# When Models Judge Themselves: Unsupervised Self-Evolution for Multimodal Reasoning

**Anonymous ACL submission**

## Abstract

Recent progress in multimodal large language models has led to strong performance on reasoning tasks, but these improvements largely rely on high quality annotated data, which is costly and difficult to scale. To address this, we propose an unsupervised self-evolution training framework for multimodal reasoning that achieves stable performance improvements without using human-annotated answers or external reward models. For each input, we sample multiple reasoning trajectories and jointly model their within group structure. We use the Actor's self-consistency signal as a training prior, and introduce a bounded Judge based modulation to continuously reweight trajectories of different quality. We further model the modulated scores as a group level distribution and convert absolute scores into relative advantages within each group, enabling more robust policy updates. Trained with Group Relative Policy Optimization (GRPO) on unlabeled data, our method consistently improves reasoning performance and generalization on five mathematical reasoning benchmarks, including MathVision and DynaMath, offering a scalable path toward self-evolving multimodal models.

## 1 Introduction

In recent years, multimodal large language models (MLLMs) have demonstrated remarkable progress in vision–language reasoning tasks. These models have achieved impressive performance on a wide range of benchmarks, including visual mathematical reasoning(Huang et al., 2025b), chart understanding(Tang et al., 2025), and complex scene inference(Liang et al., 2025).

Despite the progress discussed above, recent gains in multimodal reasoning largely rely on high-quality, large-scale training data and strong supervision(Li et al., 2025). This supervision typically comes from carefully annotated answers and reasoning traces(Safaei et al., 2025), or from evaluation models trained on costly preference data(Yasunaga et al., 2025). However, the supply of such supervision is approaching a scale limit. Especially with the growing demand for longer-horizon dependencies(Bai et al., 2024) and broader open-world diversity in reasoning tasks(Conti et al., 2025), building reliable datasets at scale becomes increasingly expensive and difficult to sustain. Under this constraint, researchers have started to explore self-evolving post-training for multimodal large models, aiming to improve reasoning ability while reducing reliance on human annotations and external supervision(Wang et al., 2025b).

MLLM self-evolution aims to further improve a model's reasoning ability using unlabeled or self-synthesized data, without relying on human annotations. However, this paradigm faces a fundamental challenge: training signals generated by the model itself are inherently noisy. Directly applying post-training optimization on top of such signals can therefore lead to unstable training dynamics. To mitigate this issue, existing methods(Wei et al., 2025; Thawakar et al., 2025) often reduce the training signal to a single scalar, for example by using majority voting to form pseudo-labels and updating model parameters accordingly. In practice, this strategy provides a "bootstrapped" approximation of stable supervision: it reduces the noise of a single sample and aligns the training objective with output patterns that are relatively consistent under the current policy distribution. Nevertheless, this simplification comes with clear limitations. As illustrated in Fig 1, high consistency does not necessarily imply high quality; it may instead reflect systematic biases of the model, which can be amplified during long-term training and suppress effective exploration. Moreover, the training signal fails to capture fine-grained differences between candidates and can further trigger response-length collapse. As training proceeds, rewards often concentrate quickly on a few dominant modes, causing

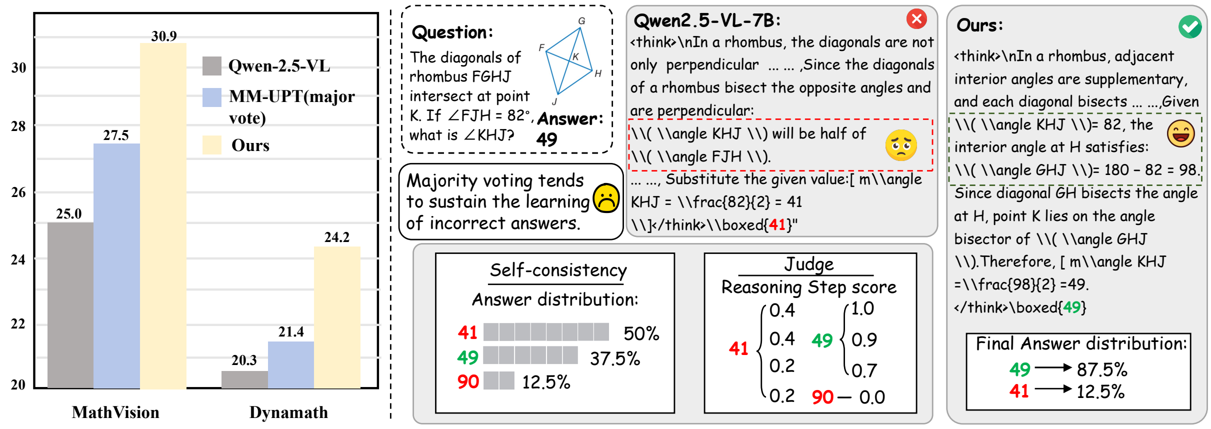


Figure 1: Limitation of majority voting in unsupervised self-evolution.Right: An example where the most frequent answer is incorrect. Majority voting reinforces this dominant error, while our method favors higher-quality reasoning paths through Judge modulation.Left: Results on MathVision(Wang et al., 2024) and DynaMath(Zou et al., 2024) show that our approach consistently outperforms majority-voting-based self-training.

optimization to saturate early and pushing the policy toward a low-entropy output distribution.

In light of these limitations,we argue that stable unsupervised self-evolution requires more than constructing a more accurate pseudo-label. It also calls for explicitly modeling and optimizing the relative structure among candidate solutions for the same input. Motivated by this insight, we propose a self-evolving training framework. Specifically, we instantiate two roles from a single multimodal model: an Actor and a Judge. Given an input, the Actor samples multiple reasoning trajectories, forming the model's current self-consistency distribution. The Judge evaluates each trajectory and maps its output to a bounded, continuously differentiable modulation signal, which reshapes the Actor's initial self-consistency distribution. On the optimization side, we further construct training rewards in a group-wise, distributional manner. For multiple trajectories generated from the same input, we apply an energy-based normalization to compare them relatively, converting absolute scores that are not directly comparable across samples into within-group relative advantages. In this way, training no longer simply amplifies early dominant modes. Instead, our framework can distinguish fine-grained quality differences among reasoning trajectories for the same input and adjust the model's output distribution accordingly, leading to more effective improvements in reasoning ability.

We conduct a series of experiments to analyze the limitations of existing paradigms for modeling training signals. Based on these observations, we further propose and validate a collaborative modeling paradigm. This paradigm leads to more stable training behavior across benchmarks, as reflected by healthier entropy trajectories and reduced response-length collapse. It also delivers more effective performance improvements. For instance, on MathVision(Wang et al., 2024), our unsupervised post-training achieves up to a +5.9 absolute improvement in accuracy (30.9% vs. 25.0%). Importantly, the entire training pipeline does not rely on ground-truth labels, additional metadata, or any external reward model at any stage. In summary, our main contributions are as follows:

1. We propose a new framework for unsupervised post-training of large multimodal models, enabling sustained self-improvement without any external supervision.
2. Through extensive empirical analysis, we identify common failure modes in unsupervised self-evolution and mitigate them by modeling and optimizing the within-input relative structure among candidate solutions.
3. We evaluate our method on multiple mathematical reasoning benchmarks and observe accuracy improvements after multiple iterations under different training data settings.

## 2 Related work

Recent advances show that reinforcement learning (RL) enhances the reasoning ability of large language models (LLMs), with representative systems including DeepSeek-R1(DeepSeek-AI et al., 2025) and Kimi-K1.5(Team et al., 2025). Under appropriately designed optimization objectives, models can gradually acquire more long-term strategic reasoning behaviors. These advances are largely enabled by effective RL-based optimization frameworks, including PPO(Schulman et al., 2017), DPO(Rafailov et al., 2023), GRPO(Shao et al., 2024a), DAPO(Yu et al., 2025), and GSPO(Zheng et al., 2025).

### 2.1 Multi-modal Reasoning

Motivated by the success of verifiable rewards in LLM reasoning, recent studies(Shen et al., 2025) have begun to explore post-training and R1-style reinforcement learning in multimodal settings. Instead of relying on subjective human preferences, these methods(Yang et al., 2025; Huang et al., 2025b) derive reward signals from objectively verifiable signals, enabling more stable reasoning optimization. Empirical results show that when rewards are verifiable or targets are well structured, RL-style post-training leads to stable improvements on multimodal reasoning tasks.

To address this limitation, another line of research(Liu et al., 2025b) explicitly introduces reflection mechanisms to improve robustness during reasoning. For example, VL-Rethinker(Wang et al., 2025a) studies self-reflection in multimodal reasoning and examines the trade-off between reasoning benefits and computational cost. Building on this idea, later work(Cheng et al., 2024; Wang et al., 2025c) integrates reflection into training by using structured reflection steps or learning an explicit critic for evaluation. Despite this progress, effective reasoning post-training still depends on high quality training signals, and designing informative reward functions remains expensive.

### 2.2 Self-Evolving In Large Language Models

Unsupervised self-evolution has been explored to some extent in large language models(Shafayat et al., 2025). A core idea is that, even without ground-truth answers, test-time scaling strategies (e.g., majority voting) can provide useful relative correctness signals(Zuo et al., 2025; Liu et al., 2025a). In parallel, some work(Zhang et al., 2025; Zhao et al., 2025b) uses reinforcement learning with internal feedback, treating model-internal signals (e.g., confidence) as rewards and eliminating the need for external annotations. Absolute Zero(Zhao et al., 2025a) studies a fully data-free setting where the model generates tasks and uses executable checkers to verify answers, providing a self-driven curriculum and RLVR signals. Building on this idea, R-Zero(Huang et al., 2025a) uses a Challenger–Solver co-evolution framework to generate suitable problems.

Recently, self-evolution has also been extended to multimodal large language models. MM-UPT(Wei et al., 2025) uses majority voting over multiple sampled answers to form pseudo-rewards, enabling continual improvement on multimodal reasoning data without ground-truth labels. EvoLMM(Thawakar et al., 2025) uses a Proposer–Solver loop and derives continuous self-rewards from internal consistency signals. VisPlay(He et al., 2025) uses a Questioner–Reasoner role split and applies GRPO with diversity rewards to balance question complexity and answer quality, enabling autonomous evolution from unlabeled images. However, most of these methods use majority voting as the main training signal, which primarily reinforces consistency under the current output distribution. Over long-term training, this can bias the model toward early dominant patterns and limit exploration.

## 3 Method

As shown in Fig. 2, we propose an unsupervised self-evolution framework for multimodal large models. By jointly modeling multiple reasoning trajectories generated from the same input, our approach enables stable and sustained improvements in reasoning ability. Specifically, Sec. 3.1 constructs a consistency-based initial reward for the Actor from repeated rollouts under the same input. Sec. 3.2 introduces a Judge to provide a bounded and continuous modulation of this reward. Finally, Sec. 3.3 models the modulated rewards as a group-wise distribution to support more robust policy updates in the unsupervised setting.

### 3.1 Consistency-Based Initial Reward for the Actor

We consider an unsupervised multimodal reasoning sample consisting of an image–question pair $x = (I, q)$. Given the current policy $\pi_\theta$, we perform $n$ rollouts for the same input $x$, resulting in a set of candidate reasoning trajectories:

$$\mathcal{T}(x) = \{\tau_i\}_{i=1}^{n}, \quad \tau_i \sim \pi_\theta(\cdot \mid x). \tag{1}$$

Each trajectory $\tau_i$ is associated with a final answer $a_i \in \mathcal{A}$, where $\mathcal{A}$ denotes the set of unique answers produced for the input $x$ under the current rollouts. For each answer $a \in \mathcal{A}(x)$, we define its count and

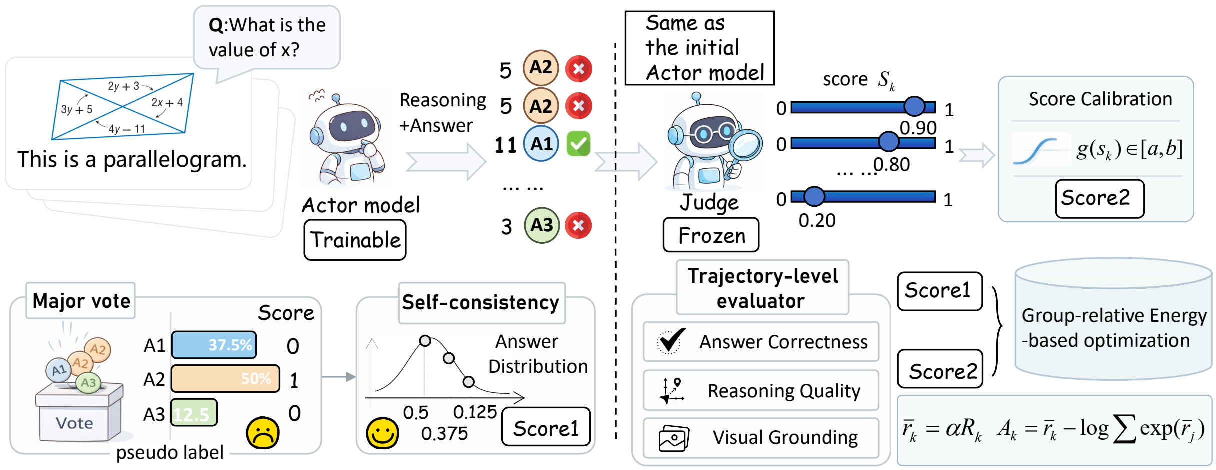


Figure 2: **Overview of the proposed unsupervised self-evolution framework.**The Actor generates multiple reasoning trajectories for the same input, while a frozen Judge provides bounded score modulation. The final rewards are optimized in a group-wise, distributional manner to enable stable policy updates without external supervision.

the corresponding empirical distribution as:

$$c(a) = \sum_{i=1}^{n} \mathbb{I}[a_i = a], \qquad \hat{p}(a) = \frac{c(a)}{n}. \tag{2}$$

We then define the initial reward of each trajectory $\tau_i$ as the empirical frequency of its answer:

$$r_i^{\text{SC}} \triangleq \hat{p}(a_i) = \frac{1}{n} \sum_{j=1}^{n} \mathbb{I}[a_j = a_i]. \tag{3}$$

Under this formulation, when multiple sampled trajectories agree on the same final answer, the corresponding empirical probability $\hat{p}(a)$ becomes larger, and all trajectories associated with that answer receive higher rewards accordingly.

**Consistency-Based Rewards vs. Majority Voting.** Unlike supervised learning, training signals in unsupervised self training are typically generated by the model itself and therefore inevitably contain noise and bias. Applying reinforcement learning based optimization on top of such signals often leads to gradient fluctuations and unstable training. The most common paradigm for unsupervised self training is majority voting, which treats the most frequent answer as the sole training signal. Formally, it selects the majority answer as:

$$a^{\star} = \arg\max_{a \in \mathcal{A}(x)} \hat{p}(a), \tag{4}$$

and assigns a binary reward to each trajectory:

$$r_i^{\text{MV}} = \mathbb{I}[a_i = a^{\star}]. \tag{5}$$

From the perspective of training signals, majority voting is often effective in unsupervised self-training because it provides a simple denoising mechanism. By aggregating multiple samples from the same input, it encourages the learning objective to align with outputs that are more consistent under the policy distribution, thereby reducing the randomness of single-sample supervision. Compared with using raw frequency-based signals, binarized pseudo-labels also offer a clearer optimization direction, making it easier for policy updates to obtain noticeable improvements in the early stage.

However, an answer that becomes dominant early in training does not necessarily correspond to a higher-quality reasoning path. At the same time, the initial answer distribution encodes rich structural information about the model's output behavior, such as the relative proximity between dominant and secondary modes. Majority voting discards this information entirely, retaining only the identity of the most frequent answer.

As a result, once an answer becomes dominant at an early stage, the binary reward further amplifies its advantage, driving the policy distribution toward that mode and suppressing exploration of alternative reasoning trajectories. Over long-term training, this mechanism encourages rapid collapse toward low-entropy, near-deterministic policies. In contrast, consistency-based rewards preserve the relative strength of the empirical distribution, leading to a smoother training signal and better maintaining effective exploration during optimization.

### 3.2 Calibrating Consistency Rewards with a Judge

The initial reward assigned to the Actor primarily reflects the degree of self-consistency under the current policy, rather than directly measuring the quality or correctness of the underlying reasoning. In practice, the model may converge to a pseudo-stable state during training.

To address this issue, we introduce a Judge module that provides a continuous quality signal for each trajectory, serving as a correction to the initial reward. Specifically, at the beginning of training, we initialize the Judge as a structurally identical copy of the current Actor policy and keep its parameters fixed throughout training. The Judge then outputs a raw score for each trajectory by jointly assessing answer correctness, reasoning quality, and visual grounding:

$$s_k = J_\phi(x, \tau_k), \quad s_k \in [0, 1]. \tag{6}$$

Importantly, the Judge score $s_k$ is not used as the final reward directly. Instead, it serves as a modulation factor to reweight the initial reward (see Sec. 3.3). To transform the raw Judge score into a stable and controllable modulation signal, we design a calibration function $g(s)$ that satisfies three desiderata: (1) it is continuously differentiable to support stable optimization; (2) it provides appropriate encouragement for high-scoring trajectories and suppression for low-scoring ones; and (3) it is bounded, preventing Judge noise from being amplified in the unsupervised training loop. Concretely, we adopt:

$$g(s) = 1 + \lambda_+ \, \sigma\Big(\frac{s - t_h}{\tau_h}\Big) - \lambda_- \, \sigma\Big(\frac{t_l - s}{\tau_l}\Big), \tag{7}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $t_h$ and $t_l$ are the high-score and low-score gating thresholds, $\tau_h, \tau_l > 0$ control the smoothness of the gating transitions, and $\lambda_+, \lambda_- > 0$ determine the maximum magnitude of reward amplification and suppression, respectively.

This design incorporates the Judge as a bounded and continuous modulation signal rather than an absolute authority, thereby mitigating pseudo-consistency while avoiding excessive reliance on the Judge's raw scale in the unsupervised training loop. More importantly, this joint modeling makes the training signal adaptive. As the policy distribution evolves, the Judge modulation continuously reshapes the reward signal, preventing optimization from simply locking into the current consensus and enabling ongoing correction during training.

Meanwhile, we also consider a more direct alternative that uses the Judge's raw score $s_k$ as the reward for optimization. This choice often leads to instability in an unsupervised closed loop: since the Judge scores are not comparable in scale across inputs, updates can be dominated by a small number of high-scoring trajectories, causing rapid shifts in the policy distribution. Such shifts can amplify the impact of Judge noise or bias in the loop, eventually resulting in premature convergence.

### 3.3 Distributional Modeling of the Final Reward

For the $k$-th trajectory corresponding to the same input $x$, the final reward is defined as:

$$R_k = r_k \cdot g(s_k) - \lambda_{\text{fmt}} \, \delta_k, \tag{8}$$

where $\delta_k \in \{0, 1\}$ indicates whether the trajectory violates the predefined output format constraints, and $\lambda_{\text{fmt}} = 0.5$ is the corresponding penalty coefficient. We adopt Group Relative Policy Optimization (GRPO)(Shao et al., 2024a) to perform relative optimization over candidate trajectories corresponding to the same input. For a given input $x$, let the reward vector of its $n$ trajectories be $r(x) = [r_1, \ldots, r_n]$. We first apply energy-based scaling to the rewards:

$$\tilde{r}_k = \alpha \, R_k, \tag{9}$$

where $\alpha$ is a temperature parameter. We then define a group-wise log-sum-exp baseline as:

$$b(x) = \log \sum_{j=1}^{n} \exp(\tilde{r}_j), \tag{10}$$

The resulting group-relative advantage is computed as:

$$A_k(x) = \tilde{r}_k - b(x). \tag{11}$$

By modeling the final scores as a group-wise distribution, policy updates no longer collapse rapidly to a deterministic mapping. Instead, the policy is encouraged to gradually shift probability toward better trajectories, while still keeping several reasonable candidates. Moreover, this distributional formulation suppresses the dominance of a small number of abnormally high-scoring trajectories over the gradient, thereby mitigating early collapse driven by extreme values in unsupervised training.

Overall, this group-wise distributional modeling shifts the optimization objective from chasing absolute high scores to continuously reshaping probability mass within each group, leading to more stable policy updates and reducing self-reinforcement of early dominant modes in the unsupervised loop. A more detailed analysis is provided in the appendix A. Finally, the GRPO objective for policy optimization can be written as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{n}\sum_{k=1}^{n} r_k^{\text{clip}} - \beta\, D_{\text{KL}}\Big(\pi_\theta(\cdot \mid x)\,\Big\|\,\pi_{\text{ref}}(\cdot \mid x)\Big)\right] \tag{12}$$

$$r_k^{\text{clip}} = \min\Big(\gamma_k(\theta)\, A_k,\ \text{clip}\big(\gamma_k(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\, A_k\Big), \tag{13}$$

$$\gamma_k(\theta) = \frac{\pi_\theta(\tau_k \mid x)}{\pi_{\theta_{\text{old}}}(\tau_k \mid x)}. \tag{14}$$

Here, the expectation is taken over training inputs $x \sim \mathcal{D}$ and the corresponding trajectories $\{\tau_k\}_{k=1}^{n}$ sampled from the behavior policy $\pi_{\theta_{\text{old}}}(\cdot \mid x)$, $A_k$ denotes the group-relative advantage for the $k$-th trajectory under input $x$, $\gamma_k(\theta)$ is the probability ratio between the current policy and the behavior policy, $\epsilon$ is the clipping threshold, and $\beta$ controls the strength of the KL regularization toward the reference policy $\pi_{\text{ref}}$.

# 4 Experiments

## 4.1 Datasets and Training Details

We train and evaluate our method in the domain of mathematical reasoning. This domain requires models to jointly perform visual perception and multi-step reasoning, and provides a natural setting for evaluating multimodal self-evolving training.

**Training Data.** We use Geometry3k(Lu et al., 2021), GeoQA(Chen et al., 2021), and MMR1(Leng et al., 2025) as training datasets. Notably, during training we only provide the model with raw images and question texts, without access to any ground-truth answers. All experiments are conducted using Qwen2.5-VL-7B-Instruct(Bai et al., 2025) as the backbone model.

**Evaluation Benchmarks.** We evaluate our method on several widely used multimodal mathematical reasoning benchmarks, including MathVision(Wang et al., 2024), MathVerse(Lu et al., 2024), WeMath(Qiao et al., 2025), LogicVista(Xiao et al., 2024), and DynaMath(Zou et al., 2024), following their standard accuracy protocols. These benchmarks cover a wide range of problem types, including geometry, charts, and tables, and span multiple disciplines with diverse visual mathematical challenges. We compare our approach against state-of-the-art multimodal unsupervised self-evolving methods, including VisionZero(Wang et al., 2025b), EvoLMM(Thawakar et al., 2025), and MM-UPT (major-vote)(Wei et al., 2025), as well as supervised training schemes such as SFT(Tong et al., 2024) and RL-based(Shao et al., 2024b) methods.

**Training Setup.** We perform multimodal unsupervised post-training using the Verl framework(Sheng et al., 2024). Specifically, both the actor model and the Judge model are initialized from Qwen2.5-VL-7B-Instruct, with the Judge kept frozen while the actor is trained using GRPO(Shao et al., 2024a) for unsupervised reasoning improvement. Training is conducted on a single node equipped with 8× NVIDIA A800 GPUs (80GB). We set the number of training epochs to 20 and use the AdamW optimizer. For the Judge, the sampling temperature is set to 1.0 with top-$p$ sampling of 0.9. The reward modulation parameters are set to $\lambda_+ = \lambda_- = 0.2$, $t_h = 0.95$, $t_l = 0.40$, and $\tau_h = \tau_l = 1$. For distributional reward modeling, the energy-based scaling coefficient is set to $\alpha = 1$. During actor training, each question is rolled out with 8 trajectories. The KL-divergence constraint coefficient in GRPO is set to $\beta = 0.01$ for training. The learning rate is set to $1 \times 10^{-6}$, with a weight decay of $1 \times 10^{-2}$ and a gradient norm of 1.0.

## 4.2 Experimental Results

**Main Results.** Table 1 summarizes comparisons between our method and three categories of baselines: (1) the Qwen2.5-VL-7B model without training; (2) state-of-the-art multimodal unsupervised self-evolving methods; and (3) supervised training schemes, including SFT and RL (GRPO). All methods are evaluated using the same Qwen2.5-VL-7B backbone under evaluation setting.

Without relying on any human-annotated answers, our method achieves consistent improvements across all three training datasets (MMR1, GeoQA, and Geo3K). Specifically, the average accuracy improves from 34.6 to 37.0 on MMR1 (+2.4), 37.0 on GeoQA (+2.4), and 37.9 on Geo3K (+3.3). The improvements are more pronounced on challenging benchmarks. For example, on Math-

| Training Data | Method | MathVision | MathVerse | WeMath | LogicVista | DynaMath | Avg. |
|---|---|---|---|---|---|---|---|
| *Performance on Qwen2.5-VL-7B* | | | | | | | |
| — | Qwen2.5-VL-7B | 25.0 | 44.2 | 37.1 | 46.3 | 20.3 | 34.6 |
| *Unsupervised self-Evolving* | | | | | | | |
| CLEVR | VisionZero | 27.6 | 46.4 | 38.8 | 48.8 | 21.7 | 36.7 |
| ImgEdit | VisionZero | 27.4 | 46.8 | 38.5 | 49.1 | 21.3 | 36.6 |
| Multi-Bench | EvoLMM | 25.8 | 44.7 | 37.6 | 46.9 | 21.0 | 35.2 |
| MMR1 | +SFT | 27.3 | 45.0 | 38.3 | 48.1 | 22.9 | 36.3 |
| | +RL(GRPO) | 29.3 | 47.4 | 39.3 | 49.4 | 23.3 | 37.7 |
| MMR1 | Major Vote | 26.4 | 46.0 | 38.6 | 47.9 | 21.8 | 36.1 |
| | Ours | **28.4** | **46.4** | **38.8** | **48.6** | **23.0** | **37.0** |
| GeoQA | +SFT | 27.6 | 45.3 | 38.6 | 47.8 | 22.6 | 36.4 |
| | +RL(GRPO) | 28.8 | 47.1 | 39.0 | 49.2 | 23.4 | 37.5 |
| GeoQA | Major Vote | 27.3 | 45.1 | 38.2 | 47.3 | 21.9 | 36.0 |
| | Ours | **28.6** | **46.5** | **38.9** | **47.9** | **23.2** | **37.0** |
| Geo3K | +SFT | 27.8 | 44.7 | 37.9 | 47.2 | 22.1 | 35.9 |
| | +RL(GRPO) | 29.1 | 46.9 | 39.1 | 49.6 | 23.8 | 37.7 |
| Geo3K | Major Vote | 27.5 | 44.0 | 37.4 | 46.9 | 21.4 | 35.4 |
| | Ours | **30.9** | **46.8** | **38.7** | **49.0** | **24.2** | **37.9** |

Table 1: **Main results on multimodal mathematical reasoning benchmarks.** We report accuracy (%) on five math benchmarks. MajorVote corresponds to the MM-UPT method. (tag icon) denotes supervised training. (flask icon) indicates VisionZero-Qwen-7B trained on CLEVR, while (camera icon) denotes VisionZero-Qwen-7B trained on real-world data.

| Method | MathVision | DynaMath | Avg. |
|---|---|---|---|
| Qwen2.5-VL-7B | 25.0 | 20.3 | 22.65 |
| + Major Vote | 27.5 | 21.4 | $24.45_{+1.8}$ |
| + Self-Consistency | 25.2 | 20.5 | $22.85_{+0.2}$ |
| + Judge Scoring | 27.3 | 21.1 | $24.20_{+1.6}$ |
| + MV + JS | 28.4 | 22.7 | $25.55_{+2.9}$ |
| + SC + JS | 30.1 | 23.7 | $26.90_{+4.3}$ |
| + SC + JS (Dist.) | **30.9** | **24.2** | **27.55**$_{+4.9}$ |

Table 2: Ablation experiments on different modules.

| Model / Method | MathVision | LogicVista | DynaMath |
|---|---|---|---|
| Qwen2.5-VL-3B | 19.5 | 39.8 | 18.2 |
| w/ major vote | 21.3 | 40.3 | 19.1 |
| w/ ours | **24.8** | **42.1** | **21.3** |
| InternVL3-8B | 20.8 | 40.7 | 23.6 |
| w/ major vote | 22.8 | 42.1 | 23.7 |
| w/ ours | **25.6** | **44.5** | **26.9** |

Table 3: Self-evolution performance comparison across different models.

| Method | MathVision | DynaMath | Avg. |
|---|---|---|---|
| Qwen2.5-VL-7B | 0.66 | 0.48 | 0.57 |
| GRPO (w. label) | 0.63 | 0.43 | $0.53_{-0.04}$ |
| Major Vote | 0.58 | 0.37 | $0.48_{-0.09}$ |
| Ours | 0.64 | 0.43 | $0.54_{-0.03}$ |

Table 4: Comparison of pass@10 across benchmarks.

Vision, our method achieves an improvement of up to 5.9 when trained on Geo3K (30.9 vs. 25.0). Compared with unsupervised self-evolving methods, our approach consistently outperforms prior work under the same training settings, and in particular surpasses MM-UPT (major-vote) by 2.5 points on Geo3K (37.9 vs. 35.4). Moreover, our method achieves performance comparable to supervised SFT and RL (GRPO) methods, and even outperforms them in some settings.

Figure 3 compares the training dynamics of different strategies on MMR1. Majority voting rapidly amplifies early dominant answers, leading to a sharp reduction in policy entropy. It also encourages the model to obtain stable rewards through shorter and more template-like responses. In contrast, our method avoids repeatedly reinforcing early dominant patterns and maintains sufficient exploration during training, resulting in more stable training behavior and more consistent performance improvements over time.

**Ablation Study.** Table 2 reports ablation results for the key components, while Fig. 4 further il-

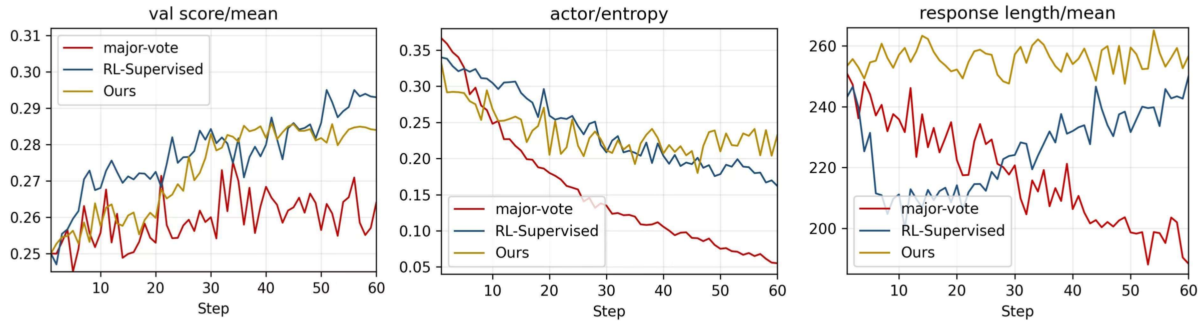


Figure 3: **Training dynamics on MMR1(Leng et al., 2025).**The figure compares majority voting, supervised reinforcement learning, and our method during training, in terms of validation accuracy on MathVision, actor entropy, and average response length.


Figure 4: **Ablation training dynamics on MMR1(Leng et al., 2025).**We compare Self-Consistency, Judge-only, and the full method in terms of validation accuracy on MathVision, actor entropy, and average response length during training.

lustrates their training dynamics. When trained on MMR1, using Self-Consistency alone can retain some output diversity, but it leads to limited improvement on MathVision because it cannot reliably distinguish between highly consistent and low-quality trajectories. In contrast, the Judge-only variant updates the policy based directly on evaluation scores. Although this introduces a quality signal, it ignores the Actor's candidate distribution within each input, making the updates more likely to be dominated by a small number of high-scoring trajectories. This leads to unstable training behavior and a noticeable increase and fluctuation in response length. This behavior is consistent with the model favoring longer and more template-like responses that receive higher scores, rather than gradually correcting its reasoning through diverse trajectories. Overall, our method achieves the best performance by continuously redistributing probability mass and correcting errors within the candidate set for each input, which helps prevent self-reinforcement of early dominant patterns and reduces training instability.

**Generalization Across Backbones and Evaluation Metrics (pass@10).** As shown in Table 3, applying our method to multiple backbone models(Zhu et al., 2025) of different scales consistently improves average performance. As shown in Table 4, our method consistently outperforms majority voting in pass@10 across benchmarks. However, compared to the base model, it leads to a certain reduction in exploration. Further analysis and experiments on the relationship between the Judge and self-consistency are provided in Appendix G.

## 5 Conclusion

We propose an unsupervised self-evolution training framework for multimodal large models. By jointly modeling multiple reasoning trajectories from the same input, our method leverages the Actor's self-consistency signal and a Judge-based modulation. It further applies group-wise distributional reward modeling to reduce mode collapse during long-term training. Experiments on multiple mathematical reasoning benchmarks show that our approach achieves stable and consistent performance improvements.

## Limitations

However, this work primarily investigates the construction of stable training signals, and leaves the question of how to further improve the self-evolving system beyond the Judge's capability limit to future study. To enable sustained unsupervised self-evolution, the Judge should be able to progressively raise its evaluation standards as training proceeds, and autonomously determine when it should be updated to remain a reliable training signal.

## 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. *ArXiv*, abs/2502.13923.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *ArXiv*, abs/2412.15204.

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric P. Xing, and Liang Lin. 2021. Geoqa: A geometric question answering benchmark towards multimodal numerical reasoning. *ArXiv*, abs/2105.14517.

Kanzhi Cheng, Yantao Li, Fangzhi Xu, Jianbing Zhang, Hao Zhou, and Yang Liu. 2024. Vision-language models can self-improve reasoning via reflection. In *North American Chapter of the Association for Computational Linguistics*.

Alessandro Conti, Massimiliano Mancini, Enrico Fini, Yiming Wang, Paolo Rota, and Elisa Ricci. 2025. On large multimodal models as open-world image classifiers. *ArXiv*, abs/2503.21851.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, and et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *ArXiv*, abs/2501.12948.

Yicheng He, Chengsong Huang, Zongxia Li, Jiaxin Huang, and Yonghui Yang. 2025. Visplay: Self-evolving vision-language models from images.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. 2025a. R-zero: Self-evolving reasoning llm from zero data. *ArXiv*, abs/2508.05004.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaoshen Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Yao Hu, and Shaohui Lin. 2025b. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *ArXiv*, abs/2503.06749.

Sicong Leng, Jing Wang, Jiaxi Li, Hao Zhang, Zhiqiang Hu, Boqiang Zhang, Yuming Jiang, Hang Zhang, Xin Li, Li Bing, Deli Zhao, Wei Lu, Yu Rong, Aixin Sun, and Shijian Lu. 2025. Mmr1: Enhancing multimodal reasoning with variance-aware sampling and open resources. *ArXiv*, abs/2509.21268.

Yunxin Li, Zhenyu Liu, Zitao Li, Xuanyu Zhang, Zhenran Xu, Xinyu Chen, Haoyuan Shi, Shenyuan Jiang, Xintong Wang, Jifang Wang, Shouzheng Huang, Xinping Zhao, Borui Jiang, Lanqing Hong, Longyue Wang, Zhuotao Tian, Baoxing Huai, Wenhan Luo, Weihua Luo, and 3 others. 2025. Perception, reason, think, and plan: A survey on large multimodal reasoning models. *ArXiv*, abs/2505.04921.

Dayong Liang, Changmeng Zheng, Zhiyuan Wen, Yi Cai, Xiao Wei, and Qing Li. 2025. Seeing beyond the scene: Enhancing vision-language models with interactional reasoning. *ArXiv*, abs/2505.09118.

Jia Liu, ChangYi He, YingQiao Lin, MingMin Yang, FeiYang Shen, and Shaoguo Liu. 2025a. Ettrl: Balancing exploration and exploitation in llm test-time reinforcement learning via entropy mechanism. *ArXiv*, abs/2508.11356.

Shuhang Liu, Zhenrong Zhang, Pengfei Hu, Jie Ma, Jun Du, Qing Wang, Jianshu Zhang, Quan Liu, Jianqing Gao, and Feng Ma. 2025b. Mmc: Iterative refinement of vlm reasoning via mcts-based multimodal critique. *Proceedings of the 3rd International Workshop on Large Generative Models Meet Multimodal Applications*.

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. In *International Conference on Learning Representations (ICLR)*.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. *arXiv preprint arXiv:2105.04165*.

Runqi Qiao, Qiuna Tan, Guanting Dong, MinhuiWu MinhuiWu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma GongQue, Shanglin Lei, Yifan Zhang, and 1 others. 2025. 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)*, pages 20023–20070.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea

Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *ArXiv*, abs/2305.18290.

Bardia Safaei, Faizan Siddiqui, Jiacong Xu, Vishal M. Patel, and Shao-Yuan Lo. 2025. Filter images first, generate instructions later: Pre-instruction data selection for visual instruction tuning. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14247–14256.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

Sheikh Shafayat, Fahim Tajwar, Ruslan Salakhutdinov, Jeff Schneider, and Andrea Zanette. 2025. Can large reasoning models self-train? *ArXiv*, abs/2505.21444.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. 2024a. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024b. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, Ruochen Xu, and Tiancheng Zhao. 2025. Vlm-r1: A stable and generalizable r1-style large vision-language model. *ArXiv*, abs/2504.07615.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Liyan Tang, Grace Kim, Xinyu Zhao, Thom Lake, Wenxuan Ding, Fangcong Yin, Prasann Singhal, Manya Wadhwa, Zeyu Leo Liu, Zayne Sprague, Ramya Namuduri, Bodun Hu, Juan Diego Rodriguez, Puyuan Peng, and Greg Durrett. 2025. Chartmuseum: Testing visual reasoning capabilities of large vision-language models. *ArXiv*, abs/2505.13444.

Kimi Team, Angang Du, Bofei Gao, and et al. 2025. Kimi k1.5: Scaling reinforcement learning with llms. *ArXiv*, abs/2501.12599.

Omkat Thawakar, Shravan Venkatraman, Ritesh Thawkar, Abdelrahman M. Shaker, Hisham Cholakkal, Rao Muhammad Anwer, Salman H. Khan, and Fahad Shahbaz Khan. 2025. Evolmm: Self-evolving large multimodal models with continuous rewards.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *Advances in Neural Information Processing Systems*, 37:7821–7846.

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. *ArXiv*, abs/2504.08837.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. 2024. Measuring multimodal mathematical reasoning with math-vision dataset. *ArXiv*, abs/2402.14804.

Qinsi Wang, Bo Liu, Tianyi Zhou, Jing Shi, Yueqian Lin, Yiran Chen, Hai Li, Kun Wan, and Wentian Zhao. 2025b. Vision-zero: Scalable vlm self-improvement via strategic gamified self-play. *ArXiv*, abs/2509.25541.

Xiyao Wang, Chunyuan Li, Jianwei Yang, Kai Zhang, Bo Liu (Benjamin Liu), Tianyi Xiong, and Furong Huang. 2025c. Llava-critic-r1: Your critic model is secretly a strong policy model. *ArXiv*, abs/2509.00676.

Lai Wei, Yuting Li, Chen Wang, Yue Wang, Linghe Kong, Weiran Huang, and Lichao Sun. 2025. First sft, second rl, third upt: Continual improving multi-modal llm reasoning via unsupervised post-training.

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

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. *ArXiv*, abs/2503.10615.

Michihiro Yasunaga, Luke S. Zettlemoyer, and Marjan Ghazvininejad. 2025. Multimodal rewardbench: Holistic evaluation of reward models for vision language models. *ArXiv*, abs/2502.14191.

Qiying Yu, Zheng Zhang, Ruofei Zhu, and et al. 2025. Dapo: An open-source llm reinforcement learning system at scale. *ArXiv*, abs/2503.14476.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and 1 others. 2024. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? *arXiv preprint arXiv:2403.14624*.

Yanzhi Zhang, Zhaoxi Zhang, Haoxiang Guan, Yilin Cheng, Yitong Duan, Chen Wang, Yue Wang, Shuxin Zheng, and Jiyan He. 2025. No free lunch: Rethinking internal feedback for llm reasoning. *ArXiv*, abs/2506.17219.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025a. Absolute zero: Reinforced self-play reasoning with zero data. *ArXiv*, abs/2505.03335.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Xiaodong Song. 2025b. Learning to reason without external rewards. *ArXiv*, abs/2505.19590.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization. *ArXiv*, abs/2507.18071.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Yue Cao, Yangzhou Liu, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Han Lv, and 29 others. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *ArXiv*, abs/2504.10479.

Chengke Zou, Xing ming Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. 2024. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. *ArXiv*, abs/2411.00836.

Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Yuchen Zhang, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. 2025. Ttrl: Test-time reinforcement learning. *ArXiv*, abs/2504.16084.

## A Why Group-wise Distributional Modeling Prevents Policy Collapse

For each input $x$, we sample a set of candidate trajectories $\mathcal{T}(x) = \{\tau_1, \dots, \tau_n\}$ from the behavior policy $\pi_{\theta_{\text{old}}}(\cdot \mid x)$. Each trajectory $\tau_k$ is assigned a final scalar reward $r_k \equiv R(\tau_k, x)$, and we apply an energy scaling $\tilde{r}_k = \alpha r_k$, where $\alpha$ is a temperature parameter. We then introduce a group-wise log-sum-exp baseline

$$b(x) = \log \sum_{j=1}^{n} \exp(\tilde{r}_j),$$

and define the group-relative advantage

$$A_k(x) = \tilde{r}_k - b(x).$$

This construction implicitly induces a target distribution over the candidate set $\mathcal{T}(x)$:

$$q_\alpha(\tau_k \mid x) \triangleq \frac{\exp(\alpha r_k)}{\sum_{j=1}^{n} \exp(\alpha r_j)}. \tag{A.1}$$

It follows immediately that

$$\log q_\alpha(\tau_k \mid x) = \alpha r_k - \log \sum_{j=1}^{n} \exp(\alpha r_j) = A_k(x), \tag{A.2}$$

which shows that the group-relative advantage equals the log-probability of $\tau_k$ under the reward-induced distribution $q_\alpha(\cdot \mid x)$.

To clarify the learning objective implied by this distributional modeling, we consider the case where clipping is ignored, and we temporarily omit the KL regularization term. In this setting, the dominant gradient term can be written as a policy-gradient form under samples from the behavior policy:

$$\nabla_\theta \mathcal{J}(\theta) \propto \mathbb{E}_{x\sim\mathcal{D},\, \tau\sim\pi_{\theta_{\text{old}}}(\cdot|x)} \big[A(\tau, x)\, \nabla_\theta \log \pi_\theta(\tau \mid x)\big]. \tag{A.3}$$

Substituting Eq. (A.2), we obtain

$$\nabla_\theta \mathcal{J}(\theta) \propto \mathbb{E}_{x,\, \tau\sim\pi_{\theta_{\text{old}}}} \big[\log q_\alpha(\tau \mid x)\, \nabla_\theta \log \pi_\theta(\tau \mid x)\big]. \tag{A.4}$$

This form suggests that the update is naturally described by matching the policy to the target distribution $q_\alpha(\cdot \mid x)$ defined on the candidate set.

Concretely, consider the following distribution-matching objective over $\mathcal{T}(x)$:

$$\max_\theta \; \mathbb{E}_{x\sim\mathcal{D}} \left[ \sum_{k=1}^{n} q_\alpha(\tau_k \mid x)\, \log \pi_\theta(\tau_k \mid x) \right]. \tag{A.5}$$

This objective is equivalent to minimizing the KL divergence on the candidate set:

$$\min_\theta \; \mathbb{E}_{x\sim\mathcal{D}} \Big[ D_{\text{KL}}\big(q_\alpha(\cdot \mid x) \,\|\, \pi_\theta(\cdot \mid x)\big) \Big], \tag{A.6}$$

since for any fixed $x$,

$$D_{\text{KL}}(q \| \pi) = \sum_k q_k \log q_k - \sum_k q_k \log \pi_k. \tag{A.7}$$

Therefore, maximizing Eq. (A.5) is equivalent to minimizing Eq. (A.6). By the basic property of KL divergence, the optimum of Eq. (A.6) satisfies

$$\pi_\theta(\cdot \mid x) = q_\alpha(\cdot \mid x) \qquad \text{on } \mathcal{T}(x). \tag{A.8}$$

This shows that distributional modeling changes the learning target from selecting a single candidate to matching a soft distribution over the candidate set, and the optimal policy approaches the reward-induced distribution $q_\alpha(\cdot \mid x)$.

The sharpness of $q_\alpha(\cdot \mid x)$ is controlled by the temperature $\alpha$ and the reward gaps within the group.

When $\alpha \to \infty$, or when one candidate has a much larger reward than the others, i.e., $r_{k^\star} \gg r_j$ for all $j \neq k^\star$, the target distribution degenerates to:

$$q_\alpha(\tau_{k^\star} \mid x) \to 1, \qquad q_\alpha(\tau_j \mid x) \to 0 \ (j \neq k^\star). \tag{A.9}$$

In this case, the optimal policy in Eq. (A.8) also becomes deterministic. In contrast, when multiple candidates have comparable rewards and no single trajectory dominates, $q_\alpha(\cdot \mid x)$ remains non-degenerate and assigns non-zero probability mass to several candidates. Eq. (A.8) then implies that learning favors a gradual reallocation of probability mass within each group, rather than an immediate collapse to a single mode.

For comparison, a one-hot target distribution over the candidate set,

$$q_{\mathrm{OH}}(\tau_k \mid x) = \mathbb{I}[k = k^\star], \tag{A.10}$$

leads to an optimal solution $\pi_\theta(\tau_{k^\star} \mid x) = 1$ when minimizing $D_{\mathrm{KL}}(q_{\mathrm{OH}} \| \pi_\theta)$. This objective directly encourages a deterministic mapping. By instead using the energy-normalized distribution $q_\alpha(\cdot \mid x)$, the learning target remains a soft distribution as long as $q_\alpha$ is non-degenerate. As a result, policy updates can be interpreted as continuously reshaping probability mass within each group, rather than concentrating all mass on a single candidate in early training.

# B Training Algorithm

This algorithm 1 presents the pseudocode of our unsupervised self-evolution training algorithm. The algorithm summarizes the overall training procedure, including multi trajectory sampling, self consistency based reward initialization, Judge based score modulation, group wise distributional reward shaping, and GRPO based policy optimization. It is provided for completeness and to facilitate reproducibility.

# C Experimental Setup and Baselines

## C.1 Training Data

Geo3K(Lu et al., 2021) is a geometry problem-solving dataset designed to support diagram-grounded symbolic reasoning. It contains 3,002 multiple-choice geometry questions, each paired with a natural language problem statement, a corresponding geometry diagram, and a ground-truth answer. Geo3K is split into 2,101 / 300 / 601 examples for training, validation, and testing, respectively. The original paper also reports basic statistics for each split, such as the numbers of sentences and words, as well as the scale of formal semantic annotations, indicating the dataset's semantic and reasoning complexity.

GeoQA(Chen et al., 2021) is a benchmark for multimodal numerical reasoning in planar geometry. Each example is a geometry problem collected from real exam or practice question banks, where the model must jointly interpret the problem text and the accompanying diagram. The original GeoQA dataset contains 5,010 problems and follows a 7:1.5:1.5 split for training, validation, and test sets. GeoQA further groups problems into three categories: angle computation, length computation, and others (e.g., area-related problems).

The MMR1 training data(Leng et al., 2025) is organized into two parts: a cold-start set with long chain-of-thought (CoT) annotations and an RL-stage set of question–answer (QA) pairs. This design aims to cover both multimodal mathematical and logical reasoning, with an emphasis on data quality, difficulty, and diversity. In our experiments, we use the RL-stage QA split as the unlabeled training set for unsupervised self-evolution.

## C.2 Baselines

Vision-Zero(Wang et al., 2025b) proposes a zero-human-in-the-loop framework for improving vision–language models (VLMs). Instead of relying on manually curated QA pairs or preference data, it formulates training as a self-play game based on visual differences, inspired by the "Who Is the Spy" setting. The key idea is that, given a pair of slightly different images (an original image and its edited counterpart), the model can generate training signals through multi-round interactions and optimize with verifiable rewards, thereby improving visual reasoning and strategic inference. In our comparisons, we use two Vision-Zero variants built on Qwen2.5-VL-7B: one trained with image pairs constructed from synthetic CLEVR scenes, and the other trained with real-world edited image pairs from datasets such as ImgEdit.

EvoLMM(Thawakar et al., 2025) trains using only raw images, without any human-annotated QA pairs or metadata. It improves multimodal reasoning through a closed-loop process where the model generates questions, produces answers, and derives rewards from its own outputs. To build the training pool, EvoLMM samples about 1,000 images from each of several widely used visual reasoning, chart, and geometry datasets, resulting in roughly 6k training images. The method adopts a Proposer–Solver framework, where the Proposer generates questions and the Solver answers them to drive self-evolution.

### C.3 Evaluation Benchmarks

MathVision(Wang et al., 2024) is a benchmark designed to evaluate large models' mathematical reasoning under visual context. The authors argue that existing visual math benchmarks, while broad, are still limited in problem diversity and subject coverage; MathVision is therefore introduced to provide a more comprehensive assessment of vision-based mathematical reasoning. Following common practice, we evaluate both our method and all baselines on the official testmini split, which contains 304 problems.

MathVerse(Zhang et al., 2024) is designed not only to measure final-answer accuracy, but also to diagnose whether multimodal large models truly use diagram information in visual math problems. The authors note that many existing benchmarks include substantial textual cues that duplicate the visual content, allowing models to answer correctly with little reliance on the image and thus overestimating genuine visual understanding. MathVerse covers three major areas—Plane Geometry, Solid Geometry, and Functions—and provides 12 fine-grained categories for capability-based evaluation. In our experiments, we evaluate on the MathVerse-mini split.

WeMath(Qiao et al., 2025) is motivated by the observation that most visual math benchmarks focus on final-answer accuracy, but provide limited insight into a model's underlying weaknesses in mathematical knowledge and generalization. It therefore introduces a textbook concept–centered evaluation framework to distinguish whether errors arise from missing knowledge or from failure to compose and generalize known concepts. WeMath categorizes model behaviors into four diagnostic types: IK (Insufficient Knowledge), IG (Inadequate Generalization), CM (Complete Mastery), and RM (Rote Memorization). We evaluate both our method and all baselines on the full WeMath benchmark, which contains 6.5K visual math problems.

LogicVista(Xiao et al., 2024) is a benchmark designed to evaluate logical reasoning of multimodal large language models (MLLMs) under visual context. The authors argue that existing multimodal evaluations often focus on perception and understanding, or emphasize math-oriented reasoning, while providing limited coverage of more general logical reasoning skills needed for tasks such as navigation and pattern inference. LogicVista consists of 448 multiple choice questions. It groups logical reasoning into five skill categories: inductive reasoning, deductive reasoning, numerical reasoning, spatial reasoning, and mechanical reasoning. We evaluate on the full LogicVista benchmark.

DynaMath(Zou et al., 2024) focuses on the robustness of mathematical reasoning. Unlike most existing visual math benchmarks that are static, it is designed to systematically test whether the same reasoning process remains valid under varying conditions. To this end, DynaMath evaluates generalization by dynamically generating problem variants. The benchmark contains 501 high-quality seed questions spanning multiple topics, and each seed is instantiated into 10 variants, resulting in 5,010 instances in total. We evaluate on the full DynaMath benchmark.

## D Prompt Design

This section presents the Judge prompt and the Actor prompt used during training to support reward modeling and output-format constraints in our unsupervised self-evolution framework.

The Judge prompt consists of two complementary components. The first component specifies the scoring and decision rules, which evaluate each candidate solution trajectory along three dimensions: answer correctness, reasoning quality, and visual grounding. The second component enforces a strict output format to standardize the Judge's scores and ensure stable reward signals. We use the combination of these two components as the full Judge prompt.

The Actor prompt is not designed to elicit better reasoning content. Instead, it enforces a unified output format. This constraint helps avoid ambiguous formatting, missing answers, or multiple final answers, and prevents the Judge from producing unstable or exploitable scores due to malformed outputs during reward modeling. Importantly, these prompts are only used in the training stage for self-evolution. For all evaluations, we use the original, official prompting setup of each model and keep the evaluation protocol identical across all methods.

**Judge Prompt**

*You are an expert evaluator for multimodal mathematical reasoning.*
All scores must be real numbers in $[0, 1]$.
**Evaluation Dimensions:**
**1. answer_correctness**

- 1.0: fully correct
- 0.5–0.9: almost correct
- 0.1–0.4: partial progress
- 0.0: wrong or missing

**2. reasoning_quality**

- 1.0: rigorous and logical
- 0.5–0.9: minor gaps
- 0.1–0.4: fragmented reasoning
- 0.0: invalid reasoning

**3. visual_grounding**

- 1.0: correct use of visual elements
- 0.5–0.9: minor misreadings
- 0.1–0.4: partial or incorrect usage
- 0.0: no grounding or hallucination

**Overall Score**

- answer_correctness: 0.50
- reasoning_quality: 0.30
- visual_grounding: 0.20

**Mandatory Validity Rule**
If the candidate solution fails to provide a clear and single final answer in the required format, **all scores are set to `0.0`**.

**Judge Output Format**

```
[BEGIN THOUGHT]
<your analysis>
[END THOUGHT]

[BEGIN SCORES]
{
  "answer_correctness": <float>,
  "reasoning_quality": <float>,
  "visual_grounding": <float>,
  "overall_score": <float>
}
[END SCORES]
```

Only valid JSON enclosed in the above tags is accepted.

**Actor Prompt**

You must first think through the reasoning process as an internal monologue. The entire reasoning process must be fully enclosed within a matching `<think>...</think>` pair, with no text outside these tags.
After the closing `</think>` tag, you must output the final answer in the exact format `\boxed{ANSWER}`.
The `\boxed{}` must contain only the final answer value, with no additional words, steps, symbols, units, or punctuation.
If the required format is not followed **exactly**, the answer is considered invalid.

## E Hyperparameters

This section summarizes the main hyperparameters and system configurations used in our unsupervised self-evolution training, to support reproducibility and clarify implementation details.Detailed settings are provided in Table 5.

## F Case study

In this section, we present qualitative case studies to illustrate the behavior of our method and analyze representative failure cases. Through these examples, we provide deeper insights into how the proposed framework influences model reasoning and where its limitations remain. Figures 5 and 6 compare the behaviors of our method and the majority-voting strategy during training. Majority voting essentially compresses the model's output distribution into a near-deterministic mapping: the model updates only with respect to the most frequent answer, while ignoring the relative structure among alternative candidates for the same question. In the early stage of training, even a slight frequency advantage of an incorrect answer over the correct one can be amplified over subsequent iterations. As the incorrect answer becomes dominant, its pseudo-label signal is repeatedly reinforced, eventually driving the model into a self-reinforcing but incorrect learning trajectory. In contrast, our method exhibits more stable learning behavior overall.

Figure 7 presents a representative failure case. In some cases, the training signal can still be misled when the model's self-consistency distribution is already strongly biased toward an incorrect answer and the frozen Judge also assigns a relatively high score to that answer. Under this "incorrect consensus" scenario, the group-relative reward modeling will continue to favor the wrong trajectory, causing the model to update in an undesired direction. Moreover, once such incorrect consistency is solidified, the output distribution becomes sharper and exploration is reduced. This effect can partially explain the drop in pass@10 reported in Table 4: with lower sampling diversity, the probability of reaching the correct solution across multiple attempts decreases.

## G Experiments and Analysis

**Analysis of pass@10.** Table 4 reports pass@10 results of different methods across multiple benchmarks. Note that pass@10 is highly sensitive to sampling diversity, as it measures the probability of obtaining at least one correct solution among multiple samples. Our method improves training stability through group-relative reward modeling and Judge-based modulation, reducing the random amplification of incorrect trajectories, but it can also make the output distribution more concentrated. In some cases, especially when both the model's self-consistency signal and the Judge score favor the same incorrect answer, the policy may develop an incorrect consensus, which further reduces sampling diversity. Such distributional contraction weakens the coverage benefit of multiple attempts and is one of the main reasons for the decrease in pass@10.

**Effect of Distributional Modeling.** Table 6 analyzes the effect of group-wise distributional modeling under different designs. Across self-consistency, Judge-only, and our final formulation, incorporating distributional modeling consistently leads to improved performance on both benchmarks. This indicates that distributional modeling serves as a general and stable refinement step, converting raw trajectory-level signals into relative group-wise advantages, which helps smooth the training signal and leads to more stable optimization.

**Relationship Between Self-Consistency and Judge Preferences.** The results are presented in Table 7. Agree@1 measures the fraction of questions for which the top-1 choice under self-consistency (i.e., the answer with the highest self-consistency) matches the top-1 answer selected by the Judge (i.e., the answer with the highest average score), reflecting the alignment between the two signals over the candidate set. SC-winner denotes the answer accuracy when selecting solely based on self-consistency, serving as a proxy for the reliability of self-consistency as a distributional prior. J-winner denotes the answer accuracy when selecting solely based on the Judge. For evaluation, we sample a batch from Geo3K and, for each input, roll out 8 trajectories using both the base model and the final model.

As shown in Table 7, the top-1 agreement between self-consistency and the Judge increases noticeably after training while remaining unsaturated. This suggests that the model's output distribution gradually shifts from being partially driven by incidental frequency patterns to being more aligned with reasoning quality. At the same time, the agreement remaining below saturation indicates that the policy does not collapse into a deterministic mapping dominated by a single signal, and still preserves room for exploration.

Moreover, the accuracy of the trajectories selected by the Judge improves, even though the Judge is not trained. This implies that the Judge guidance encourages the Actor to produce higher quality reasoning trajectories more consistently. Overall, these results highlight the synergy between self-consistency and Judge-based modulation: self-consistency provides a stable distributional prior, while the Judge steers learning toward higher quality candidates under ambiguity. They form a positive feedback loop that improves trajectory quality while mitigating distribution collapse. We present the full evaluation results of unsupervised train-

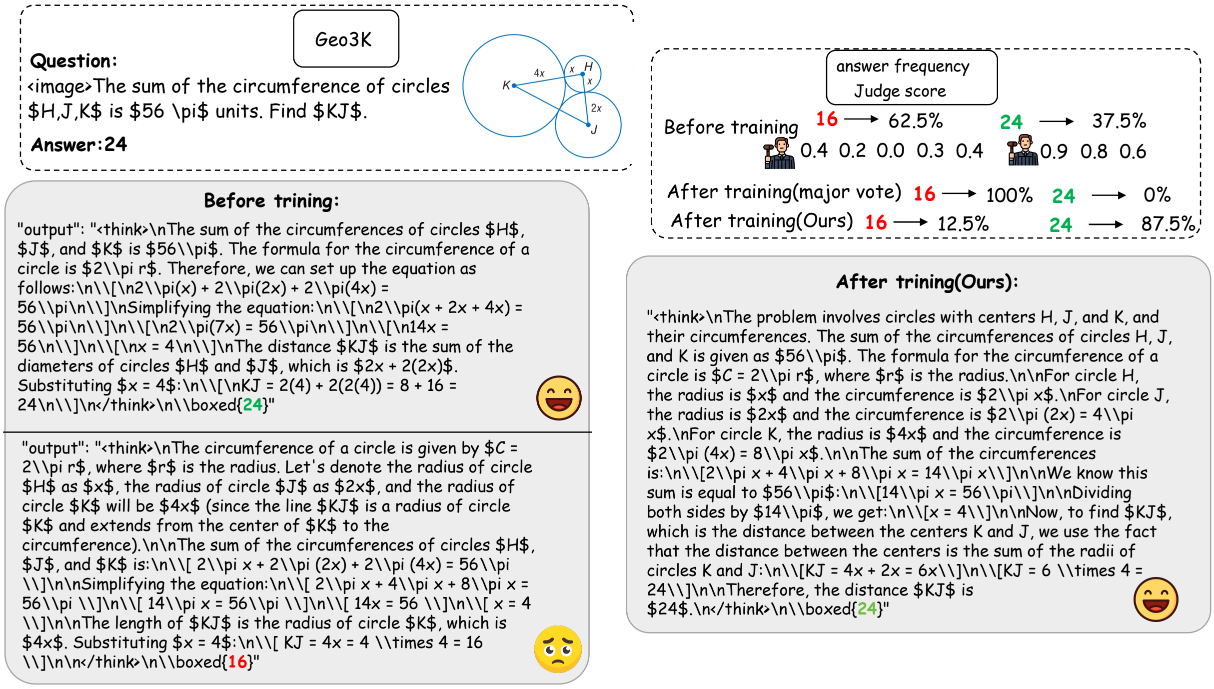


Figure 5: A case study on Geo3K(Lu et al., 2021)

ing on Geo3K across multiple benchmarks in Tables 8, 9, 10, 11.

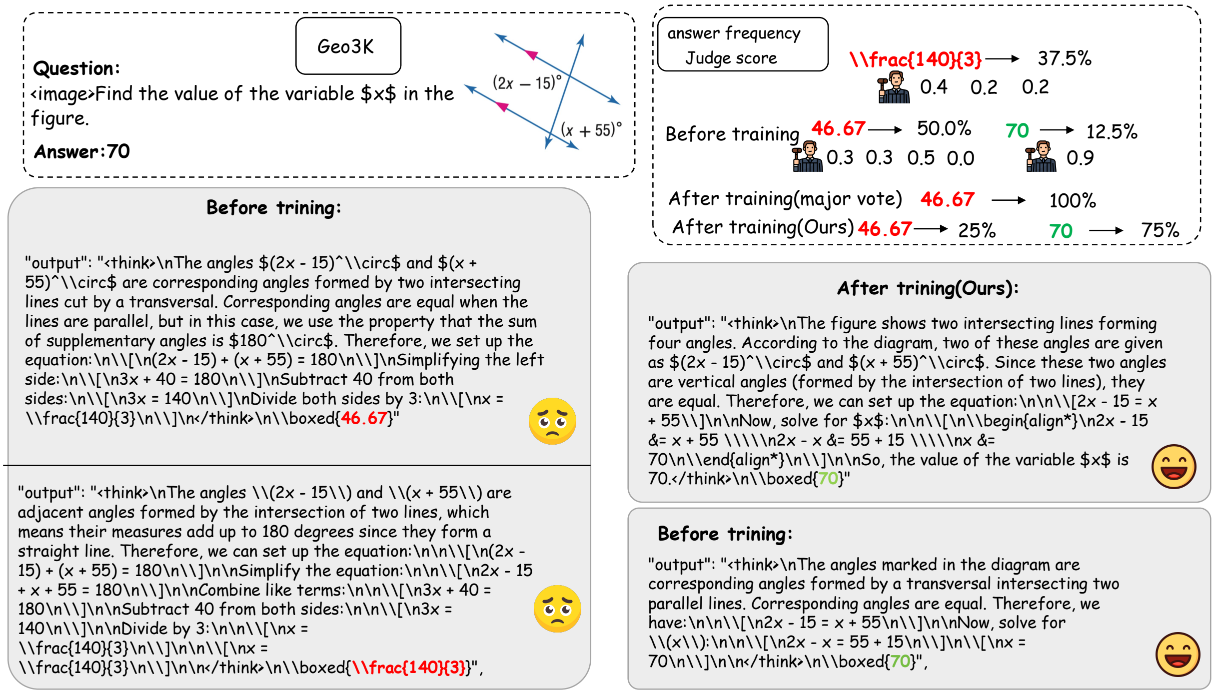


Figure 6: A case study on Geo3K(Lu et al., 2021)

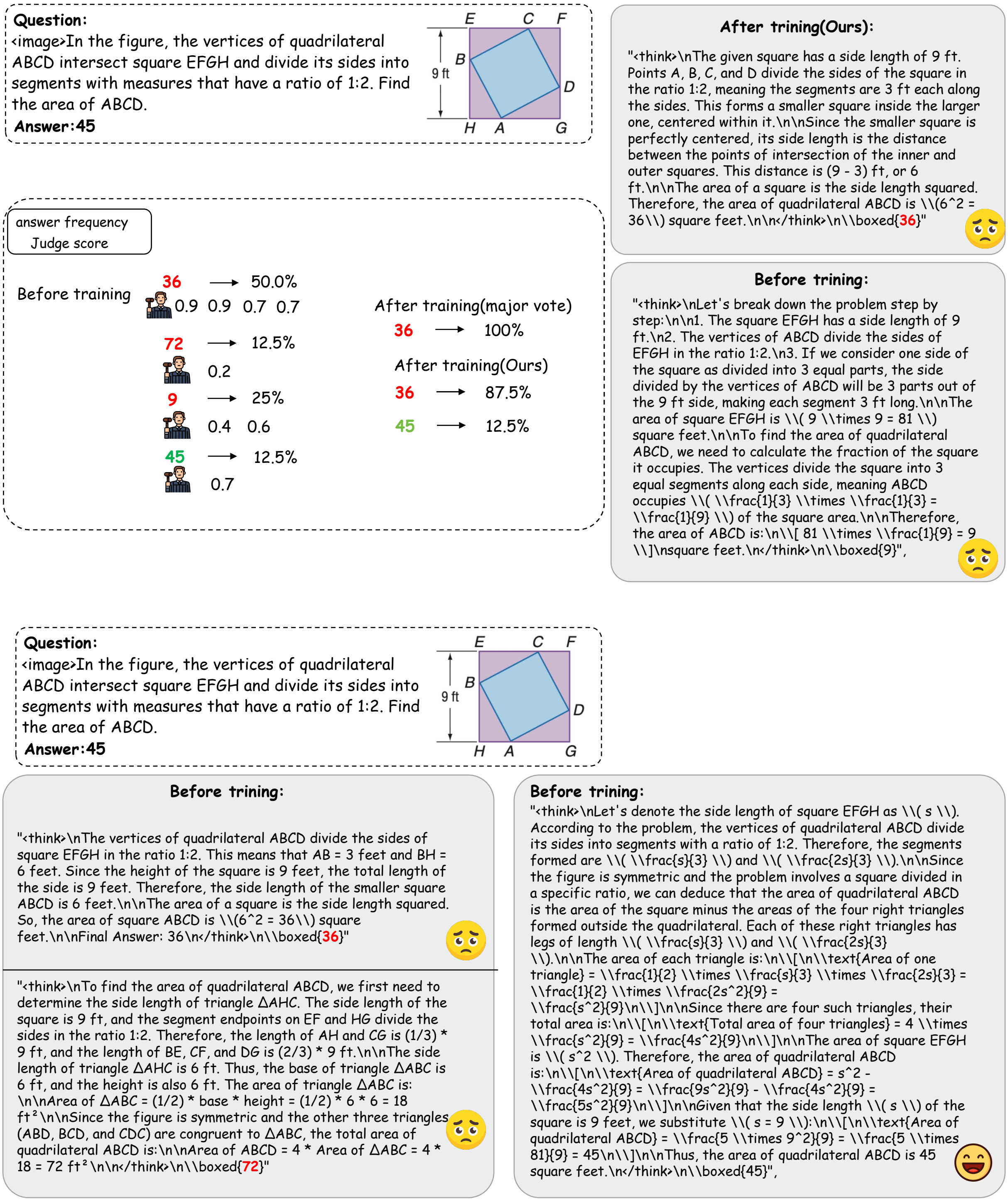


Figure 7: A qualitative failure case analysis on the Geo3K dataset (Lu et al., 2021).

**Algorithm 1:** Unsupervised Self-Evolution with Actor–Judge Joint Modeling

**Input:** Current policy $\pi_\theta$, old policy $\pi_{\theta_{\text{old}}}$, unlabeled training set $\mathcal{D}$, group size $G$, frozen Judge $J_\phi$, reference model $\pi_{\text{ref}}$, energy temperature $\alpha$, clip parameter $\epsilon$, KL coefficient $\beta$, answer extractor $E(\cdot)$, calibration params $(\lambda_+, \lambda_-, t_h, t_l, \tau_h, \tau_l)$.

**foreach** $(I, q) \sim \mathcal{D}$ **do**

- Sample a group of trajectories: $\{\tau_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(\cdot \mid I, q)$ `// // Sample multiple trajectories`
- Extract answers: $\hat{Y} = E(\mathcal{T}) = \{\hat{y}_i\}_{i=1}^{G}$ `// // Parse final answers`
- Compute self-consistency distribution: $\hat{p}(y) = \frac{1}{G}\sum_{i=1}^{G} \mathbb{I}[\hat{y}_i = y]$
- Assign SC rewards: $r_i^{\text{SC}} \leftarrow \hat{p}(\hat{y}_i)$ `// // Soft frequency reward within the group`
- Judge scoring (frozen): $s_i \leftarrow J_\phi(I, q, \tau_i) \in [0, 1]$ `// // Trajectory-level evaluation`
- Calibrate Judge scores: $g(s_i) \leftarrow 1 + \lambda_+ \sigma\left(\frac{s_i - t_h}{\tau_h}\right) - \lambda_- \sigma\left(\frac{t_l - s_i}{\tau_l}\right)$ `// // Bounded, smooth modulation`
- Compute final rewards: $R_i \leftarrow r_i^{\text{SC}} \cdot g(s_i)$ `// // Joint modeling: SC × Judge modulation`
- Group-wise distributional shaping: $\tilde{r}_i \leftarrow \alpha R_i; \quad b \leftarrow \log \sum_{j=1}^{G} \exp(\tilde{r}_j)$; $A_i \leftarrow \tilde{r}_i - b$ `// // Group-relative advantage via log-sum-exp baseline`
- Compute GRPO objective $\mathcal{J}(\theta)$ according to Eq. (12) , Eq. (13) and Eq. (14) `// // Group-relative policy optimization`
- where $\gamma_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid I, q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid I, q, o_{i,<t})}$ `// // Token-level ratio as in GRPO`
- Update policy parameters: $\theta \leftarrow \theta + \eta \nabla_\theta \mathcal{J}(\theta)$
- Update old policy: $\theta_{\text{old}} \leftarrow \theta$

**return** $\pi_\theta$

Table 5: Training hyperparameters.

| **Category** | **Hyperparameter (Value)** |
|---|---|
| **Data Configuration** | |
| Train Batch Size | 512 |
| Max Prompt Length | 1024 |
| Max Response Length | 2048 |
| Filter Overlong Prompts | True |
| Truncation Strategy | `error` |
| Image Key | `images` |
| **Model & Optimization** | |
| Base Model | Qwen2.5-VL-7B-Instruct |
| Optimizer Learning Rate | $1 \times 10^{-6}$ |
| KL Loss Coefficient ($\beta$) | 0.01 |
| KL Loss Type | Low-Var KL |
| **PPO / GRPO Settings** | |
| Algorithm | GRPO |
| PPO Mini Batch Size | 128 |
| PPO Micro Batch Size (per GPU) | 8 |
| Rollout Group Size ($n$) | 8 |
| Ref Log-Prob Micro Batch (per GPU) | 20 |
| Param Offload (Ref Model) | True |
| **Sampling / Engine** | |
| Rollout Engine | `$ENGINE` |
| Tensor Model Parallel Size | 2 |
| GPU Memory Utilization (Rollout) | 0.7 |
| GPU Memory Utilization (Reward) | 0.6 |
| **Trainer Settings** | |
| Total Epochs | 20 |
| GPUs per Node | 8 |
| Number of Nodes | 1 |

| Method | MathVision | DynaMath |
|---|---|---|
| Self-Consistency | 25.2 | 20.5 |
| w. distribution | 25.9 | 20.7 |
| Judge | 27.3 | 21.1 |
| w. distribution | 27.8 | 21.5 |
| Ours w.o. distribution | 30.1 | 23.7 |
| Ours w. distribution | 30.9 | 24.2 |

Table 6: Effect of group-wise distributional modeling.

| Method | Agree@1 | SC-winner | J-winner |
|---|---|---|---|
| Qwen2.5-VL-7B | 41.2 | 36.3 | 40.2 |
| Ours | 73.8 | 48.6 | 50.3 |

Table 7: Relationship between self-consistency and Judge preferences.

Table 8: Category-wise overall accuracy (%) on MathVision (trained on Geo3K).

| Method | Overall | Algebra | Analytic Geo. | Arithmetic | Comb. Geo. |
|---|---|---|---|---|---|
| Qwen2.5-VL-7B | 25.0 | 15.8 | 26.3 | 36.8 | 15.8 |
| Ours | 30.9 | 15.8 | 42.1 | 36.8 | 21.1 |
| **Method** | **Combinatorics** | **Counting** | **Descriptive Geo.** | **Graph Theory** | **Logic** |
| Qwen2.5-VL-7B | 10.5 | 21.1 | 26.3 | 15.8 | 15.8 |
| Ours | 21.1 | 21.1 | 21.1 | 10.5 | 21.1 |
| **Method** | **Metric (Angle)** | **Metric (Area)** | **Metric (Length)** | **Solid Geo.** | **Statistics** |
| Qwen2.5-VL-7B | 36.8 | 47.4 | 31.6 | 15.8 | 47.4 |
| Ours | 63.2 | 47.4 | 21.1 | 36.8 | 57.9 |
| **Method** | **Topology** | **Transformation Geo.** | | | |
| Qwen2.5-VL-7B | 26.3 | 10.5 | | | |
| Ours | 36.8 | 21.1 | | | |

Table 9: Category-wise overall accuracy (%) on MathVerse (trained on Geo3K).

| Method | Text Dominant | Vision Only | Vision Dominant | Vision Intensive | Text Lite |
|---|---|---|---|---|---|
| Qwen2.5-VL-7B | 53.6 | 41.0 | 40.9 | 40.9 | 44.9 |
| Ours | 56.6 | 42.5 | 42.6 | 44.8 | 47.7 |

Table 10: Category-wise overall accuracy (%) on LogicVista(trained on Geo3K).

| Method | Overall | Inductive | Deductive | Numerical | Spatial | Mechanical |
|---|---|---|---|---|---|---|
| Qwen2.5-VL-7B | 46.3 | 36.4 | 64.5 | 55.8 | 19.2 | 54.1 |
| Ours | 49.0 | 29.0 | 65.6 | 65.3 | 30.8 | 55.4 |

Table 11: Category-wise overall accuracy (%) on Wemath (trained on Geo3K).

| Category | Qwen2.5-VL-7B | Ours | Δ |
|---|---|---|---|
| Two-step (S2) | 55.28 | 58.61 | +3.33 |
| Understanding of Plane Figures | 60.22 | 64.45 | +4.23 |
| Calculation of Solid Figures | 77.26 | 79.03 | +1.77 |
| Direction | 82.86 | 90.00 | +7.14 |
| Route Map | 55.49 | 62.64 | +7.15 |