

R²PO: Decoupling Training Trajectories from Inference Responses for LLM Reasoning

Anonymous ACL submission

Abstract

Reinforcement learning has become a central paradigm for improving LLM reasoning. However, existing methods use a single policy to produce both inference responses and training optimization trajectories. The objective conflict between generating stable inference responses and diverse training trajectories leads to insufficient exploration, which harms reasoning capability. In this paper, to address the problem, we propose R²PO (Residual Rollout Policy Optimization), which introduces a lightweight Residual Rollout-Head atop the policy to decouple training trajectories from inference responses, enabling controlled trajectory diversification during training while keeping inference generation stable. Experiments across multiple benchmarks show that R²PO consistently outperforms baselines, achieving average accuracy gains of 3.1% on MATH-500 and 2.4% on APPS, while also reducing formatting errors and mitigating length bias for stable optimization. Our code is publicly available at [this link](#).

1 Introduction

In recent years, Large Language Models (LLMs) Reinforcement learning (RL) has become a central paradigm for improving the reasoning capabilities of large language models (LLMs), underpinning approaches such as Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022) and Verifiable-Reward-based fine-tuning (RLVR) (Lambert et al., 2024; DeepSeek-AI et al., 2025), and have achieved remarkable success in complex reasoning tasks, including mathematical problem solving, code generation, and multi-step decision making (DeepSeek-AI et al., 2025; Wan et al., 2025; Ouyang et al., 2022; DeepSeek-AI et al., 2025; Yu et al., 2025). By optimizing models directly against task-specific or preference-based reward signals, RL enables LLMs to move beyond pure imitation and ex-

hibit stronger performance on complex reasoning tasks (Stiennon et al., 2020; Shinn et al., 2023).

In existing RL methods such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024), a single policy network generates responses at inference time and optimization trajectories during training. However, an inherent gap exists between the *ideal response* for inference and the *ideal optimization trajectories* required for training. The former emphasizes accuracy and consistency, and thus tolerates homogeneous outputs, whereas the latter requires high diversity and broad solution-space coverage that even includes informative erroneous paths to provide effective gradient signals. However, since both objectives are enforced within the same parameter space, the optimization process is dominated by response-level correctness, which suppresses exploration for diversity. As a result, the model often falls into self-repetition during training (Moalla et al., 2024; Cui et al., 2025).

To overcome this issue, we propose R²PO (Residual Rollout Policy Optimization). Unlike traditional paradigms that compress the two conflicting objectives into a single policy, we introduce a lightweight residual module integrated atop the frozen backbone to decouple inference responses and training trajectories. The Residual Rollout-Head learns to make controlled adjustments to the base distribution, exploring candidate trajectories that the main policy would otherwise downweight. This architectural decoupling keeps the primary policy stable and high-quality at inference time, while simultaneously ensuring diverse trajectory exploration during training via the independent residual Rollout-Head. The residual Rollout-Head can be discarded at inference time, ensuring efficient and consistent generation during deployment.

We investigate the effectiveness of R²PO across mathematical reasoning and code generation tasks on Qwen2.5-3B and Qwen3-8B. Experimental results underscore that R²PO significantly outper-

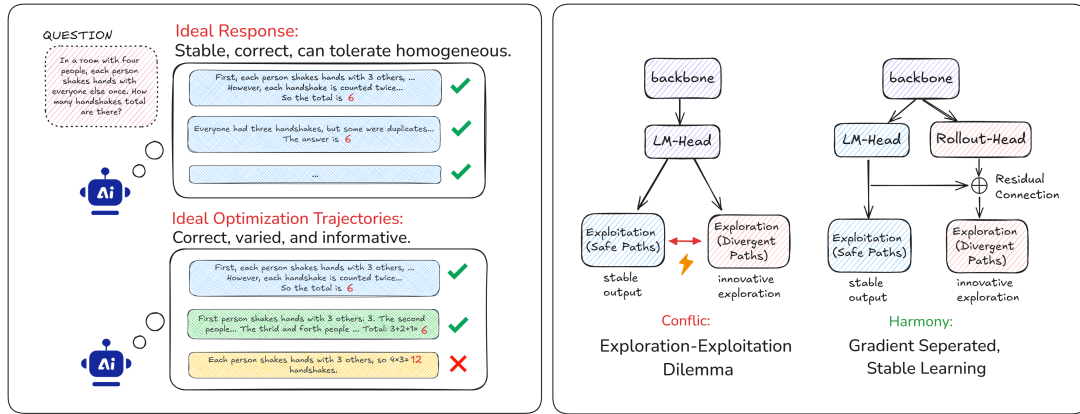


Figure 1: **Decoupling Ideal Responses and Optimization Trajectories.** Left: The mismatch between ideal inference responses and ideal training trajectories. Right: Single-head policy optimization versus R^2PO with a decoupled Rollout-Head for stable exploration.

083 forms the standard GRPO baseline, achieving an
 084 average accuracy increment of **3.1%** on MATH-
 085 500 and **2.4%** on APPS. We further reveal that
 086 our framework effectively mitigates formatting er-
 087 rors and length bias, maintaining a stable reason-
 088 ing trajectory even in the late stages of optimization.

089 The contributions of this work are as follows:

- 090 • We introduce the Rollout-Head, optimized
 091 with a Group Inverse-Frequency reward, to
 092 achieve decoupling between exploration and
 093 exploitation, supporting structured explo-
 094 ration while maintaining stable exploitation.
- 095 • We elucidate the Regularization Effect of this
 096 design, which shields the primary policy from
 097 destructive gradient noise and enhances over-
 098 all training stability in GRPO.
- 099 • We show that R^2PO consistently enhances
 100 reasoning performance and training stability
 101 across multiple benchmarks with negligible
 102 computational overhead.

103 2 Related Works

104 We review related works from three comple-
 105 mentary perspectives: (1) reinforcement learn-
 106 ing for reasoning-oriented LLMs, (2) explo-
 107 ration-exploitation mechanisms in LLM rein-
 108 forcement learning, and (3) decoupled or auxil-
 109 iary architectural designs.

110 **RL for Reasoning LLMs** Large Language Mod-
 111 els (LLMs) have demonstrated significant reason-
 112 ing improvements through Reinforcement Learn-
 113 ing from Human Feedback (RLHF) (Ouyang et al.,
 114 2022). While traditional PPO-based frameworks

(Schulman et al., 2017) rely on separate value mod-
 115 els, recent paradigms like Reinforcement Learn-
 116 ing with Verifiable Rewards (RLVR) (Lambert
 117 et al., 2025; DeepSeek-AI et al., 2025) have gained
 118 prominence in deterministic domains like mathe-
 119 matical problems and coding. However, these ver-
 120 ifiable reward landscapes are often sparse and un-
 121 forgiving, exacerbating the instability of the single-
 122 head policy architecture.

Exploration-Exploitation Dilemma The gap
 124 between the ideal response and the ideal opti-
 125 mization trajectories also reflects the exploration-
 126 exploitation dilemma. Balancing exploration and
 127 exploitation is a fundamental challenge in rein-
 128 forcement learning (RL) (Sutton and Barto, 2018).
 129 Traditional approaches like Soft Actor-Critic (SAC)
 130 (Haarnoja et al., 2018), typically employ entropy
 131 regularization to sustain exploration and prevent
 132 premature convergence. In LLM fine-tuning, most
 133 paradigms (Stiennon et al., 2020; DeepSeek-AI
 134 et al., 2025) rely on KL-divergence penalties to im-
 135 plicitly preserve diversity. However, such soft reg-
 136 ularization is often insufficient, as policy entropy
 137 can still collapse rapidly, leading to suboptimal
 138 solutions (Moalla et al., 2024; Cui et al., 2025).

140 To better understand and mitigate this issue, sev-
 141 eral works have investigated the role of entropy in
 142 LLM RL. (Park et al., 2025) and (Lehman and
 143 Stanley, 2011) analyze the interaction between
 144 PPO clipping mechanisms and entropy dynamics.
 145 (Cheng et al., 2025) examines how exploration that
 146 characterized by entropy, affects the depth and di-
 147 versity of reasoning trajectories from a reason-
 148 centric perspective. Other studies further incor-
 149 porate entropy-related signals into the optimiza-

tion objective, such as entropy-regularized policy optimization (Xu et al., 2025) or entropy-aware token-level updates (Wen et al., 2024), in order to encourage broader exploration during generation.

Decoupled and Auxiliary Architectures The idea of using auxiliary heads or decoupled structures has been explored in various domains. In computer vision, residual connections (Xu et al., 2024a) allowed for deeper training by isolating residual mappings. In RL, some architectures separate the advantage and value streams to stabilize learning. Closest to our work are efforts in Parameter-Efficient Fine-Tuning (PEFT (Xu et al., 2024b)) and auxiliary learning, where lightweight modules (like Adapters or LoRA) are used to expand model capabilities.

3 Method

The core idea of R^2PO is to isolate exploratory behaviors from the primary policy optimization process while maintaining a shared backbone representation, which is achieved by augmenting the standard language modeling architecture with a lightweight Rollout-Head.

Training proceeds through iterative optimization cycles, where each cycle consists of two coordinated stages. In the first stage, the Rollout-Head is optimized to generate diverse exploratory trajectories, while the backbone and LM-Head remain fixed to preserve representational stability. In the second stage, the Rollout-Head is frozen and serves exclusively as a behavioral sampler, providing high-quality trajectories to guide the optimization of the base policy. This iterative circle scheme allows the exploration module and the base policy to co-evolve without directly interfering with each other. Figure 2 illustrates the overall framework and the interaction between the two stages.

3.1 Motivation

Figure 1 provides an intuitive view of a structural limitation in existing single-head policy optimization. During RL fine-tuning, the same output head is simultaneously required to serve two conflicting distinct roles: producing reliable and concise responses for evaluation, and generating diverse trajectories that expose informative gradient signals for learning. While correct and stable responses are sufficient at inference time, effective optimization often relies on trajectories that are heterogeneous, exploratory, and occasionally incorrect.

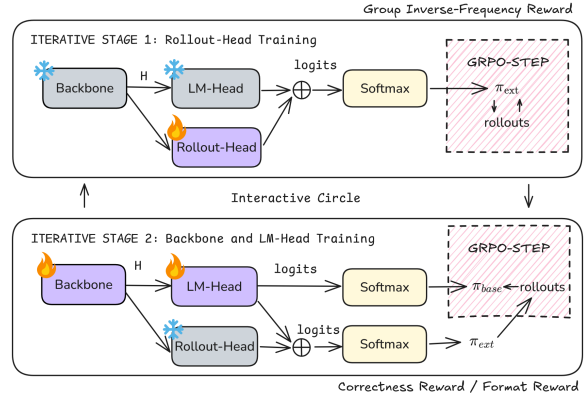


Figure 2: Overview of R^2PO . An iterative loop of two alternating stages: Stage 1 (top) optimizes the Rollout-Head using GIF reward with a frozen backbone; Stage 2 (bottom) updates the base policy on task rewards using the Rollout-Head as a fixed behavioral sampler.

In single-head architectures, these two roles are implicitly coupled. Gradients induced by response-level correctness dominate the shared output space, progressively suppressing deviations that could otherwise reveal alternative reasoning paths. As a result, the policy tends to reinforce a narrow subset of “safe” trajectories, even when broader exploration would be beneficial for optimization. This coupling manifests as conservative trajectory reuse and limits the model’s ability to discover rare but informative reasoning behaviors.

These observations motivate an architectural separation between response generation and trajectory optimization. Rather than forcing a single output head to reconcile these competing requirements, we seek a mechanism that preserves a stable primary policy while allowing controlled, structured perturbations during rollout generation. This design principle directly leads to the Rollout-Head architecture introduced in the following section.

3.2 Overall Architecture

To separate response generation and trajectory exploration, we introduce the **Rollout-Head** as a residual branch integrated atop the transformer backbone. This module is implemented as a two-layer Multi-Layer Perceptron (MLP), taking the backbone hidden states $H \in \mathbb{R}^d$ as input, and producing a logit-level offset over the vocabulary space \mathbb{R}^V , where d denotes the hidden dimension of the transformer and V is the vocabulary size. Let f_{LM} denote the original LM-Head and f_{RO} denote the Rollout-Head. Based on a shared backbone representation, we define two policies that differ

only in their output heads: the main policy (π_θ) produced by the original LM-Head and is responsible for stable exploitation, and the exploration policy (π_ϕ) that incorporated the residual perturbation introduced by the Rollout-Head.

$$\begin{aligned}\pi_\theta &= \text{Softmax}(f_{\text{LM}}(H)), \\ \pi_\phi &= \text{Softmax}(f_{\text{RO}}(H) + f_{\text{LM}}(H)).\end{aligned}\quad (1)$$

To ensure stability during the cold-start phase, we employ Zero-initialization for the Rollout-Head, making the initial exploration policy identical to the base policy at the beginning of training. By modeling exploration as a residual logit perturbation, the Rollout-Head learns to modify the base distribution in a controlled manner, enabling diverse trajectory generation without catastrophic interference with the base policy parameters.

3.3 Reward Design

Correctness and Formatting Reward. Following recent advancements in large-scale reasoning model training (Le et al., 2022; Shinn et al., 2023; Gehring et al., 2025), we utilize a rule-based reward system rather than a neural reward model to provide dense and reliable feedback. For a given query x and a generated response y_i , the total reward $r(x, y_i)$ is typically decomposed into:

$$r(x, y_i) = \mathbb{1}_{\text{correct}}(y_i) \cdot R_{\text{acc}} + \mathbb{1}_{\text{format}}(y_i) \cdot R_{\text{fmt}} \quad (2)$$

where $\mathbb{1}_{\text{correct}}$ and $\mathbb{1}_{\text{format}}$ are indicator functions checking for numerical correctness and adherence to specific formatting templates (e.g., enclosing the final answer between `<answer>` and `</answer>`). This deterministic reward mechanism ensures that the optimization signal remains consistent throughout the iterative training process.

Group Inverse-Frequency Reward. To encourage unconventional behaviors, we prioritize trajectories with lower group-relative frequencies, regardless of whether their immediate task-specific rewards are optimal. This is consistent with prior work on Intrinsic Motivation literature (Bellemare et al., 2016) and Novelty Search (Lehman and Stanley, 2011), which suggest that rewarding surprise or uniqueness can prevent premature convergence in sparse reward environments.

To formally define the Group Inverse-Frequency (GIF) Reward, let $\mathcal{G} = \{y_1, \dots, y_G\}$ be a group of G sampled trajectories. Each trajectory y_i is associated with a standard verifiable reward r_i . We partition the group into K distinct reward bins

$\mathcal{B}_1, \dots, \mathcal{B}_K$, where $\mathcal{B}_k = \{y_i \in \mathcal{G} \mid r_i = v_k\}$ and v_k is a unique reward value present in the group. The raw GIF score s_i for trajectory $y_i \in \mathcal{B}_k$ is defined as the inverse of the bin’s cardinality:

$$s_i = \frac{1}{|\mathcal{B}_k|} = \frac{1}{\sum_{j=1}^G \mathbb{1}[r_j = r_i]} \quad (3)$$

To ensure training stability and prevent gradient explosion, we apply Z-score Standardization to the scores within the group to derive the final exploration reward R_{GIF} .

3.4 Iterative Training

R²PO adopts a two-stage alternating optimization scheme to separate the update signals for exploration and exploitation. Both stages utilize the standard Group Relative Policy Optimization (GRPO) (Shao et al., 2024) objective:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) &= \mathbb{E}_{x \sim \mathcal{D}, \{y_i\}_{i=1}^G \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \right. \\ &\quad \min \left(r_{i,t}(\theta) \hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{i,t} \right) \\ &\quad \left. - \beta \mathbb{D}_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \right],\end{aligned}\quad (4)$$

where G is the number of generated responses per query x . The importance ratio $r_{i,t}(\theta)$ and the advantage $\hat{A}_{i,t}$ of token $y_{i,t}$ are given by:

$$r_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t} \mid x, y_{i,<t})} \quad (5)$$

$$\hat{A}_{i,t} = \hat{A}_i = \frac{r(x, y_i) - \text{mean}(\{r(x, y_i)\}_{i=1}^G)}{\text{std}(\{r(x, y_i)\}_{i=1}^G)}, \quad (6)$$

respectively, where all the tokens in y_i share the same advantage as \hat{A}_i . π_{ref} typically refers to the initial SFT model used for KL regularization, while π_{old} denotes the policy from the previous iteration used for importance sampling.

3.4.1 Rollout-Head Optimization

In Rollout-Head optimization stage, we only optimize the parameters of the Rollout-Head (ϕ) while keeping the backbone and the primary LM-Head frozen. Here, π_ϕ serves as both the trajectory generator and the optimization target. Let y denote the trajectories sampled from π_ϕ , the optimization objective for Stage 1 is formulated as:

$$\min \mathcal{J}_{\text{GRPO}}(\phi), \quad y \sim \pi_\phi \quad (7)$$

In our primary experiments, we employ the Group Inverse-Frequency Reward as the default exploration signal, as it explicitly encourages diversity in optimization trajectories. A detailed comparative analysis between this reward and alternative designs is provided in Section 4.3.3.

3.4.2 Main Policy Optimization

In the main policy optimization stage, we freeze the Rollout-Head and update the backbone and LM-Head (θ). Crucially, the frozen Rollout-Head acts exclusively as a **behavioral sampler** to provide diverse trajectories that the base policy might not otherwise discover. The optimization objective focuses on the base policy:

$$\min \mathcal{J}_{\text{GRPO}}(\theta), \quad y \sim \pi_{\phi} \quad (6)$$

By alternating between these two stages, the exploration module continuously adapts to the evolving base policy, while the base policy gradually absorbs high-quality exploratory behaviors under a more stabilized training signal.

3.5 Discussion

Approaches that address the exploration-exploitation dilemma discussed in Section 2 still operate within the same parameter space for exploration and exploitation, relying on modified objectives, clipping strategies, or additional sampling procedures. As a result, the gradients that promote exploration and those that drive reward maximization remain entangled. In contrast, our proposed Rollout-Head introduces a structural solution by explicitly decoupling the exploration signal from the policy update parameters. This architectural separation alleviates gradient interference between training objectives and inference objectives, yielding a more stable and scalable exploration mechanism for LLM reinforcement learning.

Moreover, our method is not merely a parameter-efficient extension where PEFT typically prioritizes parameter savings. Instead, R²PO serves as a dedicated explorer designed to shield the primary policy from the destructive noise of high-variance reasoning paths, allowing the backbone to remain focused on learning stable, high-reward reasoning patterns while the auxiliary head absorbs the volatility inherent in deep-space exploration.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of R²PO in enhancing reasoning capabilities and training stability. We begin by comparing R²PO against the competitive GRPO baseline across two representative domains: mathematical reasoning and code generation. Subsequently, we provide a series of diagnostic analyses to examine the mechanistic drivers of R²PO. Finally, we discuss the computational efficiency and parameter overhead of the proposed framework to demonstrate its practical scalability.

4.1 Experimental Setup

We evaluate the performance of R²PO across diverse reasoning tasks. This section details our base models, benchmark datasets, and evaluation protocol.

Base Models We utilize Qwen2.5-3B (Qwen et al., 2025) and the reasoning-enhanced Qwen3-8B (Yang et al., 2025) as backbones to evaluate the scalability of our method across different scales.

Datasets Our evaluation spans two representative domains: Mathematical Reasoning and Code Generation. In the tasks of mathematical reasoning, we use GSM8K (Cobbe et al., 2021) for training and in-distribution testing, and MATH-500 (Zhang et al., 2025) as an out-of-distribution benchmark to assess generalization in sophisticated logical deduction. For programming tasks, we employ MBPP (Tao et al., 2024) for optimization, while evaluating functional synthesis capabilities on HumanEval (Yadav and Mondal, 2025) and the multi-level algorithmic challenges in APPS (Hendrycks et al., 2021).

Metrics For all benchmarks, we primarily report the Pass@1 accuracy. During training, we adopt a rule-based parser to extract numerical answers from <answer> tags for mathematical tasks, while for coding tasks, generated programs are validated through a standard execution sandbox to verify functional correctness. During inference time, all benchmarks are evaluated following the protocol described in the Evaluation Protocol paragraph.

Evaluation Protocol All benchmarks are evaluated using the OpenCompass framework (Contributors, 2023). To ensure a fair comparison and eliminate potential performance discrepancies caused by prompt sensitivity, we further customized the evaluation configurations to maintain strict consistency between the inference prompts and the

Table 1: Benchmark results comparison. Our proposed R^2PO framework significantly outperforms the standard GRPO baseline across both mathematical and programming reasoning tasks.

Backbone	Method	Math (Pass@1)		Code (Pass@1)			Avg.
		GSM8K	MATH500	MBPP	HumanEval	APPS	
Qwen2.5-3B	Base (SFT)	76.12	36.00	54.20	64.63	9.02	47.99
	+ GRPO	80.74	42.20	58.60	68.90	8.22	51.73
	+ R^2PO (Ours)	83.17	45.60	59.00	68.90	9.52	53.24
Qwen3-8B	Base (SFT)	80.74	51.00	58.80	66.46	12.84	53.97
	+ GRPO	88.55	54.40	66.60	86.59	15.38	62.30
	+ R^2PO (Ours)	88.48	57.20	67.20	87.80	15.56	63.25

training templates. It should be noted that, as the rule-based metric provided by OpenCompass for the MATH500 dataset is not fully compatible with the output format generated during our training, we therefore evaluate on this specific dataset using our own tailored metric.

4.2 Main Results

The comparative results presented in Table 1 show that our proposed R^2PO framework consistently outperforms the vanilla GRPO baseline across both the 3B and 8B model scales.

Superiority in Complex Reasoning A key observation is that the performance gains of R^2PO are more pronounced on challenging, out-of-distribution benchmarks. For instance, on the MATH-500 dataset, the Qwen2.5-3B model with our method achieves an absolute improvement of **3.4%** over the standard GRPO (45.6% vs. 42.2%), and the Qwen3-8B model sees a similar gain of **2.8%**. Similarly, in the APPS coding benchmark, which requires sophisticated algorithmic synthesis, R^2PO achieves the highest Pass@1 scores across both backbones. This suggests that the decoupling provided by the Rollout-Head allows the model to explore more diverse and advanced reasoning paths, which are critical for solving complex problems that require multi-step logical deduction.

Decoupling as a Stability Buffer While standard GRPO shows a performance decline in some coding tasks (e.g., Qwen2.5-3B on APPS dropping from 9.02 to 8.22), our R^2PO maintains or even enhances the base model’s performance. This supports our hypothesis regarding the representational conflict. In vanilla GRPO, the destructive gradient noise from high-variance exploration may probably destabilizes the learned patterns in the primary head. In contrast, the Rollout-Head acts as

a structural buffer. This architectural isolation ensures that the core policy can exploit stable reasoning paradigms while the rollout branch probes the fringes of the solution space.

Scalability and Generalization Even when the base policy is already strong, the Rollout-Head continues to provide incremental gains. The fact that the base policy (π_θ) reaches higher accuracy after being trained with trajectories sampled from the exploration policy (π_ϕ) confirms the effectiveness of our *Two-Stage Iterative Training*. The exploration head successfully discovers rare but correct reasoning paths that effectively expand the cognitive boundaries of the primary backbone.

4.3 Stability and Robustness Analysis

In this section, we analyze the stability and robustness of R^2PO from three perspectives: (1) reward dynamics and stability, (2) robustness to reward misspecification, (3) length bias and verbosity.

4.3.1 Reward Dynamics and Stability

We monitor the mean and variance of rewards during the training process on GSM8K. Specifically, we distinguish between the global reward mean of all generated trajectories and the subset of Informative Rollouts defined as groups where reward variance exists, thereby providing non-zero advantage signals for policy updates. As illustrated in Figure 3, our model consistently achieves a higher mean reward with a more pronounced upward trajectory compared to the baseline GRPO. Our model, on the other hand, keeps the reward variance essentially the same as the baseline model. This suggests that the decoupled architecture facilitates a more thorough search of the solution space before converging to a stable, high-performing policy.

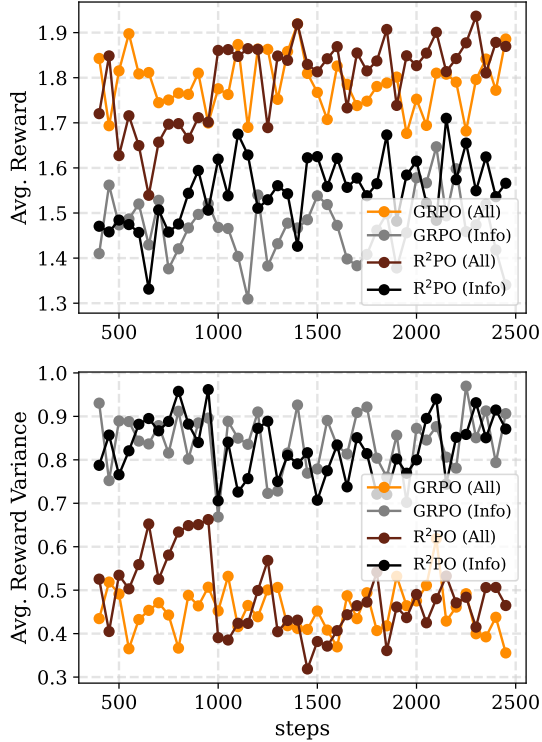


Figure 3: The mean and variance of rewards during the training process on GSM8K. The upper row shows the mean reward, while the lower row displays the variance of the mean reward. “Info” refers to Informative Rollouts, which contain varying rewards and thus supply non-zero advantage signals for policy updates.

4.3.2 Robustness to Reward Misspecification

A critical challenge in rule-based reinforcement learning is the gap between the training reward function and the actual evaluation criteria. We encountered a case of Reward Misspecification: during training, the format reward was loosely defined (requiring only the presence of `<think>` and `<answer>` tags), whereas the evaluation script strictly enforced a single occurrence of these tags.

In the baseline GRPO, we observed a sudden spike in reward variance on the test set between steps 2500 and 3000, exceeding even the initial exploration phase. As shown in A.4, diagnostic analysis revealed that the model learned to output redundant `<think>` blocks, which is a behavior that was not penalized during training but caused catastrophic failures during evaluation. According to the error evolution trends in Table 2, this noise or sub-optimal behavior is an emergent artifact of prolonged training. We utilize the GRPO model at step 100 as a baseline, as it represents the stage where the model has fundamentally mastered the necessary formatting requirements to obtain rewards.

Table 2: Evolution of Format Error Rate in Baseline GRPO. The noise (redundant `<think>` tags) emerges and accelerates during the late stages of training due to reward misspecification. Measured on the first 100 samples of the GSM8K test set.

Training Steps	100	2000	2500	3000
Error Rate (%)	0.125	2.750	6.875	46.625

To test whether our model’s resilience to this issue was coincidental, we conducted a **Perturbation Experiment**. After 2000 steps, we manually injected redundant tags into successful trajectories for 10 steps to simulate a “noise trap.” We then measured the probability of the model adopting this erroneous behavior in the subsequent 100 steps.

Table 3: Results of the Perturbation Experiment. Measured on the first 100 samples of the GSM8K test set.

Steps	GRPO	R²PO
2000 (Start)	0.125%	0.000%
2050	0.375%	0.000%
2100	0.875%	0.000%

The baseline GRPO quickly succumbed to the perturbation, with the error rate rising to 0.875%, whereas our model maintained a 0% error rate. This clearly demonstrates that our decoupled head architecture provides significantly higher **perturbation immunity**, likely because the primary policy is effectively shielded from the high-variance exploration noise by the Rollout-Head.

4.3.3 Length Bias and Verbosity

Standard GRPO often suffers from a “verbosity bias” where incorrect answers tend to increase in length to minimize token-level penalties (DeepSeek-AI et al., 2025). In contrast, our model produces more concise responses. We compare the average length of correct and incorrect answers against a baseline GRPO model from the early training phase (100 steps) to serve as a reference point, as its outputs are sufficiently standardized to avoid evaluation bias under our metrics. Our results indicate that by decoupling exploration, the model avoids the trap of using excessive tokens as a buffer for negative gradients, leading to more efficient reasoning paths.

Table 4: Average Response Length (tokens) Comparison. Our model suppresses the "length explosion" observed in late-stage GRPO, particularly for incorrect trajectories. Measured on the first 100 samples of the GSM8K test set.

Response Type	Base	GRPO	R ² PO
Correct Answers	211.61	211.95	204.12
Incorrect Answers	281.22	295.28	281.59

4.4 Ablation Study

To identify what truly drives the gains of R²PO: the decoupled architecture, the reward design, or the freeze–update schedule, we isolate these factors in two additional ablations that test (i) the role of the Group Inverse-Frequency Reward signal, and (ii) the necessity of the Rollout-Head.

Reward Signal Instead of the Group Inverse-Frequency (GIF) Reward, we train the Rollout-Head using the main reward, i.e., the same accuracy and formatting rewards as the primary policy used. This ablation study evaluate whether the improvements come mainly from the *architectural decoupling* itself rather than the *Group Inverse-Frequency Reward* signal. As shown in Table 5, the Correctness and Formatting Reward configuration achieved the highest overall performance, slightly exceeding the Group Inverse-Frequency Reward variant. This suggests that the structural separation of exploration is the primary driver of success, providing a stable sandbox for the model to refine reasoning paths. This is an encouraging finding, as it implies that R²PO can be successfully deployed without complex reward engineering.

Residual Branch To evaluate if the gains simply result from the staggered frozen training schedule, we implemented a version without the Rollout-Head, where we periodically froze the LM-Head while updating the backbone, and vice-versa. As also shown in Table 5, the optimization without the Rollout-Head outperformed vanilla GRPO but lagged behind the full R²PO, confirming that the additional parameter capacity of the residual branch is essential for effective decoupling.

4.5 Efficiency and Parameter Discussion

To evaluate the computational efficiency of R²PO, we analyze the parameter overhead and its impact on both training and inference phases. As shown in Table 6, although R²PO introduces a set of aux-

Table 5: Ablation Study on GSM8K using Qwen2.5-3B

Configuration	GSM8K (Math)
Vanilla GRPO	80.74
R ² PO (w/o Rollout-Head)	82.18
R ² PO (w/ GIF Reward)	83.17
R ² PO (w/ Main Reward)	83.55

iliary parameters (ranging from 7.8% to 10.2%), this growth is confined to the training stage. In the inference phase, the Rollout-Head is detached, resulting in zero additional latency.

Table 6: Parameter overhead across model scales.

Model	Base	Full	Added (Δ)	Growth (%)
Qwen2.5-3B	3.09 B	3.40 B	315.36 M	$\sim 10.2\%$
Qwen3-8B	8.19 B	8.83 B	639.11 M	$\sim 7.8\%$

Beyond parameter count, we further examine the peak GPU memory usage during training. Compared to vanilla GRPO, which requires 58,519 MB of GPU memory, R²PO exhibits a markedly different memory profile across its two stages. In Stage 1, where only the Rollout-Head is optimized and the backbone is frozen, the peak memory usage drops significantly to 26,343 MB, representing a reduction of over 55%. In Stage 2, where the backbone and LM-Head are updated while the Rollout-Head is frozen, the memory usage (60,563 MB) is comparable to that of vanilla GRPO. The proposed framework therefore incurs no additional inference cost and only marginal training overhead, making it suitable for large-scale deployment.

5 Conclusion

To bridge the gap between response generation and trajectory optimization, we introduce the Rollout-Head, an architectural refinement that decouples policy functions by isolating exploration into a residual branch. This approach mitigates the representational conflicts and training instabilities inherent in standard reinforcement learning, with experiments showing significant gains in robustness against reward misspecification and length bias compared to GRPO. Ultimately, such structural decoupling proves essential for building stable and efficient large-scale reasoning models in complex reward landscapes.

606 Limitation

607 R²PO is evaluated only on models of 3B and 8B pa-
608 rameters. Its behavior on substantially larger mod-
609 els (e.g., 70B+) or smaller-scale models remains to
610 be explored. For large-scale models, it is unclear
611 whether the Rollout-Head can continue to yield
612 meaningful gains with a relatively small number
613 of additional parameters. Conversely, for smaller
614 models, the parameter increase introduced by the
615 Rollout-Head, although minimized, may still con-
616 stitute a non-negligible overhead given their limited
617 capacity and vocabulary size. Moreover, our exper-
618 iments focus exclusively on single-turn reasoning.
619 Extending this decoupled exploration framework to
620 multi-turn dialogue settings or long-horizon agent
621 tasks may introduce additional challenges, which
622 we leave for future investigation.

623 Ethics Statement

624 While R²PO enhances the reasoning and problem-
625 solving capabilities of LLMs, we recognize po-
626 tential risks associated with its deployment. Im-
627 proved reasoning could theoretically be misused
628 for generating sophisticated malicious code or au-
629 tomated misinformation. To mitigate such risks,
630 our experiments rely solely on publicly available
631 datasets, and the training follows standard RLHF
632 safety alignment practices. Moreover, the structural
633 decoupling introduced in R²PO improves optimiza-
634 tion transparency and model interpretability, which
635 can help in auditing model behavior and reducing
636 unintended outputs. We emphasize that the deploy-
637 ment of R²PO should consider these risks and be
638 accompanied by appropriate monitoring and access
639 control.

640 Use of AI Assistants

641 AI assistants (specifically ChatGPT and Gemini)
642 were used in this research primarily for language
643 polishing, improving grammatical structures, and
644 refining the clarity of the manuscript. All original
645 methodology, experimental design, and data analy-
646 sis were conducted solely by the human authors.

647 References

648 Marc G. Bellemare, Sriram Srinivasan, Georg Ostro-
649 vski, Tom Schaul, David Saxton, and Rémi Munos.
650 2016. [Unifying count-based exploration and intrinsic motivation](#). In *Advances in Neural Information*

Processing Systems 29: Annual Conference on Neu- 652
ral Information Processing Systems 2016, December 653
5-10, 2016, Barcelona, Spain, pages 1471–1479. 654

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, 655
Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 656
2025. [Reasoning with exploration: An entropy per- 657](#)
[spective](#). *CoRR*, abs/2506.14758. 658

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan 659
Martic, Shane Legg, and Dario Amodei. 2017. [Deep 660](#)
[reinforcement learning from human preferences](#). In 661
Advances in Neural Information Processing Systems 662
30: Annual Conference on Neural Information Pro- 663
cessing Systems 2017, December 4-9, 2017, Long 664
Beach, CA, USA, pages 4299–4307. 665

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, 666
Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias 667
Plappert, Jerry Tworek, Jacob Hilton, Reiichiro 668
Nakano, Christopher Hesse, and John Schulman. 669
2021. [Training verifiers to solve math word prob- 670](#)
[lems](#). *CoRR*, abs/2110.14168. 671

OpenCompass Contributors. 2023. [Opencompass: A 672](#)
[universal evaluation platform for foundation models](#). 673
<https://github.com/open-compass/opencompass>. 674

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, 675
Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, 676
Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, 677
Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and 678
Ning Ding. 2025. [The entropy mechanism of rein- 679](#)
[forcement learning for reasoning language models](#). 680
CoRR, abs/2505.22617. 681

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, 682
Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, 683
Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, 684
Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhi- 685
hong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 686
2025. [Deepseek-rl: Incentivizing reasoning capa- 687](#)
[bility in llms via reinforcement learning](#). *CoRR*, 688
abs/2501.12948. 689

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingx- 690
uan Wang, Bochao Wu, Chengda Lu, Chenggang 691
Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, 692
Damai Dai, Daya Guo, Dejian Yang, Deli Chen, 693
Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, 694
and 181 others. 2025. [Deepseek-v3 technical report](#). 695
Preprint, arXiv:2412.19437. 696

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard 697
Mella, Taco Cohen, and Gabriel Synnaeve. 2025. 698
[RLEF: grounding code llms in execution feedback 699](#)
[with reinforcement learning](#). In *Forty-second Interna- 700*
tional Conference on Machine Learning, ICML 2025, 701
Vancouver, BC, Canada, July 13-19, 2025. OpenRe- 702
view.net. 703

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and 704
Sergey Levine. 2018. [Soft actor-critic: Off-policy 705](#)
[maximum entropy deep reinforcement learning with 706](#)
[a stochastic actor](#). In *Proceedings of the 35th Inter- 707*
national Conference on Machine Learning, ICML 708

709		2018, <i>Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018</i> , volume 80 of <i>Proceedings of Machine Learning Research</i> , pages 1856–1865. PMLR.	
710			
711			
712	Dan Hendrycks, Steven Basart, Saurav Kadavath, Man-		
713	tas Mazeika, Akul Arora, Ethan Guo, Collin Burns,		
714	Samir Puranik, Horace He, Dawn Song, and Jacob		
715	Steinhardt. 2021. Measuring coding challenge com-		
716	petence with APPS . In <i>Proceedings of the Neural</i>		
717	<i>Information Processing Systems Track on Datasets</i>		
718	<i>and Benchmarks 1, NeurIPS Datasets and Bench-</i>		
719	<i>marks 2021, December 2021, virtual</i> .		
720	Nathan Lambert, Jacob Morrison, Valentina Pyatkin,		
721	Shengyi Huang, Hamish Ivison, Faeze Brahma-		
722	n, Lester James V. Miranda, Alisa Liu, Nouha Dziri,		
723	Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf,		
724	Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras,		
725	Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and		
726	4 others. 2024. Tülu 3: Pushing frontiers in open lan-		
727	guage model post-training . <i>CoRR</i> , abs/2411.15124.		
728	Nathan Lambert, Jacob Morrison, Valentina Pyatkin,		
729	Shengyi Huang, Hamish Ivison, Faeze Brahma-		
730	n, Lester James V. Miranda, Alisa Liu, Nouha Dziri,		
731	Shane Lyu, Yuling Gu, Saumya Malik, Victoria		
732	Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le		
733	Bras, Oyvind Tafjord, Chris Wilhelm, Luca Sol-		
734	daini, and 4 others. 2025. Tulu 3: Pushing fron-		
735	tiers in open language model post-training . <i>Preprint</i> ,		
736	arXiv:2411.15124.		
737	Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio		
738	Savarese, and Steven Chu-Hong Hoi. 2022. Coder1:		
739	Mastering code generation through pretrained mod-		
740	els and deep reinforcement learning . In <i>Advances in</i>		
741	<i>Neural Information Processing Systems 35: Annual</i>		
742	<i>Conference on Neural Information Processing Sys-</i>		
743	<i>tems 2022, NeurIPS 2022, New Orleans, LA, USA,</i>		
744	<i>November 28 - December 9, 2022</i> .		
745	Joel Lehman and Kenneth O. Stanley. 2011. Aban-		
746	doning objectives: Evolution through the search for		
747	novelty alone . <i>Evol. Comput.</i> , 19(2):189–223.		
748	Skander Moalla, Andrea Miele, Daniil Pyatko, Razvan		
749	Pascanu, and Caglar Gulcehre. 2024. No represen-		
750	tation, no trust: Connecting representation, collapse,		
751	and trust issues in PPO . In <i>Advances in Neural In-</i>		
752	<i>formation Processing Systems 38: Annual Confer-</i>		
753	<i>ence on Neural Information Processing Systems 2024,</i>		
754	<i>NeurIPS 2024, Vancouver, BC, Canada, December</i>		
755	<i>10 - 15, 2024</i> .		
756	Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida,		
757	Carroll Wainwright, Pamela Mishkin, Chong Zhang,		
758	Sandhini Agarwal, Katarina Slama, Alex Ray, and 1		
759	others. 2022. Training language models to follow in-		
760	structions with human feedback. <i>Advances in neural</i>		
761	<i>information processing systems</i> , 35:27730–27744.		
762	Jaesung R. Park, Junsu Kim, Gyeongman Kim, Jiny-		
763	oung Jo, Sean Choi, Jaewoong Cho, and Ernest K.		
764	Ryu. 2025. Clip-low increases entropy and clip-high		
765	decreases entropy in reinforcement learning of large		
766	language models . <i>CoRR</i> , abs/2509.26114.		
	Qwen, :, An Yang, Baosong Yang, Beichen Zhang,		767
	Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan		768
	Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan		769
	Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin		770
	Yang, Jiaxi Yang, Jingren Zhou, and 25 oth-		771
	ers. 2025. Qwen2.5 technical report . <i>Preprint</i> ,		772
	arXiv:2412.15115.		773
	John Schulman, Filip Wolski, Prafulla Dhariwal, Alec		774
	Radford, and Oleg Klimov. 2017. Proximal policy		775
	optimization algorithms . <i>CoRR</i> , abs/1707.06347.		776
	Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu,		777
	Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu,		778
	and Daya Guo. 2024. Deepseekmath: Pushing the		779
	limits of mathematical reasoning in open language		780
	models . <i>CoRR</i> , abs/2402.03300.		781
	Noah Shinn, Federico Cassano, Ashwin Gopinath,		782
	Karthik Narasimhan, and Shunyu Yao. 2023. Re-		783
	flexion: language agents with verbal reinforcement		784
	learning . In <i>Advances in Neural Information Pro-</i>		785
	<i>cessing Systems 36: Annual Conference on Neural</i>		786
	<i>Information Processing Systems 2023, NeurIPS 2023,</i>		787
	<i>New Orleans, LA, USA, December 10 - 16, 2023</i> .		788
	Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M.		789
	Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford,		790
	Dario Amodei, and Paul F. Christiano. 2020. Learn-		791
	ing to summarize with human feedback . In <i>Advances</i>		792
	<i>in Neural Information Processing Systems 33: An-</i>		793
	<i>annual Conference on Neural Information Processing</i>		794
	<i>Systems 2020, NeurIPS 2020, December 6-12, 2020,</i>		795
	<i>virtual</i> .		796
	Richard S. Sutton and Andrew G. Barto. 2018. Rein-		797
	forcement Learning: An Introduction , 2 edition. MIT		798
	Press, Cambridge, MA.		799
	Ning Tao, Anthony Ventresque, Vivek Nallur, and Tak-		800
	farinas Saber. 2024. Enhancing program synthesis		801
	with large language models using many-objective		802
	grammar-guided genetic programming . <i>Algorithms</i> ,		803
	17(7):287.		804
	Ziyu Wan, Yunxiang Li, Yan Song, Hanjing Wang, Linyi		805
	Yang, Mark Schmidt, Jun Wang, Weinan Zhang,		806
	Shuyue Hu, and Ying Wen. 2025. Rema: Learning to		807
	meta-think for llms with multi-agent reinforcement		808
	learning . <i>CoRR</i> , abs/2503.09501.		809
	Muning Wen, Junwei Liao, Cheng Deng, Jun		810
	Wang, Weinan Zhang, and Ying Wen. 2024. Entropy-		811
	regularized token-level policy optimiza-		812
	tion for language agent reinforcement . <i>Preprint</i> ,		813
	arXiv:2402.06700.		814
	Huazhi Xu, Xiaoyan Luo, and Wencong Xiao. 2024a. Multi-		815
	residual unit fusion and wasserstein distance-		816
	based deep transfer learning for mill load recognition .		817
	<i>Signal Image Video Process.</i> , 18(4):3187–3196.		818
	Wujiang Xu, Wentian Zhao, Zhenting Wang, Yu-Jhe Li,		819
	Can Jin, Mingyu Jin, Kai Mei, Kun Wang, and Dim-		820
	itris N. Metaxas. 2025. EPO: entropy-regularized		821
	policy optimization for LLM agents reinforcement		822
	learning . <i>CoRR</i> , abs/2509.22576.		823

824 Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng
825 Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng
826 Zhang, and Qi Tian. 2024b. [Qa-lora: Quantization-
827 aware low-rank adaptation of large language models.](#)
828 In *The Twelfth International Conference on Learning
829 Representations, ICLR 2024, Vienna, Austria, May
830 7-11, 2024*. OpenReview.net.

831 Devansh Yadav and Shouvik Mondal. 2025. [Evaluat-
832 ing pre-trained large language models on zero shot
833 prompts for parallelization of source code.](#) *J. Syst.
834 Softw.*, 230:112543.

835 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang,
836 Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao,
837 Chengen Huang, Chenxu Lv, Chujie Zheng, Day-
838 iheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao
839 Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41
840 others. 2025. [Qwen3 technical report.](#) *Preprint,*
841 arXiv:2505.09388.

842 Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan,
843 Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu,
844 Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole
845 Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang,
846 Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others.
847 2025. [DAPO: an open-source LLM reinforcement
848 learning system at scale.](#) *CoRR*, abs/2503.14476.

849 Renrui Zhang, Chengzhuo Tong, Zhizheng Zhao, Ziyu
850 Guo, Haoquan Zhang, Manyuan Zhang, Jiaming Liu,
851 Peng Gao, and Hongsheng Li. 2025. [Let’s verify
852 and reinforce image generation step by step.](#) In
853 *IEEE/CVF Conference on Computer Vision and Pat-
854 tern Recognition, CVPR 2025, Nashville, TN, USA,
855 June 11-15, 2025*, pages 28662–28672. Computer
856 Vision Foundation / IEEE.

857 A Appendices

858 A.1 The Details of Datasets

859 We evaluate the performance of R²PO across sev-
860 eral high-quality reasoning and coding benchmarks.
861 The statistics of these datasets are summarized in
862 Table 7.

863 **GSM8K (Cobbe et al., 2021)** The Grade School
864 Math 8K dataset consists of 8.5K high-quality
865 grade school math word problems. We use the
866 training set for reinforcement learning and the test
867 set for in-distribution evaluation. This benchmark
868 tests basic multi-step arithmetic reasoning.

869 **MATH-500 (Zhang et al., 2025)** A subset of the
870 original MATH dataset, containing 500 challenging
871 problems across several subjects like algebra and
872 geometry. We use this as an out-of-distribution
873 (OOD) test set to evaluate the generalization of
874 models trained on GSM8K.

875 **MBPP (Tao et al., 2024)** The Mostly Basic
876 Python Problems dataset contains around 1,000
877 entry-level Python programming problems. It is

878 used to fine-tune the model’s basic code generation
879 capabilities through verifiable rewards.

880 **HumanEval (Yadav and Mondal, 2025)** A stan-
881 dard benchmark consisting of 164 handwritten
882 Python programming problems. It evaluates the
883 model’s ability to solve functional programming
884 tasks based on docstrings.

885 **APPS (Hendrycks et al., 2021)** The Automated
886 Programming Progress Standard contains 10,000
887 coding challenges ranging from introductory to
888 competition level. It assesses sophisticated algo-
889 rithmic synthesis and is a key metric for reasoning-
890 intensive coding tasks.

Table 7: Statistics of the datasets used for R²PO training and evaluation.

Dataset	Domain	Train Size	Test Size
GSM8K	Math	7,473	1,319
MATH-500	Math	-	500
MBPP	Code	374	500
HumanEval	Code	-	164
APPS	Code	-	5,000

891 A.2 The Details of Baselines

892 We select two models from the Qwen series to
893 evaluate R²PO across different parameter scales and
894 capability levels:

895 **Qwen2.5-3B (Qwen et al., 2025)** A highly effi-
896 cient language model with 3.09 billion parameters.
897 It serves as our primary testbed for analyzing train-
898 ing dynamics and performing ablation studies due
899 to its balanced performance and computational ef-
900 ficiency.

901 **Qwen3-8B (Yang et al., 2025)** The latest
902 reasoning-enhanced backbone with 8.19 billion pa-
903 rameters. This model features a deeper understand-
904 ing of complex logical structures and improved
905 long-context reasoning capabilities. We use this
906 model to verify the scalability of R²PO and its ef-
907 fectiveness when applied to stronger base policies.

908 A.3 Training Hyperparameters

909 All experiments are conducted on a cluster of
910 NVIDIA A800-SXM4-80GB GPUs. For both the
911 Qwen2.5-3B and Qwen3-8B backbones, we em-
912 ploy the AdamW optimizer with a constant learn-
913 ing rate of 1×10^{-6} and a cosine decay schedule.
914 To ensure a fair comparison, GRPO and R²PO share
915 identical RL configurations: a group size (G) of
916 8, a KL-divergence coefficient (β) of 0.04, and a
917 PPO clip range (ϵ) of 0.2. The maximum sequence

918 length is set to 512 for 3B models and 768 for 8B
919 models.

920 A.4 Case Study: Redundant Reasoning 921 Patterns in GRPO

922 As shown in the example, the standard GRPO
923 model frequently produces redundant `<think>` tags
924 and repetitive logic cycles. This behavior indicates
925 that without the structural decoupling provided by
926 R²PO, the single-head policy are more susceptible
927 to undesirable reward noise.

Table 8: Case Study on Formatting and Verbosity. The baseline GRPO model produces an over-optimized response with repetitive `<think>` `</think>` tags, which often leads to parsing failures in strict evaluation environments. R²PO effectively suppresses this formatting noise through its decoupled architecture, yielding a significantly shorter response (one `<think>` `</think>` block) without compromising mathematical accuracy.

Case 1
Question: Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have together if Seattle has 20 sheep?
Ground Truth: 260
GRPO: <code><think></code> First, let’s find out how many sheep Charleston has. Since Toulouse has twice as many sheep as Charleston, and Charleston has 4 times as many sheep as Seattle, we can start by finding out how many sheep Charleston has: $\text{Charleston sheep} = 20 \text{ (Seattle’s sheep)} * 4 = 80 \text{ sheep}$ <code></think></code> <code><think></code> Now, to find the number of sheep in Toulouse, we multiply Charleston’s number of sheep by 2: $\text{Toulouse sheep} = 80 \text{ (Charleston’s sheep)} * 2 = 160 \text{ sheep}$. <code></think></code> <code><think></code> Now, let’s sum up the total number of sheep in Toulouse, Charleston, and Seattle: $\text{Total sheep} = \text{Toulouse sheep} + \text{Charleston sheep} + \text{Seattle sheep} = 160 + 80 + 20 = 260 \text{ sheep}$. <code></think></code> <code><answer></code> The total number of sheep in Toulouse, Charleston, and Seattle is 260. <code></answer></code>
R ² PO: <code><think></code> Let’s start by figuring out how many sheep each city has based on the information provided. If Seattle has 20 sheep, and Charleston has 4 times as many sheep as Seattle, then Charleston has $4 * 20 = 80$ sheep. Toulouse has twice as many sheep as Charleston, so Toulouse has $2 * 80 = 160$ sheep. To find the total number of sheep in Toulouse, Charleston, and Seattle together, we add the number of sheep in each city: $20 + 80 + 160 = 260 \text{ sheep}$ <code></think></code> <code><answer></code> 260 sheep <code></answer></code>
Case 2
Question: Kylar went to the store to buy glasses for his new apartment. One glass costs \$5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?
Ground Truth: 64
GRPO: <code><think></code> First, let’s calculate the cost of one glass, which is \$5. <code></think></code> <code><think></code> For every second glass, Kylar will pay 60% of \$5, which is $0.6 * \$5 = \3 . <code></think></code> <code><think></code> Kyle wants to buy 16 glasses. In this case, he will buy 8 glasses for the first price and 8 glasses for the second price (60% of the price). <code></think></code> <code><think></code> So, the total cost for the first 8 glasses is $8 * \$5 = \40 . <code></think></code> <code><think></code> The total cost for the second 8 glasses is $8 * \$3 = \24 . <code></think></code> <code><think></code> Therefore, the total cost for Kylar’s glasses is $\$40 + \$24 = \$64$. <code></think></code> <code><answer></code> \$64 <code></answer></code>
R ² PO: <code><think></code> First, Kylar needs to buy 16 glasses. The price for the first glass is \$5. For every second glass, he pays 60% of \$5, which is $0.60 * \$5 = \3 . So, for every pair of glasses (one original and one discounted), he pays $\$5 + \$3 = \$8$. Since he wants to buy 16 glasses, half of them (8 pairs) are original, and the other half (8 pairs) are discounted. Therefore, the total cost is $8 * \$5 + 8 * \$3 = \$40 + \$24 = \$64$. <code></think></code> <code><answer></code> \$64 <code></answer></code>