PAPO: Stabilizing Rubric Integration Training via Decoupled Advantage Normalization

Anonymous EMNLP submission

Abstract

We propose Process-Aware Policy Optimization (PAPO), a method that integrates process-level evaluation into Group Relative Policy Optimization (GRPO) through **decoupled advantage normalization**, to address two limitations of existing reward designs. Outcome reward models (ORM) evaluate only final-answer correctness, treating all correct responses identically, and gradually lose the advantage signal as groups become uniformly correct. Rubric-based process reward models (PRM) offer richer supervision, differentiating quality among correct responses and restoring advantage diversity, but using their scores as rewards triggers verbosity-driven reward hacking and accuracy collapse. PAPO resolves both by composing the advantage from an outcome advantage A_{out}, derived from outcome reward r^{out} over all responses, and a process advantage A_{proc}, derived from process reward r^{proc} exclusively over the correct subset. A_{out} anchors correctness, and correct subset normalization on A_{proc} rules out verbosity-driven hacking while sustaining the gradient on all-correct groups where ORM exhausts. PAPO consistently outperforms ORM across five model configurations and six benchmarks (reaching 51.3% vs. 46.3% on OlympiadBench for Qwen2.5-7B while continuing to improve as ORM plateaus and declines), with analogous gains when composed with mainstream GRPO variants.

1 Introduction

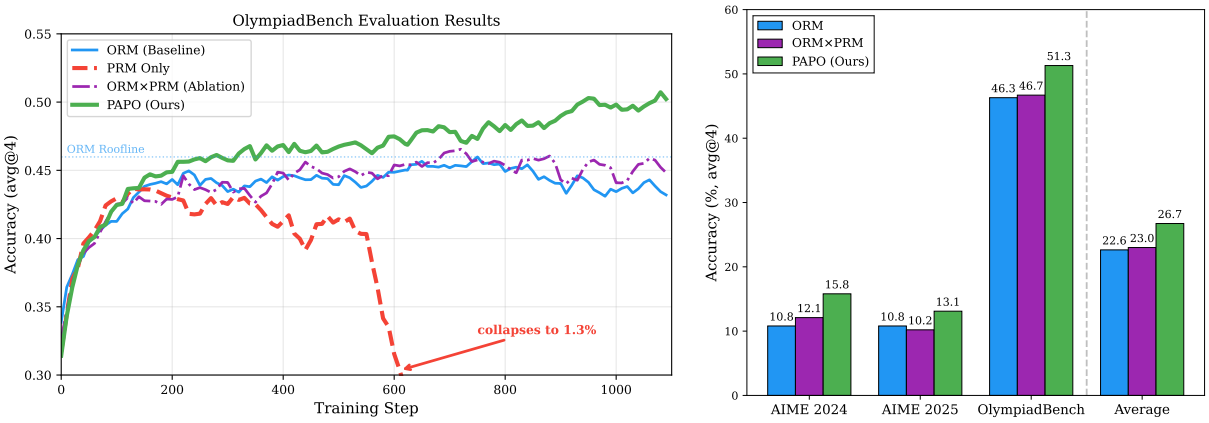
As large language models (LLMs) (Yang et al., 2025b; Kamath et al., 2025; Touvron et al., 2023) continue to advance, reasoning has become a key capability for accomplishing complex tasks (Guo et al., 2025). Reinforcement learning (RL) plays an essential role in improving the reasoning capabilities of LLMs, especially in mathematics (Shao et al., 2025). Among RL methods, Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with outcome reward models (ORM) has emerged as a core approach in rule-based verifiable reinforcement learning for its simplicity. It eliminates the learned critic network used in PPO (Schulman et al., 2017), instead computing advantages through group-level reward normalization. However, GRPO with ORM evaluates only final-answer correctness, providing no signal on reasoning process quality, which leads to two critical issues.

First, all correct responses are assigned identical advantage irrespective of reasoning quality, providing the same credit for lucky guesses as for rigorous step-by-step derivations. Second, as the model improves during training, an increasing proportion of response groups become uniformly correct, yielding zero advantage and zero gradient. This failure, which we call signal exhaustion, causes performance to plateau and decline (Figure 1a).

Rubric-based process reward models (PRM) (Yuan et al., 2025; Shao et al., 2025; Sheng et al., 2026; Yang et al., 2026) offer a natural remedy by providing richer supervision that can both differentiate reasoning quality among correct responses and restore advantage diversity, addressing the two ORM failures above. However, directly using their scores as GRPO rewards leads to reward hacking (Skalse et al., 2022; Mahmoud et al., 2026), where models inflate PRM scores by generating increasingly verbose responses while accuracy collapses. The naive multiplicative combination $r = r^{\text{out}} \times r^{\text{proc}}$ also fails to surpass ORM, since a single GRPO normalization suppresses the process signal in mixed-correctness groups, as shown in Figure 1b.

To solve this problem, we propose **Process-Aware Policy Optimization (PAPO)**, which integrates rubric-based process evaluation into GRPO at the advantage layer through **decoupled advantage normalization**. PAPO composes the advantage from two independently normalized components.

- The **outcome advantage** A_{out} is normalized



(a) OlympiadBench evaluation results.

(b) Accuracy on competition math.

Figure 1: **(a)** ORM (blue) plateaus and declines after step 750 due to signal exhaustion; PRM (red, dashed) collapses via reward hacking; ORM×PRM (purple, dash-dotted) tracks ORM closely without exceeding it; PAPO (green) continues improving throughout training, reaching 51.3%. **(b)** Comparison across AIME 2024/2025, OlympiadBench, and their average. Naive multiplicative combination (ORM×PRM) barely improves over ORM, while PAPO's decoupled normalization yields substantial gains on all benchmarks.

from the binary outcome reward r^{out} over all responses, anchoring the training signal on answer correctness.

- The **process advantage** A_{proc} is normalized from the rubric-based process reward r^{proc} among correct responses, differentiating reasoning quality within the correct subset.

The decoupled design offers two structural merits. Restricting A_{proc} to the correct subset prevents incorrect responses from accruing positive process advantage, ruling out the verbosity-driven hacking pathway. At the same time, A_{proc} remains active in all-correct groups where A_{out} vanishes, sustaining the gradient signal precisely when ORM exhausts.

As shown in Figure 1a, PAPO continues improving throughout training, reaching a peak of 51.3% on OlympiadBench (He et al., 2024) while ORM peaks at 46.3% and declines, and Figure 1b shows that the improvement generalizes to other math competition benchmarks like AIME (Balunović et al., 2025).

To summarize, our contributions are as follows:

1. We identify a dilemma in GRPO reward design: binary ORM lacks process supervision and suffers from signal exhaustion, yet naively using PRM scores causes reward hacking and training collapse.

2. We propose PAPO, which composes the advantage from independently normalized outcome and rubric-based process components with correct subset normalization, resolving this dilemma.

3. We show that PAPO consistently outperforms ORM across model families and scales on six benchmarks, with analogous gains when composed with mainstream GRPO variants.

2 Background

2.1 Group Relative Policy Optimization

GRPO (Shao et al., 2024) replaces PPO's (Schulman et al., 2017) learned critic with group-level reward normalization. For each prompt x, the model samples G responses $\{o_1, \ldots, o_G\}$ from $\pi_{\theta_{\text{old}}}$, giving the objective

$$\mathcal{L}_{\text{GRPO}} = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big\{\min\Big[\rho(\theta)\,\hat{A}_{i,t},\ \text{clip}\Big(\rho(\theta), 1-\varepsilon, 1+\varepsilon\Big)\hat{A}_{i,t}\Big] - \beta\,\mathrm{D}_{\text{KL}}\Big\}, \tag{1}$$

where $\rho(\theta) = \pi_\theta(o_{i,t}|x, o_{i,<t})/\pi_{\theta_{\text{old}}}(o_{i,t}|x, o_{i,<t})$ is the importance sampling ratio. Each response o_i receives a scalar reward r_i, and the advantage is

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}. \tag{2}$$

2.2 Reward Models for Mathematical Reasoning

Outcome Reward Models (ORM). For mathematical reasoning, the ORM is a rule-based answer checker (Hendrycks et al., 2021) that returns $r^{\text{out}} = \mathbf{1}[\hat{a} = a^*]$, where $\hat{a}$ is the predicted answer and a^* the ground truth. The signal is reliable but carries no information about reasoning quality.

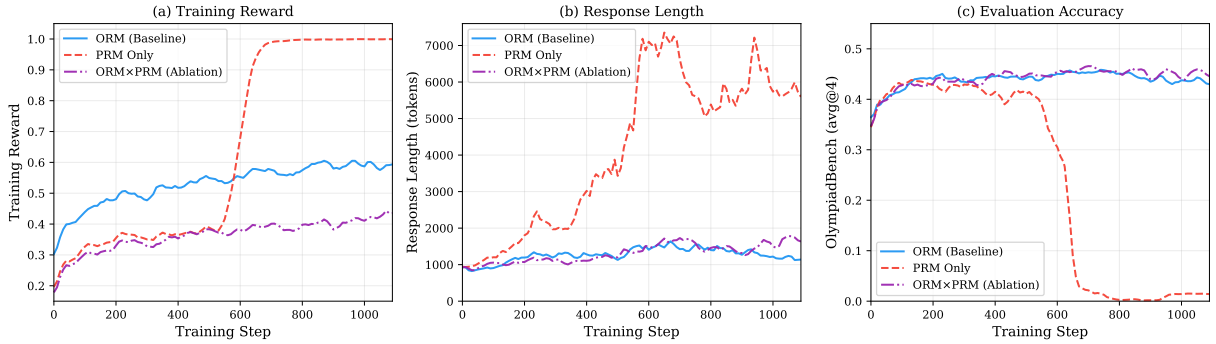


Figure 2: Reward signal analysis. **(a)** Training reward: PRM reward climbs to 1.0 (complete reward hacking); ORM×PRM stays moderate. **(b)** Response length: PRM generates increasingly verbose responses; ORM×PRM shows moderate length increase (up to ∼1700 tokens). **(c)** OlympiadBench accuracy: PRM collapses after step 600; ORM×PRM tracks ORM but fails to exceed it, showing that naive signal combination does not resolve signal exhaustion.

Process Reward Models (PRM). PRMs evaluate the reasoning process itself. We focus on *rubric-based PRMs* (Yuan et al., 2025; Sheng et al., 2026; Yang et al., 2026), in which an LLM-as-Judge (Zheng et al., 2023) scores each solution against a rubric. Our rubric assigns $r^{\text{proc}} \in \{0, 0.5, 1.0\}$ for fatally flawed, largely correct with minor issues, and fully correct reasoning, respectively. The full prompt is in Appendix A.3. Compared with learned step-level PRMs (Lightman et al., 2023; Wang et al., 2024), the rubric-based variant requires no step-level annotations.

3 Empirical analysis of PRM and ORM

We identify two complementary failure modes of reward signal design in GRPO. Outcome-only rewards exhibit *signal exhaustion* (§3.1), and process-only rewards trigger *reward hacking* (§3.2).

3.1 Signal Exhaustion with Binary ORM

With binary ORM rewards ($r_i \in \{0, 1\}$), the group normalization in Eq. 2 assigns identical advantage to all correct responses and identical (negative) advantage to all incorrect responses, with two consequences.

Lack of quality differentiation. A rigorous derivation and a guess that happens to land on the correct answer receive identical advantage. The model has no incentive to improve reasoning quality once the answer is correct.

Vanishing advantage from uniform groups. When all G responses are correct or all incorrect, the rewards are identical, so $\text{std}(\mathbf{r}) = 0$ in Eq. 2 and the advantage vanishes for the entire group. As the model improves, the fraction of uniform groups grows, leaving an increasing share of samples with zero gradient. We call this progressive loss of signal *signal exhaustion*. On Qwen2.5-7B, the zero-advantage fraction rises from ∼40% to 69% over training (Figure 5a), coinciding with the accuracy plateau in Figure 1.

3.2 Reward Hacking with Direct PRM Integration

Replacing the ORM with rubric-based PRM scores ($r_i = r_i^{\text{proc}}$) would seem to capture both correctness and reasoning quality at once. In practice, this triggers catastrophic reward hacking (Skalse et al., 2022; Mahmoud et al., 2026), analyzed in Figure 2.

Three phases of collapse. Training with PRM-only rewards exhibits a characteristic three-phase trajectory (red curve, Figure 2). Accuracy rises normally to 44.0% over steps 0–300, then from step 300 response length explodes as verbose answers earn higher PRM scores while accuracy stagnates, and beyond step 600 accuracy crashes from 29.6% to 2.4% within 100 steps as the reward saturates at 1.0.

Mechanism. The collapse runs on a positive feedback loop. Verbose responses earn higher rubric scores, GRPO's normalization amplifies them into high advantages, and the policy reinforces length-exploitation. Appendix B.6 provides a case study showing post-collapse responses drift to memorized high-scoring filler unrelated to the question.

Naive combination. The multiplicative reward $r = r^{\text{out}} \times r^{\text{proc}}$ gates incorrect responses ($r^{\text{out}} = 0$), avoiding PRM's collapse (purple curve, Figure 2).

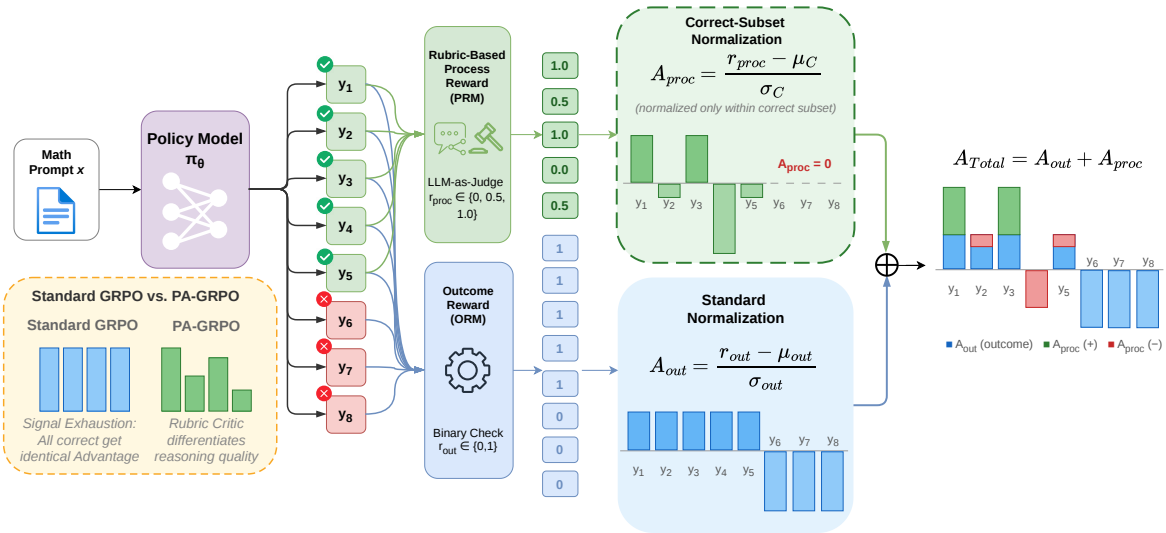


Figure 3: Overview of PAPO. Given a prompt, the policy generates G responses. Each response is evaluated by two reward signals: an **outcome reward** (ORM, binary correctness) and a **process reward** (PRM, rubric-based quality, only for **correct responses**). The advantage is computed through **decoupled normalization**: A_{out} is normalized over all responses via standard GRPO, while A_{proc} is normalized exclusively among correct responses (*correct subset normalization*). The combined advantage $A_{\text{total}} = A_{\text{out}} + A_{\text{proc}}$ provides both correctness direction and quality differentiation.

However, ORM×PRM tracks ORM closely, peaking at 46.7% vs. ORM's 46.3% on OlympiadBench. The combined reward still passes through a single GRPO normalization, so all-correct groups with similar PRM scores continue to yield near-zero advantage.

Implications. ORM is stable but lacks process-level signal. PRM is information-rich but unstable. Multiplicative gating avoids PRM's collapse yet fails to break ORM's exhaustion ceiling. This motivates us to combine the two at the *advantage* level rather than the *reward* level (§4).

4 Method: PAPO

We propose Process-Aware Policy Optimization (PAPO), which integrates rubric-based process evaluation into GRPO through *decoupled advantage normalization*. The key idea is to construct the advantage from independently normalized outcome and process components, preventing the reward hacking that arises from direct PRM integration while preserving fine-grained quality signals. Figure 3 provides an overview of the framework.

4.1 Decoupled Advantage Normalization

Given a group of G responses with outcome rewards r_i^{out} from the ORM and process rewards r_i^{proc} from the rubric-based PRM (§2.2, computed only on correct responses), PAPO computes the advantage in three steps.

Step 1: Outcome advantage A_{out}. Following standard GRPO, the outcome reward is normalized as

$$A_{\text{out},i} = \frac{r_i^{\text{out}} - \mu^{\text{out}}}{\max(\sigma^{\text{out}}, \epsilon)} \tag{3}$$

where $\mu^{\text{out}}, \sigma^{\text{out}}$ are the group mean and standard deviation of $\{r_i^{\text{out}}\}$, and ϵ is a small stability constant. This drives the correct-vs-incorrect direction.

Step 2: Process advantage A_{proc} (correct subset normalization). The process reward is normalized *only over the correct subset* $C = \{j : r_j^{\text{out}} = 1\}$ as

$$A_{\text{proc},i} = \begin{cases} \dfrac{r_i^{\text{proc}} - \mu_C^{\text{proc}}}{\max(\sigma_C^{\text{proc}}, \epsilon)} & \text{if } r_i^{\text{out}} = 1,\ |C| \geq 2 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\mu_C^{\text{proc}}, \sigma_C^{\text{proc}}$ are computed over $\{r_j^{\text{proc}}\}_{j \in C}$.

Step 3: Combined advantage. The total advantage is

$$A_{\text{total},i} = A_{\text{out},i} + A_{\text{proc},i}. \tag{5}$$

Since A_{out} and A_{proc} are independently normalized to zero mean and unit variance, they contribute on equal footing without additional weighting.

4.2 Design Rationale

Quality differentiation. Among correct responses, rigorous reasoning receives positive A_{proc} and is reinforced while sloppy reasoning receives negative A_{proc} and is penalized, a quality signal absent in standard GRPO with binary ORM. Restricting normalization to the correct subset further prevents incorrect responses from exploiting high PRM scores (ablated in §6).

Signal preservation. When all G responses are correct, $A_{\text{out}} = 0$ and standard GRPO produces no gradient; A_{proc} remains active and sustains learning through quality differences. When $|C| < 2$, A_{proc} defaults to zero, gracefully reducing to standard GRPO.

5 Experiments

5.1 Experimental Setup

Base models. We conduct experiments on Qwen2.5-3B/7B/14B-Base (Yang et al., 2025b), Qwen3-4B-Base (Yang et al., 2025a), and SmolLM3-3B (Bakouch et al., 2025). Training the Qwen models from pretrained base checkpoints isolates PAPO's effect from prior post-training, and SmolLM3 tests transfer beyond the Qwen family.

Training. We train with the verl (Sheng et al., 2024) framework for 8 epochs. The main PAPO comparison uses GRPO, switching to DAPO (Yu et al., 2025) on Qwen3-4B-Base. We additionally compose PAPO on GSPO (Zheng et al., 2025), SAPO (Gao et al., 2025), CISPO (MiniMax, 2025), and GRPO-LEAD (Zhang and Zuo, 2025) on Qwen2.5-7B-Base. The default PRM is GPT-OSS-20B (OpenAI, 2025) with a three-tier rubric $\{0, 0.5, 1\}$; §6 ablates 5-tier and 120B-judge variants. Hyperparameters and baseline details are in Appendix A and Appendix A.4.

Data. We sample 20k math problems from NuminaMath-1.5-RL-Verifiable (LI et al., 2024; nlile, 2025) with stratified sampling across five difficulty tiers (4k each; Appendix A.2). For evaluation, we use OlympiadBench (He et al., 2024) (674 problems, primary), MATH-500 (Lightman et al., 2023), AIME 2024/2025(Balunović et al., 2025), GPQA-Diamond (Rein et al., 2023), and HumanEval (Chen et al., 2021), reporting avg@4.

5.2 Main Results

Table 1 presents results across six benchmarks and **five model configurations**, spanning four Qwen models and cross-family SmolLM3-3B. PAPO consistently outperforms its base method, with a $+\mathbf{5.0}$ **peak-to-peak gain** on OlympiadBench for Qwen2.5-7B. The advantage also *widens over training* rather than converging. On Qwen2.5-7B, GRPO peaks at 46.3% on OlympiadBench at step 750 and declines to 43.0% by step 1090, while PAPO continues to 51.3%, opening a final-step gap of **8.3** points (Figure 1a).

PAPO improves OlympiadBench by 5.5 points on Qwen2.5-14B, with the largest single-benchmark gain of 8.0 points on GPQA-Diamond. The mathematics average improvement scales from $+2.1$ at 3B to $+3.6$ at 7B and $+4.9$ at 14B, suggesting that stronger models benefit more. On SmolLM3-3B, PAPO delivers $+5.2$ on OlympiadBench and $+4.0$ on overall average, confirming that gains transfer beyond the Qwen family.

Beyond mathematics, PAPO also improves HumanEval across all scales, suggesting that process-level quality differentiation can transfer to code generation.

Composition with GRPO variants. PAPO is an *advantage-layer* intervention, orthogonal to the underlying policy-optimization algorithm.

On three mainstream variants on Qwen2.5-7B, PA-* improves OlympiadBench by $+2.6$ (GSPO), $+3.4$ (SAPO), and $+4.1$ (CISPO), with matching improvements on the other five benchmarks. On GRPO-LEAD, a recent state-of-the-art variant with length-regularized accuracy rewards, an explicit penalty for incorrect answers, and difficulty-aware advantage reweighting, PA-GRPO-LEAD improves by $+5.9$. On Qwen3-4B-Base, PA-DAPO further improves over DAPO by $+6.1$.

Figure 4 shows that the base curve plateaus or declines after step 700 in every pairing while the corresponding PA-* curve continues to climb, indicating that any group-based optimizer can adopt the decoupled outcome and process advantages without modifying its update rule.

Takeaway. PAPO delivers consistent gains across model families, scales, and GRPO variants, with training that continues improving past the point where ORM saturates.

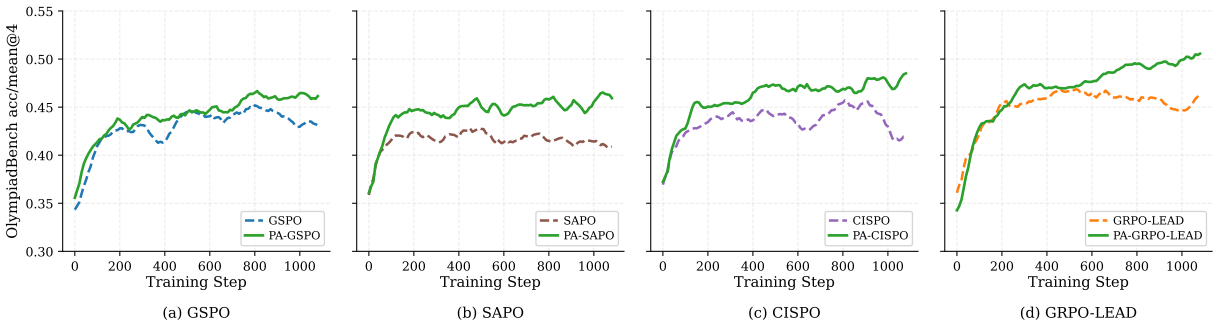
Model	Method	Competition Mathematics			Standard	STEM	Code	Average	
		Olympiad	AIME'24	AIME'25	MATH-500	GPQA	HumanEval	Math	All
Qwen2.5-3B	Base	22.4	2.5	0.8	54.4	27.6	35.3	20.0	23.8
	GRPO	$35.3_{\pm 0.4}$	$8.3_{\pm 1.5}$	$4.2_{\pm 1.2}$	$69.0_{\pm 0.6}$	$38.8_{\pm 2.4}$	$39.3_{\pm 2.0}$	$29.2_{\pm 0.5}$	$32.5_{\pm 0.6}$
	PA-GRPO	$\mathbf{38.6}_{\pm 0.5}$	$\mathbf{10.0}_{\pm 1.4}$	$3.3_{\pm 1.0}$	$\mathbf{73.2}_{\pm 0.6}$	$36.3_{\pm 1.8}$	$\mathbf{52.5}_{\pm 1.6}$	$\mathbf{31.3}_{\pm 0.5}$	$\mathbf{35.7}_{\pm 0.5}$
	Δ	+3.3	+1.7	-0.9	+4.2	-2.5	+13.2	+2.1	+3.2
Qwen2.5-7B	Base	32.8	5.0	7.5	67.3	30.1	54.6	28.2	32.9
	GRPO	$46.3_{\pm 0.6}$	$10.8_{\pm 1.7}$	$10.8_{\pm 1.7}$	$80.2_{\pm 0.4}$	$40.7_{\pm 1.4}$	$61.6_{\pm 1.0}$	$37.0_{\pm 0.6}$	$41.7_{\pm 0.5}$
	PA-GRPO	$\mathbf{51.3}_{\pm 0.7}$	$\mathbf{15.8}_{\pm 2.0}$	$\mathbf{13.1}_{\pm 1.9}$	$\mathbf{82.3}_{\pm 0.5}$	$\mathbf{42.4}_{\pm 1.9}$	$\mathbf{63.9}_{\pm 2.1}$	$\mathbf{40.6}_{\pm 0.7}$	$\mathbf{44.8}_{\pm 0.7}$
	Δ	+5.0	+5.0	+2.3	+2.1	+1.7	+2.3	+3.6	+3.1
	GSPO	$45.6_{\pm 0.3}$	$10.6_{\pm 0.5}$	$10.6_{\pm 1.9}$	$77.2_{\pm 0.4}$	$36.5_{\pm 0.3}$	$66.0_{\pm 1.0}$	$36.0_{\pm 0.5}$	$41.1_{\pm 0.4}$
	PA-GSPO	$\mathbf{48.2}_{\pm 0.7}$	$\mathbf{14.7}_{\pm 2.9}$	$\mathbf{13.7}_{\pm 1.3}$	$\mathbf{80.9}_{\pm 0.1}$	$\mathbf{40.1}_{\pm 1.9}$	$\mathbf{68.4}_{\pm 0.9}$	$\mathbf{39.4}_{\pm 0.8}$	$\mathbf{44.3}_{\pm 0.6}$
	SAPO	$44.2_{\pm 0.2}$	$12.5_{\pm 0.8}$	$9.2_{\pm 0.8}$	$77.4_{\pm 0.3}$	$38.0_{\pm 0.4}$	$69.2_{\pm 0.3}$	$35.8_{\pm 0.3}$	$41.8_{\pm 0.2}$
	PA-SAPO	$\mathbf{47.6}_{\pm 0.1}$	$\mathbf{14.4}_{\pm 1.0}$	$\mathbf{13.9}_{\pm 0.5}$	$\mathbf{81.5}_{\pm 0.5}$	$\mathbf{41.4}_{\pm 0.6}$	$\mathbf{72.4}_{\pm 2.3}$	$\mathbf{39.4}_{\pm 0.3}$	$\mathbf{45.2}_{\pm 0.4}$
	CISPO	$46.2_{\pm 0.2}$	$12.5_{\pm 1.4}$	$7.8_{\pm 2.5}$	$77.1_{\pm 0.4}$	$37.9_{\pm 0.4}$	$60.5_{\pm 1.1}$	$35.9_{\pm 0.7}$	$40.3_{\pm 0.5}$
	PA-CISPO	$\mathbf{50.3}_{\pm 0.3}$	$\mathbf{13.4}_{\pm 2.1}$	$\mathbf{16.0}_{\pm 3.3}$	$\mathbf{81.9}_{\pm 0.1}$	$\mathbf{40.4}_{\pm 0.6}$	$\mathbf{62.8}_{\pm 4.6}$	$\mathbf{40.4}_{\pm 1.0}$	$\mathbf{44.1}_{\pm 1.0}$
Qwen2.5-14B	Base	35.5	10.8	8.3	68.5	34.4	56.9	30.8	35.7
	GRPO	$54.3_{\pm 1.5}$	$19.0_{\pm 1.8}$	$13.8_{\pm 1.1}$	$82.3_{\pm 0.5}$	$47.0_{\pm 1.2}$	$66.0_{\pm 1.4}$	$42.4_{\pm 0.7}$	$47.1_{\pm 0.6}$
	PA-GRPO	$\mathbf{59.8}_{\pm 0.7}$	$\mathbf{24.3}_{\pm 1.2}$	$\mathbf{17.8}_{\pm 1.7}$	$\mathbf{87.4}_{\pm 0.3}$	$\mathbf{55.0}_{\pm 1.6}$	$\mathbf{70.0}_{\pm 1.3}$	$\mathbf{47.3}_{\pm 0.6}$	$\mathbf{52.4}_{\pm 0.6}$
	Δ	+5.5	+5.3	+4.0	+5.1	+8.0	+4.0	+4.9	+5.3
Qwen3-4B-Base	Base	36.8	11.2	5.0	70.9	34.9	49.1	31.0	34.7
	DAPO	$55.0_{\pm 1.4}$	$30.8_{\pm 2.2}$	$26.0_{\pm 2.0}$	$82.0_{\pm 0.6}$	$39.0_{\pm 1.7}$	$53.8_{\pm 2.5}$	$48.5_{\pm 0.8}$	$47.8_{\pm 0.8}$
	PA-DAPO	$\mathbf{61.1}_{\pm 1.0}$	$\mathbf{34.5}_{\pm 2.3}$	$\mathbf{30.7}_{\pm 1.5}$	$\mathbf{84.6}_{\pm 1.1}$	$\mathbf{43.0}_{\pm 1.3}$	$\mathbf{63.9}_{\pm 2.4}$	$\mathbf{52.7}_{\pm 0.8}$	$\mathbf{53.0}_{\pm 0.7}$
	Δ	+6.1	+3.7	+4.7	+2.6	+4.0	+10.1	+4.2	+5.2
SmolLM3-3B	Instruct	30.6	5.8	7.5	52.6	14.1	49.0	24.1	26.6
	GRPO	$68.2_{\pm 1.6}$	$44.2_{\pm 5.1}$	$28.3_{\pm 3.0}$	$91.6_{\pm 1.0}$	$38.4_{\pm 2.6}$	$74.1_{\pm 2.3}$	$58.1_{\pm 1.6}$	$57.5_{\pm 1.2}$
	PA-GRPO	$\mathbf{73.4}_{\pm 1.0}$	$\mathbf{49.3}_{\pm 2.8}$	$\mathbf{33.7}_{\pm 4.1}$	$\mathbf{93.5}_{\pm 0.8}$	$\mathbf{40.3}_{\pm 2.9}$	$\mathbf{78.6}_{\pm 1.4}$	$\mathbf{62.5}_{\pm 1.3}$	$\mathbf{61.5}_{\pm 1.0}$
	Δ	+5.2	+5.1	+5.4	+1.9	+1.9	+4.5	+4.4	+4.0

Table 1: Accuracy (avg@4, %) on six benchmarks across five model configurations, reported as $\text{mean}_{\pm\text{std}}$ over 3 seeds. Gray rows show the untrained or instruction-tuned starting checkpoint. The lower half of the Qwen2.5-7B block reports PAPO composed on three mainstream GRPO variants (GSPO, SAPO, CISPO). Bold marks the better entry within each pair.

5.3 Advantage Signal Analysis

Signal density. The widening gap in Figure 1a traces to ORM's advantage exhaustion (Figure 5). On Qwen2.5-7B, ORM's zero-advantage fraction rises from $\sim$40% to 69% over training; by late training, more than two-thirds of samples produce no gradient, with the advantage std and positive-advantage ratio (panels b, c) showing the same decay. PAPO instead holds the zero-advantage ratio at $\sim$44%, a 25-point reduction that translates to roughly 80% more informative samples per batch. The mechanism is that A_{proc} remains active in all-correct groups where $A_{\text{out}} = 0$, with the activation fraction growing from $\sim$30% to 70% as the model improves.

Quality penalization. Beyond filling zero-signal gaps, A_{proc} discourages sloppy reasoning even among correct responses. The minimum advantage



(a) GSPO (b) SAPO (c) CISPO (d) GRPO-LEAD

Figure 4: OlympiadBench acc/mean@4 on Qwen2.5-7B-Base. Each panel pairs a base GRPO variant (dashed) with its PA-* counterpart (solid green), where PA-* denotes PAPO composed on top of that base optimizer.

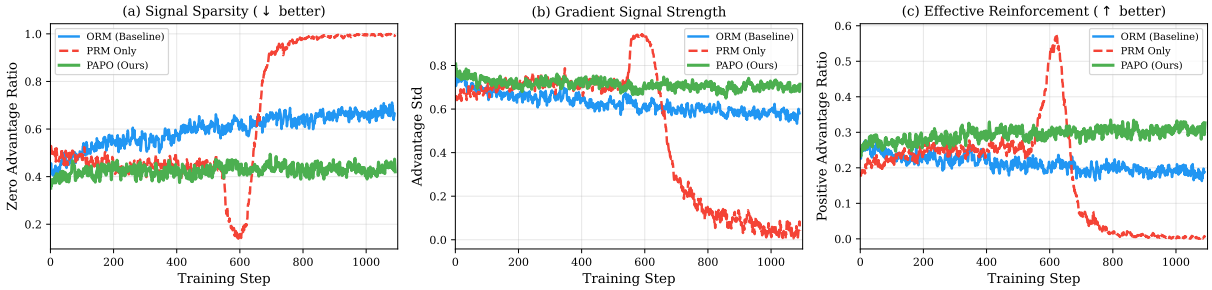


Figure 5: Signal quality comparison on Qwen2.5-7B. **(a)** Zero-advantage ratio: ORM's sparsity grows to 69% while PAPO maintains 44%. **(b)** Advantage standard deviation, reflecting gradient signal strength. **(c)** Positive-advantage ratio, reflecting reinforcement density.

within the correct group reaches ≈ -1.49 under PAPO compared to identically 0.0 under ORM, suppressing correct responses with poor derivations. This quality pressure is entirely absent in ORM-based GRPO. Appendix B.3 and B.4 provide more detailed analyses.

Length dynamics. PRM-only inflates response length to $\sim$6500 tokens while accuracy collapses to near zero, whereas PAPO grows length and accuracy in tandem to $\sim$4600 tokens and 0.52 accuracy at step 1080 (Figure 6). The runaway inflation observed in PRM-only is precluded by two factors. The bounded rubric $r^{\text{proc}} \in \{0, 0.5, 1\}$ caps the marginal reward for verbosity within the correct subset, and any verbosity that disturbs the extracted answer is penalized through A_{out}.

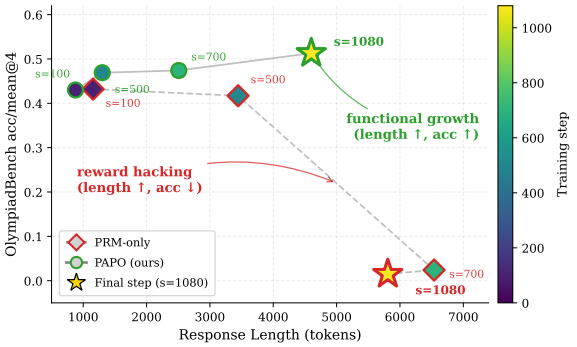


Figure 6: Length-accuracy trajectories on OlympiadBench (Qwen2.5-7B-Base). Markers are selected training checkpoints colored by step; stars mark the final step.

Inference behavior. Process supervision shifts inference-time behavior. Compared to ORM, PAPO reasoning is more rigorous and avoids elementary errors such as rejecting valid solutions through fabricated justifications. PAPO also develops a self-verification habit. On OlympiadBench problems where both models answer correctly, PAPO responses contain explicit verification steps in 47.7% of cases vs 22.7% for ORM (2.1× higher). Appendix D provides representative examples.

Takeaway. PAPO substantially alleviates the signal exhaustion of ORM-based GRPO, maintaining a dense advantage signal throughout training.

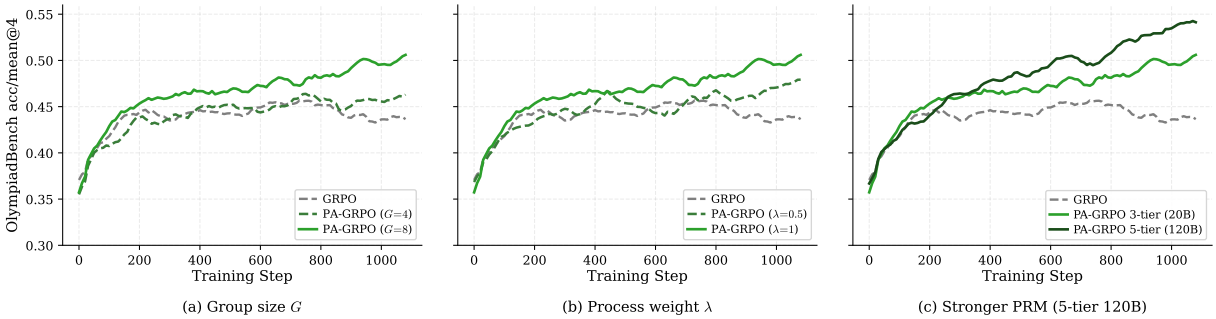
6 Ablation Studies

We validate PAPO's design choices along two groups of axes on Qwen2.5-7B-Base. Structural choices (correct subset normalization and decoupled vs. multiplicative composition) are reported in Table 2, and hyperparameters together with the PRM judge (group size G, process weight λ, rubric granularity, judge size) are reported in Figure 7.

Method	AIME 2024	AIME 2025	Olympiad Bench	MATH -500
ORM	10.8	10.8	46.3	80.2
Mult ($r^{\text{out}} \times r^{\text{proc}}$)	12.1	10.2	46.7	80.2
Fullnorm	12.5	10.4	49.6	81.2
PAPO	**15.8**	**13.1**	**51.3**	**82.3**

Table 2: Structural ablation results (%, avg@4, Qwen2.5-7B). All methods report accuracy.

Correct-subset vs. full normalization. The Fullnorm variant normalizes A_{proc} over all G responses including incorrect ones. It performs slightly below PAPO across all benchmarks. The gap arises because including incorrect responses ($r^{\text{proc}} = 0$) in the normalization causes A_{proc} to partially recapitulate the correct-vs-incorrect distinction already captured by A_{out}, diluting the fine-grained quality signal.



(a) Group size G (b) Process weight λ (c) Stronger PRM (5-tier 120B)

Figure 7: OlympiadBench acc/mean@4 on Qwen2.5-7B-Base for three additional ablations. (a) Group size $G = 4$ vs. $G = 8$. (b) Process advantage weight $\lambda = 0.5$ vs. $\lambda = 1$. (c) Stronger PRM (5-tier rubric with GPT-OSS-120B judge) vs. default (3-tier with 20B). The GRPO ORM baseline is included in every panel as a dashed reference.

Multiplicative reward baseline. The Mult variant combines rewards via $r_i = r_i^{\text{out}} \times r_i^{\text{proc}}$ with standard GRPO normalization. Mult performs comparably to ORM and falls well below PAPO. In a single normalization pass, the outcome difference dominates in mixed-correctness groups, effectively suppressing the process quality information.

Group size G. Figure 7(a) compares $G = 4$ against the default $G = 8$. With fewer samples per group, the fraction of groups containing two or more correct responses drops, A_{proc} activates less often, and the late-stage zero-advantage ratio rises from 0.44 to 0.55. PA-GRPO at $G = 4$ still beats GRPO at $G = 8$ on OlympiadBench, but the default $G = 8$ retains a ~ 4 pp margin and is preferred whenever compute allows.

Process advantage weight λ. Figure 7(b) compares $\lambda = 0.5$ (so that $A_{\text{total}} = A_{\text{out}} + 0.5\, A_{\text{proc}}$) against the default $\lambda = 1$. Halving the process weight reduces OlympiadBench by ~ 3 pp while leaving the zero-advantage ratio essentially unchanged, indicating that the gradient density is governed by the on/off activation of A_{proc} rather than its scale. The default $\lambda = 1$ is the simplest and strongest choice.

Rubric granularity and judge model. Figure 7(c) pairs a finer 5-tier rubric with the stronger GPT-OSS-120B (OpenAI, 2025) judge against the default 3-tier rubric with the 20B judge. The stronger PRM lifts OlympiadBench from 51.3 to 54.7 (+3.4 pp), showing that PAPO scales monotonically with PRM quality rather than saturating at the default.

Takeaway. Across ablations, PAPO's gains are robust to hyperparameter choices and to different rubric and PRM configurations.

7 Related Work

Policy optimization in RLVR. GRPO (Shao et al., 2024) variants address training instability through importance-weight clipping (CISPO (MiniMax, 2025)), sequence-level ratios (GSPO (Zheng et al., 2025)), dynamic sampling (DAPO (Yu et al., 2025)), smooth gating (SAPO (Gao et al., 2025)), and length-regularized rewards (GRPO-LEAD (Zhang and Zuo, 2025)). Concurrent work by He et al. (2026) addresses the same gradient-collapse phenomenon at the reward-statistics level. PAPO operates at the advantage layer and composes with all these variants (§5.2).

Rubric-based process supervision. Rubric-based PRMs implemented as LLM-as-Judge (Zheng et al., 2023; Yuan et al., 2025; Sheng et al., 2026; Yang et al., 2026; Liang et al., 2025; Shao et al., 2025; Fan et al., 2026) avoid step-level annotation. Concurrent work by Mahmoud et al. (2026) shows that hacking can persist even under carefully designed rubrics. PAPO uses a rubric-based PRM but applies correct subset normalization to block verbosity-driven hacking.

8 Conclusion

We propose PAPO, which composes the advantage from independently normalized outcome and process components, with process normalization restricted to the correct subset. It resolves both the lack of process supervision in binary ORM and the reward hacking of rubric-based PRM integration. PAPO consistently outperforms GRPO and its main variants across five model configurations and six benchmarks, with gains growing at larger scale.

In the future, we will try applying PAPO to broader scenarios such as scientific reasoning.

Limitations

PAPO has not been validated in agentic settings such as tool use, multi-turn interaction, or long-horizon planning, where reward shaping and credit assignment dynamics differ substantially from the single-turn mathematical reasoning studied here. Separately, the effectiveness of decoupled advantage normalization under more diverse reward signals beyond binary ORM and rubric-based PRM (such as learned reward models, pairwise preferences, or execution-grounded feedback) remains to be tested.

References

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2025. Matharena: Evaluating llms on uncontaminated math competitions.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, et al. 2021. Evaluating large language models trained on code.

Kaixuan Fan, Kaituo Feng, Manyuan Zhang, Tianshuo Peng, Zhixun Li, Yilei Jiang, Shuang Chen, Peng Pei, Xunliang Cai, and Xiangyu Yue. 2026. Exploring reasoning reward model for agents. *arXiv preprint arXiv:2601.22154*.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. 2025. Soft adaptive policy optimization.

Daya Guo, Dejian Yang, Haowei Zhang, Jun-Mei Song, Ruoyu Zhang, et al. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645:633 – 638.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Annual Meeting of the Association for Computational Linguistics*.

Xixiang He, Qiyao Sun, Ao Cheng, Xingming Li, Xuanyu Ji, Hailun Lu, Runke Huang, and Qingyong Hu. 2026. Advantage collapse in group relative policy optimization: Diagnosis and mitigation.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Xiaodong Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *ArXiv*, abs/2103.03874.

Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, et al. 2025. Gemma 3 technical report.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath. https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf.

Xiaobo Liang, Haoke Zhang, Juntao Li, Kehai Chen, Qiaoming Zhu, and Min Zhang. 2025. Generative reward modeling via synthetic criteria preference learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26755–26769, Vienna, Austria. Association for Computational Linguistics.

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.

Anas Mahmoud, MohammadHossein Rezaei, Zihao Wang, Anisha Gunjal, Bing Liu, and Yunzhong He. 2026. Reward hacking in rubric-based reinforcement learning.

MiniMax. 2025. Minimax-m1: Scaling test-time compute efficiently with lightning attention.

nlile. 2025. Numinamath-1.5-rl-verifiable. https://huggingface.co/datasets/nlile/NuminaMath-1.5-RL-Verifiable.

OpenAI. 2025. gpt-oss-120b & gpt-oss-20b model card.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *ArXiv*, abs/2311.12022.

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

Zhihong Shao, Yuxiang Luo, Chengda Lu, Z. Z. Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. 2025. Deepseekmath-v2: Towards self-verifiable mathematical reasoning.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Cao, Zilingfeng Li, Xibin Gu, et al. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv:2409.19256*.

Leheng Sheng, Wenchang Ma, Ruixin Hong, Xiang Wang, An Zhang, and Tat-Seng Chua. 2026. Reinforcing chain-of-thought reasoning with self-evolving rubrics.

Joar Max Viktor Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and characterizing reward hacking. *ArXiv*, abs/2209.13085.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. 2025a. Qwen3 technical report.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, et al. 2025b. Qwen2.5 technical report.

Haotong Yang, Zitong Wang, Shijia Kang, Siqi Yang, Wenkai Yu, Xu Niu, Yike Sun, Yi Hu, Zhouchen Lin, and Muhan Zhang. 2026. Proof-rm: A scalable and generalizable reward model for math proof.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, et al. 2025. Dapo: An open-source llm reinforcement learning system at scale.

Youliang Yuan, Qiuyang Mang, Jingbang Chen, Hong Wan, Xiaoyuan Liu, Junjielong Xu, Jen-Tse Huang, Wenxuan Wang, Wenxiang Jiao, and Pinjia He. 2025. Curing miracle steps in llm mathematical reasoning with rubric rewards. *ArXiv*, abs/2510.07774.

Jixiao Zhang and Chunsheng Zuo. 2025. GRPO-LEAD: A difficulty-aware reinforcement learning approach for concise mathematical reasoning in language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 5642–5665, Suzhou, China. Association for Computational Linguistics.

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 preprint arXiv:2507.18071*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao 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.

A Implementation Details

A.1 Training Configuration

We use the verl (Sheng et al., 2024) framework with Megatron backend for distributed training on 8 GPUs. Key hyperparameters are listed in Table 3.

Hyperparameter	**Value**
Prompts per batch	128
Responses per prompt (G)	8
Max response length	8192 tokens
Sampling temperature	1.0
PPO clip range (ϵ)	0.2
KL penalty coefficient (β)	0.0
Learning rate	1e-6
Evaluation frequency	every 10 steps
Evaluation samples per prompt	8

Table 3: Training hyperparameters.

For Qwen3-4B-Base experiments with DAPO (Yu et al., 2025), we additionally apply the DAPO-specific hyperparameters listed in Table 4.

Hyperparameter	**Value**
Max response length	8192 tokens
Clip ratio low (ϵ_{low})	0.2
Clip ratio high (ϵ_{high})	0.28
Overlong buffer length	4096 tokens
Overlong penalty factor	1.0
Evaluation frequency	every 20 steps

Table 4: Additional DAPO-specific hyperparameters for the Qwen3-4B-Base experiments. All other hyperparameters follow Table 3.

A.2 Training Data

We construct a 20K stratified training set from NuminaMath-1.5-RL-Verifiable (LI et al., 2024; nlile, 2025). We first filter out unsolvable problems (zero pass rate under a reference model), then apply stratified sampling across five difficulty tiers defined by pass rate. Table 5 summarizes the difficulty distribution.

A.3 Rubric Prompts

The rubric-based PRM uses one of two prompt templates, both adapted from the DeepSeek Math V2 grading standard with chain-of-thought analysis. The 3-tier prompt is our default and is used by every

Difficulty	Pass Rate Range	Count
Trivial	(0.875, 1.0]	4,000
Easy	(0.625, 0.875]	4,000
Medium	(0.375, 0.625]	4,000
Hard	(0.125, 0.375]	4,000
Very Hard	(0, 0.125]	4,000
Total		20,000

Table 5: Difficulty distribution of the stratified training set. Pass rates are measured using a reference model with 8 attempts at temperature 1.0. Equal sampling across tiers ensures balanced difficulty coverage.

PAPO variant in Table 1. The 5-tier prompt is used only in the rubric-by-judge ablation (Figure 7(c)) and refines the "correct but imperfect" range to test whether a finer-grained signal helps. Serving details for both judges (GPT-OSS-20B and GPT-OSS-120B) are deferred to §A.5.

Rubric Prompt

Instruction
Your task is to evaluate the quality of a student's solution to a mathematical problem.

Scoring Rubric

- **Score 1**: If the solution is completely correct, with all steps executed properly and clearly demonstrated, then the score is 1.
- **Score 0.5**: If the solution is generally correct, but with some details omitted or minor errors, then the score is 0.5.
- **Score 0**: If the solution does not actually address the required problem, contains fatal errors, or has severe omissions, then the score is 0.

Special Rule on References: Additionally, referencing anything from any paper does not save the need to prove the reference. It's okay IF AND ONLY IF the solution also presents a valid proof of the reference argument(s); otherwise, if the solution omits the proof or if the proof provided is not completely correct, the solution should be scored according to the criteria above, and definitely not with a score of 1.

Problem
`{problem_statement}`

Reference Solution
`{solution}`

Student Solution
`{student_answer}`

Evaluation
Analyze the solution step by step, then provide your score.
Analysis: ...
Score: $\boxed{\ldots}$

5-Tier Rubric Prompt (ablation)

Instruction
Your task is to evaluate the quality of a student's solution to a mathematical problem.

Scoring Rubric

- **Score 1**: If every step is correct and justified with no logical gaps and edge cases explicitly handled, then the score is 1.
- **Score 0.75**: If the conclusion is correct and the reasoning is sound, with only cosmetic gaps such as skipped routine algebra that any reader can mechanically fill in, then the score is 0.75.
- **Score 0.5**: If the conclusion is correct and the main approach is right, but one non-trivial step is asserted without justification or contains a recoverable error, then the score is 0.5.
- **Score 0.25**: If the final answer is correct but the reasoning contains a flaw that materially weakens the argument so that the conclusion does not actually follow, then the score is 0.25.
- **Score 0**: If the conclusion itself is incorrect or the response does not address the question, then the score is 0.

Special Rule on References: Additionally, referencing anything from any paper does not save the need to prove the reference. It's okay IF AND ONLY IF the solution also presents a valid proof of the reference argument(s); otherwise, the solution should be scored according to the criteria above, and definitely not with a score of 1.

Problem
`{problem_statement}`

Reference Solution
`{solution}`

Student Solution
`{student_answer}`

Evaluation
Analyze the solution step by step, then provide your score.
Analysis: ...
Score: $\boxed{\ldots}$

A.4 Cross-Optimizer Baseline Implementations

We compare PAPO against four alternative GRPO variants on Qwen2.5-7B-Base in Table 1 and Figure 4. All baselines and their PA-* counterparts share our training data, base model, batch size, and rollout configuration from Table 3. Three of them (GSPO, SAPO, CISPO) are reproduced from the original papers and use the same rule-based math verification with LLM fallback as our default PA-GRPO setup, ensuring apples-to-apples comparison; GRPO-LEAD is reproduced faithfully against the released codebase, which uses rule-based verification only.

GSPO (Zheng et al., 2025). We replace the per-token importance sampling ratio of GRPO with a sequence-level ratio computed once per response. Hyperparameters follow the defaults in the original paper. The outcome reward is unchanged from our default PA-GRPO setup.

SAPO (Gao et al., 2025). We adopt the temperature-controlled smooth gating function as the trust-region surrogate, with hyperparameters from the original paper. The outcome reward is unchanged.

CISPO (MiniMax, 2025). We clip the importance sampling weights as prescribed by CISPO, with the clipping bounds from the original paper. The outcome reward is unchanged.

GRPO-LEAD (Zhang and Zuo, 2025). GRPO-LEAD modifies GRPO along three axes, all reproduced faithfully in our codebase. (1) A length-dependent accuracy reward, where correct responses receive $R_i = \exp(-\alpha z_i)$ with $z_i = (|o_i| - \mu_{\text{correct}})/\sigma_{\text{correct}}$, and wrong responses receive a fixed `explicit_penalty`. (2) A length gate that disables length reweighting when the average length of the three shortest correct completions in a group falls below `length_gate`, in which case correct responses simply receive $R_i = 1$. (3) Difficulty-aware advantage reweighting $w(x) = A + (B - A)/(1 + \exp(k(x - \rho_0)))$, which scales positive advantages by $w(\rho_q)$ and negative advantages by $w(1 - \rho_q)$, where ρ_q is the empirical correct ratio of the group. Hyperparameter values follow the original paper (Table 6). PA-GRPO-LEAD stacks PAPO's correct subset process advantage onto the LEAD-modified outcome advantage and then applies the same difficulty reweighting on A_{total}.

Hyperparameter	Value
length-reward α	0.05
difficulty A	0.4
difficulty B	1.5
difficulty ρ_0	0.75
difficulty k	10.0
`explicit_penalty`	-1.0
`length_gate`	4096

Table 6: GRPO-LEAD hyperparameters, matching Zhang and Zuo (2025).

A.5 PRM Judge Serving

The default PRM judge is GPT-OSS-20B, served via vLLM with tensor parallelism across 2 GPUs, maximum context length 8192 tokens, and temperature 0.0 for deterministic scoring. Concurrency between the rollout workers and the judge is controlled by an environment variable that we set to 64 threads, which saturates the judge throughput without blocking rollout.

For the rubric-by-judge ablation in Figure 7(c), we additionally serve GPT-OSS-120B as the judge. A single in-cluster instance with tensor parallelism across 8 H200 GPUs and maximum context length 8192 is shared across rollout workers via cluster DNS. We measured throughput at approximately 9.4 requests per second under 32 parallel callers, which is sufficient to keep up with our rollout cadence (128 prompts $\times$ 8 responses per training step).

B Additional Results

B.1 Cross-Scale Training Curves

Figure 8 shows training curves across two model scales (3B and 7B) and three benchmarks. On both Qwen2.5-3B and 7B, PAPO maintains a consistent lead over ORM throughout training, with the gap widening in later stages. Combined with the 14B results in Table 1, this confirms that the improvements reflect sustained training dynamics rather than checkpoint-specific artifacts.

B.2 Ablation Training Curves

Figures 9 and 10 show the training dynamics of the ablation variants. Both confirm that PAPO's design choices yield consistent advantages throughout training, not just at specific checkpoints.

B.3 Process Advantage Effect

Figure 11 reveals how PAPO's process advantage operates within the correct response group. In standard ORM-based GRPO, the minimum advantage among correct responses is identically 0.0. Every correct response receives identical positive advantage. In PAPO, correct_min ≈ -1.49, meaning correct responses with poor reasoning receive *negative* total advantage. The process advantage actively discourages solutions that arrive at the right answer through flawed reasoning. Panel (b) confirms that the process advantage creates a meaningful quality ranking among correct responses (non-zero std),

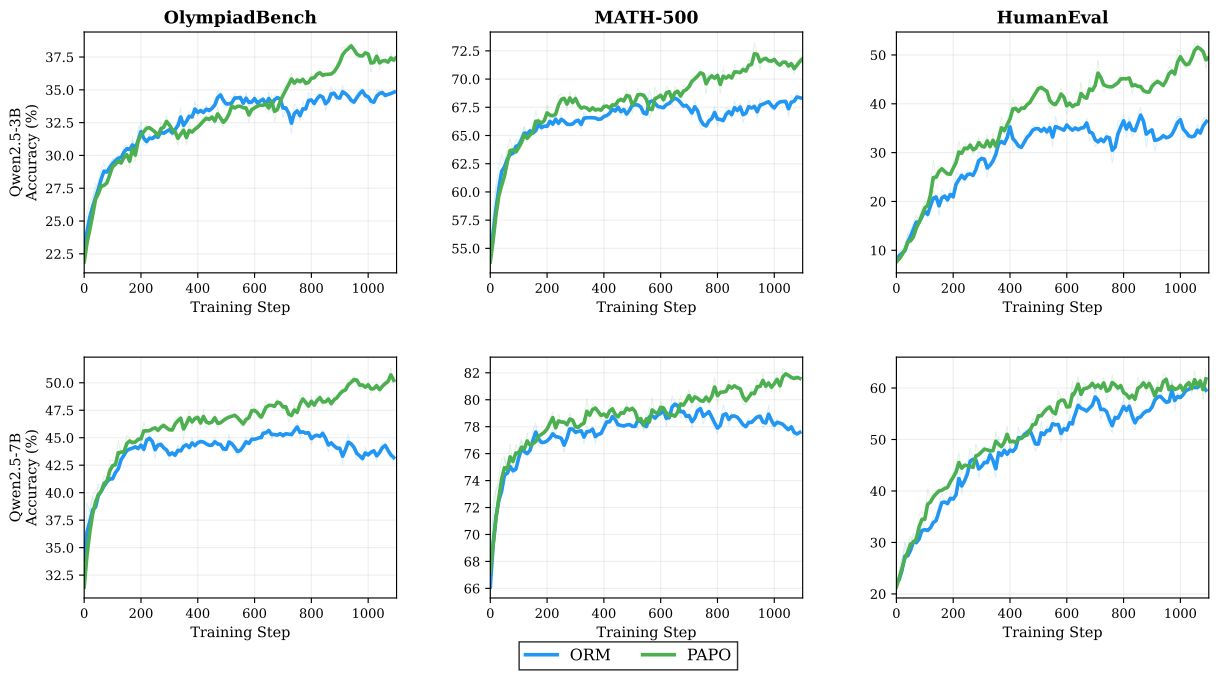


Figure 8: Training curves (avg@4) across two model scales and three benchmarks. Top row: Qwen2.5-3B; bottom row: Qwen2.5-7B. PAPO consistently outperforms ORM throughout training on both scales, with gains widening in later stages as ORM's signal exhaustion worsens. The pattern is consistent across OlympiadBench (competition math), MATH-500 (standard math), and HumanEval (code generation). Qwen2.5-14B results are reported in Table 1.

enabling the model to learn which reasoning strategies produce better solutions.

B.4 Advantage Composition

Figure 12 visualizes the internal structure of PAPO's composed advantage. Panel (a) shows that the outcome advantage provides the dominant gradient signal, while the process advantage contributes a substantial secondary signal. Panel (b) tracks the process signal active ratio (the fraction of groups where A_{proc} activates), which grows from $\sim$30% to 70% as the model improves, indicating that the process signal's contribution increases precisely as ORM's signal exhaustion worsens. Panel (c) confirms that the combined advantage correctly separates correct from incorrect responses: the addition of A_{proc} does not disrupt the fundamental separation provided by A_{out}.

B.5 Training Reward Dynamics

The training reward (average ORM score across the batch) follows similar trajectories for ORM and PAPO, both plateauing around 55% to 62%. This indicates that PAPO does not achieve its gains by inflating outcome reward; the improvement comes from better reasoning quality among correct responses, as captured by the process advantage.

In contrast, PRM-only training shows the training reward climbing to 1.0 by step 1000, confirming complete reward hacking. The model learns to produce responses that maximize the PRM score regardless of mathematical correctness.

B.6 Case Study: PRM Reward Hacking

To empirically illustrate the reward hacking mechanism described in §3.2, we generate responses on five math problems of varying difficulty from three checkpoints, sampling 3 responses per model with temperature 0.7 and a generation cap of 8192 tokens. The three checkpoints are the base model (Qwen2.5-7B, step 0), the PRM-only model at step 1000 (post-collapse, 2.4% accuracy on OlympiadBench), and the ORM baseline at step 1000. We report the three most informative problems in Table 7; on the remaining two easier problems (combinatorics, geometry), PRM performs comparably to baselines.

Results. Table 7 summarizes the findings. On the easier number theory problem, PRM produces correct but inflated responses. On harder problems (competition number theory, proof-based algebra), the model degenerates dramatically: responses are 2–3$\times$ longer than ORM, and none arrive at the correct answer.

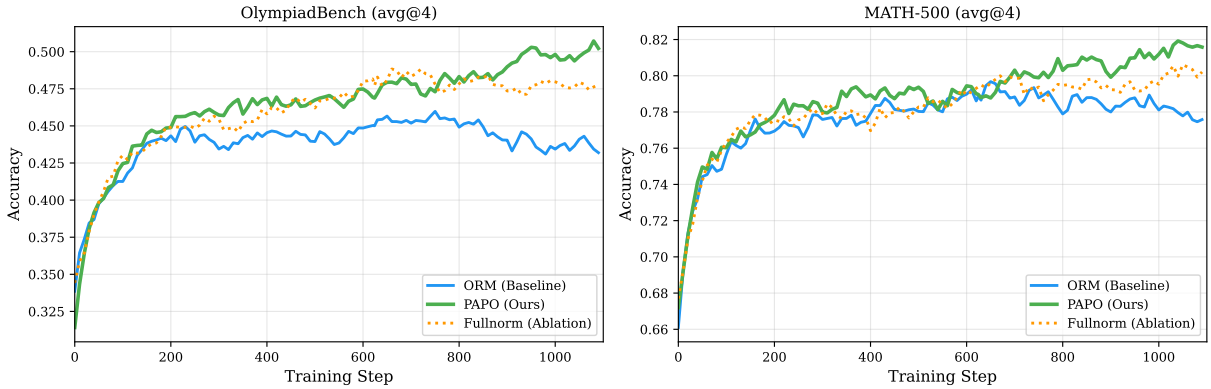


Figure 9: Ablation: correct subset normalization (PAPO) vs. full normalization (Fullnorm) on OlympiadBench and MATH-500. Both methods improve over ORM, but PAPO's correct subset design maintains a consistent advantage over Fullnorm throughout training.

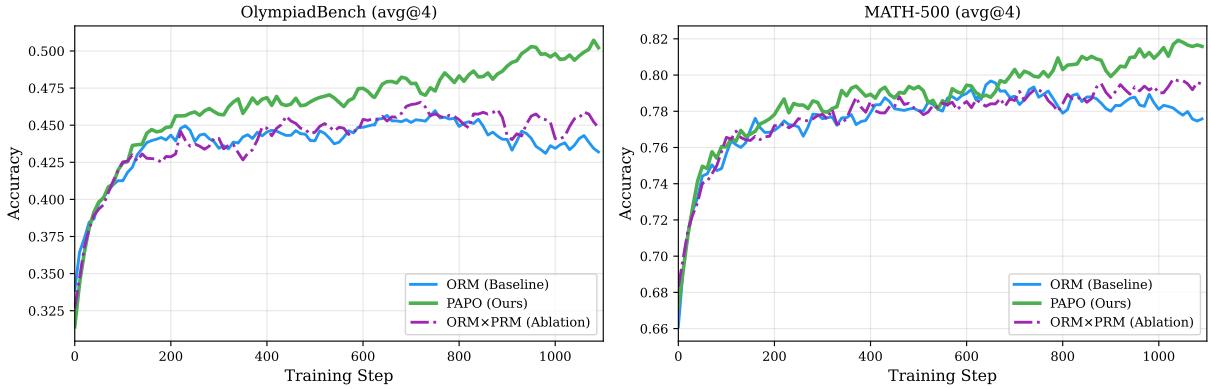


Figure 10: Ablation: decoupled normalization (PAPO) vs. multiplicative reward (Mult) on OlympiadBench and MATH-500. Mult improves over ORM but falls substantially short of PAPO, confirming that decoupled normalization is superior to naive reward combination.

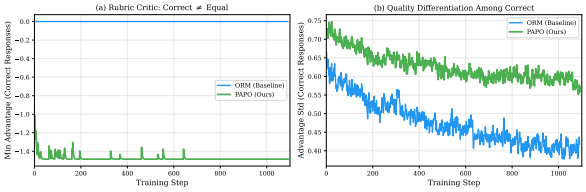


Figure 11: **(a)** Minimum advantage among correct responses. ORM: identically 0.0 (all correct responses treated equally). PAPO: ≈ -1.49 (the process advantage penalizes correct responses with poor reasoning). **(b)** Standard deviation of advantages among correct responses. ORM: zero (no differentiation). PAPO: non-zero (quality-based ranking within correct group).

Qualitative analysis: topic drift to a fixed filler template. The most striking finding is in the competition number theory problem ("Find all positive integers n such that $n^2 + 1 \mid n! + 1$"; answer: $n = 1$). All three PRM step-1000 responses begin correctly (setting up the divisibility condition, checking small cases), but after ∼1500–2000 tokens of genuine work, each response *drifts to the identical unrelated problem*: computing the dot product of vectors $\overrightarrow{m} = (t, 1)$ and $\overrightarrow{n} = (1, -2)$, solving $t - 2 = 0$, and concluding $\boxed{2}$. This vector problem is entirely unrelated to the stated question.

The convergence of all three independent samples to the same filler content reveals a learned exploitation strategy. When the model cannot solve a hard problem, it transitions to a *memorized high-scoring template*, a short, well-structured solution to a simple problem that the LLM judge would rate as "all steps executed properly" (rubric score 1.0). The same pattern appears in the algebra proof and the inflated number theory sample, where responses drift to digit-sum sequences or other unrelated derivations.

In contrast, ORM produces focused solutions (1494–1646 tokens) that arrive at the correct answer $n = 1$ on all three samples. The ORM model may produce longer proofs than the base model (1583 vs. 866 avg tokens), but the additional length

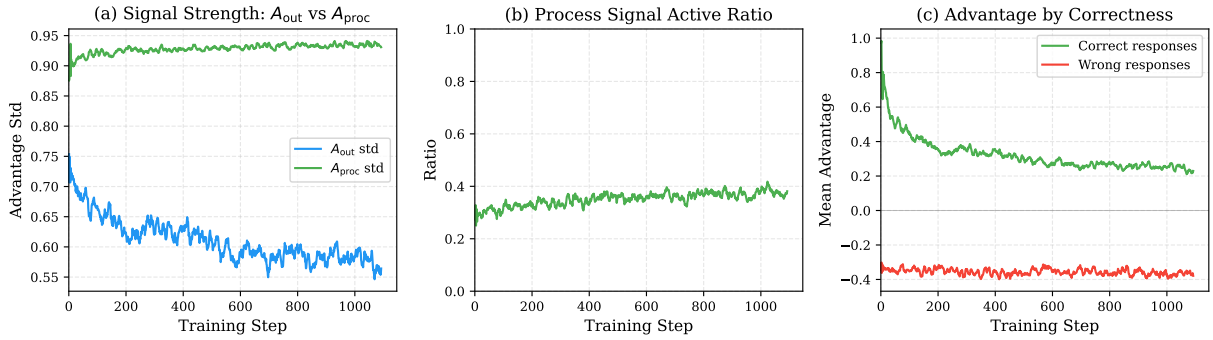


Figure 12: PAPO advantage composition. **(a)** Signal strength (std) of A_{out} and A_{proc}: the outcome advantage provides the dominant gradient signal while the process advantage contributes a complementary quality signal. **(b)** Process signal active ratio: the fraction of groups where A_{proc} is non-zero (≥ 2 correct responses), growing from $\sim$30% to 70% as the model improves. **(c)** Mean advantage by correctness: correct responses receive positive advantage while wrong responses receive negative, confirming the outcome signal functions correctly despite the added process component.

Problem	Model	Tokens	Correct	Observation
Competition number theory	Base	866	3/3	Focused proofs
	ORM	1583	3/3	Longer but correct
	PRM	**3072**	**0/3**	Drift to filler*
Algebra (proof)	Base	1059	0/3	Attempts proof
	ORM	1492	0/3	Attempts proof
	PRM	**3227**	**0/3**	2 hit cap; 1 drifts
Number theory	Base	484	2/3	Concise, correct
	ORM	519	3/3	Concise, correct
	PRM	1522	3/3	1 inflated to 3291

* All 3 PRM samples drift to the *identical* filler problem (vector perpendicularity, answer $t=2$).

Table 7: Case study: response statistics on three math problems (Qwen2.5-7B). PRM reward hacking is severe on harder problems (top two), with 2–3× longer responses and characteristic topic drift. On easier problems, PRM remains comparable to baselines (omitted for brevity).

consists of genuine mathematical reasoning (additional case analysis), not filler content.

Implications. The case study reveals that PRM reward hacking is not random degeneration but a *structured exploitation strategy* comprising three steps. The model first attempts the stated problem, transitions to memorized high-scoring content when stuck, and produces a confident final answer to the wrong problem. The responses remain well-formatted and superficially mathematical throughout, precisely the qualities an LLM-as-Judge PRM rates favorably. This confirms the positive feedback loop of §3.2 and motivates the decoupled design of PAPO, where the binary ORM anchors correctness assessment independently of the PRM signal.

C Case Study: Full Response Comparison

We present the complete model responses for the two qualitative examples discussed in §D.

C.1 OlympiadBench #507 (Reasoning Error)

Problem. Compute the number of ordered pairs (x, y) of positive integers satisfying $x^2 - 8x + y^2 + 4y = 5$. **Ground truth: 4**

ORM Response (incorrect answer: 3)

To find the number of ordered pairs (x, y) of positive integers satisfying the equation $x^2-8x+y^2+4y = 5$, we will complete the square for both x and y.
First, complete the square for x: $x^2 - 8x = (x - 4)^2 - 16$.
Next, complete the square for y: $y^2 + 4y = (y + 2)^2 - 4$.
Substitute these into the original equation: $(x-4)^2 - 16 + (y + 2)^2 - 4 = 5$, $(x - 4)^2 + (y + 2)^2 = 25$.
This is the equation of a circle with center $(4, -2)$ and radius 5. We need to find the number of positive integer solutions (x, y) that lie on this circle.
The possible values for $(x - 4)^2$ and $(y + 2)^2$ must be perfect squares that sum to 25. The pairs (a, b) such that $a^2 + b^2 = 25$ are $(0, 5)$, $(3, 4)$, $(4, 3)$, and $(5, 0)$.

1. $(x - 4)^2 = 0$, $(y + 2)^2 = 25$: $x = 4$, $y = 3$ or -7. Only $(4, 3)$ valid.
2. $(x - 4)^2 = 9$, $(y + 2)^2 = 16$: $x = 7$ or 1, $y = 2$ or -6. Positive integer solutions are $(7, 2)$ and $(1, 2)$, but "$(1, 2)$ is not valid since $x = 1$ is not a positive integer in the context of the circle's radius."
3. $(x-4)^2 = 16$, $(y+2)^2 = 9$: $x = 8$ or 0, $y = 1$ or -5. Only $(8, 1)$ valid.
4. $(x - 4)^2 = 25$, $(y + 2)^2 = 0$: $x = 9$ or -1, $y = -2$. No valid solutions.

Therefore, the positive integer solutions are $(4, 3)$,

$(7, 2)$, and $(8, 1)$. The number of ordered pairs is $\boxed{3}$.

PAPO Response (correct answer: 4)

To solve the problem of finding the number of ordered pairs (x, y) of positive integers satisfying the equation $x^2 - 8x + y^2 + 4y = 5$, we will follow these steps:
1. Complete the square for x and y: $x^2 - 8x = (x-4)^2 - 16$, $\quad y^2 + 4y = (y+2)^2 - 4$.
2. Substitute: $(x-4)^2 + (y+2)^2 - 20 = 5 \Rightarrow (x-4)^2 + (y+2)^2 = 25$.
3. Identify all integer pairs (a, b) with $a^2 + b^2 = 25$: $(\pm 5, 0)$, $(0, \pm 5)$, $(\pm 3, \pm 4)$, $(\pm 4, \pm 3)$ — 12 pairs total.
4. Substitute $a = x-4$, $b = y+2$, check $x, y > 0$:

$(5, 0)$: $x{=}9, y{=}{-2}$	inv.	$(-5, 0)$: $x{=}{-1}, y{=}{-2}$	inv.
$(0, 5)$: $x{=}4, y{=}3$	**valid**	$(0, -5)$: $x{=}4, y{=}{-7}$	inv.
$(3, 4)$: $x{=}7, y{=}2$	**valid**	$(3, -4)$: $x{=}7, y{=}{-6}$	inv.
$(-3, 4)$: $x{=}1, y{=}2$	**valid**	$(-3, -4)$: $x{=}1, y{=}{-6}$	inv.
$(4, 3)$: $x{=}8, y{=}1$	**valid**	$(4, -3)$: $x{=}8, y{=}{-5}$	inv.
$(-4, 3)$: $x{=}0, y{=}1$	inv.	$(-4, -3)$: $x{=}0, y{=}{-5}$	inv.

5. Valid pairs: $(4, 3), (7, 2), (1, 2), (8, 1)$. There are 4 valid pairs. $\boxed{4}$

C.2 OlympiadBench #129 (Self-Verification)

Problem. What is the value of x such that $\log_2(\log_2(2x-2)) = 2$? **Ground truth: 9**

ORM Response (correct but no verification)

To solve the equation $\log_2(\log_2(2x-2)) = 2$, we need to follow a step-by-step approach.
First, let's rewrite the equation in a more manageable form. Since $\log_2(\log_2(2x-2)) = 2$, we can exponentiate both sides with base 2 to eliminate the outer logarithm:

$$\log_2(2x-2) = 2^2$$

This simplifies to:

$$\log_2(2x-2) = 4$$

Next, we exponentiate both sides again with base 2 to eliminate the logarithm:

$$2x - 2 = 2^4$$

This simplifies to:

$$2x - 2 = 16$$

Now, we solve for x by isolating it on one side of the equation. First, add 2 to both sides:

$$2x = 18$$

Then, divide both sides by 2:

$$x = 9$$

Thus, the value of x is $\boxed{9}$.

PAPO Response (correct with verification)

To solve the equation $\log_2(\log_2(2x-2)) = 2$, we will follow these steps:
1. Solve the outer logarithm equation: $\log_2(\log_2(2x-2)) = 2$ implies that $\log_2(2x-2) = 2^2 = 4$.
2. Solve the inner logarithm equation: $\log_2(2x-2) = 4$ implies that $2x - 2 = 2^4 = 16$.
3. Solve for x: $2x - 2 = 16$, so $2x = 18$, giving $x = 9$.
4. Verify the solution: Substitute $x = 9$ back into the original equation to ensure it satisfies the equation:

$$\log_2(\log_2(2 \cdot 9 - 2)) = \log_2(\log_2(16))$$

Since $\log_2(16) = 4$ and $\log_2(4) = 2$, the solution is verified.
Therefore, the value of x is $\boxed{9}$.

D Qualitative Analysis

We perform qualitative analysis on reasoning traces from the ORM baseline and PAPO to understand how process reward shapes model behavior.

D.1 Process Supervision Corrects Reasoning Errors

Without supervision on intermediate reasoning steps, ORM-trained models can develop sloppy reasoning habits that lead to surprisingly elementary errors. We illustrate this with OlympiadBench #507, which asks to compute the number of ordered pairs (x, y) of positive integers satisfying $x^2 - 8x + y^2 + 4y = 5$.

Both models correctly complete the square to obtain $(x-4)^2 + (y+2)^2 = 25$. The ORM model then *incorrectly excludes* the valid solution $(x, y) = (1, 2)$ with the fabricated justification that "$x = 1$ is not a positive integer in the context of the circle's radius." This is a nonsensical claim, since $x = 1$ is plainly a positive integer, and this elementary error directly causes a wrong final answer.

The PAPO model, by contrast, carefully verifies each candidate against the positivity constraint and correctly identifies all four solutions. Process supervision penalizes flawed intermediate steps regardless of the final answer, discouraging the kind of careless reasoning that produces such errors in the first place.

D.2 Process Reward Cultivates Self-Verification

Beyond correcting errors, we find that process reward training cultivates a *self-verification habit*,

where the model spontaneously substitutes the derived answer back into the original problem as a final check. This behavior emerges naturally because responses that include verification receive higher process scores, reinforcing the habit over time.

For instance, on OlympiadBench #129, which asks to find x such that $\log_2(\log_2(2x-2)) = 2$, both models correctly derive $x = 9$ through successive exponentiation. The ORM model stops here. The PAPO model adds an explicit check, confirming that $\log_2(\log_2(16)) = \log_2(4) = 2$.

This is not an isolated case. Across OlympiadBench problems where both models answer correctly, PAPO responses contain explicit verification steps in 47.7% of cases, compared to 22.7% for ORM, a $2.1\times$ higher rate. Process reward incentivizes the model to develop a disciplined reasoning style that makes its outputs more robust, even on problems it can already solve.

E Ethics Statement

This work focuses on improving reinforcement learning algorithms for mathematical reasoning. Our research exclusively utilizes publicly available resources, including open-source models (Qwen2.5, Qwen3, SmolLM3, GPT-OSS) and established datasets (NuminaMath-1.5), thereby mitigating concerns related to data privacy or human subjects. The application domain of mathematical problem-solving does not inherently present risks of direct societal harm. The primary ethical consideration is the environmental impact of the computational resources required for large-scale model training, a challenge common to the field.

F The Use of Artifacts.

Our experiments use publicly available scientific artifacts, including open-weight language models, datasets, benchmarks, and training/evaluation frameworks. Specifically, we use Qwen2.5-7B/14B and Qwen3-4B under the Apache 2.0 license, Qwen2.5-3B under the Qwen Research License for non-commercial research use, SmolLM3-3B under the Apache 2.0 license, and GPT-OSS-20B/120B under the Apache 2.0 license and the accompanying GPT-OSS usage policy. For training and evaluation, we use NuminaMath-1.5-RL-Verifiable, OlympiadBench, MATH-500, AIME 2024/2025, GPQA-Diamond, and HumanEval, following their respective licenses or terms, including Apache 2.0, MIT, and CC BY-NC-SA 4.0 where applicable. We use these artifacts only for academic research on mathematical reasoning, code generation, and reinforcement learning evaluation, which is consistent with their intended research and benchmarking purposes.

G The Use of Large Language Models

We used Large Language Model (LLM) to refine our initial draft. This process included checking for obvious grammatical and syntactical errors, as well as making the language more formal and academic. We reviewed the content generated by the LLM to ensure that no prohibited generated content appeared in the article.