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 or teacher-model distillation, both of which are costly and difficult to scale. This raises a key question: when reliable supervision signals are unavailable, can models still improve through further training on their own generated reasoning processes? To address this, we propose an unsupervised self-evolution training framework for multimodal reasoning that improves reasoning models without using human-annotated answers or external reward models. Specifically, for each input, we sample multiple reasoning trajectories and jointly model their within-group structure as the model's initial trajectory distribution. We then introduce a bounded modulation mechanism 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 across multiple reasoning benchmarks, 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).

However, much of this progress still relies on high-quality answers and reasoning traces(Safaei et al., 2025), or on evaluators trained with expensive preference data, whose capabilities are then transferred to the target model through distillation(Huang et al., 2025b). At the same time, high-quality annotated data is becoming increasingly scarce, and the capability of existing evaluators is also approaching its practical limit(Tao et al., 2025). Motivated by this challenge, recent studies have begun to explore self-evolving post-training for multimodal models. The goal is to investigate whether models can further improve their reasoning ability by leveraging model-generated reasoning trajectories, without relying on human annotation or external supervision models(Wang et al., 2025b).

The main challenge of self-evolving post-training is the lack of reliable supervision, which makes the training signals noisy and biased. Applying reinforcement learning on top of such signals further increases the risk of gradient fluctuation and training instability. Existing approaches(Wei et al., 2025; Thawakar et al., 2025) often use model-generated intermediate results or pseudo-labels as training signals, such as sampling multiple responses and measuring their consistency. Some recent work(Zhou et al., 2025) further introduces diversity or novelty signals to encourage exploration. These strategies can make unsupervised training relatively stable: they reduce the noise of a single sample and align the training objective with output patterns that are more consistent under the current policy distribution. Nevertheless, 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 the model's ability to explore effectively. 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 optimization to saturate early and

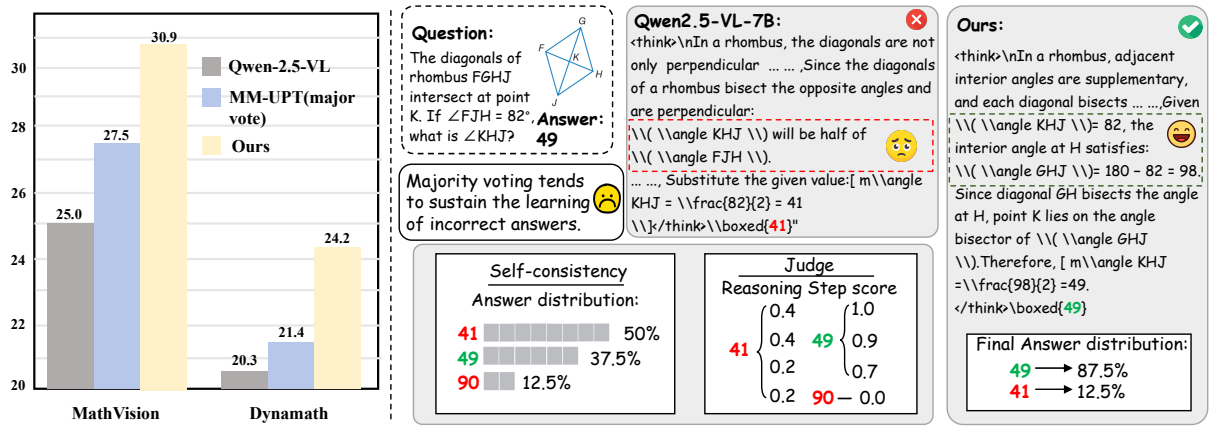


Figure 1: Limitation of majority voting in unsupervised self-evolution.Right: An example where the most frequent answer is incorrect. Majority voting reinforces this dominant error, while our method favors higher-quality reasoning paths through Judge modulation.Left: Results on MathVision(Wang et al.,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

pushing the policy distribution to collapse into low-entropy, near-deterministic output patterns.

In light of these limitations, we argue that stable unsupervised self-evolution should strike a balance between robustness and effectiveness. 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 provides a bounded and continuously differentiable modulation signal to calibrate and reshape the Actor's initial 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 method to model their rewards. 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 accordingly guide the model to adjust its output distribution. This encourages the optimization objective to better reflect the relative quality among candidate trajectories, 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, for example 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

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., 2025b; Huang et al., 2025b) derive reward signals from objectively verifiable signals, enabling more stable reasoning optimization. Despite these advances, effective reason-

ing post-training still relies on high-quality training signals or stronger teacher models.

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). 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. However, most of these methods use majority voting as the main training signal, which primarily reinforces consistency under the current output distribution.

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 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 π_θ, 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 τ_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 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 τ_i as the empirical frequency of its answer:

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

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-evolution 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. In unsupervised self-evolution, a commonly used paradigm 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:

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

From the perspective of empirical performance, majority voting is effective in unsupervised self-evolution 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 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

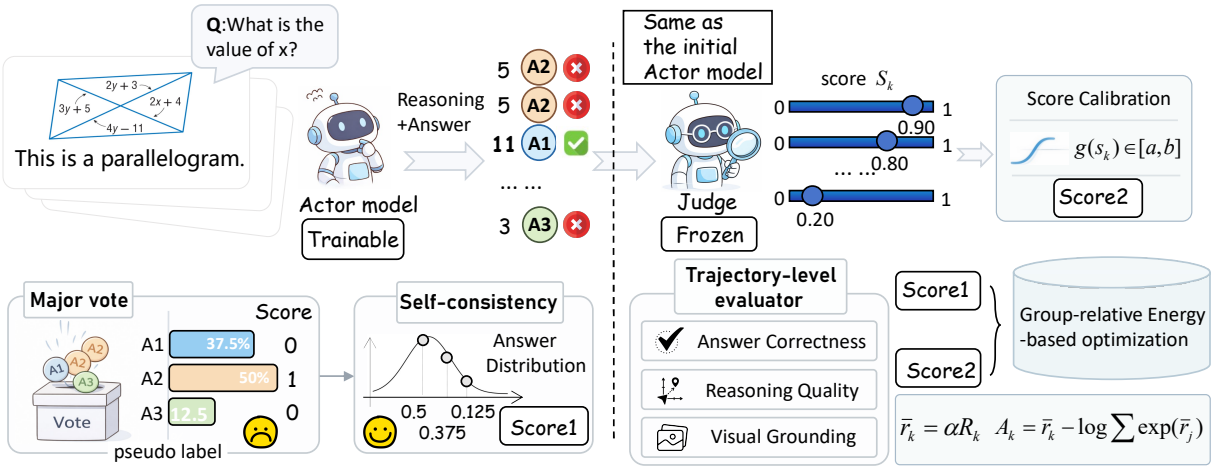


Figure 2: **Overview of the proposed unsupervised self-evolution framework.**The Actor generates multiple reasoning trajectories for the same input, while a frozen Judge provides bounded score modulation. The final rewards are optimized in a group-wise, distributional manner to enable stable policy updates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

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 mitigate this issue, we introduce a Judge module that provides a continuous quality assessment for each trajectory, which is used to calibrate the initial self-consistency-based reward. It is important to note that, to maintain the unsupervised self-evolution setting, we do not introduce any external reward model or additional supervision signal. Specifically, the Judge is initialized as a structurally identical copy of the current Actor policy and remains frozen throughout training. Therefore, the Judge's assessment inevitably contains bias, and its raw score should not be treated as an absolutely reliable reward. Based on this consideration, we only use it as a bounded modulation signal to calibrate the Actor's initial self-consistency distribution. The Judge then outputs a raw score for each trajectory by jointly assessing answer correctness, reasoning quality, and visual grounding:

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

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 and low 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.

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. This shift further amplifies the impact of Judge noise or bias in the training loop, ultimately causing the model to prematurely converge toward the Judge's preference.

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 α 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:

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

Importantly, this construction implicitly induces a reward-defined target distribution over the candidate set:

$$q_\alpha(\tau_k \mid x) = \frac{\exp(\alpha R_k)}{\sum_{j=1}^{n} \exp(\alpha R_j)}. \tag{12}$$

It then follows that:

$$A_k(x) = \log q_\alpha(\tau_k \mid x). \tag{13}$$

This shows that the group-relative advantage corresponds to the log-probability of a trajectory under the reward-induced distribution. Therefore, the policy update can be understood as gradually matching the current policy to this target distribution:

$$\min_{\theta}\ \mathbb{E}_{x\sim\mathcal{D}}\Big[D_{\text{KL}}\big(q_\alpha(\cdot \mid x) \,\|\, \pi_\theta(\cdot \mid x)\big)\Big]. \tag{14}$$

A more detailed derivation is provided in Appendix A. 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. Finally, the GRPO objective for policy optimization can be written as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{n}\sum_{k=1}^{n} r_k^{\text{clip}} - \beta\, D_{\text{KL}}\Big(\pi_\theta(\cdot \mid x) \,\Big\|\, \pi_{\text{ref}}(\cdot \mid x)\Big)\right], \tag{15}$$

$$r_k^{\text{clip}} = \min\Big(\gamma_k(\theta)\, A_k,\ \text{clip}\big(\gamma_k(\theta),\, 1-\epsilon,\, 1+\epsilon\big)\, A_k\Big), \tag{16}$$

$$\gamma_k(\theta) = \frac{\pi_\theta(\tau_k \mid x)}{\pi_{\theta_{\text{old}}}(\tau_k \mid x)}. \tag{17}$$

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, ϵ is the clipping threshold, and β controls the strength of the KL regularization toward the reference policy π_{ref}.

Overall, this group-wise distributional modeling shifts the optimization objective from simply pursuing absolute high scores to continuously reallocating probability mass within each trajectory group, leading to more stable policy updates and reducing the self-reinforcement of early dominant modes in the unsupervised loop. A more detailed analysis is provided in Appendix A.

4 Experiments

4.1 Datasets and Training Details

Training Data. We use Geometry3k(Lu et al., 2021), GeoQA(Chen et al., 2021), and MMR1(Leng et al., 2025) as training datasets. All experiments are conducted using Qwen2.5-VL-7B-Instruct(Bai et al., 2025) as the backbone.

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),

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
Supervised Training & Teacher Model Distillation							
Mixed Data🏷	R1-Onevision-7B(Yang et al., 2025b)	**29.9**	46.4	35.8	46.5	21.8	36.1
	OpenVLThinker-8B(Deng et al., 2025)	25.9	50.3	**36.5**	46.3	21.2	36.0
	Vision-R1(Huang et al., 2025b)	29.4	**52.4**	35.4	**49.7**	**25.2**	**38.4**
Unsupervised self-Evolving							
CLEVR	VisionZero(Wang et al., 2025b)	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(Thawakar et al., 2025)	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	MM-UPT(Wei et al., 2025)	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	MM-UPT	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	MM-UPT	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. 🏷 denotes supervised training.

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	DynaMath	Model / Method	MathVision	DynaMath
Qwen2-VL-2B	12.8	6.8	Qwen3-VL-8B	53.0	48.2
w/ major vote	9.6	4.2	w/ major vote	51.7	46.8
w/ ours	**16.3**	**11.4**	w/ ours	**53.5**	**50.3**
Qwen2.5-VL-3B	19.5	18.2	GLM-4.1V-9B	47.8	38.6
w/ major vote	21.3	19.1	w/ major vote	48.2	37.2
w/ ours	**24.8**	**21.3**	w/ ours	**50.4**	**40.8**
InternVL3-8B	20.8	23.6	Qwen2.5-VL-32B	38.6	35.2
w/ major vote	22.8	23.7	w/ major vote	40.2	37.1
w/ ours	**25.6**	**26.9**	w/ ours	**43.4**	**38.9**

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

WeMath(Qiao et al., 2024), LogicVista(Xiao et al., 2024), and DynaMath(Zou et al., 2024), following their standard accuracy protocols. 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., 2024b) 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 1×10^{-6}, with a weight

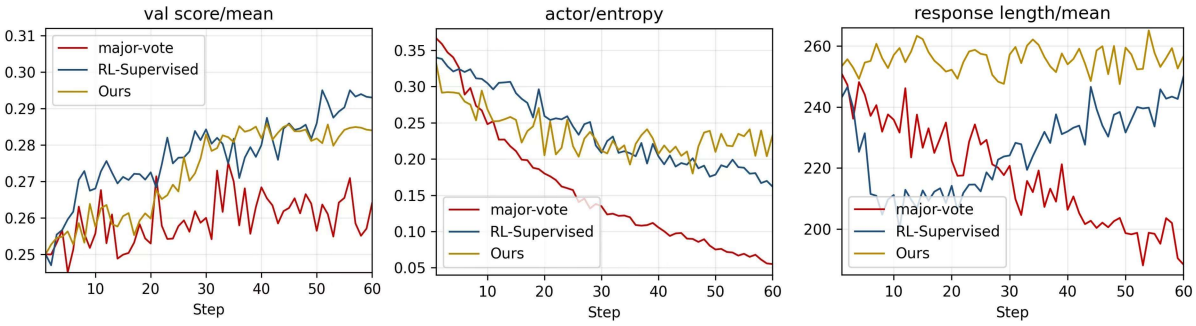


Figure 3: **Training dynamics on MMR1(Leng et al., 2025).**The figure compares majority voting, supervised reinforcement learning, and our method during training, in terms of validation accuracy on MathVision, actor entropy, and average response length.

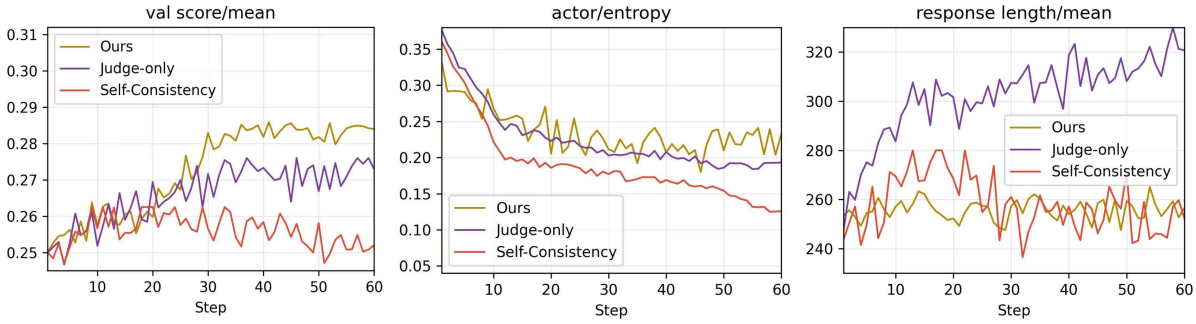


Figure 4: **Ablation training dynamics on MMR1(Leng et al., 2025).**We compare Self-Consistency, Judge-only, and the full method in terms of validation accuracy on MathVision, actor entropy, and average response length during training.

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.

Model / Method	MathVision	LogicVista	DynaMath
Vision-R1-7B	29.4	49.7	25.2
w/ major vote	30.0	49.4	24.6
w/ ours	**32.6**	**51.5**	**26.3**

Table 5: Self-evolution results on a strong baseline that has already been trained with teacher distillation.

Method	Relative Time	Accuracy
Supervised GRPO	1.0×	29.1
MM-UPT (Major Vote)	1.0×	27.5
VisionZero	2.8×	27.6
EvoLMM	2.2×	25.8
Ours	**1.4×**	**30.9**

Table 6: Comparison of relative training cost and MathVision accuracy across different methods.

decay of 1×10^{-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 methods and approaches based on strong-model distillation. Without relying on any human-annotated answers, our method consistently improves over the original model when trained on all three unsupervised training datasets(MMR1, GeoQA, and Geo3K). For example, when trained on Geo3K in an unsupervised setting, our method improves the average accuracy from 34.6 to 37.9 (+3.3) across benchmarks. The gains are more pronounced on challenging benchmarks. On MathVision, our method achieves an absolute improvement of up to 5.9 points (30.9 vs. 25.0). Compared with existing unsupervised self-evolving methods, our approach consistently outperforms prior work under the same training setting. Moreover, our

Training Setting	Method	ChartQA	MMVP
Base model	—	85.7	77.2
Geo3K	w/ MM-UPT	84.8	76.4
	w/ ours	**85.4**	**77.6**
GeoQA	w/ MM-UPT	82.6	77.4
	w/ ours	**86.3**	**77.9**
MMR1	w/ MM-UPT	85.4	74.6
	w/ ours	**87.2**	**79.8**

Table 7: Generalization to chart understanding (ChartQA) and general visual reasoning (MMVP).

method achieves performance comparable to supervised training and strong-model distillation methods, and even surpasses 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. In contrast, our method avoids repeatedly reinforcing early dominant patterns and maintains healthier entropy and response-length trajectories than supervised RL during training, resulting in more stable training behavior. Meanwhile, we evaluate three unsupervised-trained models on two non-mathematical, vision-centric benchmarks: ChartQA (chart understanding)(Masry et al., 2022) and MMVP (general visual reasoning)(Tong et al., 2024a). Table 7 shows that the proposed training paradigm generalizes beyond mathematical reasoning to broader vision-centric multimodal tasks.

4.3 Ablation Study

Table 2 reports ablation results for the key components, while Fig. 4 further illustrates 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 in response length. 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.

Table 3 shows the results of applying our method to multiple backbone models of different scales(Zhu et al., 2025; Hong et al., 2025; Yang et al., 2024). Our method remains effective on both weaker models and larger models, demonstrating good scalability across model sizes. As shown in Table 4, our method consistently outperforms majority voting on pass@10 across benchmarks. Even compared with supervised GRPO (54% vs. 53%), our method still shows more stable pass@10 performance. It is worth noting that the slight decrease in pass@10 compared with the untrained model reflects a common trade-off between performance improvement and output diversity during post-training, rather than severe mode collapse. Indeed, as shown in Fig. 4, our method exhibits a comparable or even healthier entropy trajectory than supervised GRPO, suggesting that it can improve reasoning accuracy while still maintaining effective exploration. As shown in Table 5, we further apply self-evolution training to a strong baseline that has already been improved by supervised training and teacher distillation. The results show that, even on a model already strengthened by teacher distillation, our online self-evolution mechanism can still bring further gains. Table 6 compares the computational cost of different methods. To ensure a fair comparison, we conduct all experiments under the same hardware setting and report results relative to supervised GRPO (training time $\approx$ 10.5 hours). Due to the online sampling and scoring process, our method introduces a moderate additional overhead (1.4$\times$ relative time). Further analyses and experiments on the relationship between the Judge and self-consistency, the effect of λ, the distribution of Judge scores, compute-matched comparisons, and multi-seed stability 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 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.

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.

Lili Chen, Mihir Prabhudesai, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. 2025. Self-questioning language models. *ArXiv*, abs/2508.03682.

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*.

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.

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

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

GLM-V Team Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihan Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, Aohan Zeng, and 58 others. 2025. Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning.

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.

Yang Li, Youssef Emad, Karthik Padthe, Jack Lanchantin, Weizhe Yuan, Thao Nguyen, Jason E. Weston, Shang-Wen Li, Dong Wang, Ilia Kulikov, and Xian Li. 2025. Naturalthoughts: Selecting and distilling reasoning traces for general reasoning tasks. *ArXiv*, abs/2507.01921.

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*.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, Dublin, Ireland. Association for Computational Linguistics.

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

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.

Leonardo Ranaldi, Federico Ranaldi, and Giulia Pucci. 2025. R2-MultiOmnia: Leading multilingual multimodal reasoning via self-training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8220–8234, Vienna, Austria. Association for Computational Linguistics.

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, Junmei 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.

Leitian Tao, Xuefeng Du, and Yixuan Li. 2025. Limited preference data? learning better reward model with latent space synthesis. *ArXiv*, abs/2509.26074.

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.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. 2024a. Eyes wide shut? exploring the visual shortcomings of multimodal llms. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9568–9578.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. 2024b. 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 multimodal 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.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 39 others. 2024. Qwen2 technical report. *ArXiv*, abs/2407.10671.

Wei Yang, Yiran Zhu, Zilin Li, Xunjia Zhang, and Hongtao Wang. 2025a. Self-empowering vlms: Achieving hierarchical consistency via self-elicited knowledge distillation. *ArXiv*, abs/2511.18415.

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. 2025b. R1-onevision: Advancing generalized multimodal reasoning through cross-modal formalization. *ArXiv*, abs/2503.10615.

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

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Dong Wang, Ilia Kulikov, Kyunghyun Cho, Yuandong Tian, Jason E. Weston, and Xian Li. 2025. Naturalreasoning: Reasoning in the wild with 2.8m challenging questions. *ArXiv*, abs/2502.13124.

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.

Yujun Zhou, Zhenwen Liang, Haolin Liu, Wenhao Yu, Kishan Panaganti, Linfeng Song, Dian Yu, Xiangliang Zhang, Haitao Mi, and Dong Yu. 2025. Evolving language models without labels: Majority drives selection, novelty promotes variation. *ArXiv*, abs/2509.15194.

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 $\mathcal{T}(x) = \{\tau_1, \ldots, \tau_n\}$ from the behavior policy $\pi_{\theta_{\text{old}}}(\cdot \mid x)$. Each trajectory τ_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 α 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 τ_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 $q_\alpha(\cdot \mid x)$.

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., 2024) 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 ambigu-

ous 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 8.

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 9 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 10. 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 10, 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-

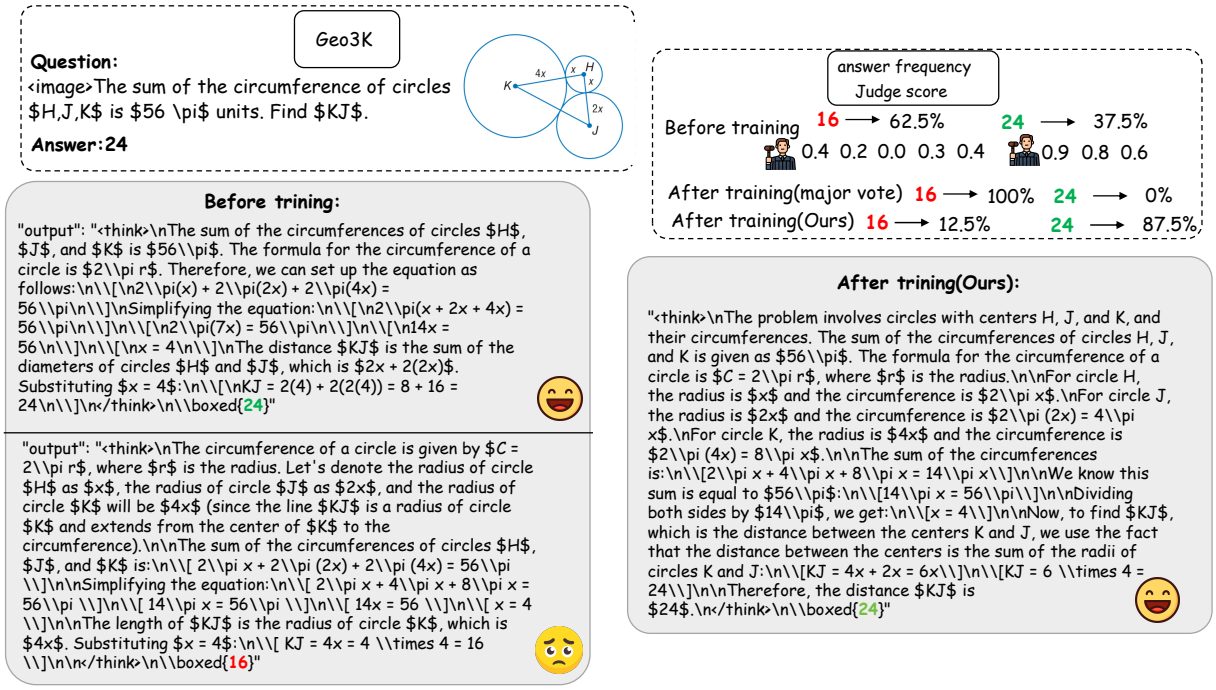


Figure 5: A case study on Geo3K(Lu et al., 2021)

ing on Geo3K across multiple benchmarks in Tables 16, 17, 18, 19.

Effect of the Judge modulation strength. We first study the effect of the modulation strength λ in the Judge calibration function. As shown in Table 11, $\lambda = 0$ reduces our method to pure self-consistency, which provides only limited improvement because it cannot correct high-frequency but low-quality answers. Increasing λ from 0.0 to 0.2 consistently improves performance, indicating that bounded Judge modulation provides useful quality information beyond self-consistency. However, further increasing λ to 0.6 or 1.0 leads to performance degradation. This suggests that a stronger Judge signal does not necessarily lead to better self-evolution: although it increases the ability to reshape the reward within each group, it also amplifies the noise and bias of the frozen zero-shot Judge. Therefore, the best performance is achieved by balancing correction ability and training stability, rather than relying on an overly strong Judge signal.

Distribution of Judge scores. We further analyze whether the frozen Judge provides meaningful trajectory-level distinctions. Table 12 reports the distribution of raw Judge scores on a mini-batch. The scores do not collapse to a constant value, but instead cover the full interval $[0, 1]$. The distribution is somewhat polarized, which is expected for an uncalibrated zero-shot evaluator, but it still provides non-trivial relative information among candidate trajectories. This supports our design choice of not treating the raw Judge score as an absolute reward, but using it only as a bounded modulation signal.

Judge scores on correct and incorrect trajectories. We also compare the average Judge scores assigned to correct and incorrect trajectories. As shown in Table 13, correct trajectories receive a substantially higher average score than incorrect ones. This result indicates that, although the Judge is not perfectly calibrated and may inherit biases from the base model, it still provides a useful relative quality signal. This further justifies our bounded modulation design: the Judge is not used as an absolute reward model, but only as a weak yet informative signal to calibrate the Actor's self-consistency prior.

Compute-matched comparison with MM-UPT. Since our method introduces additional Judge forward passes, it has about $1.4\times$ training-time overhead. To examine whether the improvement simply comes from extra computation, we construct compute-matched MM-UPT baselines by increasing the number of rollouts from $n = 8$ to $n = 12$ and $n = 16$. As shown in Table 14, increasing the rollout budget brings only marginal gains to majority-vote-based training. This suggests that

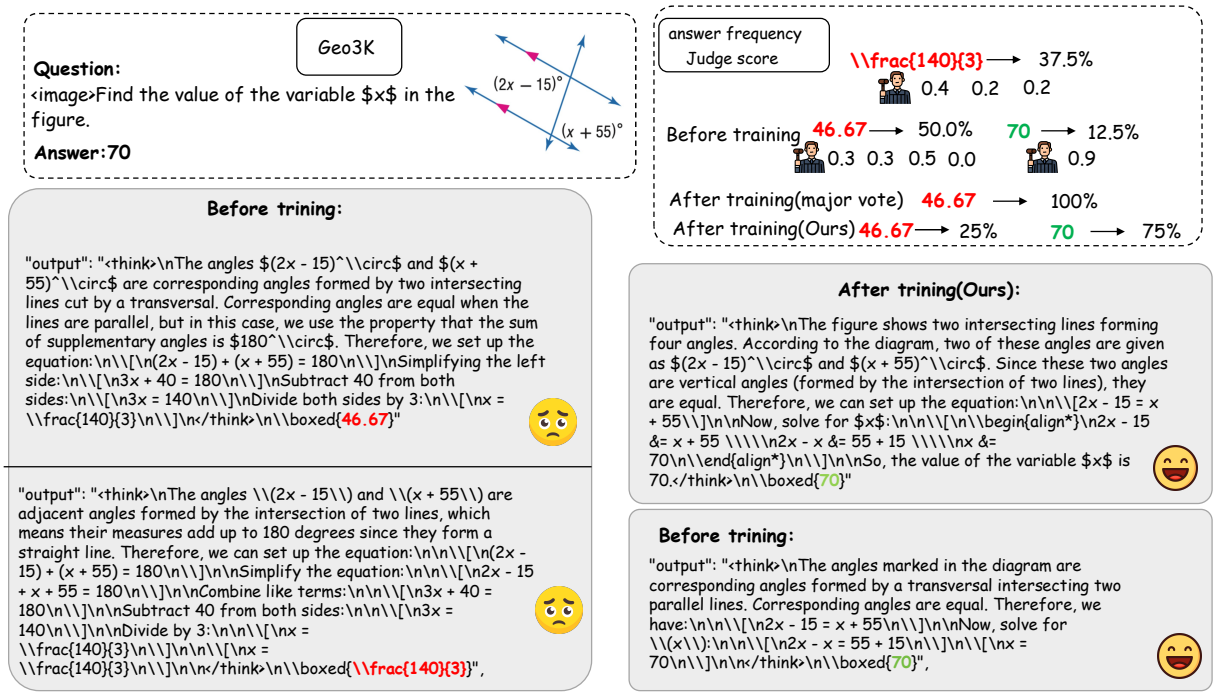


Figure 6: A case study on Geo3K(Lu et al., 2021)

the advantage of our method cannot be explained by additional sampling alone. More rollouts can reduce the sampling variance of majority voting, but they do not address its core limitation: the most frequent answer under the current policy may still be incorrect or low quality. In contrast, our method uses bounded Judge modulation and group-wise distributional modeling to reshape the reward structure within each candidate set.

Multi-seed stability. To evaluate the stability of our method, we conduct experiments with three random seeds under two representative training settings, MMR1 and Geo3K. Table 15 reports the results for each seed. Our method consistently improves over majority-vote training across different seeds. The standard deviations are small, suggesting that the observed gains are stable rather than caused by a favorable random seed. These results further support that group-wise distributional modeling and bounded Judge modulation provide a more reliable unsupervised self-evolution signal than majority voting alone.

H Related work

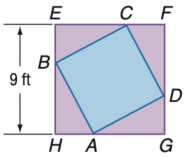
Recent advances show that reinforcement learning (RL) enhances the reasoning ability of large language models (LLMs), with representative systems including DeepSeek-R1(DeepSeek-AI et al., 2025) and Kimi-K1.5(Team et al., 2025). Under appropriately designed optimization objectives, models can gradually 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).

H.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., 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

Question:
<image>In the figure, the vertices of quadrilateral ABCD intersect square EFGH and divide its sides into segments with measures that have a ratio of 1:2. Find the area of ABCD.
Answer:45



After trining(Ours):

"<think>\nThe given square has a side length of 9 ft. Points A, B, C, and D divide the sides of the square in the ratio 1:2, meaning the segments are 3 ft each along the sides. This forms a smaller square inside the larger one, centered within it.\n\nSince the smaller square is perfectly centered, its side length is the distance between the points of intersection of the inner and outer squares. This distance is (9 - 3) ft, or 6 ft.\n\nThe area of a square is the side length squared. Therefore, the area of quadrilateral ABCD is \\(6^2 = 36\\) square feet.\n\n</think>\n\\boxed{36}"

answer frequency
Judge score

Before training

36 → 50.0%
0.9 0.9 0.7 0.7

72 → 12.5%
0.2

9 → 25%
0.4 0.6

45 → 12.5%
0.7

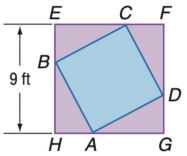
After training(major vote)
36 → 100%

After training(Ours)
36 → 87.5%
45 → 12.5%

Before trining:

"<think>\nLet's break down the problem step by step:\n\n1. The square EFGH has a side length of 9 ft.\n2. The vertices of ABCD divide the sides of EFGH in the ratio 1:2.\n3. If we consider one side of the square as divided into 3 equal parts, the side divided by the vertices of ABCD will be 3 parts out of the 9 ft side, making each segment 3 ft long.\n\nThe area of square EFGH is \\(9 \\times 9 = 81 \\) square feet.\n\nTo find the area of quadrilateral ABCD, we need to calculate the fraction of the square it occupies. The vertices divide the square into 3 equal segments along each side, meaning ABCD occupies \\(\\frac{1}{3} \\times \\frac{1}{3} = \\frac{1}{9} \\) of the square area.\n\nTherefore, the area of ABCD is:\n\\[81 \\times \\frac{1}{9} = 9 \\]\nsquare feet.\n</think>\n\\boxed{9}",

Question:
<image>In the figure, the vertices of quadrilateral ABCD intersect square EFGH and divide its sides into segments with measures that have a ratio of 1:2. Find the area of ABCD.
Answer:45



Before trining:

"<think>\nThe vertices of quadrilateral ABCD divide the sides of square EFGH in the ratio 1:2. This means that AB = 3 feet and BH = 6 feet. Since the height of the square is 9 feet, the total length of the side is 9 feet. Therefore, the side length of the smaller square ABCD is 6 feet.\n\nThe area of a square is the side length squared. So, the area of square ABCD is \\(6^2 = 36\\) square feet.\n\nFinal Answer: 36\n</think>\n\\boxed{36}"

"<think>\nTo find the area of quadrilateral ABCD, we first need to determine the side length of triangle ΔAHC. The side length of the square is 9 ft, and the segment endpoints on EF and HG divide the sides in the ratio 1:2. Therefore, the length of AH and CG is (1/3) * 9 ft, and the length of BE, CF, and DG is (2/3) * 9 ft.\n\nThe side length of triangle ΔAHC is 6 ft. Thus, the base of triangle ΔABC is 6 ft, and the height is also 6 ft. The area of triangle ΔABC is:\n\nArea of ΔABC = (1/2) * base * height = (1/2) * 6 * 6 = 18 ft²\n\nSince the figure is symmetric and the other three triangles (ABD, BCD, and CDC) are congruent to ΔABC, the total area of quadrilateral ABCD is:\n\nArea of ABCD = 4 * Area of ΔABC = 4 * 18 = 72 ft²\n\n</think>\n\\boxed{72}"

Before trining:

"<think>\nLet's denote the side length of square EFGH as \\(s \\). According to the problem, the vertices of quadrilateral ABCD divide its sides into segments with a ratio of 1:2. Therefore, the segments formed are \\(\\frac{s}{3} \\) and \\(\\frac{2s}{3} \\).\n\nSince the figure is symmetric and the problem involves a square divided in a specific ratio, we can deduce that the area of quadrilateral ABCD is the area of the square minus the areas of the four right triangles formed outside the quadrilateral. Each of these right triangles has legs of length \\(\\frac{s}{3} \\) and \\(\\frac{2s}{3} \\).\n\nThe area of each triangle is:\n\\[\n\\text{Area of one triangle} = \\frac{1}{2} \\times \\frac{s}{3} \\times \\frac{2s}{3} = \\frac{1}{2} \\times \\frac{2s^2}{9} = \\frac{s^2}{9}\n\\]\n\nSince there are four such triangles, their total area is:\n\\[\n\\text{Total area of four triangles} = 4 \\times \\frac{s^2}{9} = \\frac{4s^2}{9}\n\\]\n\nThe area of square EFGH is \\(s^2 \\). Therefore, the area of quadrilateral ABCD is:\n\\[\n\\text{Area of quadrilateral ABCD} = s^2 - \\frac{4s^2}{9} = \\frac{9s^2}{9} - \\frac{4s^2}{9} = \\frac{5s^2}{9}\n\\]\n\nGiven that the side length \\(s \\) of the square is 9 feet, we substitute \\(s = 9 \\):\n\\[\n\\text{Area of quadrilateral ABCD} = \\frac{5 \\times 9^2}{9} = \\frac{5 \\times 81}{9} = 45\n\\]\n\nThus, the area of quadrilateral ABCD is 45 square feet.\n</think>\n\\boxed{45}",

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

structured reflection steps or learning an explicit critic for evaluation. NaturalReasoning(Yuan et al., 2025) proposes a method for constructing large-scale reasoning data from real-world corpora. It shows that such large-scale in-the-wild reasoning data can support both knowledge distillation from strong teacher models and self-training with either external reward models or self-generated rewards. Building on this line of work, NaturalThoughts(Li et al., 2025) studies which teacher-generated reasoning traces are the most useful for distillation. Based on the large-scale question set introduced in NaturalReasoning, it selects high-quality reasoning traces generated by strong teacher models for distillation training. R2-MultiOmnia(Ranaldi et al., 2025) presents a self-training framework for multilingual multimodal reasoning. Despite these advances, effective reasoning post-training still relies on high-quality training signals or stronger teacher models.

H.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. Going further, some methods(Chen et al., 2025) move beyond a fixed training set by letting models generate tasks and improve themselves. Self-Empowering VLMs(Yang et al., 2025a) studies hierarchical understanding in VLMs and shows that the main challenge is not missing taxonomic knowledge, but the difficulty of maintaining cross-level consistency during step-by-step prediction. 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. Vis-Play(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.

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

Input: Current policy π_θ, old policy $\pi_{\theta_{\text{old}}}$, unlabeled training set $\mathcal{D}$, group size G, frozen Judge J_ϕ, reference model π_{ref}, energy temperature α, clip parameter ϵ, KL coefficient β, 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 $\times$ 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 π_θ

Table 8: 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	1×10^{-6}
KL Loss Coefficient (β)	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 9: 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 10: Relationship between self-consistency and Judge preferences.

Table 11: Effect of the Judge modulation strength λ.

λ	**MathVision**	**DynaMath**	**Avg.**
0.0	25.2	20.5	22.8
0.1	28.7	22.8	25.7
0.2	**30.9**	**24.2**	**27.5**
0.6	29.2	23.5	26.3
1.0	28.8	22.3	25.5
1.0 w/o SC	27.3	21.1	24.2

Table 12: Distribution of raw Judge scores on a mini-batch.

Score range	**[0.0, 0.2)**	**[0.2, 0.4)**	**[0.4, 0.6)**	**[0.6, 0.8)**	**[0.8, 1.0]**
Judge score	22.8%	12.5%	13.9%	16.7%	34.1%

Table 13: Average Judge scores for correct and incorrect trajectories.

Trajectory type	**Mean Judge score**
Correct trajectories	0.68
Incorrect trajectories	0.32

Table 14: Compute-matched comparison with MM-UPT.

Training Data	Method	MathVision	MathVerse	WeMath	LogicVista	DynaMath	Avg.
–	Qwen2.5-VL-7B	25.0	44.2	37.1	46.3	20.3	34.6
MMR1	MM-UPT rollout $n = 8$	26.4	46.0	38.6	47.9	21.8	36.1
	MM-UPT rollout $n = 12$	26.8	46.2	38.7	48.0	22.0	36.3
	MM-UPT rollout $n = 16$	27.0	45.8	38.8	48.1	21.6	36.2
	Ours	**28.4**	**46.4**	**38.8**	**48.6**	**23.0**	**37.0**
GeoQA	MM-UPT rollout $n = 8$	27.3	45.1	38.2	47.3	21.9	36.0
	MM-UPT rollout $n = 12$	27.2	45.4	38.3	47.3	22.1	36.0
	MM-UPT rollout $n = 16$	27.6	45.3	38.4	47.6	22.4	36.2
	Ours	**28.6**	**46.5**	**38.9**	**47.9**	**23.2**	**37.0**
Geo3K	MM-UPT rollout $n = 8$	27.5	44.0	37.4	46.9	21.4	35.4
	MM-UPT rollout $n = 12$	28.0	44.5	37.3	47.2	21.8	35.7
	MM-UPT rollout $n = 16$	28.2	43.7	36.9	46.8	22.5	35.6
	Ours	**30.9**	**46.8**	**38.7**	**49.0**	**24.2**	**37.9**

Table 15: Multi-seed results under two representative training settings.

Training Data	Method	Seed	MathVision	MathVerse	WeMath	LogicVista	DynaMath	Avg.
–	Qwen2.5-VL-7B	–	25.0	44.2	37.1	46.3	20.3	34.6
MMR1	Major Vote	Seed 1	26.4	46.0	38.6	47.9	21.8	36.1
		Seed 2	26.1	45.8	38.7	48.0	21.5	36.0
		Seed 3	26.6	46.3	38.3	47.5	22.0	36.1
MMR1	Ours	Seed 1	28.4	46.4	38.8	48.6	23.0	37.0
		Seed 2	28.2	46.6	38.5	48.8	23.3	37.1
		Seed 3	28.7	46.1	39.0	48.3	22.9	37.0
Geo3K	Major Vote	Seed 1	27.5	44.0	37.4	46.9	21.4	35.4
		Seed 2	27.2	44.3	37.6	46.5	21.6	35.4
		Seed 3	27.8	43.6	37.0	47.1	21.1	35.3
Geo3K	Ours	Seed 1	30.9	46.8	38.7	49.0	24.2	37.9
		Seed 2	31.1	46.5	38.9	48.7	24.4	38.0
		Seed 3	30.6	47.0	38.4	49.2	23.9	37.8

Table 16: 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 17: 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 18: 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 19: 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