DEEPER IS NOT ALWAYS WISER: EVIDENTIAL GRPO FOR OVERTHINKING IN LARGE VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing approaches to overthinking in large vision-language models (LVLMs) mainly control reasoning length at the token-sequence level, with limited attention to how entropy changes across LLM depth. We observe that entropy can reach a trough at intermediate layers but rebound as LLM depth increases, leading to unnecessary additional reasoning. Motivated by this observation, we propose Evidential GRPO (**EviGRPO**), a new GRPO variant that regulates reasoning through a *Trigger–Verify* process. It consists of an *entropy trigger*, which monitors entropy variations across LLM layers to identify the layer where entropy rebounds from its trough, and an *evidence verifier*, which detects rebound-induced token preference shifts and learns to assess whether the shift is sufficiently supported by evidence. For an evidence-unsupported preference shift, the final-layer token is replaced with the token predicted at the entropy-trough layer, while evidence-supported tokens are retained. Next, we formulate the **EviGRPO** reward based on the changes in reasoning length and final-answer correctness under trough-token substitution, while deriving evidential supervision from token-shift outcomes for verifier optimization. In this way, **EviGRPO** rectifies abnormal final-layer token shifts, thereby mitigating their propagation into redundant reasoning. Experiments across seven reasoning benchmarks demonstrate that **EviGRPO** consistently improves reasoning accuracy across 3B, 7B, and 32B models, achieving accuracy gains of 1.2–5.7 percentage points over standard GRPO while reducing reasoning length by up to 256.4 tokens, effectively balancing reasoning accuracy and efficiency.

1 INTRODUCTION

Large Vision-Language Models (LVLMs) (Bai et al., 2025a; Yang et al., 2025; Huang et al., 2026) often exhibit overthinking when solving complex multimodal tasks, producing excessively long reasoning sequences (Xiao & Gan, 2026; Diao et al., 2026). Such redundant reasoning substantially increases inference latency (Tian et al., 2026). Moreover, longer reasoning does not consistently improve performance and may reduce answer accuracy (Jiang et al., 2025).

Recent studies for alleviating overthinking in LVLMs can be broadly grouped into two categories. First, reasoning compression methods (Han et al., 2025; Luo et al., 2026a; Shen et al., 2025) reduce redundant computation by shortening long reasoning traces through predefined token budgets or length-aware objectives. While effective in improving efficiency, overly aggressive compression may also remove reasoning necessary for solving challenging problems (Shen et al., 2025; Xiang et al., 2025; Aggarwal et al., 2026). Second, adaptive reasoning methods (Xiao & Gan, 2026; Liu et al., 2026a; Ma et al., 2025) dynamically adjust reasoning length according to problem difficulty. However, deeper reasoning on challenging problems inevitably increases the risk of overthinking (Tian et al., 2026). These current methods largely regulate reasoning at the sequence level, overlooking the evolution of internal hidden states across LLM layers (Belrose et al., 2023; Schuster et al., 2022).

In this study, we analyze the evolution of token predictions across LLM layers on Qwen2.5-VL-7B and obtain two observations. **Observation (i):** *Token entropy does not always decrease monotonically with LLM depth.* As shown in Fig. 1a, the entropy of a reasoning token can reach a trough at an intermediate layer and subsequently rebound toward the final layer. Such troughs occur frequently

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

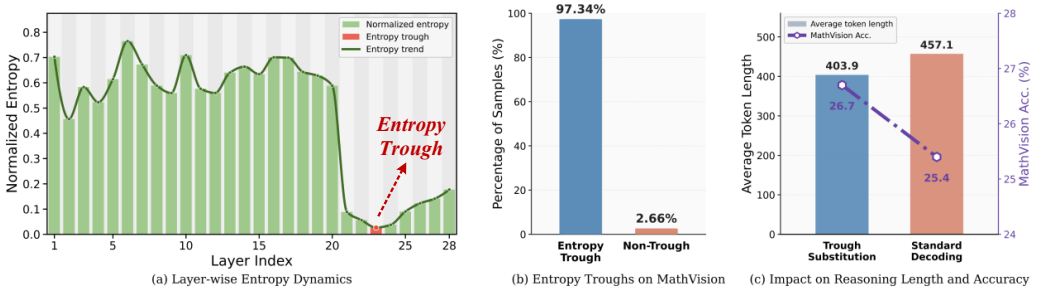


Figure 1: Analysis of entropy troughs. (a) Layer-wise token entropy dynamics, where entropy reaches a trough at an intermediate layer and rebounds toward the final layer. (b) Distribution of entropy-trough and non-trough samples. (c) Impact of replacing final-layer tokens with entropy-trough tokens on reasoning length and answer accuracy.

during reasoning (Fig. 1b). Replacing the final-layer token with the corresponding entropy-trough token reduces the average reasoning length from 457.1 to 403.9 tokens while improving accuracy from 25.4% to 26.7% (Fig. 1c). These results suggest that deeper computation after an entropy trough may induce unnecessary token shifts, prolonging reasoning and degrading answer accuracy. Therefore, entropy rebound provides a useful signal for identifying reasoning positions where deeper processing may lead to redundant reasoning and erroneous predictions. **Observation (ii):** *Entropy-trough tokens are not invariably reliable than final-layer tokens.* As illustrated in Fig. 2, replacing the final-layer token with the entropy-trough token can compromise answer correctness, indicating that some final-layer token shifts correspond to beneficial corrections. Therefore, entropy rebound can reveal potential token shifts but cannot distinguish harmful changes from beneficial corrections. This leads to a central question: *when should a final-layer token shift be retained, and when should it be replaced by the entropy-trough token?*

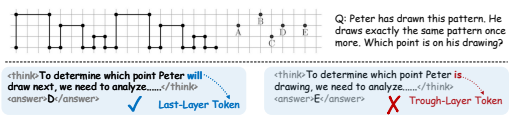


Figure 2: Comparison between final-layer and entropy-trough token predictions. Replacing the final-layer token with the trough-layer token may lead to incorrect answer.

Motivated by these observations, we propose **EviGRPO**, a GRPO (Guo et al., 2025a) variant that regulates reasoning through a *Trigger–Verify* process. The *entropy trigger* tracks token entropy across LLM layers and identifies reasoning positions where entropy rebounds from a near-final-layer trough. For the detected token shift from the entropy-trough layer to the final layer, the *evidence verifier* evaluates the evidence support for such token shift and determines whether the shift should be retained or replaced by the entropy-trough token. We further formulate the **EviGRPO** reward based on the changes in reasoning length and final-answer correctness under trough-token substitution for LVLM policy optimization, while using the resulting outcome differences to optimize the *evidence verifier*. Our main contributions are summarized as follows:

- We reveal an overlooked phenomenon in LVLM reasoning: token entropy can reach an intermediate-layer trough and rebound toward the final layer, and such rebound can affect both reasoning efficiency and answer correctness.
- We propose **EviGRPO**, which combines an *entropy trigger* for detecting trough-to-final token shifts with an *evidence verifier* for assessing the reliability of such shifts. Together with token shift rewards and evidential supervision, **EviGRPO** reduces redundant reasoning while improving answer accuracy.
- Extensive experiments across seven multimodal reasoning benchmarks demonstrate that **EviGRPO** effectively reduces redundant reasoning, achieving 1.2%–5.7% accuracy gains over standard GRPO while reducing reasoning length by up to 256.4 tokens.

2 RELATED WORK

Large Vision-Language Models. Reinforcement learning (Schulman et al., 2017) has substantially enhanced the reasoning capabilities of LLMs (Guo et al., 2025b; Team et al., 2025; Shao et al., 2024)

and has been increasingly extended to multimodal models (Liu et al., 2025b; Zhang et al., 2025; Yang et al., 2025; Huang et al., 2026; Peng et al., 2025; Wang et al., 2026a; Meng et al., 2025; Zhou et al., 2025; Liu et al., 2025a; Wang et al., 2026b). Recent methods improve multimodal reasoning through longer chains of thought, adaptive optimization, data selection, and other optimization strategies. Despite their effectiveness, they often produce unnecessarily lengthy reasoning, leading to inference overhead. More discussions are provided in the Appendix E.

Efficient Reasoning in LVLMs. Existing efficient reasoning methods mainly reduce reasoning length (Han et al., 2025; Luo et al., 2026a; Shen et al., 2025; Renze & Guven, 2024; Team et al., 2025; Yeo et al., 2025) and adjust reasoning depth according to problem difficulty (Xiao & Gan, 2026; Liu et al., 2026a; Ma et al., 2025; Chen et al., 2026). However, both methods primarily operate at the token-sequence level, which may remove necessary reasoning or induce overthinking (Han et al., 2025; Luo et al., 2026b; Liu et al., 2026b). In this work, we analyze cross-layer entropy dynamics and identify token changes that contribute to unnecessary reasoning expansion. More details are provided in the Appendix E.

3 PRELIMINARY

Notation. Given a multimodal input, the LVLM autoregressively generates a reasoning sequence $\mathbf{y} = \{y_1, \ldots, y_T\}$. Let L denote the number of LLM layers and $\mathbf{h}_t^\ell \in \mathbb{R}^d$ denote the hidden state at reasoning position t and layer ℓ. Projecting $\mathbf{h}_t^\ell$ through the final normalization layer and the shared LM head yields the token distribution $\mathbf{p}_t^\ell \in \mathbb{R}^{|\mathcal{V}|}$, where $\mathcal{V}$ denotes the vocabulary. We denote the normalized entropy of this distribution by H_t^ℓ. The top-1 predictions at the trough and final layers are denoted by v_t^{tr} and v_t^L, respectively. A token preference shift occurs when $v_t^{\mathrm{tr}} \neq v_t^L$.

Problem Formalization. Standard LVLM inference uses the final-layer prediction as the output. When entropy rebounds and changes the preferred token, the prediction at the trough layer may differ from that at the final layer. We define a token choice $a_t \in \{\mathrm{tr}, L\}$ for such position, where $a_t = \mathrm{tr}$ selects the entropy-trough token and $a_t = L$ selects the final-layer token. The resulting token-selection problem is

$$\mathbf{a}^* = \arg\max \mathcal{U}\big(\mathbf{y}(\mathbf{a})\big), \qquad a_t \in \{\mathrm{tr}, L\}, \tag{1}$$

where $\mathbf{a}$ specifies the token choice at each reasoning position, $\mathbf{y}(\mathbf{a})$ denotes the resulting response, and $\mathcal{U}(\cdot)$ measures final-answer correctness and reasoning efficiency. Setting $a_t = L$ retains v_t^L, while $a_t = \mathrm{tr}$ adopts v_t^{tr}. The objective is to preserve beneficial final-layer token shifts while replacing harmful shifts with the entropy-trough token.

4 METHODOLOGY

We aim to mitigate overthinking in LVLMs by regulating abnormal final-layer token shifts. To this end, we propose the **EviGRPO** framework (Fig. 3), which regulates reasoning through a *Trigger–Verify* process. **EviGRPO** consists of two key components. First, an *entropy trigger* tracks the entropy of each reasoning token across LLM layers and identifies where entropy rebounds from a preceding trough (§4.1). Second, an *evidence verifier* determines whether the rebound induces a token preference shift and evaluates the evidential support for this shift (§4.2). When a token shift from the entropy-trough layer to the final layer lacks sufficient evidential support, the LLM falls back to the entropy-trough token; otherwise, the final-layer token is retained. We further construct the **EviGRPO** reward by incorporating the changes in reasoning length and final answer under trough-token substitution (§4.3), while an evidential objective jointly optimizes the *evidence verifier* during GRPO training (§4.4).

4.1 ENTROPY TRIGGER

The *entropy trigger* $\mathcal{T}$ detects reasoning positions where entropy rebounds after reaching a near-final-layer trough, indicating that deeper reasoning reintroduces uncertainty after an initially confident prediction. Given a multimodal input, the LVLM autoregressively generates a reasoning sequence $\mathbf{y} = \{y_1, \ldots, y_T\}$. Let L denote the number of LLM layers and $\mathbf{h}_t^\ell \in \mathbb{R}^d$ denote the hidden state at

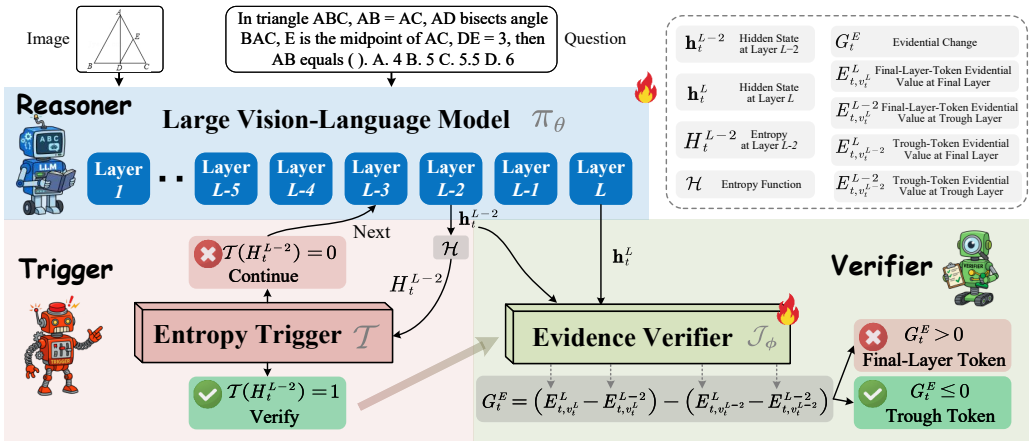


Figure 3: **Overview of EviGRPO framework.** The Reasoner processes the image-question pair with a large vision-language model and extracts intermediate- and final-layer representations. The *entropy trigger* detects entropy troughs across intermediate layers during decoding. The *evidence verifier* measures the evidential change between the trough-layer and final-layer predictions, preserving the final-layer token when the shift is evidence supported and otherwise adopting the trough-layer token.

reasoning position t and layer ℓ. To trace how entropy evolves across LLM depth, we project each intermediate hidden state through the final normalization layer and the shared LM head (Chuang et al., 2024; Chen et al., 2026), yielding $\mathbf{z}_t^\ell = \mathbf{W}_{\text{LM}}\,\text{Norm}(\mathbf{h}_t^\ell)$ and $\mathbf{p}_t^\ell = \text{Softmax}(\mathbf{z}_t^\ell)$, where $\mathbf{W}_{\text{LM}}$ denotes the LM head and $\mathcal{V}$ denotes the vocabulary. We compute the normalized entropy as

$$H_t^\ell = \mathcal{H}(\mathbf{p}_t^\ell) = -\frac{1}{\log|\mathcal{V}|}\sum_{v\in\mathcal{V}} p_{t,v}^\ell \log p_{t,v}^\ell. \tag{2}$$

A lower H_t^ℓ indicates a more concentrated token distribution, whereas a higher value reflects greater prediction uncertainty. We focus on late-layer entropy dynamics because they are most directly related to the final token prediction. Within the last K layers, starting from the final layer, we search backward toward earlier layers and identify the nearest entropy trough, where K denotes the number of layers considered for trough detection. We define $\mathcal{T}(H_t^\ell) = 1$ if $H_t^\ell < \min\left(H_t^{\ell-1}, H_t^{\ell+1}\right)$, indicating an entropy trough, and $\mathcal{T}(H_t^\ell) = 0$ otherwise. If no entropy trough is found within the last K layers, the trigger remains inactive for that reasoning position.

4.2 Evidence Verifier

The *evidence verifier* assesses whether the final-layer token shift induced by entropy rebound has evidential support, since such a shift may either reflect a beneficial correction or introduce an abnormal deviation. For each triggered reasoning position, we compare the token preferences at the entropy-trough and final layers. Their top-1 predictions are denoted by $v_t^{\text{tr}} = \arg\max_{v\in\mathcal{V}} p_{t,v}^{\ell^{\text{tr}}}$ and $v_t^L = \arg\max_{v\in\mathcal{V}} p_{t,v}^L$, respectively. If $v_t^{\text{tr}} = v_t^L$, the entropy rebound changes prediction confidence without altering the preferred token, and the final-layer prediction is directly retained. If $v_t^{\text{tr}} \neq v_t^L$, the rebound induces a token preference shift and the *evidence verifier* further determines whether the final-layer shift is supported by sufficient evidence.

Instantiation. We instantiate the *evidence verifier* as a lightweight MLP $\mathcal{J}_\phi$. For each reasoning position t activated by the *entropy trigger*, it receives the layer representation $\mathbf{h}_t^\ell$ together with the embedding $\mathbf{e}_v$ of candidate token v, and outputs a non-negative evidence value $E_{t,v}^\ell$. For each detected shift $v_t^{\text{tr}} \to v_t^L$, it evaluates both tokens at the trough and final layers to obtain the evidence values for subsequent verification.

Evidential Estimation. For each triggered preference shift, the verifier takes the layer representation $\mathbf{h}_t^\ell$ and the embedding $\mathbf{e}_v$ of candidate token v as input. The corresponding non-negative evidential support is defined as

$$E_{t,v}^\ell = \text{Softplus}\left(\mathcal{J}_\phi\left(\mathbf{h}_t^\ell, \mathbf{e}_v\right)\right). \tag{3}$$

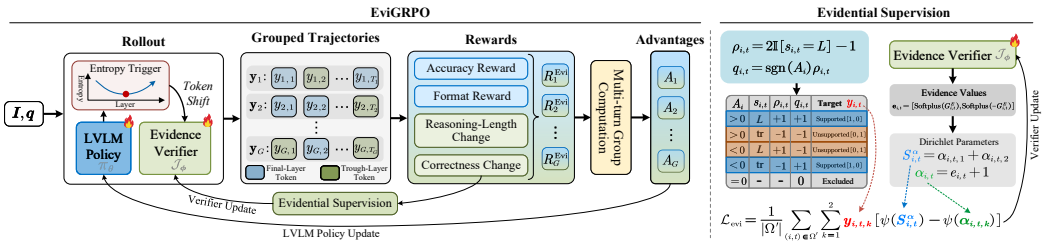


Figure 4: **Optimization for `EviGRPO`**. *Left:* **`EviGRPO`** enables the LVLM policy equipped with the *entropy trigger* and *evidence verifier* to generate grouped trajectories with final-layer and trough-layer token shifts. It combines accuracy and format rewards with reasoning-length and correctness changes to compute group advantages and optimize the LVLM policy, while the *evidence verifier* is optimized through evidential supervision. *Right:* Evidential supervision is constructed from token-shift outcomes and used to optimize the *evidence verifier* through Dirichlet evidence modeling.

where $\mathcal{J}_\phi$ denotes a lightweight learnable evidence head (i.e., MLP). For a preference shift $v_t^{\mathrm{tr}} \to v_t^L$, we measure the evidential change for the two tokens from the trough to final layers:

$$G_t^E = \left(E^L_{t,v_t^L} - E^{\ell^{\mathrm{tr}}}_{t,v_t^L}\right) - \left(E^L_{t,v_t^{\mathrm{tr}}} - E^{\ell^{\mathrm{tr}}}_{t,v_t^{\mathrm{tr}}}\right). \tag{4}$$

where $E^{\ell^{\mathrm{tr}}}_{t,v_t^{\mathrm{tr}}}$ and $E^{\ell^{\mathrm{tr}}}_{t,v_t^L}$ denote the evidential values for the trough- and final-layer tokens at the entropy-trough layer, while $E^L_{t,v_t^{\mathrm{tr}}}$ and E^L_{t,v_t^L} denote the evidential values for the trough- and final-layer tokens at the final layer. The first term in G_t^E measures the evidential change for the final-layer token, while the second measures that for the trough-layer token. Thus, G_t^E captures the relative evidential change from the trough to the final layer. A larger G_t^E indicates stronger evidential support for the final-layer token shift. The *evidence verifier* retains the final-layer preference when $G_t^E > 0$, indicating positive evidential support for the final-layer token shift. Otherwise, the final-layer token is replaced with the token predicted at the entropy-trough layer:

$$v_t = \begin{cases} v_t^{\mathrm{tr}}, & v_t^{\mathrm{tr}} \neq v_t^L,\ G_t^E \leq 0, \\ v_t^L, & \text{otherwise}, \end{cases} \tag{5}$$

Thus, trough-token substitution is applied only when the entropy rebound induces a token preference shift without positive evidential support.

4.3 EviGRPO Reward

To evaluate the reasoning change induced by entropy-trough-token substitution, we obtain two responses for each sample: a standard response $\mathbf{y}^{\mathrm{std}}$ generated with the final-layer tokens, and an entropy-trough-substituted response $\mathbf{y}^{\mathrm{tr}}$ generated by replacing the evidence-deficient final-layer tokens with the corresponding entropy-trough tokens.

Relative Answer Correctness Reward. Let $C^{\mathrm{std}}, C^{\mathrm{tr}} \in \{0, 1\}$ denote the final-answer correctness of standard response and entropy-trough-substituted response. We design the relative answer correctness reward as:

$$R^{\mathrm{ans}} = C^{\mathrm{tr}} - C^{\mathrm{std}}. \tag{6}$$

where $R^{\mathrm{ans}} = 1$ when trough-token substitution changes an incorrect standard response into a correct one, $R^{\mathrm{ans}} = -1$ when it changes a correct standard response into an incorrect one, and $R^{\mathrm{ans}} = 0$ when the two responses have the same correctness outcome. This reward directly measures whether trough-token substitution improves the final prediction.

Relative Reasoning Length Reward. Let N^{std} and N^{tr} denote their total reasoning lengths. We further evaluate the change in reasoning length. Once the entropy-trough-substituted response yields the correct final answer, we assess whether trough-token substitution reduces the overall reasoning length:

$$R^{\mathrm{len}} = \mathbb{I}\left[C^{\mathrm{tr}} = 1\right] \frac{N^{\mathrm{std}} - N^{\mathrm{tr}}}{N^{\mathrm{std}}}. \tag{7}$$

A positive R^{len} indicates that trough-token substitution yields a correct response with a shorter reasoning sequence than the standard response, while $R^{\text{len}} = 0$ when the substitution produces an incorrect response.

We additionally retain the accuracy and format rewards of the entropy-trough-substituted response (Guo et al., 2025a). The accuracy reward is defined as $R^{\text{acc}} = \mathbb{I}[\widehat{a} = a^*]$, where $\widehat{a}$ and a^* denote the predicted and ground-truth answers, respectively. The format reward is defined as $R^{\text{fmt}} = \mathbb{I}[\mathbf{y} \in \mathcal{F}]$, where $\mathcal{F}$ denotes the required response format. The complete **EviGRPO** reward is formulated as

$$R^{\text{Evi}} = \lambda_{\text{acc}} R^{\text{acc}} + \lambda_{\text{fmt}} R^{\text{fmt}} + \lambda_{\text{ans}} R^{\text{ans}} + \lambda_{\text{len}} R^{\text{len}}, \tag{8}$$

where λ_{acc}, λ_{fmt}, λ_{ans}, and λ_{len} balance the contributions of answer correctness, format compliance, answer correction, and reasoning efficiency, respectively.

4.4 EviGRPO Optimization

LVLM Policy Optimization. As shown in Fig. 4, given a multimodal input, we sample a group of G reasoning responses $\{\mathbf{y}_1, \ldots, \mathbf{y}_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$, where $\mathbf{y}_i = \{y_{i,1}, \ldots, y_{i,T_i}\}$ and T_i denotes the number of generated tokens in the i-th reasoning response. For each response, we compute the reward R_i^{Evi}. Following GRPO (Shao et al., 2024) , the reward is normalized within the group to obtain the group-relative advantage $A_i = (R_i^{\text{Evi}} - \mu_R)/(\sigma_R + \epsilon)$, where $\mu_R = \frac{1}{G}\sum_{j=1}^{G} R_j^{\text{Evi}}$ and $\sigma_R = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(R_j^{\text{Evi}} - \mu_R)^2}$. For each generated token $y_{i,t}$, the importance ratio between the current and old policies is defined as $r_{i,t}(\theta) = \pi_\theta(y_{i,t} \mid \mathbf{x}, \mathbf{y}_{i,<t}) / \pi_{\theta_{\text{old}}}(y_{i,t} \mid \mathbf{x}, \mathbf{y}_{i,<t})$. We optimize the policy using the clipped GRPO objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\, \{\mathbf{y}_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|\mathbf{x})} \left[\frac{1}{G}\sum_{i=1}^{G} \min\left(r_i(\theta)A_i,\ \text{clip}\left(r_i(\theta), 1-\epsilon_c, 1+\epsilon_c\right) A_i\right) - \beta D_{\text{KL}}\left(\pi_\theta \| \pi_{\text{ref}}\right) \right] \tag{9}$$

where ϵ_c denotes the clipping coefficient and β controls the KL regularization between the current policy π_θ and the reference policy π_{ref}. The token-level KL estimator is defined as

$$D_{\text{KL}}^{i,t} = \frac{\pi_{\text{ref}}(y_{i,t} \mid \mathbf{x}, \mathbf{y}_{i,<t})}{\pi_\theta(y_{i,t} \mid \mathbf{x}, \mathbf{y}_{i,<t})} - \log \frac{\pi_{\text{ref}}(y_{i,t} \mid \mathbf{x}, \mathbf{y}_{i,<t})}{\pi_\theta(y_{i,t} \mid \mathbf{x}, \mathbf{y}_{i,<t})} - 1. \tag{10}$$

Evidential Supervision. We optimize the *evidence verifier* $\mathcal{J}_\phi$ using the group-relative outcomes of the sampled rollouts. For the i-th reasoning response $\mathbf{y}_i$, let Ω_i denote the set of triggered preference shifts and $s_{i,t} \in \{\text{tr}, L\}$ denote the token choice at position $t \in \Omega_i$. We reuse the group-relative advantage A_i as an outcome signal and define the action indicator $\rho_{i,t} = 2\mathbb{I}[s_{i,t} = L] - 1$, where $\rho_{i,t} = 1$ corresponds to retaining the final-layer token and $\rho_{i,t} = -1$ corresponds to adopting the entropy-trough token. The rollout-derived supervision signal is then defined as

$$q_{i,t} = \text{sgn}(A_i)\rho_{i,t} \in \{-1, 0, 1\}. \tag{11}$$

A positive $q_{i,t}$ indicates that the group-relative rollout outcome supports the final-layer token shift, whereas a negative $q_{i,t}$ favors the entropy-trough token. For $q_{i,t} \neq 0$, we construct the binary evidential label $\mathbf{y}_{i,t} = [y_{i,t,1}, y_{i,t,2}]$, where $\mathbf{y}_{i,t} = [1, 0]$ for $q_{i,t} = 1$ and $\mathbf{y}_{i,t} = [0, 1]$ for $q_{i,t} = -1$, corresponding to supported and unsupported final-layer token shifts, respectively. Based on the evidential change score $G_{i,t}^E$ defined in Eq. (4), we construct the evidence vector (Sensoy et al., 2018)

$$\mathbf{e}_{i,t} = \left[\text{Softplus}(G_{i,t}^E), \text{Softplus}(-G_{i,t}^E)\right], \tag{12}$$

and obtain the Dirichlet parameters $\boldsymbol{\alpha}_{i,t} = \mathbf{e}_{i,t} + \mathbf{1}$. Let $S_{i,t}^\alpha = \alpha_{i,t,1} + \alpha_{i,t,2}$ denote the Dirichlet strength. The evidential loss is defined as

$$\mathcal{L}_{\text{evi}} = \frac{1}{|\Omega'|} \sum_{(i,t)\in\Omega'} \sum_{k=1}^{2} y_{i,t,k} \left[\psi(S_{i,t}^\alpha) - \psi(\alpha_{i,t,k})\right], \tag{13}$$

where $\Omega' = \{(i,t) \mid t \in \Omega_i,\ q_{i,t} \neq 0\}$, and $\psi(\cdot)$ denotes the digamma function. This objective provides group-relative evidential supervision to the verifier, encouraging token choices associated

with better-performing rollouts while suppressing those associated with worse outcomes. Finally, the LVLM policy and *evidence verifier* are jointly optimized as

$$\mathcal{L}_{\text{EviGRPO}} = -\mathcal{J}_{\text{GRPO}}(\theta) + \lambda_{\text{evi}}\mathcal{L}_{\text{evi}}(\phi), \tag{14}$$

where λ_{evi} balances policy optimization and evidential supervision. In this way, the *evidence verifier* learns to determine whether a final-layer token shift should be retained or replaced by the corresponding entropy-trough token, and **EviGRPO** optimizes the reasoning policy toward improved answer correctness and reduced redundant reasoning.

5 Experiments

5.1 Experimental Setup

Implementation Details. For training dataset, we adopt ViRL39K (Wang et al., 2026a), a curated dataset of 38,870 verifiable multimodal question–answer pairs spanning diverse reasoning tasks, including mathematical and geometric reasoning, chart and document understanding, science, and general visual reasoning. We conduct experiments on the Qwen2.5-VL series, including the 3B, 7B, and 32B (Bai et al., 2025b). All models are trained for 2 epochs with a learning rate of 1×10^{-6} and a global batch size of 256. We sample 8 rollouts per question with a rollout temperature of 1.0. Training is performed on 32 NVIDIA H100 GPUs. More details are provided in the Appendix A.1.

Baselines and Benchmarks. We compare **EviGRPO** with two categories of baselines: (i) *closed-source models*, including GPT-4o, Claude-3.7-Sonnet, and Gemini-2.0-Flash; (ii) *open-source models*, including Qwen2.5-VL (3B/7B/32B) (Bai et al., 2025b), OpenVLThinker-7B (Deng et al., 2026), MM-Eureka (7B/32B) (Meng et al., 2025), VL-Rethinker-7B (Wang et al., 2026a), R1-Onevision-7B (Yang et al., 2025), R1-VL-7B (Zhang et al., 2025), ThinkLite-VL-7B (Wang et al., 2026b), MM-R1-7B (Leng et al., 2025), LMM-R1-3B (Peng et al., 2025), Curr-ReFT-3B (Deng et al., 2025), FAST-GRPO (3B/7B/32B) (Xiao & Gan, 2026), and Vision-R1 (7B/32B) (Huang et al., 2026). We evaluate **EviGRPO** across three categories of multimodal reasoning benchmarks: (i) *mathematical reasoning*, including MathVista (Lu et al., 2024), MathVerse (Zhang et al., 2024), MathVision (Wang et al., 2024), LogicVista (Xiao et al., 2024), and WeMath (Qiao et al., 2025); (ii) *general reasoning*: MM-Vet (Yu et al., 2023); and (iii) *cross-domain reasoning*: MMK12 (Meng et al., 2025). Details of benchmarks are provided in the Appendix A.2.

5.2 Main Results

Accuracy. As shown in Table 1, **EviGRPO** consistently improves reasoning accuracy across model scales and task domains. On mathematical reasoning, **EviGRPO** consistently improves over both standard GRPO and the efficiency-oriented FAST-GRPO. For the 3B model, it exceeds GRPO by 2.2–5.7 points across the five benchmarks and also outperforms FAST-GRPO. The gains remain clear on more challenging benchmarks: on MathVerse and LogicVista, **EviGRPO** reaches 60.5% and 42.8%, surpassing FAST-GRPO by 3.6 and 4.3 points, respectively. Similar advantages are observed at larger models. For the 7B model, **EviGRPO** achieves 71.6% on MathVerse, higher than FAST-GRPO (70.1%) and VL-Rethinker (67.1%), while at 32B model it surpasses FAST-GRPO on all five mathematical benchmarks. On general and cross-domain reasoning, the advantage becomes more evident on MMK12, where **EviGRPO** improves over FAST-GRPO by 3.3, 3.8, and 3.7 points at 3B, 7B, and 32B, respectively. On MM-Vet, it also gains 4.7 and 2.6 points at 3B and 7B, while remaining comparable at 32B (72.9% vs. 73.4%). Overall, **EviGRPO** provides stable gains over both standard GRPO and efficient-reasoning baselines across mathematical, general, and cross-domain tasks.

Reasoning Length. Table 1 also shows that **EviGRPO** substantially reduces reasoning length across different model scales while preserving its accuracy advantage. On mathematical reasoning, **EviGRPO** reduces MathVerse from 317.9/358.6/541.2 tokens with GRPO to 227.4/211.7/350.5 at 3B/7B/32B model, respectively. Compared with FAST-GRPO, it further shortens MathVerse for the 3B, 7B, and 32B models and achieves shorter reasoning on two of five mathematical benchmarks at 32B model. The gap is larger against long-CoT methods: on MathVision with the 7B model, **EviGRPO** uses only 214.9 tokens, compared with 692.8 for R1-Onevision and 830.9 for Vision-R1.

Table 1: Main results on mathematical, general, and cross-domain reasoning benchmarks with Qwen2.5-VL (3B/7B/32B). Acc. denotes accuracy (%), and Len. denotes the average number of reasoning tokens. The best accuracy and shortest reasoning length are highlighted in **bold**, while the second-best accuracy and second-shortest reasoning length are underlined.

Model	Mathematical Reasoning										General		Cross-Domain	
	MathVista		MathVerse		MathVision		LogicVista		WeMath		MM-Vet		MMK12	
	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.
Closed-Source Models														
GPT-4o	63.8	–	49.9	–	30.4	–	52.8	–	45.8	–	80.8	–	49.9	–
Claude-3.7-Sonnet	66.8	–	52.0	–	41.3	–	58.2	–	49.3	–	70.0	–	55.3	–
Gemini-2.0-Flash	70.4	–	59.3	–	41.3	–	52.3	–	47.4	–	73.6	–	65.2	–
Open-Source Models (3B)														
LMM-R1-3B	63.2	245.0	41.8	423.9	25.2	447.8	–	–	62.9	382.5	65.9	166.3	–	–
Curr-ReFT-3B	61.9	**95.9**	36.3	**121.6**	20.1	**240.1**	–	–	57.3	**156.0**	62.0	**117.6**	–	–
Qwen2.5-VL-3B	62.6	233.1	33.1	371.4	21.5	423.4	38.2	284.4	24.1	328.0	61.8	148.7	39.1	367.4
+ GRPO	63.9	144.6	55.4	317.9	25.2	455.3	38.9	273.0	58.9	411.6	62.7	155.7	57.5	345.8
+ FAST-GRPO	65.6	125.7	56.9	265.9	27.0	354.7	38.5	**256.8**	62.0	233.6	63.6	126.2	59.1	309.5
+ **EviGRPO**	**67.7**	116.6	**60.5**	227.4	**27.4**	384.7	**42.8**	271.4	**64.6**	221.7	**68.3**	142.4	**62.4**	**273.4**
Open-Source Models (7B)														
OpenVLThinker-7B	70.2	305.7	47.9	398.4	29.6	457.2	44.3	–	64.5	326.7	68.5	312.7	60.6	561.0
MM-Eureka-7B	73.0	252.0	50.3	–	26.9	612.0	48.9	–	66.1	–	68.9	–	64.5	537.8
VL-Rethinker-7B	74.9	268.0	67.1	351.0	22.3	425.0	46.3	267.0	68.9	334.0	69.3	–	68.3	–
R1-Onevision-7B	64.1	402.5	46.4	631.5	29.9	692.8	45.6	308.0	61.8	591.9	71.6	440.7	39.8	817.5
R1-VL-7B	63.5	263.0	40.0	–	24.7	363.0	36.3	–	53.8	–	–	–	32.7	–
ThinkLite-VL-7B	75.1	247.0	52.1	–	28.5	599.0	44.3	–	69.5	–	67.8	–	57.9	–
MM-R1-7B	71.0	185.6	49.8	283.9	30.2	324.6	50.8	–	67.9	235.7	70.6	137.9	–	–
Vision-R1-7B	73.5	410.2	52.4	337.0	29.4	830.9	49.7	293.0	62.9	–	71.1	–	36.7	–
Qwen2.5-VL-7B	70.1	205.4	39.7	374.2	25.4	457.1	44.7	294.5	46.2	312.5	67.1	138.2	54.1	459.8
+ GRPO	74.4	194.0	68.3	358.6	28.6	471.3	46.5	255.2	68.3	285.1	68.2	154.7	72.8	466.7
+ FAST-GRPO	74.8	**141.4**	70.1	225.7	30.8	232.7	49.0	**179.6**	67.6	**160.5**	71.0	119.3	74.5	386.0
+ **EviGRPO**	**77.0**	166.4	**71.6**	**211.7**	**31.5**	**214.9**	**51.3**	221.4	**71.6**	178.1	**73.6**	**100.4**	**78.3**	**322.8**
Open-Source Models (32B)														
MM-Eureka-32B	74.8	352.0	56.5	560.0	34.4	639.0	53.4	–	73.4	524.0	**73.4**	344.0	72.2	857.0
Vision-R1-32B	76.4	410.0	60.9	818.0	39.1	976.0	54.2	–	74.2	637.0	72.2	384.0	64.8	1039.0
Qwen2.5-VL-32B	77.6	354.3	68.3	447.2	38.4	670.4	51.3	335.7	71.2	531.7	71.1	310.5	68.5	863.9
+ GRPO	78.1	310.3	74.5	541.2	39.7	522.0	56.1	361.4	75.4	611.5	71.7	287.4	80.4	602.7
+ FAST-GRPO	78.4	280.3	74.0	421.0	40.6	**513.8**	58.2	**258.4**	74.9	**398.3**	**73.4**	277.1	81.6	582.6
+ **EviGRPO**	**79.3**	**249.4**	**75.9**	**350.5**	**42.4**	519.2	**58.8**	281.3	**77.1**	431.4	72.9	**220.8**	**85.3**	**491.7**

On general and cross-domain reasoning, **EviGRPO** reduces MM-Vet/MMK12 from 154.7/466.7 to 100.4/322.8 tokens over GRPO at 7B model, and also outperforms FAST-GRPO on both benchmarks. For the 32B model, it further shortens MM-Vet/MMK12 to 220.8/491.7 tokens, compared with 277.1/582.6 for FAST-GRPO. Overall, **EviGRPO** consistently reduces reasoning length over GRPO, and remains competitive with FAST-GRPO.

5.3 Ablation Study

We conduct ablation studies to validate the effectiveness of each component in **EviGRPO**, including the *evidence verifier*, R^{ans}, and R^{len}. As shown in Table 2, removing the *evidence verifier* causes the average accuracy to fall from 75.1% to 72.3%, while the average reasoning length rises from 191.9 to 236.3 tokens, indicating that the token-shift verification is important for both prediction quality and efficiency. Excluding R^{ans} leads to performance drop, with accuracy decreasing to 73.0% and reasoning length increasing to 251.1 tokens, showing that correctness-aware feedback is necessary to guide beneficial token replacement. By contrast, removing R^{len} has a smaller effect on accuracy (73.8%) but still produces longer reasoning (203.7 tokens). Together, these results show that these three components provide complementary benefits to **EviGRPO**. More ablation studies are provided in the Appendix B, including sensitivity analyses of the *entropy trigger* hyperparameter K and the coefficients λ_{acc}, λ_{fmt}, λ_{ans}, λ_{len}, and λ_{evi}.

5.4 Further Analyses

Token-Shift Analysis. We further analyze different token-selection strategies in Fig. 5. Directly replacing final-layer tokens with trough tokens already improves both accuracy and reasoning efficiency across all four benchmarks. For example, on WeMath, accuracy increases from 67.5% to 69.2%, while the reasoning length decreases from 226.7 to 199.2 tokens. However, always choosing

Table 2: Ablation study of **EviGRPO** on MathVista, WeMath, MM-Vet, and MMK12.

Model	MathVista		WeMath		MM-Vet		MMK12		Avg.	
	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.
Qwen2.5-VL-7B	70.1	205.4	46.2	312.5	67.1	138.2	54.1	459.8	59.4	279.0
EviGRPO	**77.0**	**166.4**	**71.6**	**178.1**	73.6	**100.4**	**78.3**	**322.8**	**75.1**	**191.9**
w/o Verifier	74.6	195.1	67.1	239.5	72.4	119.3	75.1	391.3	72.3	236.3
w/o R^{ans}	75.2	201.6	68.2	250.7	72.7	128.8	75.9	423.3	73.0	251.1
w/o R^{len}	75.8	174.8	68.8	199.8	**74.0**	106.0	76.6	334.2	73.8	203.7

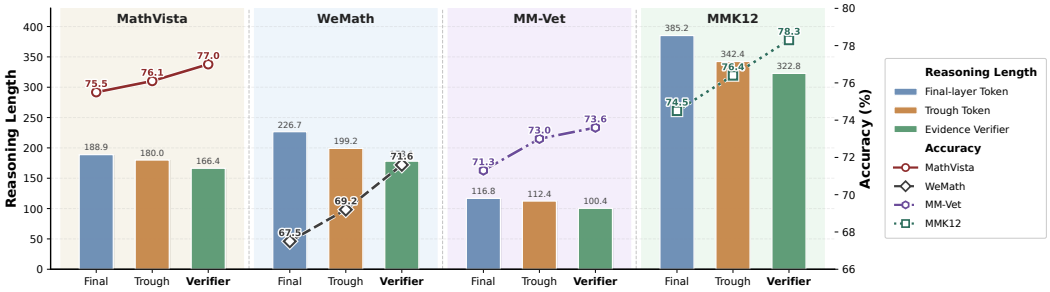


Figure 5: Token-Shift analysis of different token selection strategies on reasoning length and accuracy across MathVista, WeMath, MM-Vet, and MMK12

the trough token is still suboptimal, since some final-layer shifts correspond to useful corrections. By selectively preserving evidence-supported final-layer tokens, the *evidence verifier* further improves accuracy to 71.6% and reduces the reasoning length to 178.1 tokens. Similar trends are observed on MathVista, MM-Vet, and MMK12, where the verifier consistently achieves the highest accuracy and shortest reasoning length. These results show that effective overthinking mitigation requires not only identifying entropy troughs, but also determining whether the subsequent token shift is beneficial.

Evidential Change Analysis. Fig. 6 visualizes the evidential changes of the trough-layer and final-layer tokens. The two types of token shifts are largely separated by the decision boundary $G_t^E = 0$. When the final-layer token gains stronger evidence than the trough token, the samples mainly fall in the region $G_t^E > 0$, where the final-layer prediction is retained. In contrast, shifts with stronger support for the trough token are concentrated in the $G_t^E < 0$ region, leading to trough-token substitution. This distribution shows that the *evidence verifier* can effectively distinguish between supported and unsupported token shifts, allowing beneficial final-layer corrections to be retained while unreliable shifts are replaced by trough-layer predictions. More analyses of efficiency and training dynamics are provided in the Appendix C. Case studies are provided in the Appendix D.

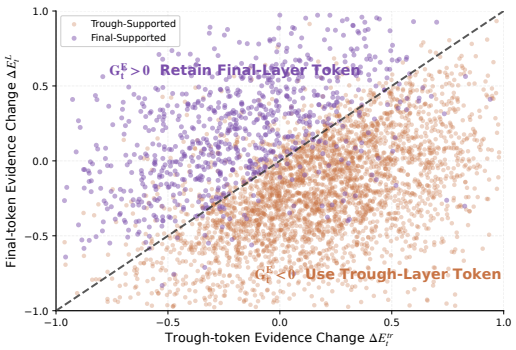


Figure 6: Evidential change distribution of entropy-trough and final-layer tokens.

6 Conclusion

In this work, we investigate LVLM overthinking from the perspective of layer-wise token dynamics and reveal that token entropy can reach an intermediate-layer trough and rebound toward the final layer, potentially affecting both the reasoning correctness and reasoning length. Based on this observation, we propose **EviGRPO**, which regulates reasoning through a *Trigger–Verify* process. The *entropy trigger* identifies entropy-trough tokens, while the *evidence verifier* determines whether the final-layer token should be replaced by the corresponding entropy-trough token. **EviGRPO** further combines relative correctness- and length-aware rewards with evidential supervision for end-to-end optimization. Experiments across 3B/7B/32B model scales and seven benchmarks demonstrate that **EviGRPO** effectively reduces redundant reasoning while maintaining superior accuracy.

AI USE STATEMENT

In preparing this work, we employed large language models (LLMs) to assist with polishing the writing, conducting literature reviews, improving the clarity of presentation, and refining the organization of the manuscript. All generated content was carefully reviewed and verified by the authors.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of **`EviGRPO`** in §4 and experimental settings in §5.1 and Appendix A. All key configurations, including Qwen2.5-VL backbones, training data, optimization hyperparameters, rollout settings, entropy-trigger and reward configurations, hardware, benchmarks, and evaluation protocols, are reported. Code and configuration files will be released to support reproducibility for public research use.

ETHICS STATEMENT

In this work, we investigate layer-wise token dynamics and evidential reasoning optimization for improving the accuracy and efficiency of Large Vision-Language Models. Our experiments are conducted using publicly available LVLMs, training data, and multimodal reasoning benchmarks. We do not collect new human-subject data and private information. Our research focuses on improving LVLM reasoning efficiency and does not involve safety-critical applications.

REFERENCES

Pranjal Aggarwal, Seungone Kim, Jack Lanchantin, Sean Welleck, Jason E Weston, Ilia Kulikov, and Swarnadeep Saha. Optimalthinkingbench: Evaluating over and underthinking in llms. In *International Conference on Learning Representations*, volume 2026, pp. 8886–8918, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

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

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Shuang Chen, Hangyu Guo, Yimeng Ye, Shijue Huang, Wenbo Hu, Jiayu Chen, Manyuan Zhang, Haoxi Li, Song Guo, and Nanyun Violet Peng. Ares: Multimodal adaptive reasoning via difficulty-aware token-level entropy shaping. In *International Conference on Learning Representations*, volume 2026, pp. 147474–147512, 2026.

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *International Conference on Learning Representations*, volume 2024, pp. 54158–54183, 2024.

Huilin Deng, Ding Zou, Xinghao Zhao, Rui Ma, Yanming Guo, Yang Cao, and Yu Kang. Curr-reft: Overcoming training bottlenecks in small-scale vision-language models via curriculum reinforcement finetuning. In *EMNLP (Findings)*, pp. 12021–12032, 2025.

Yihe Deng, Hritik Bansal, Fan Yin, Nanyun Peng, Wei Wang, and Kai-Wei Chang. Openvlthinker: Complex vision-language reasoning via iterative sft-rl cycles. *Advances in Neural Information Processing Systems*, 38:123817–123846, 2026.

Xingjian Diao, Zheyuan Liu, Chunhui Zhang, Weiyi Wu, Keyi Kong, Lin Shi, Kaize Ding, Soroush Vosoughi, and Jiang Gui. Addressing overthinking in large vision-language models via gated perception-reasoning optimization. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 4393–4410, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025b.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, 2025.

Wenxuan Huang, Bohan Jia, Shaosheng Cao, Zheyu Ye, Zhe Xu, Yao Hu, Shaohui Lin, et al. Vision-r1: Incentivizing reasoning capability in multimodal large language models. In *International Conference on Learning Representations*, volume 2026, pp. 63794–63812, 2026.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanwei Li, Yu Qi, Xinyan Chen, Liuhui Wang, Jianhan Jin, Claire Guo, Shen Yan, et al. Mme-cot: Benchmarking chain-of-thought in large multimodal models for reasoning quality, robustness, and efficiency. *arXiv preprint arXiv:2502.09621*, 2025.

Sicong Leng, Jing Wang, Jiaxi Li, Hao Zhang, Zhiqiang Hu, Boqiang Zhang, Hang Zhang, Yuming Jiang, Xin Li, Deli Zhao, et al. Mmr1: Advancing the frontiers of multimodal reasoning, 2025.

Wei Liu, Ruochen Zhou, Yiyun Deng, Yuzhen Huang, Junteng Liu, Yuntian Deng, Yizhe Zhang, and Junxian He. Learn to reason efficiently with adaptive length-based reward shaping. In *International Conference on Learning Representations*, volume 2026, pp. 43279–43308, 2026a.

Yongjiang Liu, Haoxi Li, Xiaosong Ma, Jie Zhang, and Song Guo. Think how to think: Mitigating overthinking with autonomous difficulty cognition in large reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 38105–38126, 2026b.

Zhiyuan Liu, Yuting Zhang, Feng Liu, Changwang Zhang, Ying Sun, and Jun Wang. Othink-mr1: Stimulating multimodal generalized reasoning capabilities via dynamic reinforcement learning. *arXiv preprint arXiv:2503.16081*, 2025a.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning. In *2025 IEEE/CVF International Conference on Computer Vision*, pp. 2034–2044, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, volume 2024, pp. 23439–23554, 2024.

Haotian Luo, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Xiaochun Cao, Dacheng Tao, Naiqiang Tan, and Li Shen. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14242–14257, 2026a.

Qin-Wen Luo, Sheng Ren, Xiang Chen, Rui Liu, Jun Fang, Naiqiang Tan, and Sheng-Jun Huang. Compress the easy, explore the hard: Difficulty-aware entropy regularization for efficient llm reasoning. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 3480–3491, 2026b.

Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6025–6035, 2025.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, et al. Mm-eureka: Exploring visual aha moment with rule-based large-scale reinforcement learning. *arXiv preprint arXiv:2503.07365*, 3, 2025.

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

Runqi Qiao, Qiuna Tan, Guanting Dong, MinhuiWu MinhuiWu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma Gongque, Shanglin Lei, Yifan Zhang, et al. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20023–20070, 2025.

Matthew Renze and Erhan Guven. The benefits of a concise chain of thought on problem-solving in large language models. *arXiv preprint arXiv:2401.05618*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems*, 31, 2018.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, Wenjing Zhang, Jiangze Yan, Ning Wang, Kai Wang, Zhaoxiang Liu, and Shiguo Lian. Dast: Difficulty-adaptive slow-thinking for large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2322–2331, 2025.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Xinyu Tian, Shu Zou, Zhaoyuan Yang, Mengqi He, Fabian Waschkowski, Lukas Wesemann, Peter Tu, and Jing Zhang. More thought, less accuracy? on the dual nature of reasoning in vision-language models. In *International Conference on Learning Representations*, volume 2026, pp. 133928–133965, 2026.

Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhu Chen. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *Advances in Neural Information Processing Systems*, 38:30865–30891, 2026a.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024.

Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement. *Advances in Neural Information Processing Systems*, 38: 118818–118850, 2026b.

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

Wenyi Xiao and Leilei Gan. Fast-slow thinking grpo for large vision-language model reasoning. *Advances in Neural Information Processing Systems*, 38:171601–171631, 2026.

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

Yi Yang, Xiaoxuan He, Hongkun Pan, Xiyan Jiang, Yan Deng, Xingtao Yang, Haoyu Lu, Dacheng Yin, Fengyun Rao, Minfeng Zhu, et al. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. In *2025 IEEE/CVF International Conference on Computer Vision*, pp. 2376–2385, 2025.

Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023.

Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization. In *2025 IEEE/CVF International Conference on Computer Vision*, pp. 1859–1869, 2025.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pp. 169–186. Springer, 2024.

Hengguang Zhou, Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. R1-zero's "aha moment" in visual reasoning on a 2b non-sft model. *arXiv preprint arXiv:2503.05132*, 2025.

APPENDIX

A Additional Experimental Details

A.1 Training Configuration

Beyond the training settings reported in §5.1, we provide additional implementation details for reproducibility. We optimize the policy using AdamW (Loshchilov & Hutter, 2017) with a weight decay of 1×10^{-2}. Training is performed in `bfloat16` precision with gradient checkpointing enabled. Both the vision encoder and the language model remain trainable throughout optimization. The maximum prompt length is set to 4096 tokens. The maximum response length is 2048 tokens for the 3B and 7B models and 4096 tokens for the 32B model. During rollout generation, we set top-p to 0.99. For the GRPO objective, we use a clipping coefficient of $\epsilon_c = 0.2$ and a KL regularization coefficient of $\beta = 0.01$. For the *entropy trigger*, the default search window is set to $K = 10$. The reward coefficients are $\lambda_{\text{acc}} = 0.4$, $\lambda_{\text{fmt}} = 0.3$, $\lambda_{\text{ans}} = 0.1$, and $\lambda_{\text{len}} = 0.2$. The evidential objective is weighted by $\lambda_{\text{evi}} = 0.3$.

A.2 Benchmarks

We provide additional details of the evaluation benchmarks used in the main experiments.

MathVista (Lu et al., 2024) is a comprehensive benchmark for visual mathematical reasoning, covering diverse problem types such as geometry, charts, tables, and scientific figures. It requires models to integrate visual perception with mathematical knowledge and compositional reasoning.

MathVerse (Zhang et al., 2024) is designed to examine whether multimodal models genuinely understand visual diagrams rather than relying primarily on textual cues. Each original problem is reformulated into multiple versions with different distributions of information between the text and image, ranging from text-dominant to vision-dominant settings. It enables a fine-grained evaluation of the model's dependence on visual evidence during mathematical reasoning.

MathVision (Wang et al., 2024) focuses on competition-level multimodal mathematics. The problems are collected from real mathematical competitions and span diverse mathematical subjects and difficulty levels, providing a challenging evaluation of advanced visual reasoning.

Table 3: Sensitivity analysis of the *entropy trigger* hyperparameter K on Qwen2.5-VL-7B. Acc. denotes accuracy (%), and Len. denotes the average reasoning length.

K	MathVista		WeMath		MM-Vet		MMK12		Avg.	
	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.
6	75.8	184.2	68.9	209.7	72.7	112.8	76.4	355.6	73.5	215.6
8	76.5	173.8	70.8	189.5	73.2	104.7	77.6	335.9	74.5	201.0
10	**77.0**	**166.4**	**71.6**	**178.1**	73.6	**100.4**	**78.3**	**322.8**	**75.1**	**191.9**
12	76.7	169.8	71.2	182.9	**73.8**	102.1	77.9	328.6	74.9	195.9
14	76.2	177.5	70.3	196.4	72.9	107.6	77.2	341.2	74.2	205.7

Table 4: Sensitivity analysis of the evidential loss coefficient λ_{evi} on Qwen2.5-VL-7B. Acc. denotes accuracy (%), and Len. denotes the average reasoning length.

λ_{evi}	MathVista		WeMath		MM-Vet		MMK12		Avg.	
	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.	Acc.	Len.
0.1	76.0	179.6	70.7	201.7	73.2	108.9	78.5	348.4	74.6	209.7
0.2	75.8	171.8	**71.8**	187.4	73.6	**99.8**	**78.8**	331.5	75.0	197.6
0.3	**77.0**	**166.4**	71.6	**178.1**	**73.6**	100.4	78.3	**322.8**	**75.1**	**191.9**
0.4	76.6	169.7	70.9	184.6	73.3	102.8	77.7	329.1	74.6	196.6

LogicVista (Xiao et al., 2024) evaluates general logical reasoning in visually grounded scenarios, covering inductive, deductive, numerical, spatial, and mechanical reasoning across different visual formats.

WeMath (Qiao et al., 2025) provides a diagnostic benchmark for multimodal mathematical reasoning. Rather than evaluating models only through aggregate answer accuracy, it organizes problems according to a hierarchical structure of mathematical knowledge concepts and decomposes complex problems into finer-grained components. This design enables a more detailed examination of whether failures arise from insufficient mathematical knowledge or limited generalization ability.

MM-Vet (Yu et al., 2023) evaluates general multimodal understanding and reasoning through diverse visual questions, providing a complementary assessment beyond mathematics-oriented benchmarks.

MMK12 (Meng et al., 2025) evaluates multimodal reasoning at the K–12 educational level across mathematics, physics, chemistry, and biology. Its cross-subject setting requires models to combine visual understanding, domain knowledge, and reasoning, making it suitable for assessing whether improvements learned from multimodal reasoning training transfer across different scientific domains.

B Additional Ablation Studies

B.1 Sensitivity to the Entropy Trigger Hyperparameter K

We study the sensitivity of **EviGRPO** to the search window K of the *entropy trigger* on Qwen2.5-VL-7B. As shown in Table 3, increasing K from 6 to 10 steadily improves the average accuracy from 73.5% to 75.1%, while reducing the average reasoning length from 215.6 to 191.9 tokens. Further enlarging the window brings no additional benefit. At $K = 14$, accuracy drops to 74.2% and reasoning length increases to 205.7. A small K may miss entropy troughs near the final layer, whereas an overly large window extends the search to earlier layers whose predictions are less closely related to the final token. Finally, we set $K = 10$ as the default, which achieves the highest average accuracy and the shortest reasoning length.

B.2 Sensitivity to the Evidential Loss Coefficient

We further study the sensitivity to the evidential loss coefficient λ_{evi} on Qwen2.5-VL-7B. As shown in Table 4, increasing λ_{evi} from 0.1 to 0.3 improves the average accuracy from 74.6% to 75.1% and reduces the average reasoning length from 209.7 to 191.9. The best overall performance is achieved at $\lambda_{\text{evi}} = 0.3$. Accordingly, increasing λ_{evi} to 0.4 decreases the average accuracy to 74.6% and increases the reasoning length to 196.6 tokens. These results show that **EviGRPO** is relatively stable across different values of λ_{evi}, and we use $\lambda_{\text{evi}} = 0.3$ by default.

Table 5: Sensitivity analysis of the reward coefficients on Qwen2.5-VL-7B. Acc. and Len. are averaged over MathVista, WeMath, MM-Vet, and MMK12.

(a) Sensitivity between λ_{acc} and λ_{ans}.

$\lambda_{\text{acc}} : \lambda_{\text{fmt}} : \lambda_{\text{ans}} : \lambda_{\text{len}}$	Acc.	Len.
0.25 : 0.3 : 0.25 : 0.2	74.2	203.7
0.30 : 0.3 : 0.20 : 0.2	73.9	197.6
0.35 : 0.3 : 0.15 : 0.2	74.8	205.1
0.40 : 0.3 : 0.10 : 0.2	**75.1**	**191.9**
0.45 : 0.3 : 0.05 : 0.2	74.4	199.3

(b) Sensitivity between λ_{fmt} and λ_{ans}.

$\lambda_{\text{acc}} : \lambda_{\text{fmt}} : \lambda_{\text{ans}} : \lambda_{\text{len}}$	Acc.	Len.
0.4 : 0.15 : 0.25 : 0.2	74.6	201.2
0.4 : 0.20 : 0.20 : 0.2	74.1	195.4
0.4 : 0.25 : 0.15 : 0.2	74.9	203.6
0.4 : 0.30 : 0.10 : 0.2	**75.1**	**191.9**
0.4 : 0.35 : 0.05 : 0.2	74.3	197.8

(c) Sensitivity between λ_{ans} and λ_{len}.

$\lambda_{\text{acc}} : \lambda_{\text{fmt}} : \lambda_{\text{ans}} : \lambda_{\text{len}}$	Acc.	Len.
0.4 : 0.3 : 0.25 : 0.05	74.5	205.2
0.4 : 0.3 : 0.20 : 0.10	74.0	198.7
0.4 : 0.3 : 0.15 : 0.15	74.8	202.3
0.4 : 0.3 : 0.10 : 0.20	**75.1**	191.9
0.4 : 0.3 : 0.05 : 0.25	74.4	**187.6**

(d) Sensitivity between λ_{acc} and λ_{len}.

$\lambda_{\text{acc}} : \lambda_{\text{fmt}} : \lambda_{\text{ans}} : \lambda_{\text{len}}$	Acc.	Len.
0.50 : 0.3 : 0.1 : 0.10	74.4	202.6
0.45 : 0.3 : 0.1 : 0.15	74.9	198.4
0.40 : 0.3 : 0.1 : 0.20	**75.1**	191.9
0.35 : 0.3 : 0.1 : 0.25	74.2	193.1
0.30 : 0.3 : 0.1 : 0.30	74.7	**187.8**

Table 6: Efficiency comparison of Qwen2.5-VL-7B on MathVision.

Method	Acc.	Len.	End-to-End Latency (h)	Speedup
Qwen2.5-VL-7B	25.4	457.1	11.17	1.00×
+ GRPO	28.6	471.3	12.48	0.89×
+ FAST-GRPO	30.8	232.7	8.33	1.34×
+ **EviGRPO**	**31.5**	**214.9**	**8.16**	**1.37**×

B.3 Sensitivity to Reward Coefficients

We analyze the sensitivity of **EviGRPO** to the reward coefficients λ_{acc}, λ_{fmt}, λ_{ans}, and λ_{len} on Qwen2.5-VL-7B. As shown in Table 5, **EviGRPO** remains relatively stable across different coefficient combinations, with average accuracy ranging from 73.9% to 75.1%. Among all settings, $\lambda_{\text{acc}} = 0.4$, $\lambda_{\text{fmt}} = 0.3$, $\lambda_{\text{ans}} = 0.1$, and $\lambda_{\text{len}} = 0.2$ achieve the highest average accuracy of 75.1% with a reasoning length of 191.9 tokens. Increasing the weight of R_{len} further shortens the reasoning length, for example, $\lambda_{\text{len}} = 0.25$ yields 187.6 tokens, but reduces accuracy to 74.4%. Similarly, placing more weight on R_{ans} or reducing the contributions of R_{acc} and R_{fmt} does not provide further improvement. These results show that the default coefficients provide stable accuracy while effectively controlling reasoning length, and we use $(\lambda_{\text{acc}}, \lambda_{\text{fmt}}, \lambda_{\text{ans}}, \lambda_{\text{len}}) = (0.4, 0.3, 0.1, 0.2)$ in all experiments.

C Additional Analyses

C.1 Efficiency Analysis

As shown in Table 6, **EviGRPO** achieves the best overall efficiency on MathVision, where all methods are evaluated on four NVIDIA H100 GPUs with a batch size of 1. Compared with the base model of Qwen2.5-VL-7B, **EviGRPO** improves accuracy from 25.4% to 31.5% while reducing the average reasoning length from 457.1 to 214.9 tokens and the End-to-End latency from 11.17 hours to 8.16 hours, yielding a 1.37× speedup. It also outperforms FAST-GRPO in both accuracy (31.5% vs. 30.8%) and latency (8.16 hours vs. 8.33 hours), while producing shorter reasoning sequences. In contrast, standard GRPO improves accuracy but increases both reasoning length and latency. These results demonstrate that **EviGRPO** improves reasoning accuracy while simultaneously reducing inference cost.

C.2 Training Dynamics

Accuracy Reward Dynamics. Fig. 7 shows the evolution of the accuracy reward during training. **EviGRPO** consistently achieves higher rewards than GRPO, with the difference becoming clearer as

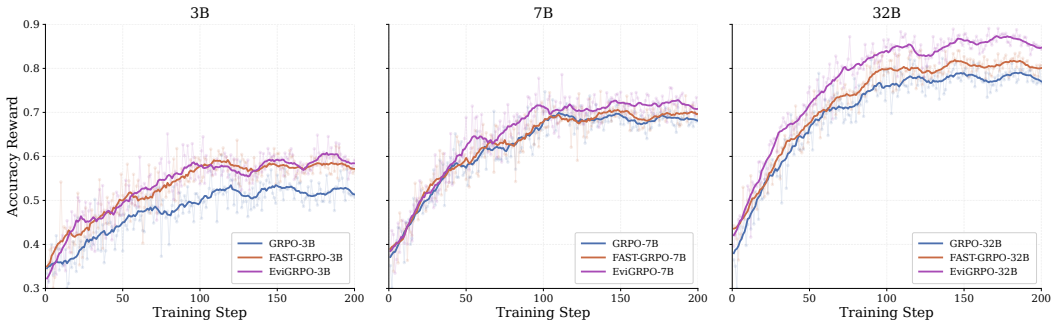


Figure 7: Training dynamics of accuracy reward for GRPO, FAST-GRPO, and **EviGRPO** across Qwen2.5-VL-3B, 7B, and 32B.

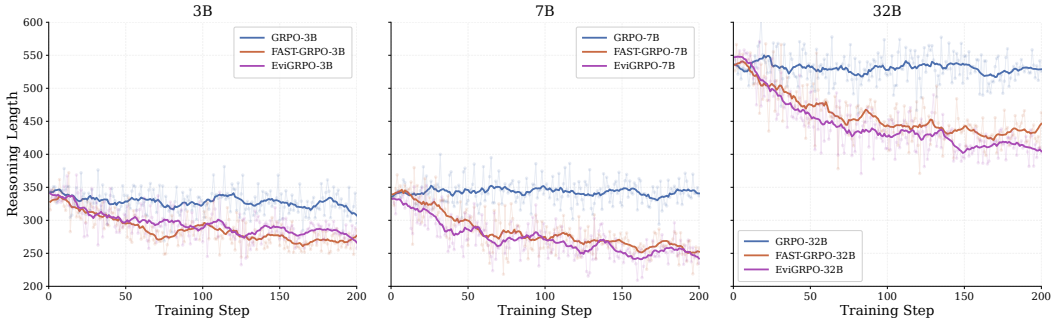


Figure 8: Training dynamics of reasoning length for GRPO, FAST-GRPO, and **EviGRPO** across Qwen2.5-VL-3B, 7B, and 32B.

training proceeds. On the 3B model, the advantage appears after the initial stage, while FAST-GRPO generally remains between GRPO and **EviGRPO**. The same trend is observed on 7B and is more pronounced on 32B, where **EviGRPO** reaches the highest final reward. These results indicate that the proposed **EviGRPO** helps improve LVLM optimization by suppressing unsupported token shifts while retaining beneficial final-layer corrections.

Reasoning-Length Dynamics. The reasoning-length curves show different trends for GRPO, FAST-GRPO, and **EviGRPO**. As shown in Fig. 8, standard GRPO keeps relatively long responses throughout training, with only small changes in reasoning length. In contrast, both FAST-GRPO and **EviGRPO** progressively shorten their reasoning sequences as optimization proceeds. The effect is especially visible for the larger models, where the gap between **EviGRPO** and GRPO continues to increase during training. Compared with FAST-GRPO, **EviGRPO** also shows a consistent decrease in reasoning length over the training process, particularly for the 7B and 32B models. Overall, these curves suggest that **EviGRPO** progressively learns shorter reasoning trajectories throughout training.

Relative-Reward Dynamics. To analyze the behavior of the two proposed rewards, Fig. 9 shows the evolution of R^{ans} and R^{len} throughout training. Both rewards increase from low initial values and gradually become stable across all three model scales, indicating that trough-token substitution becomes more effective as training proceeds. The reward values also increase as the model scale grows, with the 32B model achieving the highest values for both objectives. Notably, R^{ans} generally stays above R^{len} in the later stage, suggesting that the learned token substitutions primarily improve answer correctness while also providing consistent reductions in reasoning length. These dynamics further support the complementary roles of the correctness- and length-aware rewards in **EviGRPO**.

D Case Studies

As shown in Fig. 10, both the Base Model and FAST-GRPO produce incorrect answers, while **EviGRPO** correctly identifies square D. In addition, compared with the Base Model, **EviGRPO** reduces the reasoning length from 469 to 217 tokens (252 fewer tokens, 53.7%) and the reasoning latency from 14.06s to 7.34s (6.72s reduction, 47.8%). Compared with FAST-GRPO, it further

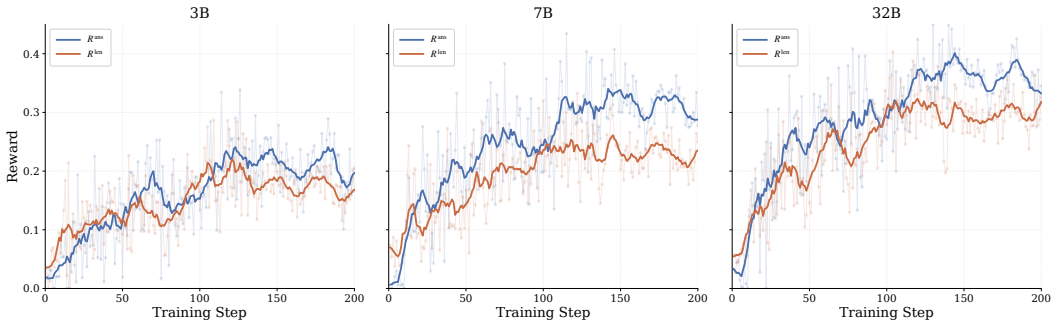


Figure 9: Training dynamics of the relative answer correctness reward R^{ans} and relative reasoning length reward R^{len} across Qwen2.5-VL-3B, 7B, and 32B.

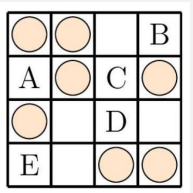
reduces the reasoning length from 263 to 217 tokens (46 fewer tokens, 17.5%) and the latency from 8.84s to 7.34s (1.50s reduction, 17.0%). We further analyze the entropy-trough token substitutions in **`EviGRPO`**. In **`EviGRPO`**, blue tokens denote predictions at the entropy-trough layer, whereas red tokens in parentheses denote the corresponding final-layer predictions. The change from "***check***" to "**preserve**" prevents the model from spending additional reasoning on rows. Replacing "***full***" with "**invalid**" makes the consequence of adding another coin explicit, because the model now treats that action as a violation rather than merely describing the row as occupied. The transition from "***Therefore***" to "**However**" introduces a clearer contrast between the completed rows and the unresolved Row 3, which helps the reasoning focus on the remaining deficiency. When "***by***" is replaced with "**without**", the model no longer follows a path of checking candidates one by one, avoiding an unnecessary expansion of the search process. The final change from "***positions***" to "**total**" shifts the reasoning from local candidate locations back to the global counting constraint, making the unique Row 3–Column 3 intersection easier to identify. These changes collectively reduce unnecessary exploration, prevent the wrong branch from being extended, and guide the model directly to square D with a shorter reasoning trace.

For another case study, as shown in Fig. 11, all three methods obtain the correct answer $4\sqrt{3}/3$, while **`EviGRPO`** reaches the same correct solution with substantially less reasoning. Compared with the Base Model, **`EviGRPO`** reduces the reasoning length from 738 to 338 tokens (400 fewer tokens, 54.2%) and the reasoning latency from 22.20s to 11.22s (10.98s reduction, 49.5%). Compared with FAST-GRPO, it further reduces the reasoning length from 391 to 338 tokens (53 fewer tokens, 13.6%) and the latency from 14.93s to 11.22s (3.71s reduction, 24.8%). In **`EviGRPO`**, using "**area**" at the beginning makes the reasoning focus directly on the target quantity $AC \cdot BD$, instead of exploring broader geometric properties. Choosing "**dividing**" gives a clear operation for the two sine-rule equations and avoids extra algebraic steps. Using "**Since**" directly links the known area $S = 1$ to the diagonal-area formula, while "*However*" may introduce an unnecessary change in reasoning direction. Together, these trough tokens make the reasoning more direct, reduce redundant intermediate steps, and preserve the correct final answer $4\sqrt{3}/3$.

For the failure case study, as shown in Fig. 12, an incorrect entropy-trough token can redirect the reasoning trajectory and cause the subsequent steps to drift progressively away from the correct solution. In this example, replacing the final-layer token "***diameter***" with the trough token "**segments**" changes the model's interpretation of the upper path from one semicircle with diameter AE to several smaller arcs. Once this geometric structure is misinterpreted, the following calculations are conditioned on the wrong representation. The later shift "***two***"→"**three**" further reinforces the error by changing the number of semicircular arcs assumed for the lower path. As a result, the model obtains internally consistent but geometrically incorrect path lengths and finally reaches the wrong ratio. This illustrates that a low-entropy intermediate prediction is not always reliable. An incorrect trough token may introduce an early error and the effect is amplified through autoregressive generation, causing the later reasoning to deviate step by step from the correct trajectory.

E RELATED WORK

Large Vision-Language Models. Reinforcement learning (Schulman et al., 2017) has substantially strengthened the problem-solving abilities of LLMs (Guo et al., 2025b; Team et al., 2025; Shao



Question: In the four squares of a row there always have to be exactly two coins. In the four squares below each other there also always have to be exactly two coins. <image1> On which square does one more coin have to be placed? Options: A. square A; B. square B; C. square C; D.

Base Model: <think>To solve this problem, we need to ensure that in every row and every column, there are exactly two coins. Let's analyze the given grid: 1. Row Analysis: Row 1: There are 2 coins (A and B). Row 2: There is 1 coin (C). Row 3: There is 1 coin (D). Row 4: There are 2 coins (E and F). 2. Column Analysis: Column 1: There is 1 coin (A). Column 2: There are 2 coins (B and C). Column 3: There is 1 coin (D). Column 4: There are 2 coins (E and F). From the analysis, we can see that: Row 2 needs 1 more coin. Row 3 needs 1 more coin. Column 1 needs 1 more coin. Column 3 needs 1 more coin. Since the question asks for the square where one more coin has to be placed, and considering the options provided, the correct choice should be the square that fits the requirement of having exactly two coins in its row and column. Looking at the options: A. square A: This square already has a coin, but it doesn't fit the requirement of having exactly two coins in its row or column. B. square B: This square already has a coin, but it doesn't fit the requirement of having exactly two coins in its row or column. C. square C: This square already has a coin, but it doesn't fit the requirement of having exactly two coins in its row or column. D. square D: This square already has a coin, but it doesn't fit the requirement of having exactly two coins in its row or column. E. square E: This square already has a coin, but it doesn't fit the requirement of having exactly two coins in its row or column. The only option that fits the requirement is square B, as it is the only square that needs an additional coin to meet the condition of having exactly two coins in its row and column. Final Answer: B. square B</think><answer>B. square B</answer>

Thinking Tokens: 469 # Thinking Latency: 14.06s

FAST-GRPO: <think>Let's analyze the given grid step by step: 1. In the first row, there are 2 coins in squares A and B. 2. In the second row, there are 2 coins in squares A and C. 3. In the third row, there are 2 coins in squares C and D. 4. In the fourth row, there are 2 coins in squares D and E. We need to ensure that each column also has exactly 2 coins. Column 1: A (1 coin), E (0 coins). Column 2: A (1 coin), C (1 coin), D (1 coin), E (0 coins). Column 3: B (1 coin), C (1 coin), D (1 coin), E (0 coins). Column 4: B (1 coin), C (1 coin), D (1 coin), E (1 coin). From the above analysis, we can see that: Column 1 needs 1 more coin in square E. Column 2 needs 1 more coin in square E. Column 3 needs 1 more coin in square E. Column 4 needs 1 more coin in square E. Therefore, the square that needs 1 more coin is square E.</think><answer>E. square E</answer>

Thinking Tokens: 263 # Thinking Latency: 8.84s

EviGRPO (Ours): <think>To solve the problem, we first examine the grid. Rows 1, 2, and 4 already have exactly two coins, so we preserve (*check*) these counts, while Row 3 has only one coin and needs one more. Adding a coin to any complete row would make it invalid (*full*). However (*Therefore*), Row 3 remains incomplete. Now check the columns. Columns 1, 2, and 4 already contain two coins, while Column 3 has only one coin and needs another. We can identify the missing square without (*by*) checking every candidate separately. The required total (*positions*) must satisfy both Row 3 and Column 3. Their intersection is square D. Placing a coin in D gives exactly two coins in Row 3 and Column 3 while leaving all other rows and columns unchanged. Therefore, the correct answer is square D.</think><answer>D. square D</answer>

Thinking Tokens: 217 # Thinking Latency: 7.34s

Figure 10: Case study of reasoning processes across the Base Model, FAST-GRPO, and **EviGRPO**. In **EviGRPO**, blue text denotes tokens predicted at the entropy-trough layer, while red text denotes the corresponding original final-layer tokens.

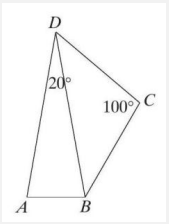
et al., 2024), prompting its adoption in multimodal models (Liu et al., 2025b; Zhang et al., 2025; Yang et al., 2025; Huang et al., 2026; Peng et al., 2025; Wang et al., 2026a; Meng et al., 2025; Zhou et al., 2025; Liu et al., 2025a; Wang et al., 2026b). VisualThinker-R1-Zero (Zhou et al., 2025) and MM-Eureka (Meng et al., 2025) demonstrate that RL can produce extended chains of thought for spatial and mathematical visual tasks. Later studies refine either optimization or data selection: OThink-MR1 (Liu et al., 2025a) dynamically adjusts KL regularization, while ThinkLite-VL (Wang et al., 2026b) selects examples according to estimated difficulty. Vision-R1 (Huang et al., 2026) learns from synthesized rationales and applies progressive thinking suppression, whereas LMM-R1 (Peng

et al., 2025) transfers text-based problem-solving skills to multimodal tasks. Nevertheless, these methods often produce unnecessarily long reasoning sequences, limiting their inference efficiency.

Efficient Reasoning in LVLMs. Existing approaches to efficient reasoning can be broadly divided into reasoning compression and adaptive reasoning. Reasoning compression methods (Han et al., 2025; Luo et al., 2026a; Shen et al., 2025; Renze & Guven, 2024; Team et al., 2025; Yeo et al., 2025) directly shorten reasoning traces through inference-stage constraints or training-stage length objectives. During inference, TALE (Han et al., 2025) specifies token budgets in prompts, whereas CCoT (Renze & Guven, 2024) provides concise in-context demonstrations to elicit shorter solutions. During training, O1-Pruner (Luo et al., 2026a) adopts a tailored RL objective to suppress verbosity, Kimi (Team et al., 2025) introduces length-penalty rewards, DAST (Shen et al., 2025) and CosineReward (Yeo et al., 2025) employ correctness-aware length rewards. In contrast, adaptive reasoning methods (Xiao & Gan, 2026; Liu et al., 2026a; Ma et al., 2025; Chen et al., 2026) dynamically allocate reasoning effort according to problem difficulty. CoT-Value (Ma et al., 2025) learns from reasoning chains of varying lengths, while FAST-GRPO (Xiao & Gan, 2026) and ARES (Chen et al., 2026) encourage deeper reasoning for challenging problems and shorter reasoning for easier ones. Both categories primarily regulate reasoning at the token-sequence level, where aggressive compression may discard reasoning steps required for difficult problems (Han et al., 2025; Luo et al., 2026b), while adaptive methods may over-encourage exploration on challenging instances and still induce overthinking (Liu et al., 2026b; Luo et al., 2026b). In this work, **`EviGRPO`** examines entropy evolution across LLM layers and verifies whether trough-to-final token shifts are sufficiently supported by evidence, thereby correcting unsupported shifts before they propagate into redundant reasoning.

F LIMITATIONS

Despite its effectiveness, **`EviGRPO`** has two main limitations. First, as shown in Fig. 12, predictions from the entropy-trough layer are not always more reliable than those from the final layer, and inaccurate verification may introduce unfavorable token substitutions. Second, the supervision signal for the *evidence verifier* is coarse-grained, since it is derived from group-relative rollout outcomes. When multiple token shifts occur within the same reasoning trajectory, the final outcome may be difficult to precisely attribute to a specific token-level decision.



Question: Let ABCD be a convex quadrilateral with an area of 1 where AB and BD are the bases of two isosceles triangles ADB and BCD respectively (as shown). The product AC · BD is equal to: Options: A. √3/3; B. 2√3/3; C. √3; D. 4√3/3; E. other answer

Base Model: <think>To solve this problem, we need to use the properties of the given isosceles triangles and the area of the quadrilateral. Let's break it down step by step: 1. Identify the angles in the triangles: In triangle ADB, since AD = AB (isosceles), let the base angles be θ. Therefore, ∠ADB = 180° - 2θ. Given ∠ADB = 20°, we have 180° - 2θ = 20°, so 2θ = 160° and θ = 80°. 2. Identify the angles in triangle BCD: Similarly, in triangle BCD, since BC = BD (isosceles), let the base angles be φ. Therefore, ∠BDC = 180° - 2φ. Given ∠BDC = 100°, we have 180° - 2φ = 100°, so 2φ = 80° and φ = 40°. 3. Sum of angles in quadrilateral ABCD: The sum of the interior angles of a quadrilateral is 360°. So, ∠ABC + ∠ADC = 360° - (∠ADB + ∠BDC) = 360° - (20° + 100°) = 240°. 4. Area of the quadrilateral: The area of the quadrilateral can be split into the areas of the two triangles. However, we know the area of the quadrilateral is 1. We can use the fact that the area of a triangle is 1/2 ab sin C, but here we will use the given information directly. 5. Product AC · BD: Using the formula for the area of a quadrilateral with diagonals and included angle, the area K is given by K = 1/2 AC · BD · sin θ, where θ is the angle between the diagonals. Since the area is 1, we have 1 = 1/2 AC · BD · sin 120°. sin 120° = sin 60° = √3/2. Therefore, 1 = 1/2 AC · BD · √3/2, so 1 = √3/4 AC · BD. Solving for AC · BD, we get AC · BD = 4/√3 = 4√3/3. The final answer is 4√3/3.</think><answer>D. 4√3/3</answer>

Thinking Tokens: 738 # Thinking Latency: 22.20s

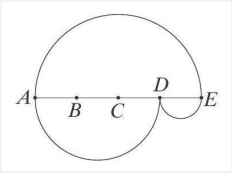
FAST-GRPO: <think>To solve this problem, we need to use the properties of isosceles triangles and the area of the quadrilateral. 1. Since triangle ADB and triangle BCD are isosceles triangles, we can use the given angles to find the angles at the vertices. 2. In triangle ADB, ∠ADB = 20° and ∠DAB = ∠ABD. Since the sum of angles in a triangle is 180°, we have: ∠DAB = ∠ABD = (180° − 20°)/2 = 80°. 3. In triangle BCD, ∠BCD = 100° and ∠DBC = ∠BDC. Since the sum of angles in a triangle is 180°, we have: ∠DBC = ∠BDC = (180° − 100°)/2 = 40°. 4. The angle ∠ABC is the sum of the angles ∠ABD and ∠DBC: ∠ABC = 80° + 40° = 120°. 5. The area of the quadrilateral ABCD can be split into the areas of the two triangles ADB and BCD. The area of a triangle can be calculated using the formula: Area = 1/2 × base × height. 6. Since the area of the quadrilateral is 1, we can use the fact that the product of the lengths of the bases AC and BD is related to the area of the quadrilateral. The product AC · BD is equal to 4 × Area for a specific configuration of the isosceles triangles. Therefore, the product AC · BD is: AC · BD = 4 × 1 = 4. However, the correct answer should be: AC · BD = 4√3/3.</think><answer>D. 4√3/3</answer>

Thinking Tokens: 391 # Thinking Latency: 14.93s

EviGRPO (Ours): <think>To solve this problem, we need to use the area (*properties*) formula for the two diagonals. We need to find the angle between AC and BD. In triangle ADB, AD = BD and ∠ADB = 20°, so the base angles are equal. This gives ∠DAB = ∠ABD = (180° − 20°)/2 = 80°. (2) In triangle BCD, BC = CD and ∠BCD = 100°, so ∠DBC = ∠BDC = (180° − 100°)/2 = 40°. Since BD lies inside ∠ABC, we have ∠ABC = 80° + 40° = 120°. (3) By the sine rule, AB/BD = sin 20°/sin 80° and BC/BD = sin 40°/sin 100°. Because sin 100° = sin 80°, dividing (*Using*) these equations gives AB/BC = sin 20°/sin 40°. Together with ∠ABC = 120°, this gives ∠BAC = 40° and ∠BCA = 20°. (4) Let O be where AC and BD meet. In triangle ABO, ∠ABO = 80° and ∠BAO = 40°, so ∠AOB = 180° − 80° − 40° = 60°. (5) Since (*However*) the area is 1, we can put the values into the diagonal area formula: 1 = 1/2 × AC × BD × sin 60°. Using sin 60° = √3/2 gives 1 = √3/4 × AC × BD. Therefore, AC · BD = 4/√3 = 4√3/3.</think><answer>D. 4√3/3</answer>

Thinking Tokens: 338 # Thinking Latency: 11.22s

Figure 11: Case study of reasoning processes across the Base Model, FAST-GRPO, and **EviGRPO**. In **EviGRPO**, blue text denotes tokens predicted at the entropy-trough layer, while red text denotes the corresponding original final-layer tokens.



Question: The segment AE is divided into four equal parts and semicircles are drawn taking AE, AD and DE as diameters, creating two paths from A to E as shown. Determine the ratio of the length of the upper path to the length of the lower path. Options: A. 1:2; B. 2:3; C. 2:1; D. 3:2; E. 1:1.

Base Model: <think>Let's denote the length of each segment AB, BC, CD, and DE as x. Therefore, the total length of AE is 4x. The upper path consists of the semicircle with diameter AE. The length of this semicircle is half the circumference of a full circle with diameter 4x: Length of upper path = 1/2 × π × 4x = 2πx. The lower path consists of the semicircles with diameters AD and DE. The length of the semicircle with diameter AD is half the circumference of a full circle with diameter 3x: Length of semicircle with diameter AD = 1/2 × π × 3x = 3πx/2. The length of the semicircle with diameter DE is half the circumference of a full circle with diameter x: Length of semicircle with diameter DE = 1/2 × π × x = πx/2. Therefore, the total length of the lower path is: Length of lower path = 3πx/2 + πx/2 = 2πx. Now, we compare the lengths of the upper and lower paths: Ratio of upper path to lower path = 2πx / 2πx = 1. Thus, the ratio of the length of the upper path to the length of the lower path is 1:1.</think><answer>E. 1:1</answer>

Thinking Tokens: 382 # Thinking Latency: 11.68s

EviGRPO (Ours): <think>Let the length of each segment AB, BC, CD, and DE be x, so AE = 4x. The upper path should be interpreted from the small arcs associated with the segments (*diameter*) along AE, so its length is computed from two equal semicircular arcs, each with diameter x, giving πx/2 + πx/2 + πx/2 = 3πx/2. For the lower path, the diagram is then read as containing three (*two*) equal semicircular arcs, each again with diameter x, so its total length becomes πx. Comparing the two paths gives (3πx/2)/πx = 3/2. Therefore, the ratio of the upper path to the lower path is 3:2.</think><answer>D. 3:2</answer>

Thinking Tokens: 211 # Thinking Latency: 7.40s

Figure 12: Failure case of **EviGRPO** reasoning. In **EviGRPO**, blue text denotes tokens predicted at the entropy-trough layer, while red text denotes the corresponding original final-layer tokens, illustrating that incorrect trough-token substitutions can lead to erroneous subsequent reasoning.