1. Visual information density
  - Number of visual elements, such as points, lines, angles, curves, regions, axes, grids, labels, arrows, colors, legends, or annotations.
  - Whether the figure has multiple subfigures, coordinate systems, objects, views, or a complex layout.
  - Whether there is overlap, occlusion, dense labeling, or visually confusing structure.

2. Figure-reading cost
  - Whether the solver must read precise values, labels, proportions, lengths, angles, areas, coordinates, or trends.
  - Whether the solver must match multiple visual elements or understand nontrivial geometric, spatial, or ordering relationships.
  - Whether careful inspection of the figure is necessary to understand the problem.

3. Text-figure coupling
  - Whether the problem statement can be understood without the figure.
  - Whether variables, objects, or conditions in the text depend heavily on the figure.
  - Whether the solver must repeatedly cross-reference the text and figure.

Scoring rules:
- The score must be in [0, 1] and rounded to two decimal places.
- 0 means an extremely simple or nearly unnecessary figure.
- 0.10–0.30 means simple: few elements and light figure-reading.
- 0.30–0.50 means moderately simple: some labels or relationships need to be read.
- 0.50–0.70 means moderately complex: many elements or substantial text-figure correspondence.
- 0.70–0.90 means complex: dense information, multiple relationships, or high risk of misreading.
- 0.90–1.00 means extremely complex: very dense, highly structured, and requiring extensive fine-grained reading.

Important constraints:
- Evaluate figure complexity, not mathematical solution difficulty.
- A hard problem with a simple figure should receive a low score.
- A visually complex figure should receive a higher score even if the solution is simple.
- Do not output explanations, reasoning steps, answers, or extra text.

The output must be exactly one line of JSON with only one key. Example:

{"complexity":0.37}

Figure 5: Prompt for estimating figure-reading complexity in multimodal mathematical problems. The evaluator is instructed to score only the visual complexity of the accompanying figure, conditioned on the problem statement, and to output a scalar $c \in [0, 1]$. The score is used offline to modulate the strength of the vision-aware length reward.

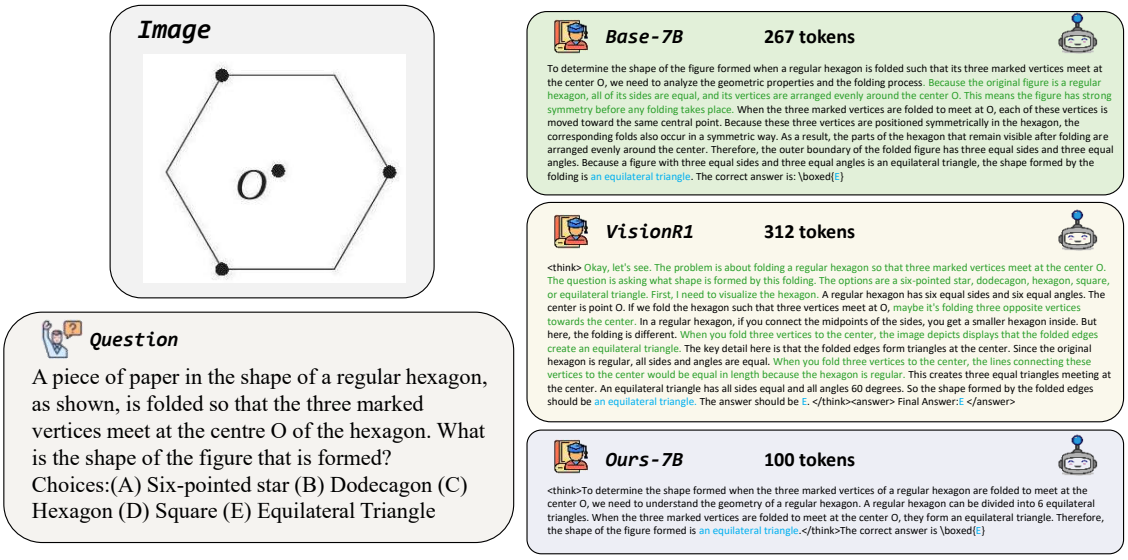

Figure 6: Qualitative comparison on a hexagon-folding problem. Green text marks redundant reasoning, and blue text marks correct key reasoning. Compared with Base-7B and VisionR1, Ours-7B removes unnecessary geometric descriptions and reaches the correct answer with fewer tokens.

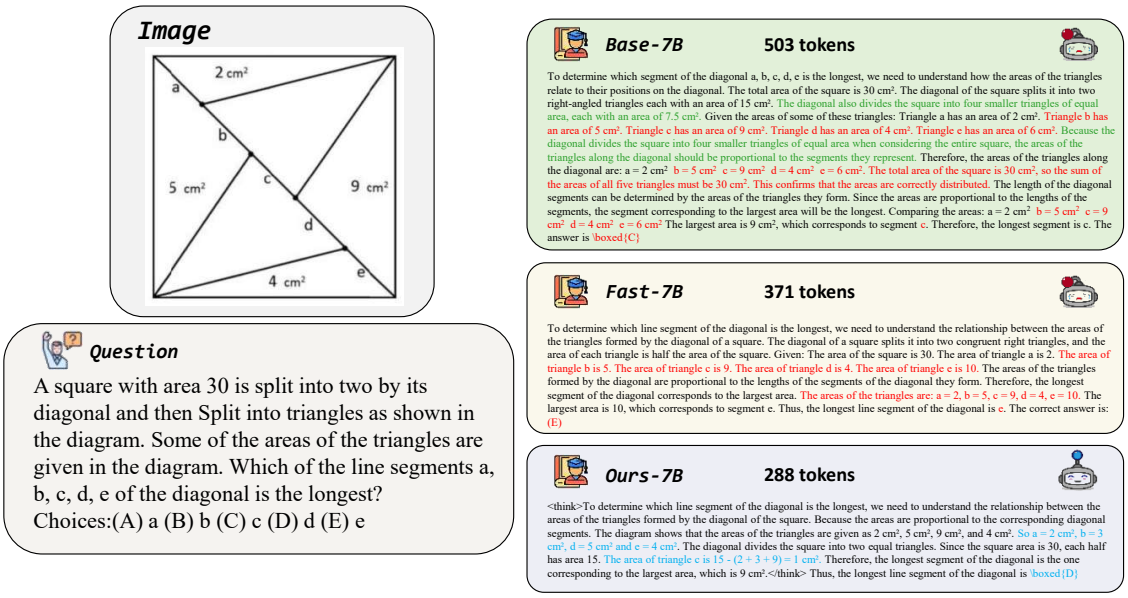

Figure 7: Qualitative comparison on a diagonal-segment problem. Red text marks incorrect reasoning, green text marks redundant reasoning, and blue text marks correct key reasoning. Ours-7B avoids misleading area assignments and produces the correct answer with a shorter response than Base-7B and FAST-7B.

We evaluate on MathVision, MathVerse, MathVista, DynaMath, MM-Math, and WeMath using VLMEvalKit. We cite the original papers or repositories for all datasets, models, and evaluation tools. Our use of these artifacts is limited to research purposes and follows their intended use as publicly released research assets. We cite all used artifacts and discuss their intended research use.

F Data Privacy

The datasets used in this work consist of multimodal mathematical reasoning problems and are not designed to contain personal information. We do not collect new human-subject data or scrape user-generated content. To check potential privacy and content-safety issues, we manually inspect a subset of the training and evaluation examples and verify that the data mainly consists of math questions, diagrams, charts, and answer annotations. We do not observe personally identifying information or offensive content in the inspected examples. Since we do not create or redistribute a new dataset, no additional anonymization procedure is applied. We also do not release any data containing personal identifiers.

G Dataset Documentation and Statistics

We provide the sources, domains, and split statistics of the training and benchmark datasets used in our experiments in Table 6 to support reproducibility and clarify the scope of our evaluation. The datasets used in this work are primarily English multimodal mathematical reasoning datasets. They mainly cover visual mathematical problem solving, diagram understanding, geometry reasoning, chart reasoning, and knowledge-aware visual reasoning. Since these datasets consist of math problems rather than user-authored personal text, demographic attributes of speakers or annotators are not applicable to our analysis.

Dataset	Domain	Split Size
ViRL39K	Multimodal RL training data	38,870
MathVision	Visual math reasoning	3,040
MathVerse	Diagram-based visual math reasoning	3,940
MathVista	Visual mathematical problem solving	1,000
DynaMath	Dynamic visual math reasoning	5,010
MM-Math	Multimodal middle-school math	5,901
WeMath	Knowledge-aware visual math reasoning	1,740

Table 6: Statistics of the training and evaluation datasets.

H Reporting Variance and Experimental Robustness

We report results from a single evaluation run due to the high cost of random seeds across six multimodal reasoning benchmarks. To improve reliability, we evaluate the trained model on multiple benchmarks and report both average accuracy and response length. We also include ablations and

Package / Toolkit	Version
Transformers	4.51.0
Accelerate	1.10.1
vLLM	0.8.3
TRL	0.24.2
FlashAttention	2.7.4
SwanLab	0.7.8

Table 7: Main packages used in our experiments.

sensitivity analyses over β, the number of progress bins, and bin-boundary update strategies.

I Packages

Our experiments are implemented with commonly used open-source libraries for multimodal model training and inference. Specifically, we use Transformers, Accelerate, vLLM, TRL, FlashAttention, and SwanLab. These packages are used for model loading, distributed training, rollout generation, efficient attention computation, and experiment tracking. The main software packages and toolkits are summarized in Table 7.

J Human Annotation and Human Subjects

Not applicable. We do not recruit human annotators or conduct human-subject experiments.

K Use of AI Assistants

We used AI writing assistants to help improve language clarity and polish paper drafts. The research ideas, experiments, analyses, and final claims were designed and verified by the authors. The authors take full responsibility for all content in the paper.