

# 000 001 002 003 004 005 FAPO: FLAWED-AWARE POLICY OPTIMIZATION FOR 006 EFFICIENT AND RELIABLE REASONING 007 008 009

010 **Anonymous authors**  
011 Paper under double-blind review  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029  
030  
031  
032  
033  
034  
035  
036  
037  
038  
039  
040  
041  
042  
043  
044  
045  
046  
047  
048  
049  
050  
051  
052  
053

## ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has emerged as a promising paradigm for enhancing the reasoning capabilities of large language models (LLMs). In this context, models explore reasoning trajectories and exploit rollouts with correct answers as positive signals for policy optimization. However, these rollouts might involve flawed patterns such as answer-guessing and jump-in-reasoning. Such flawed-positive rollouts are rewarded identically to fully correct ones, causing policy models to internalize these unreliable reasoning patterns. In this work, we first conduct a systematic study of flawed-positive rollouts in RL and find that they enable rapid capability gains during the early optimization stage, while constraining reasoning capability later by reinforcing unreliable patterns. Building on these insights, we propose **Flawed-Aware Policy Optimization (FAPO)**, which presents a parameter-free reward penalty for flawed-positive rollouts, enabling the policy to leverage them as useful shortcuts in the warm-up stage, securing stable early gains, while gradually shifting optimization toward reliable reasoning in the later refinement stage. To accurately and comprehensively detect flawed-positive rollouts, we introduce a generative reward model (GenRM) with a process-level reward that precisely localizes reasoning errors. Experiments show that FAPO is effective in broad domains, improving outcome correctness, process reliability, and training stability without increasing the token budget.<sup>1</sup>

## 1 INTRODUCTION

Large language models (LLMs) with strong reasoning capabilities, such as OpenAI o-series (OpenAI, 2024; 2025), Deepseek R1 (Guo et al., 2025), have sparked significant attention in reinforcement learning with verifiable rewards (RLVR) (Shao et al., 2024). In this paradigm, models are optimized through rule-based outcome rewards, typically a binary signal indicating whether the final answer is correct, in verifiable tasks like mathematical reasoning (Yu et al., 2025; Team et al., 2025) and code generation (Xiaomi et al., 2025). During RL training, the model explores diverse reasoning trajectories and exploits those with correct final answers as positive signals for policy optimization. This exploration-exploitation paradigm enables LLMs to evolve strong reasoning behaviors, such as planning, which in turn facilitate generalization across a wide range of domains (Huan et al., 2025).

However, certain flawed reasoning patterns could also be reinforced during policy optimization. Recent studies (Zheng et al., 2024; Kalai et al., 2025) have revealed notable flawed reasoning patterns in current LLMs, such as answer-guessing and jump-in-reasoning (Wang et al., 2025), where models reach correct answers through shortcuts. This presents a fundamental challenge for RLVR, i.e., rule-based outcome rewards assign identical positive signals to both flawed-positive and fully correct rollouts, thereby potentially reinforcing unreliable reasoning. This raises an urgent need for (1) *analyzing the distribution and impact of flawed positives throughout the RL process*, and (2) *developing effective mitigation strategies to ensure efficient and reliable reasoning*.

To this end, we first conduct a preliminary study to investigate the prevalence and impact of flawed positives in the RL process. Our findings indicate that flawed positives persist steadily throughout training: (1) in the early stages, when models are not yet capable of producing fully correct rollouts, flawed positives serve as shortcuts to correct answers, accelerating capability gains; and (2)

<sup>1</sup>Code is available at <https://anonymous.4open.science/r/FAPO-RL>.

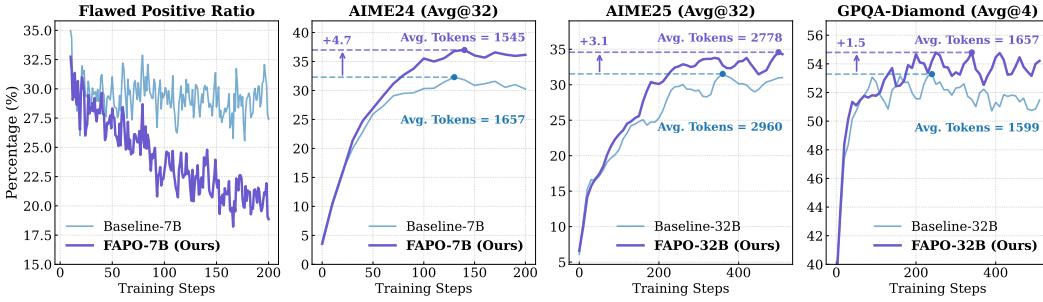


Figure 1: Flawed-positive ratio and performance comparison between FAPO models and their baselines, showing steadily decreasing flawed-positive ratios and consistent performance gains during RL training. Experiments cover both 7B and 32B models on AIME24, AIME25, and GPQA-Diamond. For evaluation, we repeat the evaluation set  $k$  times and report Avg@ $k$  for results stability.

once the model can generate fully correct rollouts, these flawed positives may hinder learning by reinforcing unreliable reasoning patterns. Thus, the optimal role of flawed positives is to act as stepping stones toward reliable reasoning. Building on these insights, we propose Flawed-Aware Policy Optimization (FAPO), which presents a parameter-free reward penalty to flawed-positive rollouts. FAPO establishes a natural self-exploration learning trajectory: the model initially exploits flawed positives for knowledge, but as its capabilities advance, the training objective gradually shifts toward genuine problem-solving, improving both the training efficiency and reasoning reliability. Furthermore, to accurately and comprehensively detect these flawed positives, we introduce a generative reward model with a process reward, which enables the model to locate intermediate process errors.

Experimental results highlight the strong potential of FAPO. For flawed positive detection, our trained model, FAPO-GenRM-4B, achieves substantial gains on both our newly constructed benchmark, FlawedPositiveBench, and the public ProcessBench (Zheng et al., 2024). When integrated into the final RL process (results in Figure 1), FAPO effectively penalizes flawed-positive rollouts, reducing unreliable reasoning patterns (left subfigure, the decreasing flawed positive ratio), while delivering remarkable improvements across AIME24, AIME25, and GPQA-Diamond (Rein et al., 2024) (other three subfigures), **with clear advantages at nearly all intermediate evaluation checkpoints**. Overall, FAPO offers clear advantages: it enhances outcome correctness, improves process reliability, and training efficiency and stability, all without increasing the token budget.

## 2 PRELIMINARY: UNVEILING FLAWED POSITIVES IN RL TRAINING

### 2.1 PROBLEM DEFINITION AND MOTIVATION

**Group Relative Policy Optimization (GRPO)** GRPO (Shao et al., 2024) is an efficient policy gradient method for LLM reinforcement learning that estimates advantages in a group-relative manner without relying on a learned value model. For a given question  $q$ , the behavior policy generates a group of  $G$  rollouts  $\{o_i\}_{i=1}^G$ , evaluates their rewards  $\{R_i\}_{i=1}^G$ , and normalizes them to obtain per-token advantage estimates  $\hat{A}_{i,t}$ :

$$\hat{A}_{i,t} = \frac{r_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}. \quad (1)$$

The policy model is then updated by maximizing the following clipped surrogate objective:

$$\begin{aligned} \mathcal{J}_{\text{GRPO}}(\theta) &= \mathbb{E}_{(q,a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot|q)} \\ &\quad \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[ \frac{\pi_{\theta}(o_t|q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q, o_{<t})} \hat{A}_{i,t}, \text{clip} \left( \frac{\pi_{\theta}(o_t|q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t|q, o_{<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] \right\}, \end{aligned} \quad (2)$$

where  $(q, a)$  denotes a question-answer pair sampled from the data distribution  $\mathcal{D}$ ,  $\pi_{\theta_{\text{old}}}$  is the old policy, and  $\epsilon$  controls the clipping range in importance sampling (Schulman et al., 2017) for stability.

108 In this work, we adopt several effective strategies such as *clip-higher*, *token-level loss*, and *overlong*  
 109 *reward shaping* (Yu et al., 2025), to ensure stable and efficient policy optimization.  
 110

$$\begin{aligned} 111 \quad \mathcal{J}(\theta) &= \mathbb{E}_{(q, a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | q)} \\ 112 \quad &\frac{1}{\sum_{i=1}^G |o_i|} \sum_{i=1}^G \sum_{t=1}^{|o_i|} \left\{ \min \left[ \frac{\pi_{\theta}(o_t | q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t | q, o_{<t})} \hat{A}_{i,t}, \text{clip} \left( \frac{\pi_{\theta}(o_t | q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t | q, o_{<t})}, 1 - \epsilon_l, 1 + \epsilon_h \right) \hat{A}_{i,t} \right] \right\}. \quad (3) \\ 113 \end{aligned}$$

114 In these algorithms, the reward  $R$  is the primary supervision signal that guides the policy optimization,  
 115 and existing RLVR approaches (Yang et al., 2025; Liu et al., 2025b) commonly employ a  
 116 rule-based outcome reward to mitigate reward hacking (Gao et al., 2023; Weng, 2024), i.e.,  
 117

$$118 \quad R_{\text{RLVR}} = R_{\text{rule}}(o, a^*) = \begin{cases} 1, & \text{If } \mathcal{I}(o, a^*) \\ -1, & \text{Otherwise} \end{cases}, \quad (4)$$

119 where  $\mathcal{I}(o, a^*)$  is an indicator function that returns `True` if the predicted answer extracted from  
 120 rollout  $o$  matches the ground-truth answer  $a^*$ , and `False` otherwise.  
 121

122 **Flawed Positive Issues** Recent studies (Zheng et al., 2024; Zhang et al., 2025b) have identified  
 123 notable flawed-positive issues in current LLMs, in some cases even accounting for a ratio of up to  
 124 50%, where models may reach correct final answers through unreliable reasoning patterns such as  
 125 answer-guessing and jump-in-reasoning (Wang et al., 2025). This poses a fundamental challenge for  
 126 reinforcement learning: rule-based reward functions assign positive signals to flawed-positive roll-  
 127 outs, thereby reinforcing unreliable reasoning patterns and ultimately limiting the model’s perfor-  
 128 mance ceiling. Formally, given a question  $q$  and reasoning trajectory  $\mathbf{x} = [x_1, x_2, \dots, x_n]$  generated  
 129 by policy  $\pi$ , with predicted answer  $\hat{a}_\pi$ , the rollout is *flawed positive* if  
 130

$$131 \quad \hat{a}_\pi = a^* \text{ and } \exists t \in \{1, 2, \dots, n\} \text{ s.t. step } x_t \text{ is logically invalid.} \quad (5)$$

132 While prior works have primarily revealed the prevalence of these issues in benchmark evaluations,  
 133 their underlying mechanisms and impact on the RL process remain largely underexplored.  
 134

## 135 2.2 FLAWED POSITIVE ANALYSIS IN REINFORCEMENT LEARNING

136 **Flawed Positives are Prevalent in Initial Checkpoints** We first examine flawed positive issues  
 137 in current LLMs, as they establish the starting conditions for subsequent RL optimization. We  
 138 evaluate three representative models: Qwen2.5-Math-7B-Base (Yang et al., 2024b), Llama3.3-70B-  
 139 Instruct (Dubey et al., 2024), and Qwen3-1.7B (Yang et al., 2025), on the DAPO-Math dataset.  
 140 Following Zhang et al. (2025b), we employ Qwen3-32B to determine whether the reasoning trajec-  
 141 tory contains unreliable reasoning patterns. As shown in Figure 2 (a), flawed positives are prevalent  
 142 across various LLMs, accounting for 20%–40% of correct rollouts, highlighting the severity of this  
 143 issue. Beyond the automatic LLM-as-a-judge evaluation, we also conduct a manual case study of  
 144 flawed-positive samples and analyze their underlying causes, which are provided in Appendix G.  
 145

146 **Flawed Positives are Stepping Stones in Learning** Reinforcement learning is often formulated as  
 147 an end-to-end optimization process driven by self-exploration towards self-improvement. To better  
 148 understand this process, and in particular the role of flawed-positive rollouts, we design a simulated  
 149 experiment in which the learning stage of each sample is approximated by its rollout accuracy.  
 150 Specifically, we use a model to generate multiple rollouts per sample, compute the corresponding  
 151 rollout accuracy, and then group all samples into different learning stages, as illustrated in Figure 2  
 152 (b). The results reveal a clear trend: flawed positives are most prevalent during the early learning  
 153 stages but diminish significantly as training progresses. This highlights their expected role as natural  
 154 stepping stones in the learning trajectory, allowing the model to initially reach correct answers before  
 155 gradually evolving the capability to produce fully correct solutions.  
 156

157 **Flawed Positives Persist and Exert Twofold Effects** We further train a pre-trained model,  
 158 Qwen2.5-Math-7B, with RL on DAPO-Math, and track its learning trajectory, as shown in Figure 2  
 159 (c). While the model’s rollout accuracy steadily improves, the flawed-positive ratio remains almost  
 160 constant at around 30%. This indicates that the optimization process struggles to shift from unre-  
 161 liable reasoning to genuine problem-solving. A major concern is that flawed positives receive the

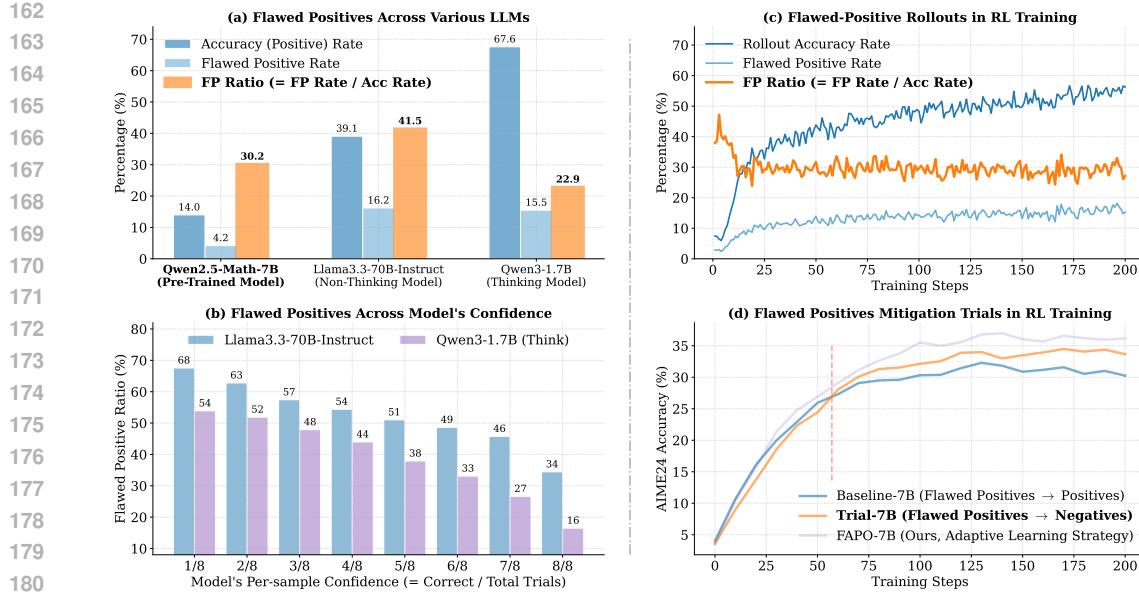


Figure 2: Preliminary experiment results of flawed positives.

same rewards as correct solutions, thereby reinforcing unreliable reasoning patterns and hindering progress. To further explore this, we conduct preliminary trials using Qwen3-32B to detect flawed positives and assign them negative signals (same as negative rollouts) during training. Figure 2 (d) reports performance on AIME24 throughout the training process. Compared with the baseline RLVR setting (blue line), penalizing flawed positives (orange line) yields significant performance gains, though improvements emerge more gradually in the early stages. From the above findings, we find flawed positives persist throughout training, and exert a twofold effect: **(1) flawed positives act as stepping stones, enabling the model to achieve rapid capability gains in the early stages, and (2) their improper reward assignment can trap optimization in unreliable reasoning.**

### 3 FAPO: FLAWED-AWARE POLICY OPTIMIZATION

Building on these insights, we propose **Flawed-Aware Policy Optimization** (FAPO) algorithm. For flawed-positive detection, directly employing a strong LLM like Qwen3-32B is impractical and computationally inefficient. Instead, we propose an RL algorithm to train a compact yet effective generative reward model (GenRM). We then present an adaptive learning algorithm that dynamically adjusts advantage assignment towards the current suitable optimization direction.

#### 3.1 FLAWED POSITIVE DETECTION

**Evaluating Flawed Positive Detection Capabilities of LLMs** To identify a suitable LLM that can detect flawed positives both effectively and efficiently, we construct an evaluation set, *Flawed-PositiveBench*, by collecting positive samples (including flawed ones) from ProcessBench (Zheng et al., 2024). We then quantify the detection capability with the following three metrics:

$$precision = \frac{\#\{\hat{y}_\theta = y^* = \text{FP}\}}{\#\{\hat{y}_\theta = \text{FP}\}}, \quad recall = \frac{\#\{\hat{y}_\theta = y^* = \text{FP}\}}{\#\{y^* = \text{FP}\}}, \quad F_1 = \frac{2}{1/precision + 1/recall}, \quad (6)$$

where  $\hat{y}_\theta$  indicates whether the judge model  $\theta$  predicts a response as a flawed positive (FP), and  $y^*$  is the ground-truth label. Precision reflects the correctness of FP predictions, recall measures the coverage of true FPs, and the  $F_1$  score provides a balanced summary of both. As shown in Figure 3, we observe that many models, such as Qwen3-4B-Instruct (Yang et al., 2025) and Qwen2.5-Math-PRM-72B (Zhang et al., 2025b), exhibit an over-critic phenomenon: they achieve high recall but suffer from low precision. Closer inspection reveals that these models often overemphasize minor or unnecessary errors like unsimplified fractions. Overall, lightweight models struggle to provide

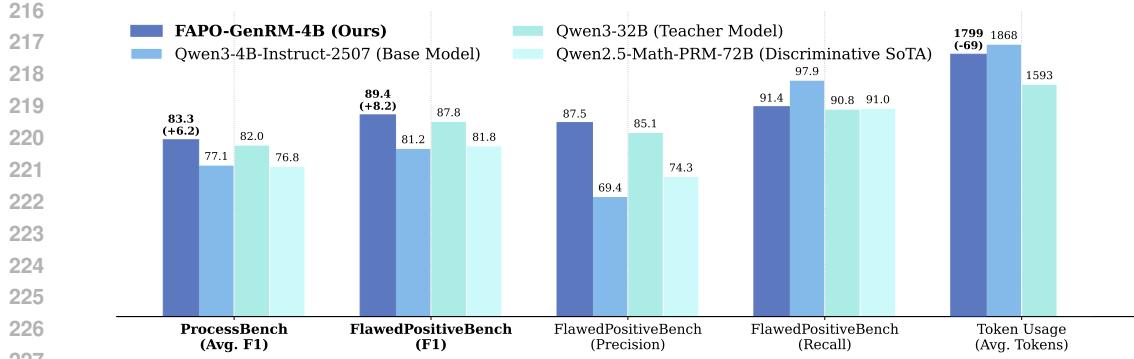


Figure 3: Performance of current state-of-the-art (SoTA) generative models and discriminative PRMs. Detailed subset-level results and additional models are reported in Table 2.

appropriate criticisms to detect flawed positives, while stronger models achieve better accuracy but remain impractical for online RL use due to slow inference. **These findings suggest that existing models are not well-aligned in both detecting capabilities and inference efficiency.**

**Enhancing Detection Capabilities via Step-wise RL Optimization** To enhance the detection capability, we introduce a step-wise RL reward formulation. Starting from a generative model, we develop the following RL strategies for training a generative reward model (GenRM):

$$R_{\text{FAPO-GenRM}} = R_{\text{Outcome}} + R_{\text{Process}}$$

$$\text{where } R_{\text{Outcome}} = \begin{cases} 1, & \text{If } \hat{y}_\theta = y^* \\ -1, & \text{Otherwise} \end{cases}, \quad R_{\text{Process}} = \begin{cases} -\frac{|\hat{t}_\theta - t^*|}{n}, & \text{If } \hat{y}_\theta = y^* = \text{FP} \\ 0, & \text{Otherwise} \end{cases}. \quad (7)$$

Building upon the outcome reward, we introduce a step-wise penalty,  $R_{\text{Process}}$ , for fine-grained and step-wise optimization. Here,  $\hat{t}_\theta$  and  $t^*$  denote the predicted and ground-truth error indices, and  $n$  is the total number of steps, ensuring  $R_{\text{Process}} \in [-1, 0]$ . In flawed-positive cases, the penalty is distance-sensitive: predictions closer to the true error receive higher rewards, while those farther away incur stronger penalties. This design guides the model toward precise error localization and fosters genuine error-detection ability, rather than mere guessing, based on the two key points:

- **Learning beyond guessing:** Flawed-positive rollouts also exist in the error detection task, particularly in the yes/no setting, where the model can often guess the label without truly identifying errors. Such guessing rollouts offer little optimization benefit. To mitigate this, we introduce the penalized step reward that guides the model toward genuine critic capabilities rather than guessing.
- **Natural reward shift:** In early training, the reward design naturally emphasizes prediction correctness, as improving  $R_{\text{Baseline}}$  yields substantial gains ( $-1 \rightarrow 1$ ), whereas  $R_{\text{Process}}$  provides only limited gains ( $R_{\text{Process}} \in [-1, 0]$ ). As correctness saturates, process optimization becomes increasingly prioritized. This enables a smooth transition without additional controlling hyperparameters.

### 3.2 FLAWED POSITIVE PENALIZATION

With the GenRM detecting flawed positives, we then regulate their roles in the final RL optimization. As discussed in Section 2.2, flawed positives should ideally facilitate rapid warm-up and then be penalized to enable genuine problem-solving. However, the key challenge lies in *how to balance their encouragement and suppression and when to shift between these roles*. To address this, we introduce a reward-penalization mechanism with a group-relative advantage estimation:

$$R_{\text{FAPO}}(o, a^* | \theta) = R_{\text{RLVR}}(o, a^*) + R_{\Delta}(\mathbf{o}, \mathbf{a}^* | \theta),$$

$$\text{where } R_{\Delta}(o, a^* | \theta) = \begin{cases} -\lambda, & \text{If } \mathcal{I}(o, a^*) \text{ and } \hat{y}_\theta(o, a^*) = \text{FP} \\ 0, & \text{Otherwise} \end{cases}, \quad (8)$$

$$\hat{A}_{i,t} = [r_i - \text{mean}(\{R_i\}_{i=1}^G)] / \text{std}(\{R_i\}_{i=1}^G).$$

where  $R_{\text{RLVR}}$  denotes the standard baseline (defined in Equation 4), and  $\lambda$  controls the penalization strength. To better characterize the entire learning dynamics of FAPO, we provide a theoretical

analysis in Appendix A, which demonstrates how FAPO enables the natural optimization shift while further stabilizing the RL training process. Concretely, when the current rollout stage contains  $\alpha$  proportion of positive samples and  $\beta$  proportion of negative samples, the optimization shifts from the warm-up stage to the refinement stage once the learning progress  $\rho = \frac{\alpha}{\beta}$  reaches  $\frac{2}{\lambda} - 1$ . Moreover, as optimization continues, when  $\rho > \frac{4}{\lambda} - 1$ , the estimated advantage for positive samples becomes downscaled, making the optimization more stable. In the process, the value of  $\lambda$  determines the timing of this optimization shift. We adopt a majority-guided strategy, where the optimization direction is determined by whether positive or negative samples dominate. This majority-guided strategy yields  $\rho_{\text{shift}} = 1$ , further determining  $\lambda = 1$ . We set  $\lambda = 1$  as the default setting. Overall, FAPO provides a principled mechanism for guiding the optimization process, aligning with the ideal learning trajectory where the focus initially lies in producing correct solutions when model capability is limited, and naturally shifts toward refining reliability once correct rollouts surpass incorrect ones.

## 4 EXPERIMENTS

### 4.1 TRAINING DETAILS

In this work, we validate the effectiveness of FAPO on Qwen2.5-Math-7B (Yang et al., 2024b) and Qwen2.5-32B (Yang et al., 2024a). We adopt GRPO (Shao et al., 2024) with several commonly used strategies, including *clip-higher*, *token-level loss*, and *overlong reward shaping* (Yu et al., 2025), as our baseline algorithm. Notably, FAPO can be easily transferred to any other RLVR method as a drop-in replacement for rule-based outcome rewards. We conduct RL training using verl framework (Sheng et al., 2025), and develop an asynchronous architecture that decouples rollout inference and generative reward modeling, which substantially improves training efficiency.

**FAPO-GenRM** To train the GenRM model via reinforcement learning, we construct a flawed-positive dataset, FAPO-Critic-85K. To ensure broad coverage, we employ a series of models from the LLaMA and Qwen families, ranging from 7B to 70B, to generate multiple responses to questions drawn from DAPO-Math-17K (Yu et al., 2025). Based on these responses, we select the samples with correct final answers and then employ Qwen3-32B to identify the inherent step-level error location. This yields the final process-error dataset:  $\mathcal{D}_{\text{FAPO-Critic}} = \{(q_i, r_i, t_i)\}_{i=1}^N$ , where  $t_i$  denotes the first error index of response  $r_i$ , and fully correct responses are included with  $t_i = +\infty$  for convenience. This dataset is then used to train Qwen3-4B-Instruct (Yang et al., 2025) with the reward defined in Equation 7, and additional hyperparameter settings are provided in Appendix B.

**FAPO-Reasoning** The trained critic model, FAPO-GenRM-4B, is then used to detect and penalize flawed positives in reinforcement learning for reasoning tasks, optimized with the reward defined in Equation 8. In practice, we deploy the GenRM as an external LLM service on a computing cluster, where process rewards are obtained via remote API requests during RL training. To ensure efficiency, we launch multiple server workers and employ a router to distribute requests with balanced load across workers. This decoupled design enables asynchronous interaction between GenRM and other RL components, substantially improving training efficiency and making the integration of GenRM into large-scale RL training practically feasible. Further details on infrastructure design and hyperparameter configurations are provided in Section 4.5 and Appendix B, respectively.

### 4.2 EVALUATION SETUP

**Flawed-Positive Detection** We primarily evaluate GenRM on FlawedPositiveBench, whose construction procedure and evaluation metrics are detailed in Section 3.1, as this benchmark directly aligns with our research purpose. In addition, we also include ProcessBench (Zheng et al., 2024), which focuses on recognizing fully correct samples and precisely locating errors in incorrect responses. We report the harmonic mean of the accuracies on correct and erroneous samples. Furthermore, we incorporate several state-of-the-art (SoTA) discriminative and generative models as strong baselines for comprehensive comparison.

**Reasoning Evaluation** We conduct a comprehensive evaluation covering AIME24 (Math), AIME25 (Math), and GPQA-Diamond (General Domain) (Rein et al., 2024). Rather than limiting the analysis to a single selected checkpoint, we present all intermediate evaluation outcomes

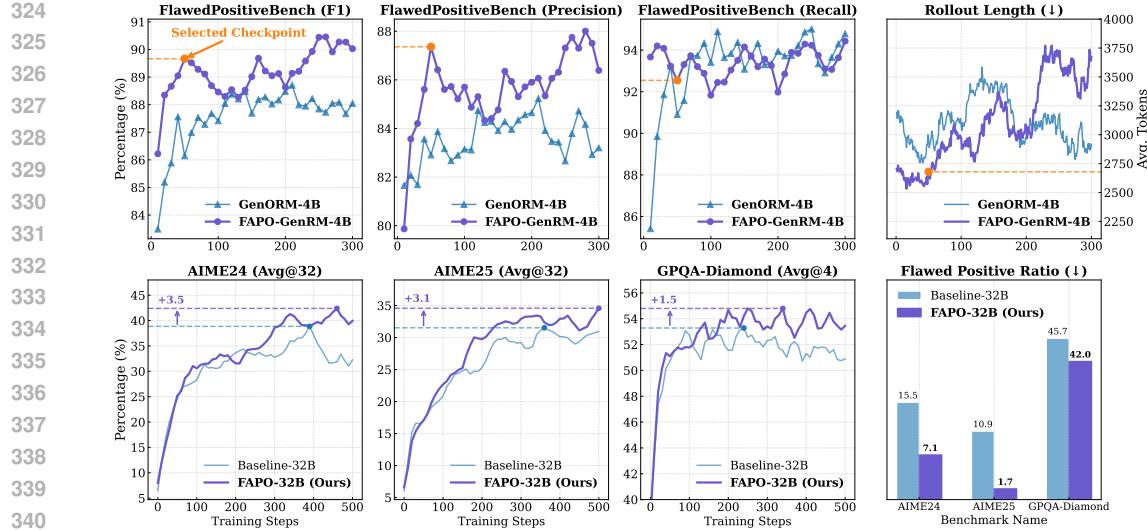


Figure 4: Performance of FAPO-GenRM and FAPO-Reasoning during training. **Top row:** comparison between FAPO-GenRM and the baseline outcome reward models (setup in Equation 7). **Bottom row:** comparison between FAPO-Reasoning and the baseline setting (setup in Equation 8). **Detailed results in a broader domain can be seen in Table 2 and Table 3.**

throughout the RL process. This not only illustrates the performance gains achieved during the training process but also highlights the stability and scalability of the optimization procedure, thereby providing stronger evidence of the effectiveness and robustness of our approach.

### 4.3 MAIN RESULTS

**FAPO-GenRM Performance** The top row of Figure 4 illustrates the training dynamics of FAPO-GenRM. The model exhibits significant performance gains in the early stages and continues to improve as training progresses. For subsequent use, however, we select an early checkpoint, as it delivers strong results with shorter responses, which is crucial for maintaining efficiency when integrating GenRM into RL training. Figure 3 compares our trained model against state-of-the-art discriminative and generative baselines. Built upon Qwen3-4B-Instruct, our approach achieves substantial improvements on both FlawedPositiveBench and ProcessBench, even outperforming the teacher model Qwen3-32B, further demonstrating the effectiveness of our approach. Additional results of our model and other strong baselines can be checked in Table 2.

**FAPO-Reasoning Performance** Figure 1 and the bottom row of Figure 4 summarize the overall performance of the FAPO reasoning models, which can be highlighted in the following aspects:

- **Outcome Correctness:** Across benchmarks, FAPO consistently maintains a clear advantage over the baselines in both mathematical and general-domain tasks, demonstrating that detecting and penalizing flawed positives leads to broad improvements in problem-solving ability.
- **Process Reliability:** We also measure the proportion of flawed positives. The results show that FAPO responses exhibit a substantially lower flawed-positive ratio. Beyond the LLM-as-a-judge approach using Qwen3-32B, we also launch a manual verification of unreliable reasoning patterns, with details and results in Table 6, demonstrating the effectiveness of FAPO.
- **Training Stability:** By mitigating the impact of flawed positives, training stability is significantly enhanced. The overall learning curves are smoother, and unlike the baselines, FAPO does not exhibit a notable performance drop in the later stages of training.
- **Token Budget:** The improvements from FAPO do not require longer responses. While prior work (Luo et al., 2025; An et al., 2025) has shown that scaling up response length can yield substantial gains, FAPO achieves improvements without relying on this factor.

### 4.4 ABLATION STUDY

378  
 379  
 380  
 381  
 382  
 383  
 384  
 385  
 386  
 387  
 388  
 389  
 390  
 391  
 392  
 393  
 394  
 395  
 396  
 397  
 398  
 399  
 400  
 401  
 402  
 403  
 404  
 405  
 406  
 407  
 408  
 409  
 410  
 411  
 412  
 413  
 414  
 415  
 416  
 417  
 418  
 419  
 420  
 421  
 422  
 423  
 424  
 425  
 426  
 427  
 428  
 429  
 430  
 431  
**Effectiveness of FAPO-GenRM method** Previous results have already demonstrated that our GenRM design achieves state-of-the-art (SoTA) performance on the error detection task (see Figure 3). Here, we further evaluate its impact on the final RL process. Figure 5 compares FAPO-GenRM with the base model (Qwen3-4B-Instruct) during RL training. The results indicate that stronger detection capability ultimately translates into improved performance. This highlights two key points: (1) the proposed FlawedPositiveBench provides a reliable measure of detection ability that is well aligned with final performance, and (2) flawed positive detection plays a crucial role, where even small improvements can yield substantial performance gains.

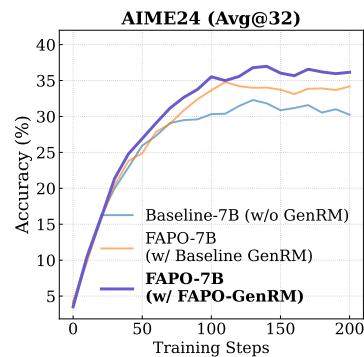


Figure 5: GenRM effectiveness.

392  
 393  
 394  
 395  
 396  
 397  
 398  
 399  
 400  
 401  
 402  
 403  
 404  
 405  
 406  
 407  
 408  
 409  
 410  
 411  
 412  
 413  
 414  
**Impact of Self-Correction Capability** Self-correction is an important evolved mechanism in RL, allowing reasoning models to recover from initial mistakes and eventually reach correct answers. Nevertheless, when correctness depends heavily on long rollouts, it could also be regarded as a form of flawed positives. While self-correction facilitates progress in the early stages of learning, it becomes less desirable once the model can solve problems directly, where fully correct rollouts not only ensure reliability but also promote more efficient reasoning. To illustrate this phenomenon, Figure 6 visualizes rollout length during training: both FAPO and the baseline initially depend on self-correction, but over time, FAPO shifts toward fully correct rollouts, resulting in shorter rollouts, more efficient reasoning, and consistent performance gains. These results indicate that FAPO preserves the benefit of learning from self-corrected rollouts at the beginning, but gradually shifts toward prioritizing fully correct rollouts in later stages.

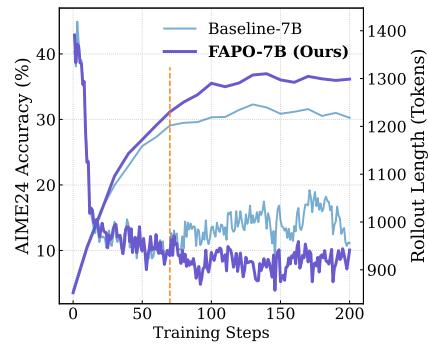


Figure 6: Self-correction Analysis.

#### 4.5 DISCUSSION: GENRM APPLICATION IN FUTURE RL SYSTEMS

411  
 412  
 413  
 414  
 Introducing generative reward models (GenRMs) may have a considerable impact on the whole RL process, influencing both algorithmic effectiveness and infrastructure efficiency. In this section, we discuss the application potential of GenRMs (with FAPO as an example) in future RL systems, considering perspectives from both algorithmic development and infrastructure design.

415  
 416  
 417  
 418  
 419  
 420  
 421  
 422  
 423  
 424  
 425  
 426  
 427  
 428  
 429  
 430  
 431  
**Algorithmic Challenge: Reward Hacking** The primary algorithmic challenge of deploying GenRM in large-scale RL systems lies in *reward hacking*, where the policy exploits imperfections in the reward signal to achieve high scores without genuinely performing the intended reasoning task. This issue is especially pronounced with complex, fine-grained reward signals produced by reward models, as they provide more opportunities for the policy to discover shortcuts for maximizing rewards. For example, we experiment with a process-based reward that assigns scores according to the proportion of correct steps before the first detected error. However, this design results in a form of reward hacking: the model tends to output only those reasoning steps in which it has very high confidence, while skipping uncertain ones altogether. As shown in Figure 7, although the step-ratio reward delivers some capability gains at the early stage compared to the baseline, the subsequent progress stalls due to reward hacking. This behavior causes an obvious *jump-in-reasoning* phenomenon, which is undesirable as it undermines the reliability of the reasoning process.

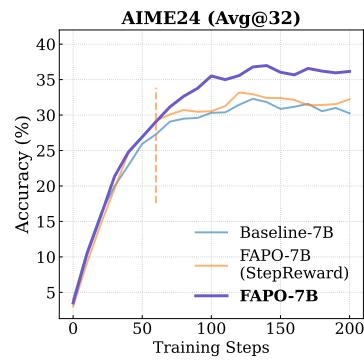


Figure 7: Step reward ablation.

432 **Infrastructure Challenge: Long-tail Problem**  
 433 The long-tail problem has long been a key bottleneck in scaling large-scale RL systems (Liang et al.,  
 434 2018), where GPUs often remain idle during the  
 435 generation of long-tail samples. A concern with  
 436 GenRM is that it introduces an additional genera-  
 437 tion stage, further aggravating this inefficiency. To  
 438 make FAPO practical for large-scale RL systems, we  
 439 adopt the following efforts: (1) **Asynchronous de-**  
 440 **sign:** We decouple GenRM from rollout inference  
 441 and actor training, reducing GPU idle time. While a  
 442 fully synchronous design may offer a better system-  
 443 level solution (Fu et al., 2025), this lies beyond the  
 444 primary research focus of our work, and we leave  
 445 this as an important future work. (2) **GenRM training:** FAPO employs an overlong reward strategy  
 446 in training and treats token budget as a key criterion in final checkpoint selection. With these efforts,  
 447 the training time of FAPO is increased by less than 20% relative to the baseline.

## 5 RELATED WORK

451 **LLM Reinforcement Learning** Reinforcement learning has emerged as a promising paradigm for  
 452 advancing LLM reasoning capabilities (Xu et al., 2025a; Zhang et al., 2025a), with the long-term  
 453 vision of Artificial Superintelligence. Recent milestones (Guo et al., 2025; OpenAI, 2025) demon-  
 454 strate that RL in verifiable tasks (Liu et al., 2025c; He et al., 2025; Feng et al., 2025) enables models  
 455 to develop reasoning patterns such as planning and tool usage, which in turn foster generalization  
 456 across broader domains (Huan et al., 2025; Seed et al., 2025). While verifiable rewards provide  
 457 only binary feedback, learning progresses gradually. Building on this direction, we explore the role  
 458 of flawed positives in this process and introduce the Flawed-Aware Policy Optimization (FAPO)  
 459 algorithm, which promotes a more natural learning trajectory and enables efficient and reliable RL.

460 **Reward Models in Reinforcement Learning** Reward models offer a promising approach to ad-  
 461 dressing the limitations of rule-based rewards, which can be broadly categorized into generative  
 462 (GenRMs) and discriminative reward models (DisRMs). Current GenRMs mainly serve as flexible  
 463 verifiers (Xu et al., 2025b; Chen et al., 2025) that augment rule-based systems when correctness  
 464 cannot be reliably assessed by predefined rules (Liu et al., 2025a; Zhao et al., 2025). In addition to  
 465 verifiable reasoning tasks, GenRMs are also applied in non-verifiable tasks, providing subjective and  
 466 rubric-based rewards (Mahan et al., 2024; Zhang et al., 2024; Zhou et al., 2025). In contrast, DisRMs  
 467 typically output fine-grained, dense rewards at every decision step, such as token-level (Rafailov  
 468 et al., 2023; Cui et al., 2025), step-level (Lightman et al., 2023; Wang et al., 2023; Ding et al., 2025),  
 469 and turn-level (Qian et al., 2025; Dong et al., 2025), to support more precise optimization. However,  
 470 the complexity of such dense rewards also makes them vulnerable to reward hacking (Gao et al.,  
 471 2023), as policies may exploit spurious shortcuts instead of learning the intended behaviors (Weng,  
 472 2024), limiting their application in large RL systems. To address this challenge, FAPO introduces  
 473 an interpretable framework that trains GenRMs to detect flawed positives and provide nuanced, in-  
 474 terpretable rewards. Both empirical experiments (Appendix 4) and theoretical analysis (Section A)  
 475 demonstrate that FAPO exhibits strong robustness against reward hacking and scales effectively.

## 6 CONCLUSION

476 This paper introduces the Flawed-Aware Policy Optimization (FAPO) algorithm to enhance effi-  
 477 ciency and reliability in LLM RL. We first reveal the role of flawed-positive rollouts: they accelerate  
 478 capability gains in the early stage but ultimately constrain reasoning quality by reinforcing unreliable  
 479 patterns. To reconcile this trade-off, FAPO applies a parameter-free reward adjustment that lever-  
 480 ages flawed positives as shortcuts during warm-up while progressively steering optimization toward  
 481 reliable reasoning. In addition, we develop a generative reward model (GenRM) with process-level  
 482 signals to accurately detect and localize reasoning errors. Both empirical experiments and theoreti-  
 483 cal analysis demonstrate the effectiveness of FAPO in future large-scale RL systems.

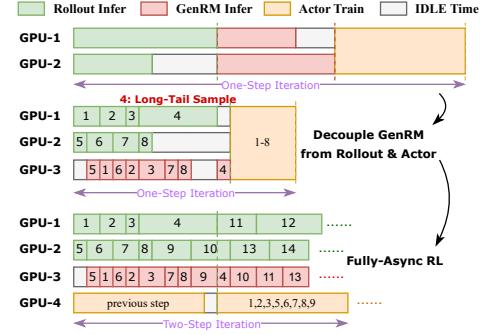


Figure 8: GenRM infrastructure design.

486 ETHICS STATEMENT  
487488 We confirm that this work adheres to ethical research practices. All data and LLMs used are publicly  
489 available (including API access) and properly cited, with no involvement of human subjects. The  
490 Use of LLM statement is illustrated in Appendix E.  
491492 REPRODUCIBILITY STATEMENT  
493494 We have made every effort to ensure the reproducibility of the results reported in this paper. Details  
495 of the algorithm design are provided in Section 3, while the infrastructure setup is described in  
496 Section 4. Additional information, including the dataset, training hyperparameters, and evaluation  
497 setups, is presented in Section 4 and Appendix B. All these details can be tracked in our code  
498 implementation (the supplementary material and the anonymous GitHub repository).  
499500 REFERENCES  
501

502 Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing  
503 Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. Polaris: A post-training recipe for scal-  
504 ing reinforcement learning on advanced reasoning models, 2025. URL <https://hkunlp.github.io/blog/2025/Polaris>.  
505

506 Ding Chen, Qingchen Yu, Pengyuan Wang, Wentao Zhang, Bo Tang, Feiyu Xiong, Xinchi Li,  
507 Minchuan Yang, and Zhiyu Li. xverify: Efficient answer verifier for reasoning model evalua-  
508 tions. *arXiv preprint arXiv:2504.10481*, 2025.  
509

510 Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu  
511 Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint*  
512 *arXiv:2502.01456*, 2025.  
513

514 Yuyang Ding, Xinyu Shi, Juntao Li, Xiaobo Liang, Zhaopeng Tu, and Min Zhang. Scan:  
515 Self-denoising monte carlo annotation for robust process reward learning. *arXiv preprint*  
516 *arXiv:2509.16548*, 2025.  
517

517 Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia  
518 Chen, Jiazen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization.  
519 *arXiv preprint arXiv:2507.19849*, 2025.  
520

521 Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha  
522 Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models.  
523 *arXiv e-prints*, pp. arXiv–2407, 2024.  
524

524 Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang,  
525 Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms.  
526 *arXiv preprint arXiv:2504.11536*, 2025.  
527

528 Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun  
529 Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for  
530 language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.  
531

532 Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In  
533 *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.  
534

534 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,  
535 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms  
536 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.  
537

538 Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang  
539 Zhang, Jiacheng Xu, Wei Shen, et al. Skywork open reasoner 1 technical report. *arXiv preprint*  
*arXiv:2505.22312*, 2025.

540 Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song,  
 541 and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*,  
 542 2021.

543 Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Pooven-  
 544 dran, Graham Neubig, and Xiang Yue. Does math reasoning improve general llm capabilities?  
 545 understanding transferability of llm reasoning. *arXiv preprint arXiv:2507.00432*, 2025.

546 Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando  
 547 Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free  
 548 evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

549 Adam Tauman Kalai, Ofir Nachum, Santosh S Vempala, and Edwin Zhang. Why language models  
 550 hallucinate. *arXiv preprint arXiv:2509.04664*, 2025.

551 Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gon-  
 552 zalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning.  
 553 In *International conference on machine learning*, pp. 3053–3062. PMLR, 2018.

554 Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan  
 555 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth  
 556 International Conference on Learning Representations*, 2023.

557 Shudong Liu, Hongwei Liu, Junnan Liu, Linchen Xiao, Songyang Gao, Chengqi Lyu, Yuzhe Gu,  
 558 Wenwei Zhang, Derek F Wong, Songyang Zhang, et al. Compassverifier: A unified and robust  
 559 verifier for llms evaluation and outcome reward. *arXiv preprint arXiv:2508.03686*, 2025a.

560 Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee,  
 561 and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint  
 562 arXiv:2503.20783*, 2025b.

563 Zihan Liu, Zhuolin Yang, Yang Chen, Chankyu Lee, Mohammad Shoeybi, Bryan Catanzaro, and  
 564 Wei Ping. Acereason-nemotron 1.1: Advancing math and code reasoning through sft and rl  
 565 synergy. *arXiv preprint arXiv:2506.13284*, 2025c.

566 Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai,  
 567 Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview  
 568 with a 1.5b model by scaling rl, 2025. Notion Blog.

569 Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato,  
 570 Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *arXiv preprint  
 571 arXiv:2410.12832*, 2024.

572 OpenAI. Learning to reason with llms. <https://openai.com/index/learning-to-reason-with-llms/>, 2024.

573 OpenAI. Introducing openai o3 and o4-mini. <https://openai.com/index/introducing-o3-and-o4-mini/>, 2025.

574 Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan  
 575 Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*,  
 576 2025.

577 Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea  
 578 Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances  
 579 in neural information processing systems*, 36:53728–53741, 2023.

580 David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Di-  
 581 rani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a bench-  
 582 mark. In *First Conference on Language Modeling*, 2024.

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

594 ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang,  
 595 Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, et al. Seed1. 5-thinking: Advancing superb rea-  
 596 soning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*, 2025.

597

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

601

602 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,  
 603 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings  
 604 of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

605

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

609

610 Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang  
 611 Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv  
 612 preprint arXiv:2312.08935*, 2023.

613

614 Yu Wang, Nan Yang, Liang Wang, and Furu Wei. Examining false positives under inference scaling  
 615 for mathematical reasoning. *arXiv preprint arXiv:2502.06217*, 2025.

616

617 Lilian Weng. Reward hacking in reinforcement learning. *lilianweng.github.io*, Nov 2024. URL  
 618 <https://lilianweng.github.io/posts/2024-11-28-reward-hacking/>.

619

620 LLM Xiaomi, Bingquan Xia, Bowen Shen, Dawei Zhu, Di Zhang, Gang Wang, Hailin Zhang,  
 621 Huaqiu Liu, Jiebao Xiao, Jinhao Dong, et al. Mimo: Unlocking the reasoning potential of lan-  
 622 guage model—from pretraining to posttraining. *arXiv preprint arXiv:2505.07608*, 2025.

623

624 Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan,  
 625 Jiahui Gong, Tianjian Ouyang, Fanjin Meng, et al. Towards large reasoning models: A survey of  
 626 reinforced reasoning with large language models. *arXiv preprint arXiv:2501.09686*, 2025a.

627

628 Zhangchen Xu, Yuetai Li, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen  
 629 Lin, and Radha Poovendran. Tinyv: Reducing false negatives in verification improves rl for llm  
 630 reasoning. *arXiv preprint arXiv:2505.14625*, 2025b.

631

632 An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li,  
 633 Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang,  
 634 Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin  
 635 Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,  
 636 Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan,  
 637 Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint  
 638 arXiv:2412.15115*, 2024a.

639

640 An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jian-  
 641 hong Tu, Jingren Zhou, Junyang Lin, et al. Qwen2. 5-math technical report: Toward mathematical  
 642 expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024b.

643

644 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu,  
 645 Chang Gao, Chengan Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint  
 646 arXiv:2505.09388*, 2025.

647

648 Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian  
 649 Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system  
 650 at scale. *arXiv preprint arXiv:2503.14476*, 2025.

651

652 Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai  
 653 Tian, Guoli Jia, Pengfei Li, et al. A survey of reinforcement learning for large reasoning models.  
 654 *arXiv preprint arXiv:2509.08827*, 2025a.

648 Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh  
649 Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint*  
650 *arXiv:2408.15240*, 2024.

651 Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu,  
652 Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical  
653 reasoning. *arXiv preprint arXiv:2501.07301*, 2025b.

654 Yulai Zhao, Haolin Liu, Dian Yu, SY Kung, Haitao Mi, and Dong Yu. One token to fool llm-as-a-  
655 judge. *arXiv preprint arXiv:2507.08794*, 2025.

656 Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu,  
657 Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical rea-  
658 soning. *arXiv preprint arXiv:2412.06559*, 2024.

659 Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Jiale Zhao, Jingwen Yang, Jianwei Lv,  
660 Kongcheng Zhang, Yihe Zhou, Hengtong Lu, et al. Breaking the exploration bottleneck: Rubric-  
661 scaffolded reinforcement learning for general llm reasoning. *arXiv preprint arXiv:2508.16949*,  
662 2025.

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

## 702 A THEORETICAL ANALYSIS OF FAPO ALGORITHM

704 To better understand the effectiveness of FAPO and the whole learning process, we present a theoretical  
 705 analysis. We begin by comparing FAPO with baseline settings to illustrate how the optimization  
 706 direction shifts during the RL process and how the corresponding advantage estimation evolves. In  
 707 this context,  $\lambda$  plays a crucial role in controlling the optimization dynamics. Building on this, we  
 708 further introduce a parameter-free, majority-guided optimization strategy.

709 In the context of LLM RL, we typically maximize the following clipped surrogate objective:

$$711 \quad \mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{(q, a) \sim \mathcal{D}, \{o_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | q)} \\ 712 \quad 713 \quad 714 \quad 715 \quad \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[ \frac{\pi_{\theta}(o_t | q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t | q, o_{<t})} \hat{A}_{i,t}, \text{clip} \left( \frac{\pi_{\theta}(o_t | q, o_{<t})}{\pi_{\theta_{\text{old}}}(o_t | q, o_{<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right] \right\}, \quad (9)$$

716 with advantage estimated in a group-relative manner:

$$717 \quad 718 \quad 719 \quad R_i = \begin{cases} 1, & \text{If } o_i \text{ is correct} \\ -1, & \text{Otherwise} \end{cases}, \quad \hat{A}_{i,t} = \frac{r_i - \text{mean}(\{R_i\}_{i=1}^G)}{\text{std}(\{R_i\}_{i=1}^G)}, \quad (10)$$

720 In this context,  $\mu_{\text{FAPO}} = \text{mean}(\{R_i\}_{i=1}^G)$  determines the sign of the advantage  $\hat{A}_{i,t}$ , which in turn  
 721 dictates the current optimization direction while  $\text{std}(\{R_i\}_{i=1}^G)$  can be regarded as a scaling factor.  
 722 FAPO introduces the following modification to the reward function:

$$724 \quad 725 \quad 726 \quad 727 \quad R_{\text{GRPO}} = \begin{cases} 1, & \text{If } o \text{ is correct} \\ -1, & \text{Otherwise} \end{cases} \xrightarrow{\text{penalty}} R_{\text{FAPO}} = \begin{cases} 1, & \text{If } o \text{ is fully correct} \\ 1 - \lambda & \text{If } o \text{ is correct yet flawed} \\ -1, & \text{Otherwise} \end{cases}. \quad (11)$$

728 We assume that the current sample contains  $n$  rollouts, with a proportion of  $\alpha$  fully correct positives  
 729 and a proportion of  $\beta$  negatives, leaving  $1 - \alpha - \beta$  as flawed positives. Since GRPO does not  
 730 distinguish between fully correct and flawed positives, the resulting advantage estimation is:

$$731 \quad 732 \quad 733 \quad \mu_{\text{GRPO}} = \text{mean}(\{R_i\}_{i=1}^n) = \frac{1 \times (1 - \beta)n + (-1) \times \beta n}{n} = 1 - \beta - \beta = 1 - 2\beta, \\ 734 \quad 735 \quad 736 \quad \sigma_{\text{GRPO}}^2 = \text{std}(\{R_i\}_{i=1}^n)^2 = \frac{(1 - \mu_{\text{GRPO}})^2 \times (1 - \beta)n + (-1 - \mu_{\text{GRPO}})^2 \times \beta n}{n} \\ = (1 - \beta)(1 - \mu_{\text{GRPO}})^2 + \beta(1 + \mu_{\text{GRPO}})^2, \quad (12)$$

737 while the advantage estimation of FAPO is

$$738 \quad 739 \quad 740 \quad 741 \quad 742 \quad 743 \quad 744 \quad 745 \quad 746 \quad 747 \quad \mu_{\text{FAPO}} = \text{mean}(\{R_i\}_{i=1}^n) = \frac{1 \times (\alpha n) + (1 - \lambda) \times (n - \alpha n - \beta n) + (-1) \times \beta n}{n} \\ = \alpha + (1 - \lambda)(1 - \alpha - \beta) - \beta \\ = \alpha - \alpha(1 - \lambda) - \beta - \beta(1 - \lambda) + 1 - \lambda \\ = 1 - 2\beta - (1 - \alpha - \beta)\lambda \\ = \mu_{\text{GRPO}} - (1 - \alpha - \beta)\lambda, \\ \sigma_{\text{FAPO}}^2 = \text{std}(\{R_i\}_{i=1}^n)^2 = \frac{(1 - \mu_{\text{FAPO}})^2 \times \alpha n + (1 - \lambda - \mu_{\text{FAPO}})^2 \times (n - \alpha n - \beta n) + (-1 - \mu_{\text{FAPO}})^2 \times \beta n}{n} \\ = \alpha(1 - \mu_{\text{FAPO}})^2 + (1 - \alpha - \beta)(1 - \lambda - \mu_{\text{FAPO}})^2 + \beta(1 + \mu_{\text{FAPO}})^2, \quad (13)$$

748 For  $\sigma_{\text{FAPO}}^2$ , we let  $\gamma = 1 - \alpha - \beta$ , so  $\sigma_{\text{FAPO}}^2 = \sigma_{\text{GRPO}}^2 - \gamma\lambda$ , then

$$749 \quad 750 \quad 751 \quad 752 \quad 753 \quad 754 \quad 755 \quad \sigma_{\text{FAPO}}^2 = \alpha(1 - \mu_{\text{FAPO}})^2 + (1 - \alpha - \beta)(1 - \lambda - \mu_{\text{FAPO}})^2 + \beta(1 + \mu_{\text{FAPO}})^2 \\ = \alpha(1 - \mu_{\text{GRPO}} + \gamma\lambda)^2 + \gamma(1 - \mu_{\text{GRPO}} + \gamma\lambda - \lambda)^2 + \beta(1 + \mu_{\text{GRPO}} - \gamma\lambda)^2 \\ = [\alpha(1 - \mu_{\text{GRPO}})^2 + \gamma(1 - \mu_{\text{GRPO}})^2 + \beta(1 + \mu_{\text{GRPO}})^2] \\ + [2\alpha\gamma\lambda(1 - \mu_{\text{GRPO}}) + 2\gamma(1 - \mu_{\text{GRPO}})(\gamma\lambda - \lambda) - 2\beta(1 + \mu_{\text{GRPO}})\gamma\lambda] \\ + [\alpha\gamma^2\lambda^2 + \gamma(\gamma - 1)^2\lambda^2 + \beta\gamma^2\lambda^2] \\ = \textcolor{red}{A} + \textcolor{blue}{B} + \textcolor{green}{C} \quad (14)$$

756 We break  $\sigma_{\text{FAPo}}^2$  down into three sub-expressions ( $\textcolor{red}{A}$ ,  $\textcolor{blue}{B}$ ,  $\textcolor{green}{C}$ ) and simplify them separately.  
 757

$$\begin{aligned}
 \textcolor{red}{A} &= \alpha(1 - \mu_{\text{GRPO}})^2 + \gamma(1 - \mu_{\text{GRPO}})^2 + \beta(1 + \mu_{\text{GRPO}})^2 \\
 &= (1 - \beta)(1 - \mu_{\text{GRPO}})^2 + \beta(1 + \mu_{\text{GRPO}})^2 \rightarrow \because \gamma = 1 - \alpha - \beta \\
 &= \sigma_{\text{GRPO}}^2 \\
 \textcolor{red}{B} &= 2\alpha\gamma\lambda(1 - \mu_{\text{GRPO}}) + 2\gamma(1 - \mu_{\text{GRPO}})(\gamma\lambda - \lambda) - 2\beta(1 + \mu_{\text{GRPO}})\gamma\lambda \\
 &= 2\gamma\lambda[\alpha(1 - \mu_{\text{GRPO}}) - (1 - \mu_{\text{GRPO}})(1 - \gamma) - \beta(1 + \mu_{\text{GRPO}})] \\
 &= 2\gamma\lambda[2\alpha\beta - 2\beta(\alpha + \beta) - \beta(2 - 2\beta)] \rightarrow \because \gamma = 1 - \alpha - \beta, \mu_{\text{GRPO}} = 1 - 2\beta \\
 &= 2\gamma\lambda \cdot (-2\beta) = -4\gamma\lambda\beta \\
 \textcolor{blue}{C} &= \alpha\gamma^2\lambda^2 + \gamma(\gamma - 1)^2\lambda^2 + \beta\gamma^2\lambda^2 \\
 &= (1 - \gamma)\gamma^2\lambda^2 + \gamma(1 - \gamma)^2\lambda^2 \rightarrow \because \gamma = 1 - \alpha - \beta \\
 &= \gamma(1 - \gamma)\lambda^2 \\
 \textcolor{red}{B} + \textcolor{blue}{C} &= -4\gamma\lambda\beta + \gamma(1 - \gamma)\lambda^2 = \gamma\lambda(\lambda(\alpha + \beta) - 4\beta) \rightarrow \because \lambda = 1 - \alpha - \beta \\
 &= \lambda\gamma(1 - \gamma)(\lambda - \frac{4\beta}{\alpha + \beta}) = \lambda\gamma(1 - \gamma)(\lambda - \frac{4}{\alpha/\beta + 1})
 \end{aligned} \tag{15}$$

775 Thus,  $\mu_{\text{FAPo}}$  and  $\sigma_{\text{FAPo}}^2$  can be expressed in terms of  $\mu_{\text{GRPO}}$  and  $\sigma_{\text{GRPO}}^2$ :

$$\begin{cases} \mu_{\text{FAPo}} = \mu_{\text{GRPO}} - \lambda\gamma \\ \sigma_{\text{FAPo}}^2 = \sigma_{\text{GRPO}}^2 + \lambda\gamma(1 - \gamma)(\lambda - \frac{4}{\alpha/\beta + 1}) \end{cases} \tag{16}$$

780 **When Optimization Direction Shift** We assume a complete learning process that begins with the  
 781 model unable to solve any problems, i.e.,  $\beta = 1$ . As training progresses,  $\beta$  gradually decreases  
 782 while  $\alpha$  increases. The shift in optimization direction occurs when:

$$\begin{aligned}
 \hat{A}_{\text{Flawed}} &= \frac{1 - \lambda - \mu_{\text{FAPo}}}{\sigma_{\text{FAPo}}} < 0 \\
 \Leftrightarrow \mu_{\text{FAPo}} &> 1 - \lambda \\
 \Leftrightarrow \mu_{\text{FAPo}} &= \alpha\lambda - \beta(2 - \lambda) + 1 - \lambda > 1 - \lambda \\
 \Leftrightarrow \lambda &> \frac{2\beta}{\alpha + \beta} = \frac{2}{\alpha/\beta + 1} \\
 \Leftrightarrow \frac{\alpha}{\beta} &> \frac{2}{\lambda} - 1
 \end{aligned} \tag{17}$$

793 **How Scaling Factor Changes** The scaling factor  $\sigma_{\text{FAPo}}^2$  changes over  $\sigma_{\text{GRPO}}^2$  is:

$$\begin{aligned}
 \sigma_{\text{FAPo}}^2 - \sigma_{\text{GRPO}}^2 &= \lambda\gamma(1 - \gamma)(\lambda - \frac{4}{\alpha/\beta + 1}) \\
 \text{when } \frac{\alpha}{\beta} &< \frac{4}{\lambda} - 1 \Rightarrow \sigma_{\text{FAPo}}^2 < \sigma_{\text{GRPO}}^2 \\
 \text{when } \frac{\alpha}{\beta} &> \frac{4}{\lambda} - 1 \Rightarrow \sigma_{\text{FAPo}}^2 > \sigma_{\text{GRPO}}^2
 \end{aligned} \tag{18}$$

803 **Summary: The Whole Optimization Process of FAPo** We introduce  $\rho = \frac{\alpha}{\beta}$  to characterize  
 804 the current optimization state, which increases monotonically from 0. For a pre-defined and fixed  
 805  $\lambda$ , the learning process drives  $\rho$  upward. Once  $\rho$  exceeds  $\frac{2}{\lambda} - 1$ , the optimization direction shifts  
 806 from reaching the correct answer (warm-up stage) toward reinforcing reliable reasoning (refinement  
 807 stage). As  $\rho$  continues to increase and surpasses  $\frac{4}{\lambda} - 1$ , the scaling factor  $\sigma$  rises accordingly, making  
 808 the advantage estimation more conservative (i.e.,  $|\hat{A}_{\text{FAPo}}| < |\hat{A}_{\text{GRPO}}|$ , for flawed-positive and fully  
 809 correct rollouts). This conservativeness helps to stabilize training by preventing overly aggressive  
 updates, while still ensuring that correct and reliable rollouts are consistently prioritized.

810    **Determining  $\lambda$**  From the above analysis, the reward assignment parameter  $\lambda$  plays a central role in  
 811    determining when the optimization shift is applied, whether conservative or aggressive. In practice,  
 812    we adopt a majority-guided strategy, which provides both intuitiveness and effectiveness. In the  
 813    early stage of training, when negative samples with proportion  $\beta$  dominate, flawed positives are  
 814    assigned positive advantages, enabling the model to acquire the ability to produce correct answers  
 815    quickly. As training progresses and fully correct rollouts become the majority (i.e.,  $\alpha > \beta$ ), the  
 816    optimization naturally shifts toward reinforcing reliable reasoning. Formally:

$$\rho_{\text{shift}} = \frac{\alpha}{\beta} = 1 \Rightarrow \lambda = \frac{2}{\rho_{\text{shift}} + 1} = 1. \quad (19)$$

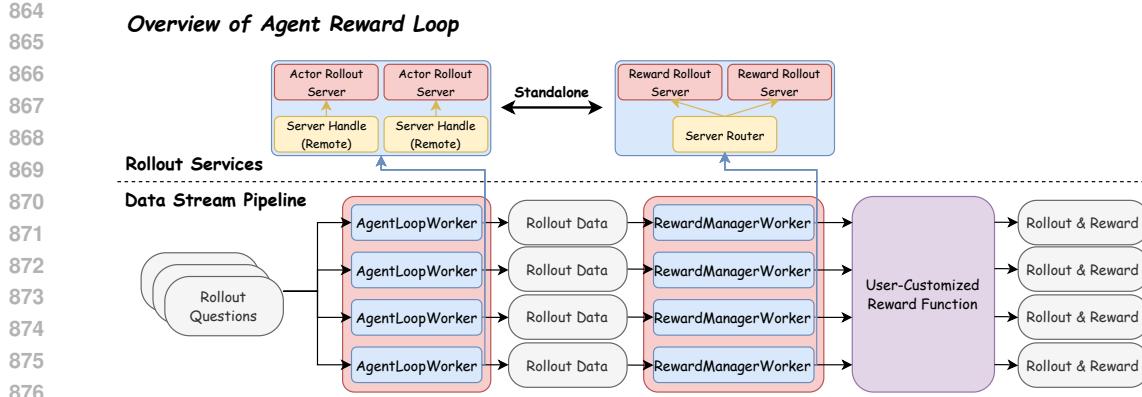
820    Therefore, we set  $\lambda = 1$  as the default configuration in FAPO.

## 823    B IMPLEMENTATION DETAILS

825    **FAPO-Reasoning** Table 1 summarizes the training configurations and hyperparameters of our  
 826    generative reward model (GenRM) and final reasoning models. For GenRM training, we follow the  
 827    practice of An et al. (2025) by using a higher rollout temperature to encourage exploration. For  
 828    reasoning model training, most settings are consistent with DAPO (Yu et al., 2025), except that we  
 829    reduce the number of rollouts from 16 to 8 to accelerate overall training speed.

831    Table 1: Training configurations and hyperparameters of our experiments.

	FAPO-GenRM-4B	Baseline & FAPO-7B	Baseline & FAPO-32B
<b>Data Configuration</b>			
Global Batch Size	512	512	512
Base Model	Qwen3-4B-Instruct	Qwen2.5-Math-7B	Qwen2.5-32B
<b>Rollout Inference</b>			
Rollout Num per Prompt	16	8	8
Temperature	1.2	1.0	1.0
Top-p	1.0	1.0	1.0
Top-k	-1	-1	-1
Max Prompt Length	5120	2048	2048
Max Response Length	8192	8192	20480
Overlong Buffer Length	4096	4096	4096
Overlong Penalty Factor	1.0	1.0	1.0
<b>Actor Training</b>			
PPO Mini Batch Size	32	32	32
Advantage Estimation Type	GRPO	GRPO	GRPO
Clipping $\epsilon_{\text{low}}$	0.2	0.2	0.2
Clipping $\epsilon_{\text{high}}$	0.28	0.28	0.28
Optimizer	Adam	Adam	Adam
Learning Rate	$10^{-6}$	$10^{-6}$	$10^{-6}$
Weight Decay	0.1	0.1	0.1
$(\beta_1, \beta_2)$	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Gradient norm clipping	1.0	1.0	1.0
Learning Rate Scheduler	constant	constant	constant
Warmup Steps	10	10	10
<b>Evaluation Setup</b>			
Temperature	0.6	1.0	1.0
Top-p	0.95	0.7	0.7
Top-k	-1	-1	-1
Max Generation Length	8192	8192	20480



918 Table 2: FAPO-GenRM results in FlawedPositiveBench and ProcessBench.  
919

920 Model	921 FlawedPositiveBench			922 ProcessBench
	Precision	923 Recall	F <sub>1</sub>	Avg. F <sub>1</sub>
<b>Discriminative Process Models (7B-70B)</b>				
Qwen2.5-Math-PRM-7B	76.6	83.5	79.9	70.5
Qwen2.5-Math-PRM-72B	74.3	91.0	81.8	76.8
<b>Generative Critic Models (7B-70B)</b>				
Qwen2.5-Math-7B-Instruct	58.0	43.8	49.9	19.9
Qwen2.5-7B-Instruct	50.0	66.2	57.0	38.9
Qwen3-1.7B (Think)	73.4	75.1	74.2	56.0
Qwen3-4B-Instruct	69.4	97.9	81.2	77.1
Qwen3-4B (Think)	89.2	84.3	82.0	79.6
Qwen3-32B (Think)	85.1	90.8	87.8	82.0
<b>FAPO-GenRM-4B (Ours)</b>	<b>87.5</b>	<b>91.4</b>	<b>89.4</b>	<b>83.3</b>

935 Table 3. FAPO outperforms the baseline in code reasoning tasks and achieves consistent improvements across a broad range of tasks.  
936938 Table 3: FAPO-Reasoning results in more evaluation benchmarks.  
939

940 Model	941 Math				942 LiveCodeBench	943 General	944 Avg.
	AIME24	AIME25	AMC	MATH			
Baseline-32B	38.9	29.5	85.0	72.8	28.6	51.0	51.0
<b>FAPO-32B</b>	<b>42.4</b>	<b>33.5</b>	<b>91.6</b>	<b>74.6</b>	<b>33.6</b>	<b>53.1</b>	<b>54.8</b>

945  
946  
947 **FAPO application in large-scale RL systems** Introducing an extra generative reward model will  
948 bring a burden to the systems. We then quantify the burden and explore the potential application of  
949 FAPO in large-scale RL systems. We provide a more detailed breakdown of the time distribution  
950 across different RL stages in Table 4.  
951952 Table 4: Time distribution across different RL stages in different settings.  
953

954 Model	955 Rollout (Infer)	956 Rollout Len	957 FAPO-GenRM (Infer)	958 GenRM Len	959 Policy Update (Train)	960 Nodes
FAPO-7B	42%	1.1k	<b>18%</b>	3.0k	33%	<b>4</b>
FAPO-32B	60%	2.3k	<b>14%</b>	3.2k	20%	<b>8</b>
Qwen3-4B	72%	12.0k	<b>10%</b>	3.8k	14%	<b>16</b>

961 The relative inference cost for long-cot models actually decreases, specifically: (1) Rollout: Long-  
962 CoT models exhibit a strong long-tail issue: the generation time is bounded by the longest trajectory  
963 in the batch. Therefore, rollout time increases significantly as the model generates longer traces  
964 (e.g., 12k tokens for Qwen3-4B). (2) GenRM: In contrast, GenRM inference time does not increase  
965 with longer trajectories. We observe that the GenRM output length stays nearly constant across  
966 models (e.g., 3.2k → 3.8k). (3) Policy update: This stage accounts for a relatively small portion of  
967 the total cost and primarily scales with model size rather than trajectory length. Overall, the root  
968 cause is that the critic task for FAPO-GenRM does not require a very long response length.  
969970 **Ablation study of hyper-parameter  $\lambda$**   $\lambda$  is the only key hyper-parameter introduced in the FAPO  
971 algorithm. We explore the parameter tuning based on the FAPO-7B setting, with results illustrated  
972 in Table 5

972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
Table 5: Hyper-parameter tuning of  $\lambda$ .

Setting	Performance
Baseline-7B ( $\rho = +\infty \Rightarrow \lambda = 0$ )	32.1
FAPO with $\rho = 2 \Rightarrow \lambda = 1/3$	34.6
FAPO with $\rho = 1 \Rightarrow \lambda = 1$ (default setting)	36.8
FAPO with $\rho = 1/2 \Rightarrow \lambda = -1/3$	39.6

We can conclude that:

- Flaw-aware learning consistently improves performance, as all configurations outperform the baseline.
- Achieving the best performance require tuning the parameter  $\lambda$ . In the 7B setting, a more aggressive strategy leads to larger gains. The configuration with  $\rho = 1/2$ , which corresponds to an optimization shift where about one third of the rollouts are fully correct, achieves the best performance.

So overall, flaw-aware learning leads to performance gains, but achieving the best performance requires tuning the parameter  $\lambda$ . That said, FAPO introduces only this single additional parameter, which makes the tuning process relatively easy.

**Human Verification on the reliability of FAPO-32B** Beyond the previous LLM-as-a-judge evaluation with Qwen3-32B, we further conduct a detailed human verification to assess the process reliability of FAPO-32B. Specifically, we randomly sample 20 positive cases with correct final answers and manually examine whether they contain unreliable reasoning patterns, with the results summarized in Table 6. Through this analysis, we observe that (1) multiple-choice questions exhibit a noticeably higher proportion of flawed positives compared to math word problems, where the final answer is often a symbolic expression; and (2) our method consistently outperforms the baseline under both LLM-as-a-judge and human verification, further demonstrating its effectiveness in mitigating flawed-positive issues.

Table 6: LLM-as-a-judge and human verification of flawed positive ratio.

Model	AIME24 (Math Word Problem)	AIME25 (Math Word Problem)	GPQA-Diamond (Multi-Choice Problem)
Baseline-32B (LLM-as-a-judge)	15.5	10.9	45.7
FAPO-32B (LLM-as-a-judge)	<b>7.1 (-8.4)</b>	<b>1.7 (-9.2)</b>	<b>42.0 (-3.7)</b>
Baseline-32B (Human-Verification)	4 / 20	1 / 20	10 / 20
FAPO-32B (Human-Verification)	<b>2 / 20</b>	<b>0 / 20</b>	<b>7 / 20</b>

**How Flawed Learning contributes to Performance Gains** In section 2.2, we demonstrate that the flawed positive acts as a role of stepping stones to performance gains, and Figure 2 (d) illustrates the correlation between them. Here, we provide a more complete explanation of how mitigating flawed positives leads to performance gains.

- **Early Stages: Flawed Behavior → More Correct Rollouts:** Prior studies (Zheng et al., 2024; Wang et al., 2025) have shown that flawed reasoning often acts as a shortcut to the correct final answer. This bias, inherited from pre-training (Kalai et al., 2025), leads to certain flawed rollouts. Thus, in the early phase of RL, flawed positives naturally increase the number of correct final answer rollouts.

- **Early Stages: More Correct Rollouts → Early performance gains:** A larger pool of correct rollouts yields more positive rewards, providing stronger supervision and driving exploitation early in training. This effect can be directly observed in train-time reward statistics in Table 7.
- **Later Stages: Penalizing flawed behavior → Fewer flawed rollouts:** Our theoretical analysis demonstrates the optimization distribution shift: flawed rollouts receive negative advantage, progressively reducing the model’s tendency to produce flawed processes. This trend is reflected in Figure 1 (left), where the proportion of flawed positives decreases steadily as training progresses.
- **Later Stages: Fewer flawed rollouts → Performance gains:** As flawed rollouts diminish, the model allocates more rollout chances to fully correct trajectories. Consequently, the RL loop receives a higher density of genuinely useful reward signals, as reflected in Table 8, improving final performance.

Table 7: Early-stage train-time rewards (view flawed rollout as positive rollout).

Step	10	20	30	40	50
Baseline-7B	0.14	0.27	0.29	0.34	0.36
FAPO-7B	<b>0.18</b>	<b>0.33</b>	<b>0.37</b>	<b>0.42</b>	<b>0.44</b>

Table 8: Later-stage train-time rewards (view flawed rollout as negative rollout).

Step	120	140	160	180	200
Baseline-7B	0.35	0.37	0.36	0.40	0.39
FAPO-7B	<b>0.38</b>	<b>0.40</b>	<b>0.41</b>	<b>0.44</b>	<b>0.45</b>

**Model Selection in FAPO Experiments** We explain the model selection in both FAPO-GenRM and FAPO-Reasoning:

- **Model Selection in FAPO-GenRM:** We adopt Qwen3-4B-Instruct as the base model for GenRM training, considering three factors: (1) it demonstrates strong instruction-following and basic error-detection capability (as shown in Table 2), making it a suitable initialization for RL; (2) its relatively small size ensures efficient training and faster inference, with Instruct models producing shorter responses than think-style models; (3) it avoids potential concerns of knowledge leakage into the final RL process that may arise if the base model is overly strong.
- **Model Selection in FAPO-Reasoning:** We follow the same setup as DAPO (Yu et al., 2025) and use Qwen2.5-Math-7B and Qwen2.5-32B as base models, as (1) the learning curves of pre-trained models clearly illustrate the entire exploration-exploitation trajectory starting from near-zero performance, whereas RL-finetuned Instruct models often exhibit instability; (2) these two models also strike a good balance between response length and training speed.

## D LIMITATIONS AND FUTURE WORK

Our work presents several limitations that point to promising future directions, both in algorithmic effectiveness and infrastructure design. On the algorithmic side, although this work trains on mathematical reasoning tasks, FAPO has strong potential in broader settings such as multi-choice tasks, multi-turn interactions, and agent-based RL, where flawed processes are often more pronounced and problematic. We will also further validate the effectiveness of FAPO across a wider range of model architectures (e.g., MoE) and larger model scales. On the infrastructure side, while our decoupled design improves efficiency and inference speed, its applicability to fully asynchronous RL systems remains uncertain, as we discussed in Section 4.5, specifically Table 1. We regard these as important research directions in our future work.

## E LLM USAGE

During the writing of this paper, AI assistants are employed to assist with minor language refinement. Their suggestions are limited to enhancing clarity and readability, without influencing the research design, experiments, or conclusions. All content was carefully reviewed, validated, and revised by the authors to ensure accuracy and fidelity to the research.

1080 F PROMPTS  
1081

1082     1083     Prompts for Flawed Positive Detection

1084     **Prompt for Outcome Reward Model (ORM):**  
1085     The following is a math problem with its ground truth answer, along  
1086     with an AI solution:

1087     **[Math Problem]**  
1088     *{problem statement}*

1089     **[Ground Truth]**  
1090     *{ground truth answer}*

1091     **[AI Solution]**  
1092     *{AI Solution}*

1093     Your task is to review and critique the solution step by step, and  
1094     output whether the AI solution is correct.  
1095     Please reason step by step, put your final answer (i.e., 'True' or  
1096     'False') in \boxed{ }  
1097

1098     **Prompt for Process Reward Model (PRM):**  
1099     The following is a math problem with its ground truth answer, along  
1100     with an AI solution (split into paragraphs, enclosed with tags and  
1101     indexed from 0):

1102     **[Math Problem]**  
1103     *{problem statement}*

1104     **[Ground Truth]**  
1105     *{ground truth answer}*

1106     **[AI Solution]**  
1107     *<paragraph\_0>...</paragraph\_0>*  
1108     *<paragraph\_1>...</paragraph\_1>*  
1109     *.....*  
1110     *<paragraph\_(n-1)>...</paragraph\_(n-1)>*

1111     Your task is to review and critique the solution paragraph by  
1112     paragraph. Once you identify an error in a paragraph, return the  
1113     index of the paragraph where the earliest error occurs. Otherwise,  
1114     return the index of -1 (which typically denotes 'not found').  
1115     Please reason step by step, put your final answer (i.e., the index)  
1116     in \boxed{ }  
1117

1118     1119     Prompts for Reasoning

1120     **Prompt for DAPO-Math-17K, AIME24, and AIME25:**  
1121     *{problem statement}*

1122     Please reason step by step, and put your final answer within  
1123     \boxed{ }.

1124     **Prompt for GPQA-Diamond:**  
1125     *{problem statement}*

1126     A. *{Choice\_A}*  
1127     B. *{Choice\_B}*  
1128     C. *{Choice\_C}*  
1129     D. *{Choice\_D}*

1130     Please reason step by step, and put your final answer (only the  
1131     choice letter) within \boxed{ }.

1134 **G FLAWED POSITIVE SAMPLES WITH HUMAN ANALYSIS**  
11351136 **Flawed Positive Example 1**  
11371138 **Question Source: AIME24**1139 **Question:**

1140 Alice chooses a set  $A$  of positive integers. Then Bob lists all  
 1141 finite nonempty sets  $B$  of positive integers with the property that  
 1142 the maximum element of  $B$  belongs to  $A$ . Bob's list has 2024 sets.  
 1143 Find the sum of the elements of  $A$ .

1144 **Generation Model: Qwen2.5-Math-7B (Pre-Trained Base Model)**1145 **Response:**

1146 We are given that Bob lists all finite nonempty sets  $B$  of positive  
 1147 integers where the maximum element of  $B$  belongs to  $A$ . Bob's list  
 1148 contains 2024 such sets. We need to find the sum of the elements of  
 1149  $A$ .

1149 **Step-by-Step Reasoning:**1150 **1. Understanding the Problem:**

- Let  $A = \{a_1, a_2, \dots, a_n\}$  where  $a_1 < a_2 < \dots < a_n$ .
- For each  $a_i \in A$ , Bob lists all finite nonempty sets  $B$  such that the maximum element of  $B$  is  $a_i$ .

1151 **2. Counting the Sets for Each  $a_i$ :**

- For a given  $a_i$ , the sets  $B$  can include any subset of the integers less than  $a_i$ , plus  $a_i$  itself.
- The number of subsets of the integers less than  $a_i$  is  $2^{a_i-1}$  (since there are  $a_i-1$  integers less than  $a_i$ ).
- However, since  $B$  must be nonempty, we subtract 1 from this count (the empty set).

1152 **Therefore, the number of sets  $B$  for a given  $a_i$  is:**

$$2^{a_i-1} - 1$$

1162 **3. Total Number of Sets:**

- The total number of sets listed by Bob is the sum of the number of sets for each  $a_i$ :

$$\sum_{i=1}^n (2^{a_i-1} - 1) = 2024$$

- Simplifying, we get:

$$\sum_{i=1}^n 2^{a_i-1} - n = 2024$$

$$\sum_{i=1}^n 2^{a_i-1} = 2024 + n$$

1173 **4. Finding  $n$ :**

- We need to find  $n$  such that the sum  $\sum_{i=1}^n 2^{a_i-1}$  is minimized.
- The smallest possible values for  $a_i$  are  $1, 2, 3, \dots, n$ .

1174 Let's test with small values of  $n$ :

- For  $n = 1$ :

$$2^{1-1} = 1 \quad (\text{too small})$$

- For  $n = 2$ :

$$2^{1-1} + 2^{2-1} = 1 + 2 = 3 \quad (\text{too small})$$

- For  $n = 3$ :

$$2^{1-1} + 2^{2-1} + 2^{3-1} = 1 + 2 + 4 = 7 \quad (\text{too small})$$

- For  $n = 4$ :

$$2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} = 1 + 2 + 4 + 8 = 15 \quad (\text{too small})$$

- For  $n = 5$ :

$$2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} = 1 + 2 + 4 + 8 + 16 = 31 \quad (\text{too small})$$

```

1188
1189 - For  $n = 6$ :
1190  $2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} + 2^{6-1} = 1 + 2 + 4 + 8 + 16 + 32 = 63$  (too small)
1191
1192 - For  $n = 7$ :
1193  $2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} + 2^{6-1} + 2^{7-1} = 127$  (too small)
1194
1195 - For  $n = 8$ :
1196  $2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} + 2^{6-1} + 2^{7-1} + 2^{8-1} = 255$  (too small)
1197
1198 - For  $n = 9$ :
1199  $2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} + 2^{6-1} + 2^{7-1} + 2^{8-1} + 2^{9-1} = 511$  (too small)
1200
1201 - For  $n = 10$ :
1202  $2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} + 2^{6-1} + 2^{7-1} + 2^{8-1} + 2^{9-1} + 2^{10-1} = 1023$  (too small)
1203
1204 - For  $n = 11$ :
1205  $2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} + 2^{6-1} + 2^{7-1} + 2^{8-1} + 2^{9-1} + 2^{10-1} + 2^{11-1} = 2047$  (too large)
1206 Therefore,  $n = 10$  is the correct value because:
1207  $2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} + 2^{6-1} + 2^{7-1} + 2^{8-1} + 2^{9-1} + 2^{10-1} = 1023$ 
1208
1209 And:
1210  $1023 + 10 = 1033$  (which is too large)
1211
1212 So, we need to adjust  $n$  to be 10 and check the sum:
1213  $2^{1-1} + 2^{2-1} + 2^{3-1} + 2^{4-1} + 2^{5-1} + 2^{6-1} + 2^{7-1} + 2^{8-1} + 2^{9-1} + 2^{10-1} = 1023$ 
1214
1215 And:
1216  $1023 + 10 = 1033$  (which is too large)
1217 Therefore, the correct  $n$  is 10, and the sum of the elements of  $A$  is:
1218  $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 = 55$ 
1219
1220 Thus, the sum of the elements of  $A$  is:
1221 55
1222
1223 Human Analysis: There are two clear mistakes in the AI's response:
1224 1. The number of sets  $B$  for a  $a_i$  should be  $2^{a_i-1}$  (not  $2^{a_i-1} - 1$ ). This incorrect assumption leads to a completely flawed derivation throughout the rest of the solution.
1225 2. The correct set  $A$  should be  $\{4, 6, 7, 8, 9, 10, 11\}$ , instead of  $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ . From the solution process, it's clear that the AI guessed a value close to the target, rather than identifying the exact condition required.
1226
1227 In summary, although the final answer happens to be correct, the overall reasoning process is fundamentally flawed.
1228
1229
1230
1231
1232
1233
1234 Flawed Positive Example 2
1235
1236 Question Source: DAPO-Math-17K
1237 Question:
1238 Let  $a, b, c$  be the three roots of the equation  $x^3 - (k+1)x^2 + kx + 12 = 0$ , where  $k$  is a real number. If  $(a-2)^3 + (b-2)^3 + (c-2)^3 = -18$ , find the value of  $k$ .
1239
1240 Response Generation Model: Llama-3.3-70B-Instruct (ShortCoT Model)
1241 Response:

```

1242

**Step 1: Recall Vieta's formulas for a cubic equation**

1243

Vieta's formulas for a cubic equation  $x^3 + px^2 + qx + r = 0$  with roots  $a, b, c$  state that:  $a + b + c = -p$ ,  $ab + bc + ac = q$ , and  $abc = -r$ . Applying these formulas to the given equation  $x^3 - (k+1)x^2 + kx + 12 = 0$ , we get  $a + b + c = k + 1$ ,  $ab + bc + ac = k$ , and  $abc = -12$ .

1244

**Step 2: Expand the given expression using the identity for**

1245

$$(a - 2)^3 + (b - 2)^3 + (c - 2)^3$$

1246

The identity for  $(a - 2)^3 + (b - 2)^3 + (c - 2)^3$  can be expanded as  $(a^3 - 6a^2 + 12a - 8) + (b^3 - 6b^2 + 12b - 8) + (c^3 - 6c^2 + 12c - 8)$ . This simplifies to  $(a^3 + b^3 + c^3) - 6(a^2 + b^2 + c^2) + 12(a + b + c) - 24$ .

1247

**Step 3: Use the given equation to find  $a^3 + b^3 + c^3$** 

1248

Since  $a, b, c$  are roots of the equation  $x^3 - (k+1)x^2 + kx + 12 = 0$ , we can express  $a^3, b^3$ , and  $c^3$  in terms of  $a, b, c$ , and  $k$ . Substituting  $x = a, b, c$  into the equation gives  $a^3 = (k+1)a^2 - ka - 12$ ,  $b^3 = (k+1)b^2 - kb - 12$ , and  $c^3 = (k+1)c^2 - kc - 12$ . Summing these gives  $a^3 + b^3 + c^3 = (k+1)(a^2 + b^2 + c^2) - k(a + b + c) - 36$ .

1249

**Step 4: Find  $a^2 + b^2 + c^2$  using the sum of roots**

1250

From Vieta's formulas, we know  $a + b + c = k + 1$ . Squaring both sides gives  $(a + b + c)^2 = (k + 1)^2$ , which expands to  $a^2 + b^2 + c^2 + 2(ab + bc + ac) = k^2 + 2k + 1$ . Since  $ab + bc + ac = k$ , we can substitute this in to get  $a^2 + b^2 + c^2 + 2k = k^2 + 2k + 1$ , simplifying to  $a^2 + b^2 + c^2 = k^2 + 1$ .

1251

**Step 5: Substitute  $a^2 + b^2 + c^2$  into the expression for  $a^3 + b^3 + c^3$** 

1252

Substituting  $a^2 + b^2 + c^2 = k^2 + 1$  into  $a^3 + b^3 + c^3 = (k+1)(a^2 + b^2 + c^2) - k(a + b + c) - 36$  gives  $a^3 + b^3 + c^3 = (k + 1)(k^2 + 1) - k(k + 1) - 36$ . Simplifying yields  $a^3 + b^3 + c^3 = k^3 + k + k^2 + 1 - k^2 - k - 36 = k^3 - 35$ .

1253

**Step 6: Substitute known values into the expanded expression for**

1254

$$(a - 2)^3 + (b - 2)^3 + (c - 2)^3$$

1255

Given that  $(a - 2)^3 + (b - 2)^3 + (c - 2)^3 = -18$ , and using the expansion from Step 2, we substitute  $a^3 + b^3 + c^3 = k^3 - 35$  and  $a + b + c = k + 1$  into the expression. We also need  $a^2 + b^2 + c^2 = k^2 + 1$  from Step 4. The expression becomes  $(k^3 - 35) - 6(k^2 + 1) + 12(k + 1) - 24 = -18$ .

1256

**Step 7: Solve for  $k$** 

1257

**Simplifying the equation**  $(k^3 - 35) - 6(k^2 + 1) + 12(k + 1) - 24 = -18$  **gives**  $k^3 - 6k^2 + 12k - 35 - 6 + 12k + 12 - 24 = -18$ . **Combining like terms yields**  $k^3 - 6k^2 + 24k - 53 = -18$ . **Adding 18 to both sides gives**  $k^3 - 6k^2 + 24k - 35 = 0$ .

1258

This is a cubic equation in terms of  $k$ .

1259

**Step 8: Solve the cubic equation for  $k$** 

1260

To solve  $k^3 - 6k^2 + 24k - 35 = 0$ , let's try to factor it or find a rational root. Checking for simple integer solutions,  $k = 5$  is found to be a root (through trial and error or using the Rational Root Theorem), as  $5^3 - 6(5)^2 + 24(5) - 35 = 125 - 150 + 120 - 35 = 60$ . However, this was an error in calculation: the correct calculation for  $k = 5$  should be  $5^3 - 6(5)^2 + 24(5) - 35 = 125 - 150 + 120 - 35 = 60$ , which is not correct. The correct approach is to actually solve the equation correctly. Let's correct the step: We need to solve  $k^3 - 6k^2 + 24k - 35 = 0$ . Trying  $k = 5$  correctly:  $5^3 - 6(5)^2 + 24(5) - 35 = 125 - 150 + 120 - 35 = 60$ , which does not equal 0, indicating an error in identifying  $k = 5$  as a root. The correct factorization or solution method must be applied.

1261

**Step 9: Correctly identify and solve for  $k$** 

1262

The error in Step 8 indicates a miscalculation in identifying a root. The correct approach involves factoring or applying a method like synthetic division if a rational root is suspected, or using numerical methods. However, given the mistake in calculation and approach, let's reconsider the equation and directly evaluate potential rational roots or use given conditions more effectively.

1263

**The final answer is:** 5

1264

**Human Analysis:** The correct simplified equation is  $k^3 - 6k^2 + 12k - 35 = 0$ , not  $k^3 - 6k^2 + 24k - 35 = 0$ . This typo leads to an incorrect solution of  $k$ . However, what's quite peculiar is that despite the model's multiple failed attempts to correct itself, it ultimately guessed  $k = 5$ .

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296  
1297

## Flawed Positive Example 3

1298

**Question Source:** DAPO-Math-17K

1299

**Question:**

1300

Let  $a, b, c$  be distinct numbers such that the equations  $x^2 + ax + 1 = 0$  and  $x^2 + bx + c = 0$  have a common real root, and the equations  $x^2 + x + a = 0$  and  $x^2 + cx + b = 0$  also have a common real root. Compute the sum  $a + b + c$ .

1302

**Response Generation Model:** Owen3-1.7B (LongCoT Model, Think Mode)

1303

**Response:**

1304

&lt;think&gt;...&lt;/think&gt; (Think Content Omitted Here)

1305

1306

To solve the problem, we are given three distinct numbers  $a, b, c$  such that:

1307

- The equations  $x^2 + ax + 1 = 0$  and  $x^2 + bx + c = 0$  share a common real root,
- The equations  $x^2 + x + a = 0$  and  $x^2 + cx + b = 0$  also share a common real root.

1310

We are to compute the value of  $a + b + c$ .

1311

**Step 1: Use the Common Root Condition**

1313

Let  $\alpha$  be a real root of both equations  $x^2 + ax + 1 = 0$  and  $x^2 + bx + c = 0$ .

1314

Substituting  $\alpha$ :

1315

$$\alpha^2 + a\alpha + 1 = 0 \quad (1) \quad \alpha^2 + b\alpha + c = 0 \quad (2)$$

1316

Subtracting (1) from (2):

1317

$$(b - a)\alpha + (c - 1) = 0 \Rightarrow \alpha = \frac{1 - c}{b - a} \quad (3)$$

1319

Now, substitute  $\alpha$  into (1) and (2) to express relationships between  $a, b, c$ .

---

1322

**Step 2: Use the Second Pair of Equations**

1323

Let  $\beta$  be a real root of both  $x^2 + x + a = 0$  and  $x^2 + cx + b = 0$ .

1324

Substituting  $\beta$ :

1325

$$\beta^2 + \beta + a = 0 \quad (4) \quad \beta^2 + c\beta + b = 0 \quad (5)$$

1326

Subtracting (4) from (5):

1327

$$(c - 1)\beta + (b - a) = 0 \Rightarrow \beta = \frac{a - b}{c - 1} \quad (6)$$

---

1330

**Step 3: Use the Relationship Between  $\alpha$  and  $\beta$** 

1331

From the problem, it is known that  $\beta = \frac{1}{\alpha}$ . Therefore:

1332

$$\frac{a - b}{c - 1} = \frac{1}{\alpha} \Rightarrow \alpha = \frac{1 - c}{b - a}$$

1333

Also, from (6):

1334

$$\beta = \frac{a - b}{c - 1}$$

1335

So:

1336

$$\beta = \frac{a - b}{c - 1} = \frac{1}{\alpha} = \frac{1 - c}{b - a}$$

1337

This gives a relationship between  $a, b, c$ , and we proceed by assuming  $\alpha = 1$ , which simplifies the algebra.

---

1343

**Step 4: Find  $a, b, c$  with  $\alpha = 1$** 

1344

Suppose  $\alpha = 1$  is a common root of both pairs of equations.

1345

From equation (1):

1346

$$1^2 + a \cdot 1 + 1 = 0 \Rightarrow 1 + a + 1 = 0 \Rightarrow a = -2$$

1347

Now, since  $\alpha = 1$  is a root of equation (2):

1348

$$1^2 + b \cdot 1 + c = 0 \Rightarrow 1 + b + c = 0 \Rightarrow c = -1 - b$$

1349

1350

1351 Now, from the second pair of equations,  $\beta = 1$  is a root of both  $x^2 +$   
 1352  $x + a = 0$  and  $x^2 + cx + b = 0$ . Substitute  $x = 1$  into both equations:

1353  $- 1^2 + 1 + a = 0 \Rightarrow 1 + 1 + (-2) = 0 \Rightarrow 0 \checkmark$

1354  $- 1^2 + c \cdot 1 + b = 0 \Rightarrow 1 + c + b = 0 \Rightarrow c + b = -1 \checkmark$

1355 Thus, consistent.

1356 Now, the sum:

$$a + b + c = -2 + b + (-1 - b) = -3$$

1357  $\cdots$

1358 **Final Answer**

1359 -3

1360 **Human Analysis:** The AI makes an unwarranted assumption that  $\alpha = 1$ ,  
 1361 whereas in fact, deriving this requires a complex reasoning process.  
 1362 Although the final answer is correct, the reasoning is flawed and  
 1363 the correct value is reached coincidentally.

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403