R^{2}PO: Decoupling Rollout and Inference Policies for LLM Reasoning

Anonymous ACL submission

Abstract

Existing reinforcement learning methods for LLM reasoning implicitly assume that the policy generating training trajectories should coincide with the one producing inference responses. We argue that this is a misleading inductive bias: the *optimization-optimal* trajectory distribution favors informative gradients, whereas the *inference-optimal* response distribution emphasizes accuracy and consistency. Forcing both into a single policy entangles their gradients and suppresses exploration. We propose **R^{2}PO** (**R**esidual **R**ollout **P**olicy **O**ptimization), which attaches a lightweight Residual Rollout-Head atop the policy to decouple training trajectories from inference responses, diversifying rollouts during training while keeping inference generation intact. Experiments show that R^{2}PO consistently outperforms baselines, with average accuracy gains of **3.4%** on MATH-500 and **1.3%** on APPS, alongside more diverse rollouts and reduced length bias. Our code is available at this link.

1 Introduction

In recent years, Reinforcement Learning (RL) for Large Language Models (LLMs) has become a central paradigm for improving their reasoning capabilities. It underpins 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 has achieved remarkable success in complex reasoning tasks such as mathematical problem solving ang code generation (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 exhibit stronger performance on complex reasoning tasks (Stiennon et al., 2020; Shinn et al., 2023).

A common but rarely questioned assumption underlies these methods (e.g., PPO (Schulman et al., 2017), GRPO (Shao et al., 2024)): the policy used to generate *training rollouts* is identical to the policy used to produce *inference responses*. We argue that this is a misleading inductive bias for reasoning RL. As illustrated in Figure 1 (left), the *inference-optimal* distribution emphasizes accuracy and consistency and thus tolerates homogeneous outputs, whereas the *optimization-optimal* distribution requires diversity and broad solution-space coverage, including informative erroneous paths that provide effective gradient signals. Tying these two roles to a single set of parameters forces the optimization process to be dominated by response-level correctness, which suppresses effective exploration for diversity. As a result, the model frequently collapses into self-repetition during training (Moalla et al., 2024; Cui et al., 2025).

To overcome this, we propose **R^{2}PO** (Residual Rollout Policy Optimization). Unlike paradigms that compress the two conflicting objectives into a single policy, we introduce a lightweight residual module 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 decoupling keeps the primary policy stable at inference time, while ensuring diverse trajectory exploration during training via the residual Rollout-Head. The Rollout-Head can be discarded at inference, ensuring efficient and consistent deployment.

We investigate the effectiveness of R^{2}PO across mathematical reasoning and code generation tasks on Qwen2.5-3B and Qwen3-8B. Experimental results underscore that R^{2}PO outperforms the standard GRPO baseline, achieving an average accuracy increment of **3.4%** on MATH-500 and **1.3%** on APPS. We further reveal that our framework

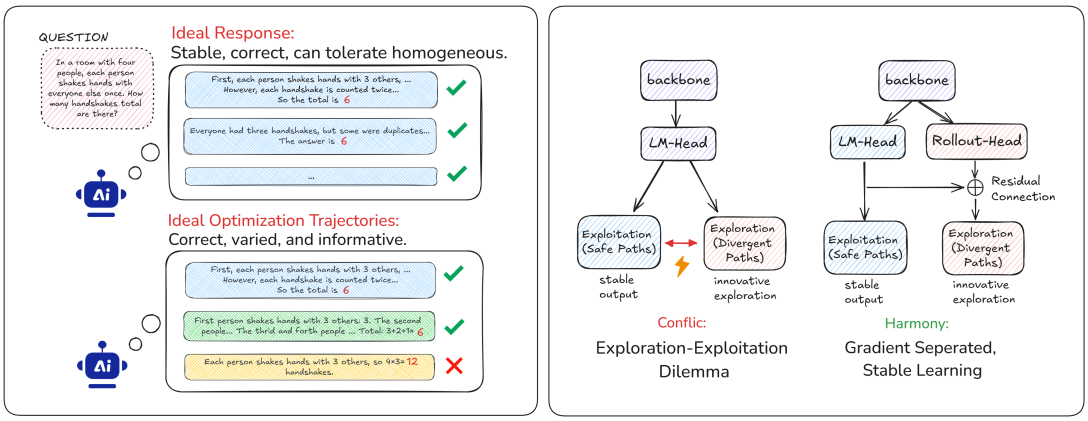

Figure 1: **Decoupling Ideal Responses and Optimization Trajectories.** Left: The mismatch between ideal inference responses and ideal training trajectories. Right: Single-head policy optimization versus R^2PO with a decoupled Rollout-Head for stable exploration.

produces more diverse training rollouts and mitigates length bias, maintaining a stable reasoning trajectory even in the late stages of optimization.

The contributions of this work are as follows:

- **A new perspective on reasoning RL.** We identify and challenge a misleading inductive bias in existing methods: the rollout policy need not coincide with the inference policy. We articulate why the *optimization-optimal* and *inference-optimal* trajectory distributions are intrinsically different.
- **A lightweight framework, R^2PO.** We realize this principle with a lightweight residual Rollout-Head atop a frozen backbone, jointly trained but discarded at inference, fully compatible with standard GRPO.
- **Consistent empirical gains.** Across mathematical reasoning and code generation benchmarks, R^2PO improves accuracy, increases rollout diversity, controls length bias, and stabilizes training, with negligible overhead.

2 Related Works

We review related work from tow perspectives: (1) RL for reasoning-oriented LLMs, and (2) exploration mechanisms in LLM RL.

RL for Reasoning LLMs LLMs have demonstrated significant reasoning improvements through Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022). While traditional PPO-based frameworks (Schulman et al., 2017) rely on separate value models, recent paradigms like Reinforcement Learning with Verifiable Rewards (RLVR) (Lambert et al., 2025; DeepSeek-AI et al., 2025) have gained prominence in domains like mathematics and coding. However, these verifiable reward landscapes are often sparse and unforgiving, posing significant exploration and optimization challenges for reasoning RL.

Exploration Mechanism in LLM RL Balancing exploration and exploitation is a fundamental challenge in reinforcement learning (Sutton and Barto, 2018). Traditional approaches such as Soft Actor-Critic (SAC) (Haarnoja et al., 2018) employ entropy regularization to sustain exploration and prevent premature convergence, and in LLM fine-tuning most paradigms (Stiennon et al., 2020; DeepSeek-AI et al., 2025) rely on KL-divergence penalties to implicitly preserve diversity. However, such soft regularization is often insufficient, as policy entropy can still collapse rapidly, leading to suboptimal solutions (Moalla et al., 2024; Cui et al., 2025). Several recent studies further investigate the role of entropy in LLM RL: (Park et al., 2025) analyzes the interaction between PPO clipping and entropy dynamics; (Cheng et al., 2025) examines how entropy-characterized exploration affects the depth and diversity of reasoning trajectories; and other works incorporate entropy-related signals into the objective itself, such as entropy-regularized policy optimization (Xu et al., 2025) and entropy-aware token-level updates (Wen et al., 2024). All of these methods, however, modify the objective while still using a single policy for both rollout and inference, which indeed mirrors the gap between the ideal optimization trajectory and the ideal inference response that motivates our work.

3 Method

Motivated by the gap between the ideal inference response and the ideal training trajectory, we propose R^{2}PO, a framework that decouples these two roles instead of forcing them into a single shared policy. We first analyze why a single policy cannot reconcile both objectives, and then introduce the overall framework, policy implementation, reward design, and iterative training procedure of R^{2}PO.

3.1 Motivation

In standard RL fine-tuning for LLM reasoning, a single policy π_θ actually plays two roles at the same time. During training, π_θ generates rollouts whose rewards drive parameter updates; and during inference, the same π_θ produces responses that are scored as the system's output.

These two roles (illustrated in Figure 1), however, are governed by fundamentally different objectives. The inference role aims to maximize the expected task reward of the deployed response,

$$\mathcal{O}_{\text{infer}}(\pi) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi(\cdot|x)}\big[\, r(x,y)\,\big], \tag{1}$$

which favors low-entropy, mode-seeking behavior. The training role aims to maximize the magnitude of the gradient signal produced by rollouts, defined as $\mathcal{O}_{\text{train}}$, which under group-relative advantages requires rollout distributions that spread across trajectories with different rewards so that the per-group advantage

$$\hat{A}_i = \frac{r(x,y_i) - \text{mean}\big(\{r(x,y_j)\}_{j=1}^{G}\big)}{\text{std}\big(\{r(x,y_j)\}_{j=1}^{G}\big)} \tag{2}$$

does not collapse to zero. Consequently, training often benefits from rollout distributions that remain diverse, exploratory, and tolerant of occasionally incorrect trajectories.

When the same parameters carry both objectives, gradients induced by response-level correctness progressively suppress deviations that could otherwise reveal alternative reasoning paths. As a result, the policy tends to reinforce a narrow subset of safe trajectories, limiting the discovery of rare but informative behaviors.

These observations suggest that the two roles should not be carried by a fully shared parameter set. Rather than insisting on one policy that simultaneously satisfies both objectives, we allocate a small set of parameters to absorb $\mathcal{O}_{\text{train}}$, while keeping the deployed response policy dedicated to inference. This principle of partial parameter decoupling is what R^{2}PO instantiates.

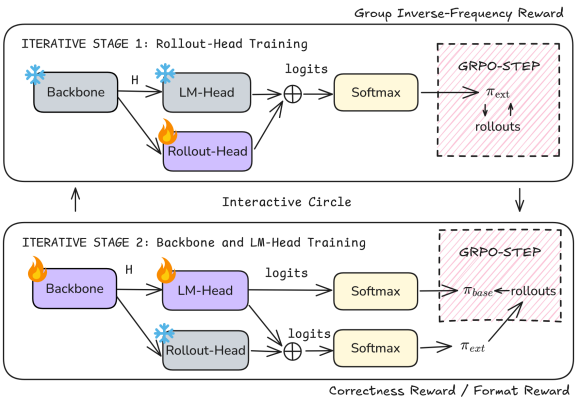

Figure 2: Overview of **R^{2}PO**. An iterative loop of two alternating stages: Stage 1 (top) optimizes the Rollout-Head using GIF reward with a frozen backbone; Stage 2 (bottom) updates the base policy on task rewards using the Rollout-Head as a fixed behavioral sampler.

3.2 Overall Framework

To decouple the two optimization objectives identified above, we propose R^{2}PO, whose overall framework is clearly illustrated in Figure 2. R^{2}PO attaches a lightweight residual head, called the **Rollout-Head**, that is optimized only against the training objective $\mathcal{O}_{\text{train}}$, while original backbone and language modeling head (LM head) remain responsible for the inference objective $\mathcal{O}_{\text{infer}}$. This yields two policies sharing the same backbone: main policy π_θ, produced by the original LM Head, which serves as the deployed inference policy; and rollout policy π_ϕ, produced by combining the LM Head with the Rollout-Head, which serves as the behavioral sampler during training. To coordinate the two policies, R^{2}PO adopts an iterative training scheme that alternates between optimizing the Rollout-Head for exploration and updating the main policy from the resulting rollouts.

3.3 Dual-Policy Implementation

We implement the Rollout-Head as a residual branch attached on top of the transformer backbone. Concretely, it is a two-layer Multi-Layer Perceptron (MLP) that takes the backbone hidden states $H \in \mathbb{R}^d$ as input and produces a logit-level offset over the vocabulary space $\mathbb{R}^V$, where d is the hidden dimension and V is the vocabulary size. Let f_{LM} denote the original LM Head and f_{RO} the Rollout-Head. The two policies share the backbone and differ only in their output heads:

$$\begin{aligned} \pi_\theta &= \text{Softmax}\big(f_{\text{LM}}(H)\big), \\ \pi_\phi &= \text{Softmax}\big(f_{\text{RO}}(H) + f_{\text{LM}}(H)\big). \end{aligned} \tag{3}$$

This concrete formulation operationalizes the two-stage scheme described in Section 3.2: during Stage 1, we optimize the residual parameters ϕ of f_{RO} while anchoring it around the frozen base logits; during Stage 2, f_{RO} is fixed to output stable exploratory perturbations that guide the optimization of the backbone and θ.

In our framework, the main policy π_θ is responsible for stable exploitation and is the only policy used at inference time, while the rollout policy π_ϕ incorporates the residual perturbation introduced by the Rollout-Head and is used exclusively during training. To stabilize the cold-start phase, we zero-initialize the Rollout-Head, so that π_ϕ coincides with π_θ at the beginning of training and gradually deviates from it as exploration is learned. Because the perturbation is restricted to a separate residual branch, the Rollout-Head can be discarded after training, leaving the deployed model architecturally identical to a standard LLM.

3.4 Reward Design

The two policies are trained with different rewards, reflecting their different objectives.

Group Inverse-Frequency Reward. Because the Rollout-Head is responsible for the training-time objective $\mathcal{O}_{\text{train}}$, its reward should encourage rollouts that yield non-trivial group-relative advantages: a group whose rollouts collapse into a single reward value provides no learning signal under group-relative advantage estimation. We therefore upweight trajectories whose reward outcome is rare within the group, reflecting the intuition of intrinsic-motivation and novelty-search literature (Bellemare et al., 2016; Lehman and Stanley, 2011). Let $\mathcal{G} = \{y_1, \ldots, y_G\}$ be a group of rollouts sharing the same query with verifiable rewards r_i, and let $\mathcal{B}_k = \{y_i \in \mathcal{G} \mid r_i = v_k\}$ denote the bin of rollouts sharing the unique reward value v_k. The Group Inverse-Frequency (GIF) score for $y_i \in \mathcal{B}_k$ is the inverse of the bin's cardinality:

$$s_i = \frac{1}{|\mathcal{B}_k|} = \frac{1}{\sum_{j=1}^{G} \mathbf{1}[r_j = r_i]}. \tag{4}$$

To prevent gradient explosion from groups with extreme bin imbalance, we apply Z-score standardization within each group to obtain the final exploration reward R_{GIF}.

Correctness and Formatting Reward. Because the main policy π_θ is responsible for the inference-time objective $\mathcal{O}_{\text{infer}}$, its reward must reflect the actual quality of the deployed response on downstream task. Following recent advances in reasoning model training (Le et al., 2022; Shinn et al., 2023; Gehring et al., 2025), we adopt a rule-based reward that provides dense and verifiable feedback without a learned reward model. For a query x and generated response y_i, the total task reward is

$$r(x, y_i) = \mathbf{1}_{\text{correct}}(y_i) \cdot R_{\text{acc}} + \mathbf{1}_{\text{format}}(y_i) \cdot R_{\text{fmt}}, \tag{5}$$

where $\mathbf{1}_{\text{correct}}$ and $\mathbf{1}_{\text{format}}$ are indicator functions checking numerical correctness and adherence to the formatting template (e.g., enclosing the final answer between `<answer>` and `</answer>`). This deterministic reward keeps the optimization signal of the main policy consistent during training.

3.5 Iterative Training

R^{2}PO 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:

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x\sim\mathcal{D},\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}} \Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \\ \min\Big(r_{i,t}(\theta)\hat{A}_{i,t}, \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{i,t}\Big) \\ - \beta \mathbb{D}_{KL}(\pi_\theta \| \pi_{ref}) \Bigg],\end{aligned} \tag{6}$$

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:

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

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

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.

3.5.1 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:

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

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 4.4. The Rollout-Head is reward-agnostic; GIF is one effective instantiation, but any task-appropriate exploration signal can be substituted.

3.5.2 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:

$$\max \mathcal{J}_{\text{GRPO}}(\theta), \quad y \sim \pi_\phi \tag{8}$$

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.

3.6 Discussion

The premise of R^{2}PO is that the rollout policy and the inference policy should be treated as two distinct objects with different design objectives: the inference policy is optimized for accurate, consistent, deployed responses, whereas the rollout policy is optimized for the diverse, exploratory, occasionally erroneous trajectories that yield informative gradients. Insisting on a single parameter set for both is the default in RL for LLM reasoning, but we view it as a default rather than a justified choice.

Approaches addressing the exploration-exploitation dilemma in Section 2, including modified objectives, clipping strategies, and auxiliary sampling, still operate within a single policy whose parameters jointly govern training rollouts and inference responses, so the gradients that encourage exploration and those driving reward maximization remain entangled. R^{2}PO instead confines the exploration signal to a separate Rollout-Head, so that gradients shaping the training distribution do not directly perturb the parameters producing deployed responses. We further discuss the relationship between R^{2}PO and parameter-efficient fine-tuning (PEFT) in Appendix A.4.

4 Experiments

In this section, we conduct extensive experiments to evaluate the effectiveness of R^{2}PO in enhancing reasoning capabilities and training stability. We begin by comparing R^{2}PO against competitive baselines including GRPO and recent entropy-based exploration methods across two representative domains: mathematical reasoning and code generation. Subsequently, we provide a series of diagnostic analyses to examine the mechanistic drivers of R^{2}PO. 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 R^{2}PO 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: both 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 (Lightman et al., 2024) as an out-of-distribution benchmark to assess generalization in more sophisticated logical deduction. For programming tasks, we employ MBPP (Austin et al., 2021) for training, while evaluating functional synthesis capabilities on HumanEval (Chen et al., 2021) and the multi-level algorithmic challenges posed by APPS (Hendrycks et al., 2021).

Baselines Beyond GRPO, we include two entropy-based exploration methods closely related to our work. **EPO** (Xu et al., 2025) augments GRPO with a policy-level entropy regularization term to prevent premature convergence. **ETPO** (Wen et al., 2024) instead applies entropy regularization at the token level, incorporating per-token entropy into advantage estimation for finer-grained exploration. Both modify the optimiza-

Table 1: Benchmark results comparison. Our proposed **R^{2}PO** framework achieves consistent improvements over GRPO and entropy-based exploration baselines 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	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
	+ EPO (Xu et al., 2025)	81.27	41.20	55.20	66.46	9.04	50.63
	+ ETPO (Wen et al., 2024)	81.80	42.40	54.80	67.68	8.62	51.06
	+ R^{2}PO (Ours)	**83.17**	**45.60**	**59.00**	**68.90**	**9.52**	**53.24**
Qwen3-8B	Base	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
	+ EPO (Xu et al., 2025)	**89.92**	56.80	63.80	85.98	15.50	62.40
	+ ETPO (Wen et al., 2024)	89.46	55.40	66.60	87.20	15.14	62.76
	+ R^{2}PO (Ours)	88.48	**57.20**	**67.20**	**87.80**	**15.56**	**63.25**

tion objective and thus serve as direct comparisons against our structural decoupling approach.

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.

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 training templates. [1]

4.2 Main Results

The comparative results presented in Table 1 show that our proposed **R^{2}PO** 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^{2}PO 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^{2}PO achieves the highest Pass@1 scores across both backbones. This suggests that the decoupled Rollout-Head successfully encourages diverse, advanced reasoning paths critical for complex multi-step logical deduction. R^{2}PO also outperforms EPO and ETPO on both backbones, demonstrating the superiority of architectural decoupling over objective-level entropy regularization.

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^{2}PO 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 destabilize the learned patterns in the primary head. In contrast, the Rollout-Head acts as a structural buffering mechanism. 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.

[1] The rule-based metric provided by OpenCompass for MATH-500 is not fully compatible with the output format generated during our training; we therefore evaluate this dataset with a tailored metric.

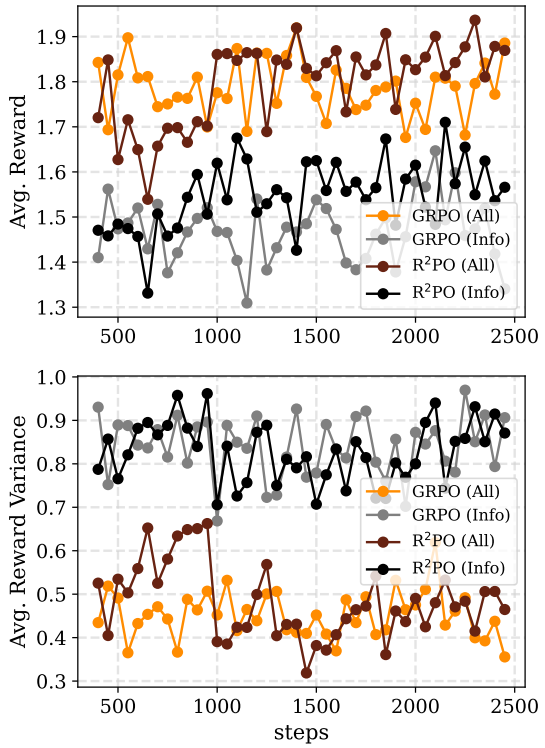

Figure 3: The mean and variance of rewards during the training process on GSM8K. The upper row shows the mean reward, while the lower row displays the variance of the mean reward. "Info" refers to Informative Rollouts, which contain varying rewards and thus supply non-zero advantage signals for policy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

4.3 Stability and Diversity Analysis

In this section, we analyze the stability and diversity of R^{2}PO from three perspectives: (1) reward dynamics and stability, (2) rollout diversity, (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, while keeping the reward variance essentially comparable. This suggests that the decoupled architecture facilitates a more thorough search of the solution space before converging to a stable, high-performing policy.

4.3.2 Rollout Diversity

To examine whether decoupled training actually shapes a more diverse trajectory distribution, we cluster rollouts on GSM8K in batches of 64 with HDBSCAN (McInnes et al., 2017) over sentence embeddings, and average per-batch statistics over the full set of rollouts using BAAI/bge-large-en-v1.5 (Xiao et al., 2024). We report three quantities: **clusters**, the number of valid HDBSCAN clusters per batch (excluding the noise label); **noise**, the number of samples per batch that fit no dense mode; and **entropy**, the Shannon entropy $H = -\sum_c \frac{|c|}{N} \log \frac{|c|}{N}$ of the cluster-size distribution (with N the total batch size including noise). We compare the Qwen2.5-3B base model, GRPO at step 2500, and R^{2}PO at step 2500, drawing 64 rollouts per prompt over the entire test set.

Table 2: Rollout diversity on GSM8K (Qwen2.5-3B). R^{2}PO produces more semantic clusters, fewer unstructured noise points, and higher cluster entropy than GRPO at the same training step.

Model	clusters	noise	entropy
BASE	2.459	34.659	0.5429
GRPO	2.522	26.346	0.5782
R^{2}PO	**2.707**	**24.942**	**0.6136**

R^{2}PO simultaneously increases the number of semantic modes, reduces unstructured noise, and raises cluster entropy, while GRPO improves over the base only marginally. This is direct evidence for our thesis: routing exploration through a separate Rollout-Head expands the *optimization-optimal* trajectory distribution into more, better-populated modes rather than collapsing onto a narrow set of dominant responses.

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 3: 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

Beyond the three analyses above, we further verify in Appendix A.5 that early-stage Pass@8 is not degraded by the additional exploration mechanism, and in Appendix A.6 that R^{2}PO is immune to a reward-misspecification perturbation that GRPO succumbs to (0% vs 0.875% format-error rate).

4.4 Ablation Study

To identify what truly drives the gains of **R^{2}PO**, we isolate its structural and reward-level components via two ablations: (i) the role of the GIF 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 evaluates whether the improvements come mainly from the *architectural decoupling* itself rather than the *Group Inverse-Frequency Reward* signal. As shown in Table 4, 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 R^{2}PO 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 4, the optimization without the Rollout-Head outperformed vanilla GRPO but lagged behind the full R^{2}PO, confirming that the additional parameter capacity of the residual branch is essential for effective decoupling.

Table 4: 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**

4.5 Efficiency and Parameter Discussion

To evaluate the computational efficiency of R^{2}PO, we analyze the parameter overhead and its impact on both training and inference phases. As shown in Table 5, R^{2}PO introduces a minor training parameter overhead (7.8%–10.2%), which is detached during inference to ensure zero additional latency.

Table 5: 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, R^{2}PO 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

We argue that tying the rollout policy to the inference policy is a misleading inductive bias in RL for LLM reasoning: the *optimization-optimal* and *inference-optimal* distributions serve different roles and should not share one parameter set. R^{2}PO realizes this view minimally via a discardable residual Rollout-Head atop a frozen inference policy. Across mathematical reasoning and code generation, this decoupling increases rollout diversity, controls length bias, and improves accuracy and stability over GRPO, suggesting future methods treat the two policies as distinct design objects.

Limitation

R^{2}PO is evaluated only on models of 3B and 8B parameters. Its behavior on substantially larger models (e.g., 70B+) or smaller-scale models remains to be explored. For large-scale models, it is unclear whether the Rollout-Head can continue to yield meaningful gains with a relatively small number of additional parameters. Conversely, for smaller models, the parameter increase introduced by the Rollout-Head, although minimized, may still constitute a non-negligible overhead given their limited capacity and vocabulary size.

Moreover, our experiments focus exclusively on single-turn reasoning. Extending this decoupled exploration framework to multi-turn dialogue settings or long-horizon agent tasks may introduce additional challenges, which we leave for future investigation.

Finally, the GIF reward used in our experiments is tailored to discrete reward settings, and its extension to dense or continuous reward landscapes remains open. We note that the decoupled architecture itself is reward-agnostic: our ablations show that even when the Rollout-Head is trained with the main task reward alone, R^{2}PO retains its advantage over GRPO. Designing exploration rewards better matched to continuous-reward regimes is an interesting direction for future work.

Ethics Statement

While R^{2}PO enhances the reasoning and problem-solving capabilities of LLMs, we recognize potential risks associated with its deployment. Improved reasoning could theoretically be misused for generating sophisticated malicious code or automated misinformation. To mitigate such risks, our experiments rely solely on publicly available datasets, and the training follows standard RLHF safety alignment practices. Moreover, the structural decoupling introduced in R^{2}PO improves optimization transparency and model interpretability, which can help in auditing model behavior and reducing unintended outputs. We emphasize that the deployment of R^{2}PO should consider these risks and be accompanied by appropriate monitoring and access control.

Use of AI Assistants

AI assistants (specifically ChatGPT and Claude) were used in this research primarily for language polishing, improving grammatical structures, and refining the clarity of the manuscript. All original methodology, experimental design, and data analysis were conducted solely by the human authors.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. 2016. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1471–1479.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *CoRR*, abs/2506.14758.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. 2025. The entropy mechanism of reinforcement learning for reasoning language models. *CoRR*, abs/2505.22617.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang,

Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 81 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. 2025. RLEF: grounding code llms in execution feedback with reinforcement learning. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021. Measuring coding challenge competence with APPS. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, Proceedings of Machine Learning Research, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024. Tülu 3: Pushing frontiers in open language model post-training. *CoRR*, abs/2411.15124.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training. *Preprint*, arXiv:2411.15124.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven Chu-Hong Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Joel Lehman and Kenneth O. Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.*, 19(2):189–223.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations (ICLR)*.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205.

Skander Moalla, Andrea Miele, Daniil Pyatko, Razvan Pascanu, and Caglar Gulcehre. 2024. No representation, no trust: Connecting representation, collapse, and trust issues in PPO. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Jaesung R. Park, Junsu Kim, Gyeongman Kim, Jinyoung Jo, Sean Choi, Jaewoong Cho, and Ernest K. Ryu. 2025. Clip-low increases entropy and clip-high decreases entropy in reinforcement learning of large language models. *CoRR*, abs/2509.26114.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, 2 edition. MIT Press, Cambridge, MA.

Ziyu Wan, Yunxiang Li, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. 2025. Rema: Learning to meta-think for llms with multi-agent reinforcement learning. *CoRR*, abs/2503.09501.

Muning Wen, Junwei Liao, Cheng Deng, Jun Wang, Weinan Zhang, and Ying Wen. 2024. Entropy-regularized token-level policy optimization for language agent reinforcement. *Preprint*, arXiv:2402.06700.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 641–649.

Wujiang Xu, Wentian Zhao, Zhenting Wang, Yu-Jhe Li, Can Jin, Mingyu Jin, Kai Mei, Kun Wang, and Dimitris N. Metaxas. 2025. EPO: entropy-regularized policy optimization for LLM agents reinforcement learning. *CoRR*, abs/2509.22576.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, and 16 others. 2025. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476.

A Appendices

A.1 The Details of Datasets

We evaluate the performance of R^2PO across several high-quality reasoning and coding benchmarks. The statistics of these datasets are summarized in Table 6.

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

MATH-500 (Lightman et al., 2024) A subset of the original MATH dataset, containing 500 challenging problems across several subjects like algebra and geometry. We use this as an out-of-distribution (OOD) test set to evaluate the generalization of models trained on GSM8K.

MBPP (Austin et al., 2021) The Mostly Basic Python Problems dataset contains around 1,000 entry-level Python programming problems. It is used to fine-tune the model's basic code generation capabilities through verifiable rewards.

HumanEval (Chen et al., 2021) A standard benchmark consisting of 164 handwritten Python programming problems. It evaluates the model's ability to solve functional programming tasks based on docstrings.

APPS (Hendrycks et al., 2021) The Automated Programming Progress Standard contains 10,000 coding challenges ranging from introductory to competition level. It assesses sophisticated algorithmic synthesis and is a key metric for reasoning-intensive coding tasks.

Table 6: Statistics of the datasets used for R^2PO 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

A.2 The Details of Baselines

We select two models from the Qwen series to evaluate R^2PO across different parameter scales and capability levels:

Qwen2.5-3B (Qwen et al., 2025) A highly efficient language model with 3.09 billion parameters.

It serves as our primary testbed for analyzing training dynamics and performing ablation studies due to its balanced performance and computational efficiency.

Qwen3-8B (Yang et al., 2025) The latest reasoning-enhanced backbone with 8.19 billion parameters. This model features a deeper understanding of complex logical structures and improved long-context reasoning capabilities. We use this model to verify the scalability of R^{2}PO and its effectiveness when applied to stronger base policies.

A.3 Training Hyperparameters

All experiments are conducted on a cluster of NVIDIA A800-SXM4-80GB GPUs. For both the Qwen2.5-3B and Qwen3-8B backbones, we employ the AdamW optimizer with a constant learning rate of 1×10^{-6} and a cosine decay schedule. To ensure a fair comparison, GRPO and R^{2}PO share identical RL configurations: a group size (G) of 8, a KL-divergence coefficient (β) of 0.04, and a PPO clip range (ϵ) of 0.2. The maximum sequence length is set to 512 for 3B models and 768 for 8B models. During the first 2000 training steps, we use 150 steps for Stage 1 and 200 steps for Stage 2 per cycle; afterwards, the cycle is lengthened to 200 and 300 steps respectively. This schedule is fixed across all tasks without task-specific tuning.

A.4 Comparison with Parameter-Efficient Fine-Tuning

The Rollout-Head in R^{2}PO is architecturally close to lightweight modules used in Parameter-Efficient Fine-Tuning (PEFT) approaches such as Adapters (Houlsby et al., 2019) and LoRA (Hu et al., 2022), in that all three attach a small trainable branch to a frozen or partially frozen backbone. Its functional role, however, is fundamentally different along three axes.

Optimization objective. PEFT methods aim to reduce the number of trainable parameters while preserving the original optimization objective and trajectory distribution. The adaptation modules learn to approximate a full fine-tune of the backbone, and the trajectories used to update them are sampled from the same policy that the adapted model will deploy. In contrast, the Rollout-Head in R^{2}PO is introduced precisely to reshape the sampling distribution used for policy optimization, so that the effective optimization objective is altered even though the loss function remains the standard group-relative policy optimization objective. Whereas PEFT moves the parameter footprint, R^{2}PO moves the trajectory distribution.

Role at training versus inference. In PEFT, the adaptation module is trained jointly with (or on top of) the backbone and remains active at inference; it is part of the deployed model. In R^{2}PO, the Rollout-Head is used only during training as the behavior policy that generates exploratory rollouts, and is discarded at inference, so the deployed model is architecturally identical to a standard LLM. The Rollout-Head therefore changes the optimization geometry but leaves no inference-time footprint, which is the opposite of what PEFT is designed for.

Exploration versus exploitation specialization. The Rollout-Head and the backbone in R^{2}PO are deliberately specialized: the Rollout-Head, trained with a rarity-driven exploration signal, expands the trajectory distribution toward informative-gradient regions, while the backbone, trained on these trajectories with the task reward, consolidates stable high-reward reasoning patterns. PEFT modules are not designed to support such a role separation; they share gradients with the backbone under a single objective. The two paradigms are therefore orthogonal rather than competing: R^{2}PO can in principle be combined with PEFT (for example, by realizing the Rollout-Head itself as an adapter), but they address different axes, namely parameter efficiency versus trajectory-distribution control.

A.5 Early-Stage Pass@k Comparison

Replay-based exploration methods are known to occasionally degrade early-stage Pass@k performance, because reused off-policy trajectories may distort the current policy's sampling distribution. Since R^{2}PO generates trajectories with an on-policy behavioral sampler (the Rollout-Head) rather than reusing past data, it does not suffer from this issue. To verify this empirically, we report Pass@8 at two early training checkpoints on GSM8K.

Table 7: Early-stage Pass@8 on GSM8K (Qwen2.5-3B). R^{2}PO maintains comparable or slightly higher Pass@8 throughout early training.

Method	200 steps	400 steps
GRPO	0.9204	0.9287
R^{2}PO	0.9227	0.9295

The results confirm that the additional explo-

ration mechanism does not harm early-stage sampling diversity, which is consistent with the on-policy nature of R^2PO.

A.6 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.7, 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 8, 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 8: 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 9: Results of the Perturbation Experiment. Measured on the first 100 samples of the GSM8K test set.

Steps	**GRPO**	**R^2PO**
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.

A.7 Case Study: Redundant Reasoning Patterns in GRPO

As shown in the example, the standard GRPO model frequently produces redundant <think> tags and repetitive logic cycles. This behavior indicates that without the structural decoupling provided by R^2PO, the single-head policy are more susceptible to undesirable reward noise.

Table 10: 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. R^2PO 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: <think> 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: Charleston sheep = 20 (Seattle's sheep) * 4 = 80 sheep </think> <think> Now, to find the number of sheep in Toulouse, we multiply Charleston's number of sheep by 2: Toulouse sheep = 80 (Charleston's sheep) * 2 = 160 sheep. </think> <think> Now, let's sum up the total number of sheep in Toulouse, Charleston, and Seattle: Total sheep = Toulouse sheep + Charleston sheep + Seattle sheep = 160 + 80 + 20 = 260 sheep. </think> <answer> **The total number of sheep in Toulouse, Charleston, and Seattle is 260.** </answer>
R^2PO: <think> 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 sheep </think> <answer> **260 sheep** </answer>

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: <think> First, let's calculate the cost of one glass, which is $5. </think> <think> For every second glass, Kylar will pay 60% of $5, which is 0.6 * $5 = $3. </think> <think> 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). </think> <think> So, the total cost for the first 8 glasses is 8 * $5 = $40. </think> <think> The total cost for the second 8 glasses is 8 * $3 = $24. </think> <think> Therefore, the total cost for Kylar's glasses is $40 + $24 = $64. </think> <answer> **$64** </answer>
R^2PO: <think> 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. </think> <answer> **$64** </answer>