

Prioritizing the Best: Incentivizing Reliable Multimodal Reasoning by Rewarding Beyond Answer Correctness

Mengzhao Jia Zhihan Zhang Meng Jiang

University of Notre Dame

mjia2@nd.edu

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) improves multimodal reasoning in answer-verifiable settings by rewarding final answers that can be checked. Yet answer-correct trajectories may still rely on incomplete derivations, weak evidence, or statements that contradict their conclusions. This gap between answer correctness and reasoning validity, which we call *reasoning-answer inconsistency*, motivates trajectory supervision in multimodal RLVR. We compare two main approaches: reward models (RMs) and generative rewards (GRs). RMs are efficient and help early in training, but their gains weaken as the policy distribution shifts; GRs improve performance, but they may yield unstable rewards and incur high pointwise judging cost. We therefore propose **Groupwise Ranking Reward**, which ranks verifier-passed trajectories for the same prompt in one pass and redistributes reward accordingly. Groupwise comparison better separates stronger and weaker correct trajectories with lower judge overhead than pointwise GRs. Experiments show that RLVR aggravates reasoning-answer inconsistency, while trajectory supervision alleviates it. Groupwise Ranking Reward performs best overall, improving reliability-conditioned accuracy from 47.4% to 54.7% over RLVR. The code is available at <https://github.com/Jill0001/GroupRank>.

1 Introduction

Multimodal Large Language Models (MLLMs) are increasingly capable of visually grounded reasoning (Shi et al., 2024; Wang et al., 2025c), and Reinforcement Learning with Verifiable Rewards (RLVR) (DeepSeek-AI et al., 2025) has become a standard way to push this progress further by directly optimizing verifiable outcomes such as final answer correctness (Zhang et al., 2025b; Wang et al., 2025e; Wei et al., 2025). While reasoning

faithfulness has been studied in text-only chain-of-thought settings (Wang et al., 2026), recent studies have noted that under outcome-only multimodal supervision, MLLMs’ reasoning process can become less reliable as training continues, even when the final answer is correct (Chen et al., 2025; Kan et al., 2025; Huang et al., 2025; Wang et al., 2025b; Jia et al., 2025). A closer look at the training process shows that answer-correct responses increasingly rely on incomplete derivations, weak evidence, or even statements that conflict with their own conclusions. As shown in Figure 1(a): the reasoning trajectory derives $X = -11$, but the final boxed answer is 10, the response would still be rewarded by outcome-only supervision despite an unfaithful reasoning chain. We trace this problem to a basic limitation of answer-verifiable multimodal RLVR: it supervises whether the final answer is correct, but not whether the visual-textual reasoning process actually justifies that answer. We refer to this failure mode as *reasoning-answer inconsistency*. Left unresolved, this failure mode can make MLLMs unreliable in applications such as medical decision support, autonomous driving, and scientific analysis, where a correct-looking answer with contradictory reasoning is not acceptable.

A natural way to address this problem is to add supervision over the validity of the reasoning trajectory. However, how to introduce such supervision signals into multimodal RL in a way that is both effective and computationally efficient remains underexplored. In this paper, we compare two main families of trajectory supervision methods: **Reward Models** (RMs) and **Generative Rewards** (GRs). RMs directly assign scalar quality scores to the reasoning process. We use the fine-grained **Process Reward Model** (PRM) that assigns a score to each intermediate reasoning step (Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2023). On the other hand, GRs use a Large Language Model (LLM) to follow a predefined evaluation

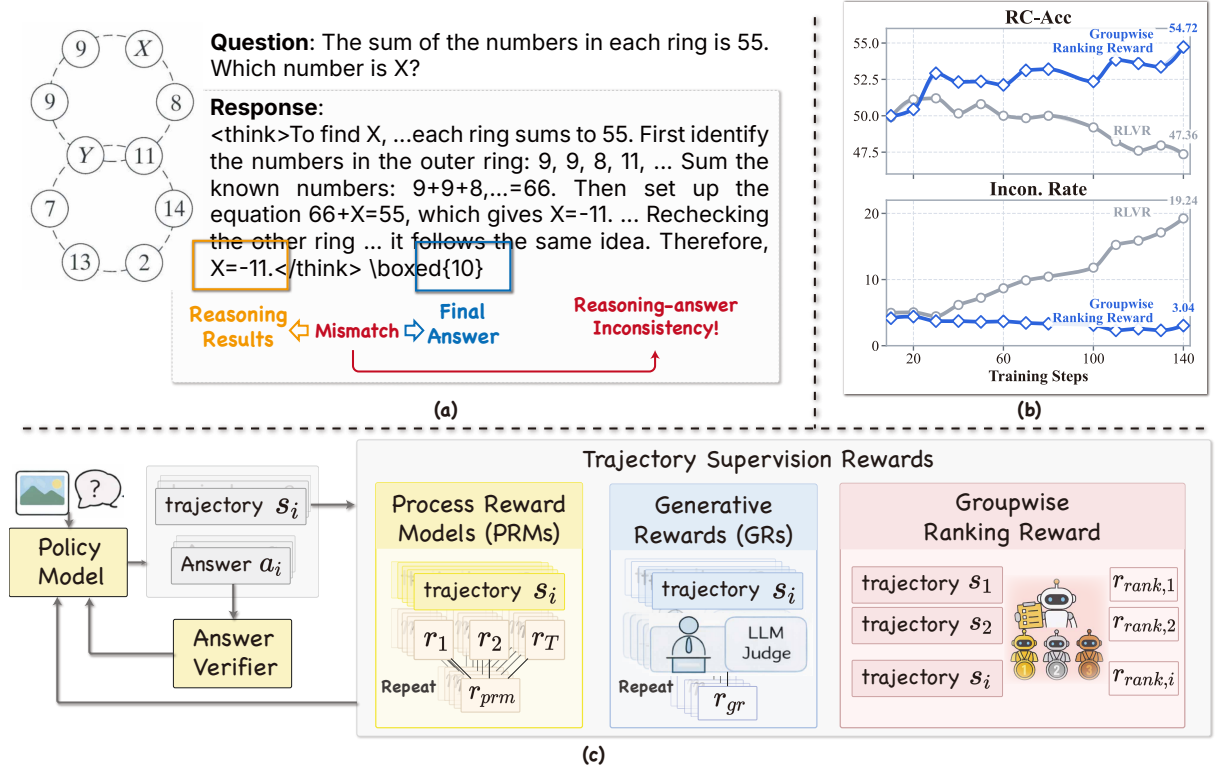


Figure 1: **Reasoning-answer inconsistency under outcome-only RLVR and the motivation for Groupwise Ranking Reward.** (a) An example of reasoning-answer inconsistency: the trajectory derives $X = -11$ in its reasoning, but the final boxed answer is 10. (b) Training dynamics under outcome-only RLVR and Groupwise Ranking Reward. Outcome-only RLVR steadily increases inconsistency and eventually reduces reliability-conditioned accuracy (RC-Acc), whereas Groupwise Ranking Reward keeps inconsistency stable and continues improving RC-Acc. (c) Comparison of trajectory supervision schemes. PRMs and pointwise GRs score each trajectory independently, while Groupwise Ranking Reward compares verifier-passed trajectories for the same prompt jointly and assigns ranking-based rewards according to their relative quality.

instruction, generate a textual judgment then assign a scalar score (Zheng et al., 2023; Yuan et al., 2024; Wang et al., 2025d). Our experiments show that RMs are computationally efficient and help in the early stages of training, but their benefits weaken as the policy distribution shifts; GRs improve performance to some extent, but their gains are limited by instability and come with severe efficiency costs. Based on these observations, we propose a **Groupwise Ranking Reward**, which ranks verifier-passed trajectories of the same problem in a single pass, and redistributes rewards based on that ranking. Figure 1(c) highlights the key distinction: PRMs and pointwise GRs score each trajectory independently, whereas our method compares multiple trajectories jointly and rewards them according to relative quality. By using groupwise comparison instead of independent scoring, the method better distinguishes stronger from weaker correct trajectories with lower judge computation overhead than pointwise GRs. It achieves the strongest performance among all trajectory supervision methods

tested.

We conduct extensive experiments with different judge models, reward designs, and training data to systematically analyze this problem and evaluate the proposed method. The results show that RLVR can aggravate reasoning-answer inconsistency, whereas trajectory supervision alleviates this effect. Figure 1(b) previews this trend: under outcome-only RLVR, inconsistency rises over training and eventually reduces Reliability-Conditioned Accuracy (RC-Acc), while Groupwise Ranking Reward keeps inconsistency stable and continues improving RC-Acc. Among the approaches we study, the Groupwise Ranking Reward achieves both the best accuracy and reasoning faithfulness, improving RC-Acc from 47.4% to 54.7% over RLVR. These gains indicate that even when answer correctness is already verified, multimodal RL still lacks a way to prefer better-grounded correct trajectories.

2 Related Work

Reinforcement Learning for Multimodal Reasoning. Chain-of-Thought prompting and RLHF established explicit multi-step reasoning in LLMs (Wei et al., 2022; Ouyang et al., 2022). Recent MLLMs extend this idea to visually grounded tasks through reasoning-oriented supervision and post-training, especially multimodal mathematical and general-purpose visual reasoning (Shi et al., 2024; Wang et al., 2025c; Zhang et al., 2025b; Wang et al., 2025e; Wei et al., 2025). RLVR is natural in this setting because verifiable rewards optimize answer correctness without dense trajectory annotations; GRPO and DeepSeek-R1 further show that this paradigm scales beyond narrow domains (Shao et al., 2024; DeepSeek-AI et al., 2025; Su et al., 2025). Our work complements these efforts by studying how RLVR should assign differential credit among verifier-passed rollouts so policies prefer better-grounded reasoning trajectories.

Reasoning-Answer Inconsistency in MLLMs. Reasoning-answer inconsistency relates to unfaithful chain-of-thought reasoning in text-only LLMs (Wang et al., 2026); we study its dynamics under multimodal RLVR, where answer accuracy can improve while answer-reasoning coherence degrades. Existing fixes follow three directions: reference-policy calibration (Chen et al., 2025; Kan et al., 2025), which can weaken as the online policy drifts; consistency under semantics-preserving perturbations (Huang et al., 2025; Wang et al., 2025b), most natural for closed-answer settings; and rubric-based generative rewards (Jia et al., 2025), which offer richer supervision but add judge overhead and are not updated with the policy.

Trajectory Supervision. Another line improves reasoning by supervising intermediate steps rather than only final answers, first in text-only mathematical reasoning (Uesato et al., 2022; Lightman et al., 2023) and recently in multimodal settings (Zhang et al., 2025a). Related work uses language models as evaluators or reward models (Zheng et al., 2023; Yuan et al., 2024; Gu et al., 2024), and shows that listwise or distribution-aware judge inference can be more reliable than independent absolute scoring (Zhao et al., 2024; Wang et al., 2025d). Closely related, RLRR replaces absolute reward shaping with relative ranking in group-based RL and trains a ranking reward model for groupwise preferences (Niu et al., 2026), but mainly uses ranking to improve reward modeling and optimization.

We instead study RMs and GRs for multimodal trajectory supervision and propose groupwise ranking to improve multimodal reasoning reliability.

3 Method

RLVR improves answer correctness, but it does not distinguish among verifier-passed trajectories whose reasoning quality differs substantially. As a result, the policy model can still be rewarded for correct answers supported by incomplete or contradictory traces. To address this limitation, we study three trajectory supervision reward designs within the same RLVR pipeline. The first is a **Reward Model (RM)**, which assigns scalar scores; in our experiments, this RM is instantiated as a fine-grained **Process Reward Model (PRM)** that scores intermediate reasoning steps (Uesato et al., 2022; Lightman et al., 2023; Wang et al., 2023; Zhang et al., 2025a). The second is a standard **Generative Reward (GR)**, which uses an LLM-as-a-Judge to produce a textual judgment and score each trajectory independently (Zheng et al., 2023; Yuan et al., 2024; Gu et al., 2024). The third is our **Groupwise Ranking Reward**, a groupwise variant of generative reward that compares verifier-passed trajectories for the same prompt and redistributes reward according to their relative quality ranking (Zhao et al., 2024; Wang et al., 2025d).

3.1 Problem Setup

Let $\mathcal{D}$ be a multimodal training set. Each training example is denoted by (x, a^*) , where $x = (v, q)$, v is the input image, q is the textual question, and a^* is the ground-truth answer. Given x , the policy $\pi_\theta(\cdot | x)$ generates a structured response

$$y = [z, a], \quad (1)$$

where z is the reasoning trajectory enclosed in `<think>...</think>` and a is the final answer extracted from the `\boxed{...}` expression. In the RLVR pipeline, each sampled complete response $y = [z, a]$ is referred to as a *rollout*.

Following standard RLVR pipelines for reasoning (Shao et al., 2024; DeepSeek-AI et al., 2025), a rule-based verifier provides a deterministic verification reward

$$r_{\text{ver}}(y) = \mathbb{I}[\text{Verify}(a, a^*) = 1], \quad (2)$$

where $\mathbb{I}[\cdot]$ is the indicator function. RLVR maximizes $\mathbb{E}_{(x, a^*) \sim \mathcal{D}, y \sim \pi_\theta} [r_{\text{ver}}(y)]$. This objective

gives identical reward to all verifier-passed responses; however, their reasoning quality may differ substantially. We use the standard GRPO (Shao et al., 2024) optimization pipeline throughout: for each prompt x , we sample a rollout group $Y = \{y_i = [z_i, a_i]\}_{i=1}^N$ from $\pi_\theta(\cdot | x)$ and optimize over this group using verifier and trajectory rewards. Here we focus on how different methods define the trajectory supervision signal; the GRPO optimization objective and equations are given in Appendix A.

3.2 Trajectory Reward Variants

Beyond the verifier reward $r_{\text{ver}}(y_i)$, we add an auxiliary trajectory reward $r_{\text{aux},i}$ to provide extra supervision on rollout $y_i = [z_i, a_i]$. Common trajectory rewards include RMs and GRs.

RMs. RMs assign scalar quality scores without generating extra natural language justifications. They can operate at different supervision granularities. In this paper, we instantiate the RM as a fine-grained PRM that decomposes the reasoning trajectory into intermediate steps and assigns a step-level quality signal along the trajectory. Let $\mathcal{S}_{\text{prm}}$ denote the reward model and let $\{s_{i,t}^{\text{prm}}\}_{t=1}^{T_i} = \mathcal{S}_{\text{prm}}(x, z_i)$ be the scores for the T_i reasoning steps in rollout i . The rollout-level PRM reward is obtained by aggregating these step scores,

$$r_{\text{prm},i} = \text{Agg}\left(\{s_{i,t}^{\text{prm}}\}_{t=1}^{T_i}\right), \quad (3)$$

where $\text{Agg}(\cdot)$ denotes the aggregation rule. PRM provides dense supervision over the reasoning process.

GRs. GRs use an LLM judge that follows an evaluation instruction with explicit scoring rubrics. For each rollout, the judge generates a textual judgment together with a rubric-grounded scalar score, from which the reward is parsed:

$$r_{\text{gr},i} = \text{ParseScore}(\mathcal{S}_{\text{gr}}(q, z_i, a_i)). \quad (4)$$

Compared with RMs, GRs do not directly emit scores; instead, the judge first produces an evaluative judgment, and the scalar reward is then extracted from that judgment. In our setup, the rubric casts the judge as an expert evaluator of reasoning quality, defines anchored score levels in $[0, 1]$, and asks it to assess dimensions such as logical coherence, calculation quality, completeness of the conclusion, and honest handling of uncertainty. The

judge returns a short feedback string together with a JSON-formatted scalar score, which we parse as the reward. We refer to the setting where each rollout is judged independently as *pointwise GR*, to distinguish it from the *groupwise GR* setting introduced next.

RMs and GRs are both widely used trajectory supervision signals, but they also have limitations: PRM can weaken under policy shift, while pointwise GR depends on the absolute score calibration and requires separate judging for each rollout. Motivated by this, we further introduce Groupwise Ranking Reward as a groupwise variant of GRs.

Groupwise Ranking Reward. Groupwise Ranking Reward extends GRs from independent scoring to within-group ranking. For each prompt x , we keep only the verifier-passed subset $I^+ = \{i : r_{\text{ver}}(y_i) = 1\}$ within the sampled rollout group Y , where $|I^+|$ denotes the number of verifier-passed rollouts, because incorrect rollouts are already handled by the verifier. We focus on finding which correct rollout is best in reliability and quality.

The groupwise judge $\mathcal{S}_{\text{cmp}}$ takes the shared question q together with all verifier-passed reasoning-answer pairs $\{(z_i, a_i)\}_{i \in I^+}$, compares them side-by-side, and outputs M ordered tie-aware tiers

$$(\tau_1, \tau_2, \dots, \tau_M) = \mathcal{S}_{\text{cmp}}(q, \{(z_i, a_i)\}_{i \in I^+}), \quad (5)$$

where $\tau_1 \succ \tau_2 \succ \dots \succ \tau_M$, τ_m contains the candidates assigned to the m -th rank tier, and τ_1 is the top tier.

We design a rank-to-score conversion based on a simple intuition: a rollout should receive a higher score if it is ranked ahead of more verifier-passed rollouts, and a lower score if it is ranked ahead of fewer. We also preserve ties instead of forcing arbitrary tie-breaking, avoiding injecting artificial noise when the judge views several traces as equally rigorous. Concretely, for any rollout $i \in \tau_m$, let $b_i = \sum_{\ell=m+1}^M |\tau_\ell|$ be the number of lower-ranked correct rollouts and let $c_i = |\tau_m| - 1$ be the number of tied rollouts. The raw score is

$$\tilde{s}_i = \frac{b_i + \frac{1}{2}c_i}{|I^+| - 1}, \quad (6)$$

where each lower-ranked rollout contributes one point, each tie contributes half a point, and the result is normalized by the number of available comparisons. Thus, candidates in τ_1 receive the highest raw score and candidates in τ_M the lowest, while prompts with different $|I^+|$ remain on

the same $[0, 1]$ scale. This normalization makes the rank-to-score mapping well-defined for prompt groups with any number of verifier-passed rollouts.

The final ranking reward is obtained by centering these raw scores within the verifier-passed set. Let $\bar{s} = \frac{1}{|I^+|} \sum_{j \in I^+} \tilde{s}_j$ denote the mean raw score over I^+ .

$$r_{\text{rank},i} = \begin{cases} \tilde{s}_i - \bar{s}, & \text{if } i \in I^+, |I^+| \geq 2, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

This centering makes the ranking reward zero-mean within the verifier-passed set. Under this mapping, $r_{\text{rank},i}$ is bounded in $[-0.5, 0.5]$. Thus, the judge acts as a pure redistribution mechanism: better-supported correct rollouts receive positive ranking reward, while weaker correct rollouts can still have positive verifier reward but negative ranking reward, which effectively penalizes weaker correct reasoning among answer-correct rollouts. Incorrect rollouts remain untouched and still receive zero total reward. If all verifier-passed rollouts fall into the same tier, the centered reward is again zero. Appendix B gives the full implementation details and summarizes alternative rank-to-score mappings.

3.3 Training Objective

The final rollout reward combines verification with one auxiliary trajectory reward:

$$\mathcal{R}_i = r_{\text{ver}}(y_i) + \lambda r_{\text{aux},i}, \quad (8)$$

where $\lambda \geq 0$ controls the strength of trajectory supervision and $r_{\text{aux},i} \in \{r_{\text{prm},i}, r_{\text{gr},i}, r_{\text{rank},i}\}$. In our experiments, we use $\lambda = 1$ for all three variants. The optimization algorithm is unchanged.

4 Experiments

4.1 Experimental Setup

Training Hyperparameters. All RL training runs start from the same base checkpoint, Qwen2.5-VL-7B-Instruct (Bai et al., 2025). Across all RL variants, we use rollout budget $n = 8$, auxiliary-reward coefficient $\lambda = 1$, learning rate 1×10^{-6} , KL coefficient 1×10^{-2} . All experiments use the same ViRL (Wang et al., 2025a) training data and are run for 2 epochs.

Methods to Compare. Our main experiment reports the untrained Qwen2.5-VL-7B-IT base model, outcome-only RLVR, a reference-based baseline, and three consistency-oriented reward variants on ViRL: w/ PRM, w/ Pointwise GR, and

w/ Groupwise Ranking Reward. All trained methods use the same GRPO optimization pipeline; the reference-based baseline and three reward variants differ in how they add trajectory-level supervision beyond outcome-only RLVR. The reference-based method follows GRPO-CARE (Chen et al., 2025): a smoothed copy of earlier policy states serves as the reference model, scoring whether each answer-correct reasoning trace supports the correct answer and giving high-scoring rollouts within the same prompt a consistency bonus. The PRM variant uses a visual process-reward model, Qwen-VL-PRM-7B (Ong et al., 2025), that scores each trajectory step with a binary true/false decision and averages rewards across all steps. The two GR-based variants both use gpt-oss-20b (OpenAI, 2025) as the judge over textualized question-trajectory-answer tuples: w/ Pointwise GR gives a scalar judge_score in $[0, 1]$, while w/ Groupwise Ranking Reward assigns tie-aware ranks for verifier-passed candidates and maps them to centered groupwise rewards. Appendix G lists the judge prompts.

Benchmarks. We evaluate final checkpoints on five multimodal reasoning benchmarks: three math-focused benchmarks, MathVision (Wang et al., 2024), MathVista (Lu et al., 2024), and WeMath (Qiao et al., 2025), and two broader general-reasoning benchmarks, MMMU (Yue et al., 2024a) and MMMU-Pro (Yue et al., 2024b). To make the reported average reasoning-answer inconsistency across all benchmarks fair, we randomly sample 500 instances from each dataset to construct our final test set. In this way, each benchmark contributes equally to the overall inconsistency rate.

Evaluation Metrics. Accuracy (Acc) measures whether the final answer matches the ground truth. However, Acc alone does not capture a failure mode that we frequently observe: *reasoning-answer inconsistency*, where the final answer may be correct but the trajectory fails to justify the boxed answer or even supports a different conclusion. To measure this failure mode, we evaluate reasoning-answer consistency with an LLM judge based on gpt-oss-20b (OpenAI, 2025). Given the question, trajectory, and final answer, the judge extracts the answer implied by the trajectory and the final boxed answer, then labels the response as consistent or inconsistent according to whether the two extracted answers agree. From these labels, we track the Inconsistency Rate (IncR), defined as the percentage of all examples labeled inconsistent,

Method	RC-Acc Avg.	Acc Avg.	MathVision	MathVista	MMMU	MMMU-Pro	WeMath
<i>Reference and baseline methods</i>							
Qwen2.5-VL-7B-IT	47.3	49.0	24.8	67.2	52.6	35.2	56.6
RLVR	47.4	53.6	22.8	70.8	48.2	36.2	58.8
w/ Reference-Based Method	51.7	53.5	27.6	<u>72.0</u>	53.2	38.0	<u>67.6</u>
<i>Controlled comparison with different reward methods</i>							
w/ PRM	46.8	49.1	26.4	66.4	50.0	30.6	60.8
w/ Pointwise GR	<u>52.3</u>	<u>53.8</u>	31.0	71.0	<u>56.6</u>	39.0	63.8
w/ Groupwise Ranking Reward	54.7	55.9	31.4	74.8	57.8	<u>38.8</u>	70.8

Table 1: Model performance comparison results under a matched training budget. The second and third columns report five-benchmark average RC-Acc and Acc, respectively, while the remaining benchmark columns report RC-Acc. Groupwise Ranking Reward achieves the best overall performance, yielding the highest average RC-Acc and leading on four of the five benchmarks. We bold the best value in each numeric column and underline the second best.

regardless of final answer correctness. We further isolate inconsistent cases that would otherwise be counted as correct: Correct-But-Inconsistent Rate (CBIR) is the percentage of all examples with a correct final answer but an inconsistent trajectory. We then define reliability-conditioned accuracy (RC-Acc) as the portion of examples whose final answer is both correct and supported by the reasoning trajectory. We use RC-Acc as our primary metric because it preserves final-answer correctness while additionally requiring consistency between the reasoning trajectory and the final answer. Equivalently, $\text{RC-Acc} = \text{Acc} - \text{CBIR}$.

To validate the judge reliability, we conduct a 400-response human audit. The judge agrees with human labels on 387 samples, yielding 96.8% alignment and Cohen’s $\kappa = 0.755$. Appendix C provides the more details of the audit, e.g. full prompt protocol and disagreement analysis.

4.2 Main Results

Table 1 reports the average RC-Acc and Acc, together with the benchmark-wise RC-Acc, after training with different methods. Appendix D provides the complete score breakdown and the corresponding multi-seed robustness results for the controlled reward variants.

RLVR improves answer accuracy but not reasoning reliability. Compared to the base checkpoint, RLVR raises average answer accuracy, but its RC-Acc remains unchanged because correct-but-inconsistent predictions become much more frequent (see Appendix D for CBIR scores). This reveals a significant side effect of outcome-only RL: the policy gets the final answer right more often, but the reward still cannot tell whether the trajectory actually supports that answer. **GR-based rewards re-**

cover this gap, and Groupwise Ranking Reward gives the best overall trade-off. The reference-based method improves over RLVR, while both GR-based variants further outperform it under the same training budget. Groupwise Ranking Reward reaches the strongest average RC-Acc at 54.7%, compared with 52.3% for Pointwise GR, 51.7% for the reference-based method, and 47.4% for RLVR. This suggests that comparing correct candidates within the same prompt gives a more reliable signal than scoring each rollout in isolation. **The gains are broad rather than benchmark-specific.** Groupwise Ranking Reward performs best on four of the five benchmarks and remains competitive on MMMU-Pro, with especially large gains on WeMath. The improvement is therefore not tied to one dataset or one answer format.

4.3 Analysis

4.3.1 Training Dynamics

Figure 2 shows the five-benchmark average trends for Acc, RC-Acc, IncR, and CBIR, so we can see how accuracy and inconsistency evolve throughout training rather than only at the final checkpoint.

Outcome-only reward harms reasoning reliability as training progresses. Although RLVR improves Acc from 51% to 54%, as shown in the first panel of Figure 2, this gain does not translate into more reliable or faithful reasoning trajectories. Its RC-Acc peaks early at around 51.20% between 20 and 30 training steps, but then declines to 47.36% by step 140. This deterioration is driven by a substantial increase in reasoning-answer inconsistency during the later stages of training: IncR rises from 6% at step 60 to 19% at step 140, while CBIR increases steadily from 2% to 6% over the full training process. These results suggest that

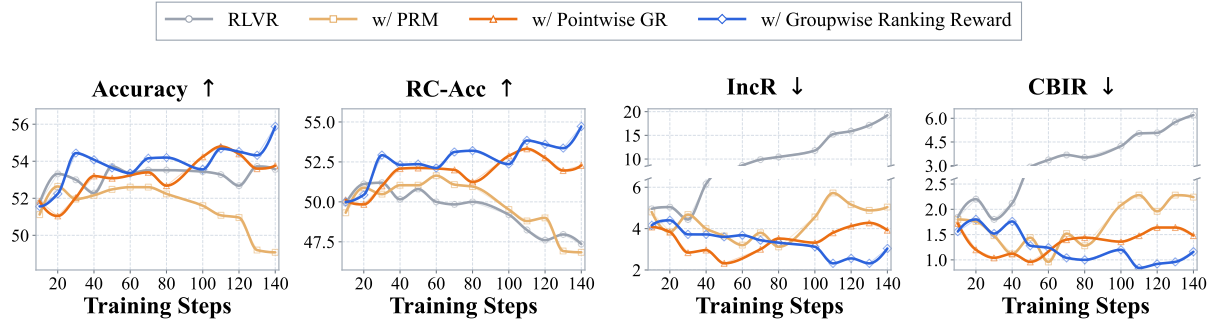


Figure 2: Training dynamics across reward variants. The four panels report Accuracy, RC-Acc, IncR, and CBIR, and all reported values are percentages. The last two panels use a broken y-axis so the large RLVR values remain visible while differences among the lower-valued methods are still readable. Although outcome-only RL can improve final-answer correctness, it simultaneously degrades reasoning reliability in multimodal reasoning tasks. By contrast, Groupwise Ranking Reward is the only method that consistently improves both answer accuracy and RC-Acc while simultaneously reducing IncR and CBIR.

while outcome-only RL can improve final-answer correctness, it simultaneously degrades reasoning reliability in multimodal reasoning tasks.

PRM offers only limited benefits for reducing reasoning-answer inconsistency. Although PRM has been reported to be effective for some text-only reasoning tasks, incorporating a PRM reward does not yield a substantial improvement in RC-Acc under our multimodal reasoning setup. Instead, its trend resembles that of vanilla RLVR: RC-Acc plateaus during the middle stage of training and then declines afterward. PRM-based training is able to prevent inconsistency rates from growing markedly, but it does not reduce them in a meaningful way either. This finding suggests that PRMs remain constrained by their generalization ability, which is a common criticism of such approaches, because the policy model can generate responses that differ substantially from the PRM’s training distribution.

Our group-wise ranking reward is the most effective at reducing inconsistent reasoning trajectories. It is the only method that delivers stable and continuous improvement across all four panels: it consistently improves both answer accuracy and RC-Acc, while simultaneously reducing IncR and CBIR. These results indicate that our method provides more stable and reliable reward signals. More importantly, they offer clear evidence that our generative reward alters the direction of optimization, rather than merely stacking training compute.

4.3.2 Effect of Model Size on Inconsistency

We next ask whether the significance of the reasoning-answer inconsistency problem is related

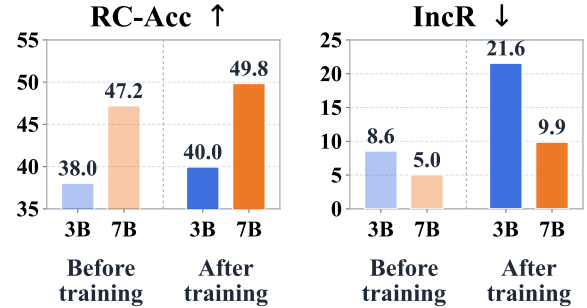


Figure 3: Qwen2.5-VL-3B and 7B models trained with vanilla RLVR: RC-Acc (left) and IncR (right). Each panel shows the results before and after training. Larger model capacity reduces inconsistency, but IncR still increases substantially after training.

to the size of the model. To keep this auxiliary analysis aligned with the rank-to-score ablation, Figure 3 reports the same two summary metrics, RC-Acc and IncR, for Qwen2.5-VL 3B and 7B checkpoints under the same ViRL training setup. We find that **scaling mitigates inconsistency, but it does not remove the incentive that creates it.** The 7B model starts with higher RC-Acc and lower IncR than the 3B model, and it remains better after training. However, outcome-only RL still increases IncR for both model sizes, from 8.56% to 21.56% for 3B and from 5.04% to 9.88% for 7B. The result suggests that model capacity changes the severity of the failure mode, whereas the underlying cause remains the reward signal.

4.3.3 Effect of the Rank-to-Score Mapping

Groupwise Ranking Reward compares verifier-passed rollouts within the same prompt and maps the resulting order to scalar rewards. This ablation asks whether the gains depend on this post-

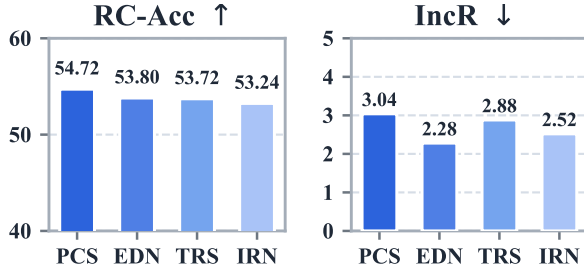


Figure 4: Ablation on the rank-to-score mapping under the matched training setup. Stable results across mappings suggest that groupwise comparison matters more than the exact monotonic conversion.

ranking conversion or mainly on groupwise comparison itself. Under the same training setup, we compare four deterministic mappings. Our default **Pairwise-Comparison Score (PCS)** converts the K -candidate ranking into a tie-aware pairwise win rate, giving full, half, or no credit against lower-ranked, tied, or higher-ranked candidates and normalizing by $K - 1$. The alternatives are **Exponential-Decay Normalization (EDN)**, which applies an exponentially decaying rank score; **Tiered-Rank Score (TRS)**, which assigns evenly spaced rewards to rank tiers; and **Inverse-Rank Normalization (IRN)**, which min-max normalizes reciprocal ranks. Figure 4 reports five-benchmark average RC-Acc and IncR.

The results are stable across mappings: average RC-Acc ranges from 53.24% to 54.72%, a spread of 1.48 points, while IncR remains between 2.28% and 3.04%. PCS gives the best RC-Acc and stays in the same low-inconsistency range as the alternatives, so we keep it as the default for its simple tie-aware interpretation. Overall, the key design choice is to compare correct rollouts within the prompt, rather than tune a fragile monotonic rank-to-reward mapping.

4.3.4 Stability Comparison Between Pointwise GRs and Groupwise Ranking

Groupwise Ranking Reward judges all rollouts for the same prompt in one pass under a shared comparison standard. In contrast, pointwise GR scores each response independently, so judging N rollouts requires N separate judge calls whose internal standards may vary. We therefore expect Groupwise Ranking Reward to be more stable. We test this with a controlled simulation: stable reward assignment should give similar rewards when repeated on the same fixed responses.

Specifically, we randomly sample 500 prompts,

generate 8 rollouts for each prompt as in training, and rerun reward assignment 4 times at a relatively low temperature (0.7). Each repeat follows the training procedure: pointwise GR scores responses independently, whereas Groupwise Ranking Reward jointly ranks verifier-passed responses within the prompt and maps the ranking to scalar rewards. We compute the variance of repeated reward values and average it over the sampled set; because responses are fixed, this variance isolates randomness in reward assignment. Lower variance indicates more stable assignment. We find that **groupwise ranking yields substantially lower reward variance**. The mean variance drops from 0.026 under pointwise GR to 0.016 under Groupwise Ranking Reward, a 37.6% reduction. This suggests that relative comparison reduces sensitivity to judge randomness and provides a more stable supervision signal in this setting.

4.3.5 Efficiency Comparison

We compare reward-and-advantage construction time across RLVR, PRM, Pointwise GR, and Groupwise Ranking Reward, with absolute stage-wise timings reported in Appendix E. Outcome-only RLVR has negligible reward-construction cost because it only uses a rule-based verifier, but it does not provide trajectory-level supervision. Among trajectory-supervision variants, Groupwise Ranking Reward has the lowest reward-stage overhead: it is about $2.7\times$ faster than PRM and about $7.6\times$ faster than Pointwise GR, mainly because one groupwise judge call replaces step-wise or independent rollout-level reward calls.

5 Conclusion

We study reasoning-answer inconsistency in multimodal RLVR and show that outcome-only optimization can worsen this mismatch even as answer accuracy improves. Trajectory supervision mitigates this effect, indicating that answer correctness alone is an incomplete objective and that trajectory-level signals remain useful after verifiable rewards improve end accuracy. Among the compared methods, **Groupwise Ranking Reward** performs best by jointly ranking verifier-passed trajectories for the same prompt. This within-prompt comparison gives the judge a shared context for distinguishing stronger and weaker correct rollouts, rather than relying on separately calibrated absolute scores. It improves average accuracy from 53.6% to 55.9% and reliability-conditioned accuracy from 47.4%

to 54.7% over RLVR, while requiring less judge computation than pointwise GRs. Its higher stability further suggests that shared within-prompt comparison provides a cleaner optimization signal.

Limitations

Our study focuses on a specific phenomenon that becomes visible in answer-verifiable RLVR settings: the model can receive reward for a correct final answer even when the reasoning trajectory does not actually support it. This framing fits domains where correctness can be checked reliably, but it does not directly cover less verifiable generation settings without a single canonical answer, such as creative writing, free-form visual dialogue, subjective caption revision, or brainstorming tasks. In such settings, the same failure mode may not appear in the same explicit form, and both inconsistency measurement and reward design may need to be reformulated. Extending the analysis to these less verifiable multimodal tasks is an important direction for future work.

Acknowledgments

This work was partially supported by NSF IIS-2119531, IIS-2137396, IIS-2142827, IIS-2234058, and Coefficient Giving. We also appreciate the support from the Foundation Models and Applications Lab of Lucy Institute and ND-IBM Tech Ethics Lab.

References

- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao 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 preprint arXiv:2502.13923*.
- Yi Chen, Yuying Ge, Rui Wang, Yixiao Ge, Junhao Cheng, Ying Shan, and Xihui Liu. 2025. [Grpo-care: Consistency-aware reinforcement learning for multimodal reasoning](#). *arXiv preprint arXiv:2506.16141*.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shiron Ma, Xiao Bi, and 1 others. 2025. [Deepseek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning](#). *arXiv preprint arXiv:2501.12948*.
- Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, and 1 others. 2024. [A survey on llm-as-a-judge](#). *arXiv preprint arXiv:2411.15594*.
- Minbin Huang, Runhui Huang, Chuanyang Zheng, Jingyao Li, Guoxuan Chen, Han Shi, and Hong Cheng. 2025. [Answer-consistent chain-of-thought reinforcement learning for multi-modal large language models](#). *arXiv preprint arXiv:2510.10104*.
- Mengzhao Jia, Zhihan Zhang, Ignacio Cases, Zheyuan Liu, Meng Jiang, and Peng Qi. 2025. [Autorubric-rlv: Rubric-based generative rewards for faithful multimodal reasoning](#). *arXiv preprint arXiv:2510.14738*.
- Zhehan Kan, Yanlin Liu, Kun Yin, Xinghua Jiang, Xin Li, Haoyu Cao, Yinsong Liu, Deqiang Jiang, Xing Sun, Qingmin Liao, and Wenming Yang. 2025. [Taco: Think-answer consistency for optimized long-chain reasoning and efficient data learning via reinforcement learning in llms](#). *arXiv preprint arXiv:2505.20777*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. [Let’s verify step by step](#). *arXiv preprint arXiv:2305.20050*.
- 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 *The Twelfth International Conference on Learning Representations*.
- Wenzhe Niu, Wei He, Zongxia Xie, Jinpeng Ou, Huichuan Fan, Yuchen Ge, Yanru Sun, Ziyin Wang, Yizhao Sun, Chengshun Shi, Jiuchong Gao, Jinghua Hao, and Renqing He. 2026. [From absolute to relative: Rethinking reward shaping in group-based reinforcement learning](#). *arXiv preprint arXiv:2601.23058*.
- Brandon Ong, Tej Deep Pala, Vernon Toh, William Chandra Tjhi, and Soujanya Poria. 2025. Training vision-language process reward models for test-time scaling in multimodal reasoning: Key insights and lessons learned. *arXiv preprint arXiv:2509.23250*.
- OpenAI. 2025. [gpt-oss-120b & gpt-oss-20b model card](#). OpenAI model card. Accessed 2026-03-13.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, and 1 others. 2022. [Training language models to follow instructions with human feedback](#). *arXiv preprint arXiv:2203.02155*.

- Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma GongQue, Shanglin Lei, Yifan Zhang, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Xiao Zong, Yida Xu, Peiqing Yang, Zhimin Bao, Muxi Diao, Chen Li, and Honggang Zhang. 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.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models.](#) *arXiv preprint arXiv:2402.03300*.
- Wenhao Shi, Zhiqiang Hu, Yi Bin, Junhua Liu, Yang Yang, See-Kiong Ng, Lidong Bing, and Roy Ka-Wei Lee. 2024. [Math-LLaVA: Bootstrapping mathematical reasoning for multimodal large language models.](#) In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4663–4680.
- Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. 2025. [Crossing the reward bridge: Expanding rl with verifiable rewards across diverse domains.](#) *arXiv preprint arXiv:2503.23829*.
- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. [Solving math word problems with process- and outcome-based feedback.](#) *arXiv preprint arXiv:2211.14275*.
- Fuxin Wang, Amr Alazali, and Yiqiao Zhong. 2026. [How does unfaithful reasoning emerge from autoregressive training? a study of synthetic experiments.](#) *arXiv preprint arXiv:2602.01017*.
- Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhui Chen. 2025a. [VL-Rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning.](#) *arXiv preprint arXiv:2504.08837*.
- Jiahao Wang, Weiye Xu, Aijun Yang, Wengang Zhou, Lewei Lu, Houqiang Li, Xiaohua Wang, and Jinguo Zhu. 2025b. [Enhancing the outcome reward-based rl training of mllms with self-consistency sampling.](#) *arXiv preprint arXiv:2511.10648*.
- Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. 2024. [Measuring multimodal mathematical reasoning with MATH-Vision dataset.](#) *arXiv preprint arXiv:2402.14804*.
- Ke Wang, Junting Pan, Linda Wei, Aojun Zhou, Weikang Shi, Zimu Lu, Han Xiao, Yunqiao Yang, Houxing Ren, Mingjie Zhan, and Hongsheng Li. 2025c. [Mathcoder-VL: Bridging vision and code for enhanced multimodal mathematical reasoning.](#) In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2505–2534.
- Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2023. [Math-shepherd: Verify and reinforce llms step-by-step without human annotations.](#) *arXiv preprint arXiv:2312.08935*.
- Victor Wang, Michael JQ Zhang, and Eunsol Choi. 2025d. [Improving LLM-as-a-judge inference with the judgment distribution.](#) In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23173–23199, Suzhou, China. Association for Computational Linguistics.
- Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, and 2 others. 2025e. [Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency.](#) *arXiv preprint arXiv:2508.18265*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models.](#) *arXiv preprint arXiv:2201.11903*.
- Yana Wei, Liang Zhao, Jianjian Sun, Kangheng Lin, Jisheng Yin, Jingcheng Hu, Yinmin Zhang, En Yu, Haoran Lv, Zejia Weng, Jia Wang, Chunrui Han, Yang Peng, Qi Han, Zheng Ge, Xiangyu Zhang, Daxin Jiang, and Vishal M. Patel. 2025. [Open vision reasoner: Transferring linguistic cognitive behavior for visual reasoning.](#) *arXiv preprint arXiv:2507.05255*.
- Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. [Self-rewarding language models.](#) *arXiv preprint arXiv:2401.10020*.
- Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, and 3 others. 2024a. [MMM: A massive multi-discipline multimodal understanding and reasoning benchmark for expert AGI.](#) In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
- Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, Yu Su, Wenhui Chen, and Graham Neubig. 2024b. [MMM-Pro: A more robust multi-discipline multimodal understanding benchmark.](#) *arXiv preprint arXiv:2409.02813*.
- Jianghangfan Zhang, Yibo Yan, Kening Zheng, Xin Zou, Song Dai, and Xuming Hu. 2025a. [Gm-prm: A generative multimodal process reward model for multimodal mathematical reasoning.](#) *arXiv preprint arXiv:2508.04088*.

Kaichen Zhang, Keming Wu, Zuhao Yang, Bo Li, Kairui Hu, Bin Wang, Ziwei Liu, Xingxuan Li, and Lidong Bing. 2025b. [Openmmreasoner: Pushing the frontiers for multimodal reasoning with an open and general recipe](#). *arXiv preprint arXiv:2511.16334*.

Yang Zhao, Yixin Wang, and Mingzhang Yin. 2024. [Permutative preference alignment from listwise ranking of human judgments](#). *arXiv preprint arXiv:2410.04346*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhonghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-bench and chatbot arena](#). *arXiv preprint arXiv:2306.05685*.

A Standard GRPO Objective

For completeness, we summarize the standard GRPO objective used in DeepSeekMath and later RLVR systems (Shao et al., 2024; DeepSeek-AI et al., 2025). Given rollout rewards $\{\mathcal{R}_i\}_{i=1}^N$ for one sampled group, GRPO first standardizes them within the prompt:

$$A_i = \frac{\mathcal{R}_i - \bar{\mathcal{R}}}{\text{std}(\{\mathcal{R}_j\}_{j=1}^N) + \epsilon_A}, \quad \bar{\mathcal{R}} = \frac{1}{N} \sum_{j=1}^N \mathcal{R}_j, \quad (9)$$

where ϵ_A is a small constant for numerical stability. Let

$$\rho_i(\theta) = \frac{\pi_\theta(y_i | x)}{\pi_{\theta_{\text{old}}}(y_i | x)}. \quad (10)$$

The clipped GRPO surrogate is

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim \mathcal{D}, \{y_i\}_{i=1}^N \sim \pi_{\theta_{\text{old}}}(\cdot | x)} \left[\frac{1}{N} \sum_{i=1}^N \ell_{\text{clip}}(\theta; x, y_i, A_i) - \beta \mathbb{D}_{\text{KL}}(\pi_\theta(\cdot | x) \| \pi_{\text{ref}}(\cdot | x)) \right] \quad (11)$$

where

$$\ell_{\text{clip}}(\theta; x, y_i, A_i) = \min \left(\rho_i(\theta) A_i, \text{clip}(\rho_i(\theta), 1 - \epsilon, 1 + \epsilon) A_i \right) \quad (12)$$

Our method does not modify this optimizer; it only changes how $\mathcal{R}_i$ is constructed.

B More Details of Groupwise Ranking Reward

Groupwise Ranking Reward. For the same prompt, let $I^+ = \{i : r_{\text{ver}}(y_i) = 1\}$ be the verifier-passed set and let $K = |I^+|$. The groupwise judge reads all textualized candidates $\{t_i\}_{i \in I^+}$ together and returns ordered tie-aware tiers $(\tau_1, \dots, \tau_M)$.

We then apply exactly the tie-aware pairwise-win-rate mapping and within-group centering defined in Section 3.2; if $K < 2$, the auxiliary reward is set to zero. This is only a post-processing of the same K -way ranking, not a separate round of pairwise judge calls. A candidate is treated as beating every lower tier, tying candidates in its own tier, and losing to every higher tier. For example, if $K = 4$ and the judge outputs $\tau_1 = \{A\}$, $\tau_2 = \{B, C\}$, and $\tau_3 = \{D\}$, then the raw scores before centering are $\tilde{s}_A = 1$, $\tilde{s}_B = \tilde{s}_C = 0.5$, and $\tilde{s}_D = 0$. If all verifier-passed rollouts tie, the centered reward becomes zero for all of them.

C Inconsistency Judge Design and Reliability

Human validation protocol. We construct a 400-response audit set to quantify the reliability of the automatic inconsistency diagnostic. The sample is stratified across four systems, with 100 responses from each system, and is drawn uniformly from all training stages so that the audit covers different methods, model states, and inconsistency levels. A human volunteer who is a graduate student annotates the audit set independently of the paper authors. The automatic judge label is produced by the four-step extraction-and-comparison procedure described in Section 4.1: it extracts the answer implied by the trajectory, extracts and normalizes the final boxed answer, compares the two under lightweight equivalence rules, and abstains when there is no valid answer. Responses without a valid final answer are labeled N/A and are not counted as inconsistent. The volunteer’s label follows the same binary decision rule: whether the reasoning-implied answer and the final answer are inconsistent.

Audit results. Across the 400 audited responses, gpt-oss-20b judge marks 31 samples as inconsistent and 369 as consistent, while the volunteer’s labels mark 26 samples as inconsistent and 374 as consistent. Treating inconsistency as the positive class, the audit gives 22 true positives, 9 false positives, 4 false negatives, and 365 true negatives. Therefore, the judge agrees with the volunteer’s labels on 387 of 400 samples, giving 96.8% alignment and Cohen’s $\kappa = 0.755$. On the inconsistent class, the judge obtains 71.0% precision, 84.6% recall, and 77.2% F1. Among judge-consistent cases, 365 of 369 are confirmed by the volunteer.

Observed residual errors. There are 13 judge-human disagreements in total, consisting of 9 false positives and 4 false negatives. The false positives mainly come from two cases. First, 4 disagreements involve missed option-value mapping: the response states the semantic answer and then maps it to the correct option, but the judge compares the surface forms too literally, such as comparing a derived value with an option letter. Second, 4 disagreements involve overly strict numeric or unit normalization: the reasoning gives a value with units, decimals, or an equivalent scale, while the final answer follows the problem’s requested rounding or unit convention. The false negatives mostly come from unsupported self-correction: in 3 cases, the response changes an earlier derivation without sufficient justification, often by altering an assumption or inserting an unsupported correction, and then forces the final answer to match an option. The remaining 2 disagreements are low-frequency mixed cases involving formatting, ambiguous notation, or hard-to-classify revisions. These patterns show that the diagnostic’s residual errors are concentrated in interpretable option-mapping, normalization, and unsupported-revision cases rather than arbitrary judge failures.

D Additional Results for Main Experiment

Complete metric breakdown. Table 2 provides the full metric breakdown for the main results. Each cell reports the three-number breakdown used throughout the paper: RC-Acc, Acc, and CBIR. We place this detailed table in the appendix so the main text can emphasize the faithfulness-aware metric directly, while still making the complete accuracy-versus-inconsistency trade-off explicit.

Multi-seed robustness. We rerun the main experiment with three random seeds under the same backbone, training data, rollout budget, optimizer, hyperparameters, training length, and evaluation split. Table 3 reports the mean and standard deviation of average RC-Acc across seeds. The bottom panel also reports the matched-seed margin between Groupwise Ranking Reward and Pointwise GR.

E Timing Efficiency

Table 4 reports measured per-step timings averaged over training steps for RLVR, PRM, Pointwise GR, and Groupwise Ranking Reward, with values

rounded to the nearest second. The measurement separates rollout generation, reward-and-advantage construction, and actor update, so the source of overhead is explicit. Outcome-only RLVR remains the cheapest setting because it uses only the rule-based verifier. PRM adds step-wise reward scoring and therefore spends about 175 seconds in reward-and-advantage construction. Pointwise GR is the slowest variant because it invokes the judge independently for rollouts in the group, raising reward-and-advantage construction to about 499 seconds. Groupwise Ranking Reward uses one groupwise call over the verifier-passed set, reducing that stage to about 66 seconds and giving a full-step time of about 487 seconds. These timings are measured on 8 NVIDIA H100 GPUs using the ver1¹ training framework, with vLLM² acceleration applied to both rollout generation and the PRM/generative-judge reward-generation stage. Under this same training setup, Groupwise Ranking Reward is more expensive than outcome-only RLVR but cheaper than the PRM and Pointwise GR variants.

F More Results on Factors Affecting Inconsistency

F.1 Question Type and Training Objective

The five-benchmark evaluation suite is not balanced by question type, so raw inconsistency counts are hard to interpret. Across the 2,500 evaluated problems, 1,990 are multiple-choice and 510 are open-ended. The open-ended subset appears only in MathVision (247), MathVista (233), and a small 30-example subset of MMMU, while MMMU-Pro and WeMath are entirely multiple-choice. We therefore normalize every rate in this subsection by the total number of examples within the same question type rather than by the global benchmark total.

Training objective matters more than the raw question-type split. Table 5 and Figure 5 show that the low open-ended inconsistency of the reference checkpoint does not survive outcome-only RL. Before training, IncR is 5.88% for multiple-choice and 1.76% for open-ended. Under RLVR, these rates jump to 20.40% and 14.71%, and CBIR rises to 6.98% and 3.14%. Question type still changes the absolute level, but the dominant effect is the reward design: outcome-only training pushes

¹<https://github.com/volcengine/ver1>

²<https://github.com/vllm-project/vllm>

Method	Avg.	Math Vision	Math Vista	MMMU	MMMU- Pro	WeMath
<i>Reference and baseline methods</i>						
Qwen2.5-VL-7B-IT	47.3	24.8	67.2	52.6	35.2	56.6
	49.0 / 1.7	26.2 / 1.4	68.6 / 1.4	55.0 / 2.4	37.4 / 2.2	58.0 / 1.4
RLVR	47.4	22.8	70.8	48.2	36.2	58.8
	53.6 / 6.2	28.6 / 5.8	75.6 / 4.8	55.8 / 7.6	40.4 / 4.2	67.4 / 8.6
	51.7	27.6	<u>72.0</u>	53.2	38.0	<u>67.6</u>
w/ Reference-Based Method	53.5 / 1.8	28.8 / 1.2	74.0 / 2.0	54.8 / 1.6	39.2 / 1.2	70.8 / 3.2
<i>Controlled comparison with different reward methods</i>						
	46.8	26.4	66.4	50.0	30.6	60.8
w/ PRM	49.1 / 2.3	26.8 / 0.4	70.0 / 3.6	51.6 / 1.6	35.2 / 4.6	61.8 / 1.0
	<u>52.3</u>	31.0	71.0	<u>56.6</u>	39.0	63.8
w/ Pointwise GR	53.8 / 1.5	32.4 / 1.4	72.2 / 1.2	58.0 / 1.4	40.0 / 1.0	66.2 / 2.4
	54.7	31.4	74.8	57.8	<u>38.8</u>	70.8
w/ Groupwise Ranking Reward	55.9 / 1.2	33.2 / 1.8	76.4 / 1.6	58.6 / 0.8	39.6 / 0.8	71.6 / 0.8

Table 2: Detailed controlled comparison at global step 140. Each cell reports [Acc / CBIR]. We bold the best RC-Acc in each column and underline the second best.

Method / Quantity	Value
<i>Three-seed method summary</i>	
RLVR	47.40 $\pm$ 0.30
w/ PRM	46.17 $\pm$ 0.60
w/ Pointwise GR	51.70 $\pm$ 0.60
w/ Groupwise Ranking Reward	54.17 $\pm$ 0.84
<i>Matched-seed Groupwise vs. Pointwise comparison</i>	
Groupwise-over-Pointwise margin	+2.47 $\pm$ 0.40
Seed-wise margins	+2.4, +2.9, +2.1
Paired two-sided <i>t</i> -test	<i>t</i> = 10.57, <i>p</i> = 0.0088

Table 3: Multi-seed robustness for the main experiment. Method rows report mean $\pm$ standard deviation of five-benchmark average RC-Acc over three random seeds; the margin rows use matched seeds.

both question types toward much larger reasoning-answer mismatch.

Groupwise Ranking Reward recovers faithfulness without giving up answer gains. On multiple-choice problems, Groupwise Ranking Reward improves Acc from 51.76% to 58.74% while driving IncR below the starting checkpoint, from 5.88% to 3.12%. The open-ended comparison is even cleaner: RLVR and Groupwise Ranking Reward reach the same Acc of 44.71%, yet Groupwise Ranking Reward cuts IncR from 14.71% to 2.75% and CBIR from 3.14% to 0.39%, which lifts RC-Acc from 41.57% to 44.31%. In other words, the open-ended improvement comes almost entirely from better faithfulness rather than better answer accuracy.

The qualitative failure mode still differs by question type. For multiple-choice problems, the

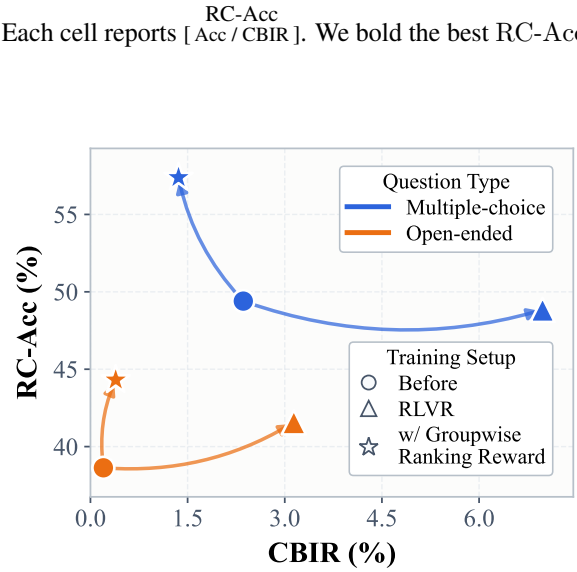


Figure 5: Question-type trade-off trajectories under three training settings. Color denotes question type and marker denotes training setup. For each question type, both trained checkpoints are connected back to the shared before-training model, so the figure shows how RLVR and w/ Groupwise Ranking Reward move the model in RC-Acc versus CBIR space. The panel plots RC-Acc against CBIR. Every percentage is normalized within the same question type.

dominant pattern is still option-selection drift: the reasoning converges to one content answer, but the final boxed output switches to a different option letter or to a nearest-choice guess. For open-ended problems, the mismatch is more often a late revision or answer-type collapse: the model derives one free-form result and then boxes a different scalar, count, or endpoint without a new derivation. These mechanisms differ, but both are strongly amplified by RLVR and sharply reduced by Groupwise Ranking Reward.

Method	Reward pattern	Time per training step (s)			
		Rollout gen.	Reward & advantage	Actor update	Full step
RLVR	Rule-based verifier only	38	0	235	282
w/ PRM	N step-wise PRM scoring passes	55	175	244	590
w/ Pointwise GR	N independent pointwise generative reward calls	62	499	282	1056
w/ Groupwise Ranking Reward	One groupwise generative reward call over verifier-passed rollouts	52	66	245	487

Table 4: Per-step timing comparison across reward designs under the main training setup. Times are measured averages in seconds per training step, rounded to the nearest second and broken down into rollout generation, reward-and-advantage construction, actor update, and full-step time.

Training Setup	Multiple-choice ($n = 1,990$)				Open-ended ($n = 510$)			
	Acc	RC-Acc	IncR	CBIR	Acc	RC-Acc	IncR	CBIR
Before training	51.76%	49.40%	5.88%	2.36%	38.82%	38.63%	1.76%	0.20%
RLVR	55.83%	48.84%	20.40%	6.98%	44.71%	41.57%	14.71%	3.14%
w/ Groupwise Ranking Reward	58.74%	57.39%	3.12%	1.36%	44.71%	44.31%	2.75%	0.39%

Table 5: Question-type breakdown under three training settings. Every percentage is normalized by the total number of examples within the same question type, not by the global benchmark total.

F.2 Training Data Distribution: ViRL vs. GSM8K

A second factor is the RL training data distribution. To study it, we compare two runs that start from the same initialization and are evaluated with the same five-benchmark multimodal inconsistency pipeline, but are trained on two very different data sources: a multimodal image-grounded RL corpus *ViRL* (Wang et al., 2025a), and *GSM8K*, a pure text arithmetic dataset (Cobbe et al., 2021). Both runs use the same GRPO training framework, but the underlying task distributions and correctness signals are not identical across the two datasets. We therefore present this as an empirical cross-dataset comparison rather than a clean modality-only causal ablation.

Figure 6 shows that GSM8K keeps inconsistency low while still improving multimodal evaluation. Relative to the shared initialization, training on GSM8K improves average Acc from 49.12% to 51.96% and RC-Acc from 47.20% to 50.72%. At the same time, IncR drops from 5.04% to 3.44%, and CBIR drops from 1.92% to 1.24%. Across training, the inconsistency curves remain in a relatively tight band and stay well below the shared initialization. This means that text-only RL data can still transfer useful reasoning discipline to multimodal evaluation, even though it does not

directly train image-grounded skills.

ViRL offers higher upside, but the faithfulness trade-off is much less stable. On the same evaluation pipeline, ViRL reaches higher answer accuracy and a stronger best RC-Acc than GSM8K. However, this gain is accompanied by substantially higher inconsistency, and the later part of training degrades further rather than remaining stable. At the end of training, ViRL still keeps Acc above the shared initialization at 52.88%, but overall inconsistency rises to 8.24% and RC-Acc falls to 49.88%. The same pattern appears broadly across benchmarks rather than in one isolated dataset: at the final checkpoints, ViRL has higher inconsistency than GSM8K on all five evaluation sets, for example 10.0% vs. 3.0% on MathVision and 7.8% vs. 3.2% on WeMath.

The training logs support the same asymmetry. We also inspect the judge-flagged inconsistent samples saved from training-time logs. On GSM8K, these cases are nearly absent: only 0.0% – 0.4% of the 500 inspected rollouts per checkpoint are flagged inconsistent, and high-score inconsistent samples never exceed 0.2%. On ViRL, the corresponding numbers are much larger, ranging from 6.2% to 10.8% for all flagged inconsistencies and from 2.4% to 5.6% for inconsistent samples with score > 0.5 . This does not prove

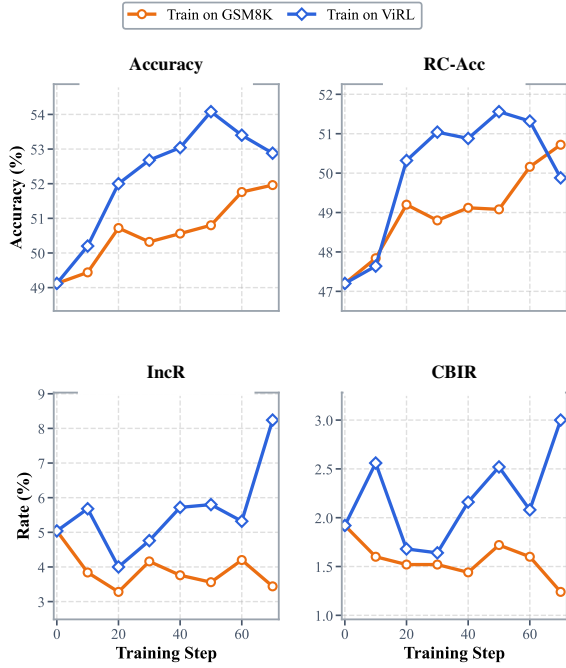


Figure 6: Average evaluation dynamics when the same RL framework is trained on text-only GSM8K or multimodal ViRL. Both runs start from the same initialization and are evaluated with the same five-benchmark multimodal inconsistency pipeline. ViRL reaches higher answer accuracy, but its inconsistency grows much faster during training.

that multimodality alone causes inconsistency, but it does show that the ViRL training distribution exposes the policy to many more internally mismatched yet nontrivially rewarded rollouts, which is consistent with the evaluation-side drift in Figure 6.

Overall, this comparison suggests that the training data distribution has a first-order effect on faithfulness dynamics under RL. GSM8K behaves like a conservative regularizer: it transfers some answer-format and reasoning discipline, keeps inconsistency low, and improves RC-Acc steadily, but its multimodal accuracy ceiling is limited. ViRL has a higher ceiling on multimodal tasks, yet it also produces a much stronger late-stage inconsistency drift. The practical implication is not that text-only data is universally preferable, but that multimodal RL appears to need stronger inconsistency control if we want to keep its accuracy gains from turning into reasoning-answer mismatch.

G Reward Judge Prompts

To make the controlled comparison concrete, we collect the prompt families used by the three trajectory-supervision variants. The PRM follows a visual stepwise process-reward setup, while the two GR-based variants use the same gpt-oss-20b judge backbone with different user instructions.

PRM instruction. For each trajectory step, the PRM uses the following prompt pair and asks the model to emit exactly one token for the current step:

PRM Prompt

System

You are a process reward model. Given the current reasoning step and prior context, output exactly one token: + or -. Use + only if the current step is valid and consistent with the question/context.

User

Question

{question}

Solution Process

{step}

When the trajectory has multiple steps, previous steps are replayed as earlier user turns paired with positive assistant replies, and the current step is queried last. The rollout-level PRM reward is then obtained by averaging the per-step rewards.

Pointwise GR instruction. Pointwise GR uses an instruction that asks the judge to score one textualized trajectory independently:

Pointwise GR Instruction

You are an expert evaluator for reasoning quality. Evaluate the assistant response based on reasoning-quality assessment.

Scoring Rubric

Score the overall reasoning quality on a scalar in [0, 1], where:

- 1.0: clear, logically coherent, and mathematically sound derivation.
- 0.8: mostly solid reasoning with only minor slips that do not break the core logic.
- 0.5: partially valid reasoning; noticeable gaps or mistakes reduce reliability.

- 0.2: largely flawed reasoning with major logical/calculation issues.
- 0.0: reasoning is missing, nonsensical, or clearly fails to solve the task.

Evaluation Guidance

When assigning the single score, consider the overall logical coherence of the steps, correct use of given conditions or visual facts, calculation and transformation quality, clarity/completeness of the conclusion, and whether uncertainty is handled honestly without unsupported guessing.

Note. The original problem may include image input, but you cannot access the image. If the reasoning references information that could plausibly come from the image and is not contradicted by available text, do not treat that as an error by itself; score based on textual reasoning quality.

Output Schema

Return JSON only with this schema:

```
{ "reasoning_feedback":  
  "<short explanation of strengths,  
  weaknesses, and key issues>",  
  "judge_score": <float between 0 and 1>  
}
```

Generate "reasoning_feedback" first and place "judge_score" last.

If some information is missing, still provide the best-effort score from available text.

Problem

{problem}

Response

{response}

Groupwise Ranking Reward instruction. The groupwise variant uses the same judge backbone but replaces scalar scoring with a tie-aware ranking instruction over all verifier-passed candidates for the same prompt:

Groupwise Ranking Instruction

You are ranking multiple candidate solutions to the same reasoning problem.

Problem

{problem}

Reference Answer

```
{reference_answer}
```

Candidate Solutions

```
{candidate_solutions}
```

Ranking Principles

1. Audit the reasoning steps, not just the final answer; flag leaps, contradictions, or missing justification.
2. Prioritise solutions whose final answers align with the reference answer.
3. If solutions are equally correct and use materially similar reasoning, give them the same rank; do not force an ordering without a clear qualitative difference. If all are equally correct, all ranks must be 1.
4. When reasoning differs, choose the mathematically valid, well-supported argument even if multiple solutions reach the same final answer.
5. Penalise calculation mistakes, invalid assumptions, or logical gaps relative to error-free solutions.
6. Keep the comparison focused on mathematical correctness and clarity.

Output Format

Return **strict JSON** with no commentary:

```
{
  "solutions": [
    {
      "index": 1,
      "rank": 1,
      "justification": "Short comparison that explains the placement.",
      "agreement_with_reference": "match" | "different" | "unknown",
      "errors": ["optional list of key mistakes"]
    },
    ...
  ]
}
```

Rules

- Lower rank numbers correspond to better solutions.
- Use the same rank for tied solutions that should be treated equally.
- Include every candidate exactly once.
- If you assign the same rank, the numeric rank fields must be identical (e.g., all 1s for a full tie); do *not* output 1, 2, 3... when you describe them as equal.
- Keep the justification concise, citing key reasoning differences or errors that justify the rank.