RPO:Reinforcement Fine-Tuning with Partial Reasoning Optimization

Anonymous ACL submission

Abstract

Within the domain of large language models, reinforcement fine-tuning algorithms necessitate the generation of a complete reasoning trajectory beginning from the input query, which incurs significant computational overhead during the rollout phase of training. To address this issue, we analyze the impact of different segments of the reasoning path on the correctness of the final result and, based on these insights, propose Reinforcement Fine-Tuning with Partial Reasoning Optimization (**RPO**), a plug-and-play reinforcement fine-tuning algorithm. Unlike traditional reinforcement fine-tuning algorithms that generate full reasoning paths, RPO trains the model by generating suffixes of the reasoning path using experience cache. During the rollout phase of training, RPO reduces token generation in this phase by approximately 95%, greatly lowering the theoretical time overhead. Compared with full-path reinforcement fine-tuning algorithms, RPO reduces the training time of the 1.5B model by 90% and the 7B model by 72%. At the same time, it can be integrated with typical algorithms such as GRPO and DAPO, enabling them to achieve training acceleration while maintaining performance comparable to the original algorithms.

1 Introduction

In recent years, large language models (LLMs) (OpenAI et al., 2024b; Touvron et al., 2023; Zeng et al., 2023) have achieved remarkable breakthroughs in reasoning and generalization capabilities (Wang et al., 2025b), particularly after the introduction of reinforcement learning during the post-training stage (Ouyang et al., 2022). Pioneering works such as OpenAI's O1 (OpenAI et al., 2024a) and DeepSeek-R1 (DeepSeek-AI et al., 2025) have demonstrated impressive reasoning-time efficiency, primarily due to the synergistic combination of reinforcement learning and chain-of-thought (CoT) reasoning (Wei et al., 2023). This paradigm shift highlights the transformative potential of Reinforcement Learning-based post-training in pushing the boundaries of LLM performance.

Despite its promising prospects, applying reinforcement learning in post-training remains immature. In terms of time overhead, Reinforcement Learning fine-tuning typically generates a large number of samples during the sampling stage. However, parameter updates cannot proceed until all samples are completed, resulting in severe underutilization of computational resources. Although some asynchronous Reinforcement Learning approaches exist, the off-policy nature of the optimization process requires more training iterations. Furthermore, during Reinforcement Learning fine-tuning of language models, rewards are usually computed only after generating the final token based on task-specific criteria. This paradigm, known as Reinforcement Learning with Verifiable Rewards (Lambert et al., 2025; Wang et al., 2025a), lacks intermediate feedback and produces sparse rewards. Such sparsity hinders the model's ability to learn optimal policies and contributes to training instability (Lightman et al., 2023).

We observe that a significant underlying issue stems from the policy model's need to identify a reasoning trajectory from the beginning of the problem to the correct answer. This approach—comparing entire reasoning paths using policy gradients—leads to excessive randomness during the sampling phase. Although it expands the search space, it often fails to find suitable reasoning paths, resulting in inefficient sampling and high variance.

An alternative perspective arises: since exploring a complete reasoning path from the beginning of the problem introduces various drawbacks, why not train the policy model to complete a reasoning path based on **partially correct reasoning process hint** instead? We found that this is feasible. Through our experiments, enabling the model to

complete correct reasoning paths can still effectively teach it to generate whole reasoning trajectories from the initial problem statement. Based on this insight, we propose the ***R**eplay-based **P**olicy **O**ptimization*(RPO). Our method is grounded in a reasonable assumption: the early tokens of a reasoning path that leads to the correct answer are more likely to guide the model toward the correct reasoning trajectory. Furthermore, we investigate the relationship between the length of the truncated trailing tokens and the model's generation accuracy. As shown in Figure 1, the initial tokens of correct answers play a crucial role in guiding the model toward correct solutions, and longer prefix lengths are positively correlated with higher generation accuracy.

Specifically, we construct a cache pool for reinforcement fine-tuning to store previously generated reasoning paths and continuously update it during training. After completing the sampling generation stage for each question, we add the reasoning path that leads to the correct answer into the cache. When the same question is encountered again, we retrieve the first n tokens of the corresponding reasoning path from the cache, prepend them to the prompt, and then perform sampling generation. At the same time, we design a reward function that adaptively adjusts based on response length to improve the accuracy of gradient estimation in the experience caching scenario.

Experimental results show that this method is plug-and-play, concise and effective, enhances training stability during the reinforcement learning stage, significantly reduces the policy model's sampling time cost, and achieves performance improvements.

We propose RPO, a novel framework for reinforcement fine-tuning of LLMs, introducing an experience replay mechanism in the sampling stage. Key advantages are: **plug-and-play**: allowing easy integration into other Reinforcement Learning fine-tuning methods; **reduced resource consumption**: achieving up to 92.6% faster training; **strong stability**: mitigating common Reinforcement Learning instability in reasoning models.We evaluated RPO on Deepseek-R1-Distill-Qwen 1.5B and 7B models across six datasets. The results show that training time was reduced by approximately 90%, and compared to full-path exploration in GRPO and DAPO, performance improved by about 2%.

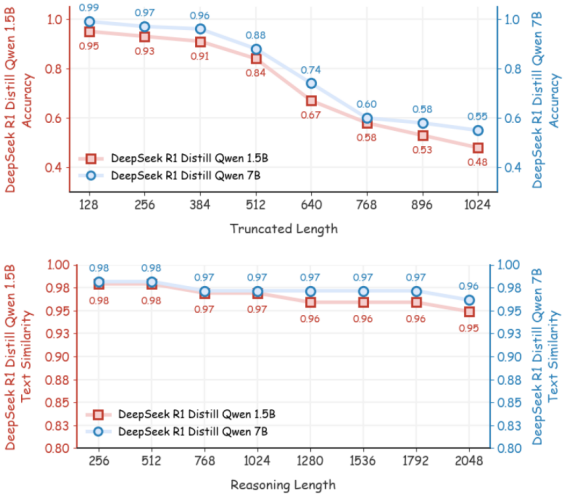

Figure 1: The top figure shows the **DeepSeekR1-Qwen-Distill-7b** and **DeepSeekR1-Qwen-Distill-1.5b** models. For each question, an initial answer is generated and then truncated; from the truncation point, 256 answers are subsequently generated, and the relationship between truncation length and the overall average accuracy is analyzed. The bottom figure shows 256 answers generated for each training question. Answers exceeding 2048 tokens are selected, and **BERT** is used to measure the similarity between equal-length prefix segments. The similarity metric is defined as: $\text{sim} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{\text{BERT}(s_i) \cdot \text{BERT}(s_j)^\top}{\|\text{BERT}(s_i)\| \, \|\text{BERT}(s_j)\|}$.

2 Related Work

Reinforcement Fine-Tuning. Reinforcement Fine-Tuning (RFT) guides the model fine-tuning process through the reward mechanisms of reinforcement learning, greatly enhancing generalization and accuracy. Kimi v1.5 (Team et al., 2025) and ReFT (Luong et al., 2024) employ traditional Proximal Policy Optimization (PPO) (Schulman et al., 2017) for RFT and have demonstrated excellent performance. DeepSeek-R1 (DeepSeek-AI et al., 2025) adopts GRPO and uses verifiable reward strategies to compute policy gradients directly. DAPO (Yu et al., 2025) further optimizes GRPO to improve training stability.

Experience Replay. Recently, a small portion of work has also explored experience replay. For example, TreePO(Li et al., 2025) organizes sequences into tree structures, accelerating the rollout phase and achieving some performance improvement. SegmentPO (Guo et al., 2025) leverages subtree information to provide process supervision for each node. BREAD (Zhang et al., 2025) combines the SFT and RFT stages and attempts to incorporate

prefixes of correct answers when reasoning fails. However, these approaches focus on tree construction, resulting in relatively complex algorithms, and the acceleration in reasoning is not particularly significant. RPO focuses on improving the accuracy of gradient estimation through a minimal revision of the reward function, and combines this with experience caching to achieve simultaneous improvements in training speed and performance.

3 Method

The overall framework of our method is illustrated in Figure 2 and consists of three main components. The first component involves cache initialization. In the second component, subsequent responses are rolled out based on the initialized cache and subsequently optimized. The third component updates the cache pool using an ϵ-greedy strategy.

3.1 Cache Pool Initialization

First, we denote the dataset of samples as $\mathcal{D} = \{q_k\}_{k=1}^{N}$, where q_k represents the k-th question in the dataset. We denote the initial model parameters as θ_0, and we represent the model's answering policy by π_{θ_0}. Before training begins, we initialize the cache pool as $\mathcal{C}^{(0)}$ as follows:

$$\mathcal{C}^{(0)} = \{(q_k, a_k) \mid a_k \sim \pi_{\theta_0}(\cdot|q_k), \forall q_k \in \mathcal{D}\}. \tag{1}$$

This stage uses the initial model policy to sample the dataset $\mathcal{D}$. To retrieve the response a_k corresponding to question q_k from the cache pool, we define the retrieval operation as:

$$a_k \coloneqq \{a \mid (q_k, a) \in \mathcal{C}\}. \tag{2}$$

Here, a_k denotes the answer associated with question q_k in the cache pool $\mathcal{C}$.

3.2 Rollout And Optimization

Rollout. At each rollout stage, we use the RPO strategy to retrieve the historical response a_k for each question q_k from the cache pool $\mathcal{C}$. We then remove the last m tokens and concatenate the remaining prefix with q_k to construct the input instruction, thereby generating a new response o. We express this process as:

$$o = a_k^{[0:-m]} \| \pi_\theta\big(\cdot \mid q_k, a_k^{[0:-m]}\big), \tag{3}$$

where $a_k \coloneqq \{a \,|\, (q_k, a) \in \mathcal{C}\}$, $m \sim \mathcal{U}\{0,1,...,L\}$.,$L$ is the maximum truncation length, $\mathcal{U}\{0, 1, ..., L\}$ samples a truncation point uniformly from $[0, L]$, $a_k^{[0:-m]}$ truncates the last m tokens of a_k, and $\pi_\theta(\cdot|q_k, a_k^{[0:-m]})$ generates a new continuation based on the question and prefix. In this paper, L is either fixed or set dynamically based on the shortest response in a sampling group G, denoted as ℓ, where:

$$\ell = \min\{\text{len}(o_1), \text{len}(o_2), \ldots, \text{len}(o_G)\}. \tag{4}$$

Replay-based Policy Optimization. After completing the sampling generation, RPO adopts **Group Relative** estimation of advantage. For a given question-answer pair (q_k, a_k), the behavioral policy $\pi_{\theta_{t-1}}$ samples a group of G individual responses $\{o_i\}_{i=1}^{G}$ from the model. Then, by normalizing the group rewards $\{R_i\}_{i=1}^{G}$, the advantage of each response is computed as:

$$\mathcal{J}_{\text{RPO}}(\theta_t) = \mathbb{E}_{(q,a)\sim\mathcal{C}^{(t-1)},\, o_{1:G}\sim\pi_{\theta_t}} \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{j=1}^{|o_i|}\ell_{i,j}(\theta_t)\right] - \beta D_{\text{KL}}(\pi_{\theta_t}\|\pi_{\text{ref}}), \tag{5}$$

where the token-level RPO loss is defined as:

$$\ell_{i,j}(\theta_t) = \min\Big(r_{i,j}(\theta_t)\hat{A}_{i,j}, \; \text{clip}\big(r_{i,j}(\theta_t), 1\pm\epsilon\big)\hat{A}_{i,j}\Big). \tag{6}$$

The probability ratio and the normalized advantage are given by

$$r_{i,j}(\theta_t) = \frac{\pi_{\theta_t}(o_{i,j}\,|\,q, o_{i,<j})}{\pi_{\theta_{t-1}}(o_{i,j}\,|\,q, o_{i,<j})}, \; \hat{A}_{i,j} = \frac{R_i - \mu_R}{\sigma_R}, \tag{7}$$

where μ_R and σ_R denote the mean and standard deviation of $\{R_i\}_{i=1}^{G}$, respectively.

Since RPO is policy-agnostic, we propose a unified forward reinforcement learning paradigm based on experience replay. We can then write the policy gradient function of RPO in a more general form as:

$$\nabla_\theta \mathcal{J}_{\text{RPO}}(\theta) = \underbrace{\mathbb{E}_{(q,o)\sim\mathcal{C}}}_{\text{Data Source}} \left(\frac{1}{|o|}\sum_{j=1}^{|o|}\underbrace{\mathcal{G}(q,o,j,\pi_{ref})}_{\text{Gradient Coefficient}}\nabla_\theta \log \pi_\theta(o_j|q,o_{<j})\right) \tag{8}$$

Equation 8 is derived from the standard policy gradient formulation. The above equations indicate that only the sampling stage is affected by RPO, while the policy gradient function remains

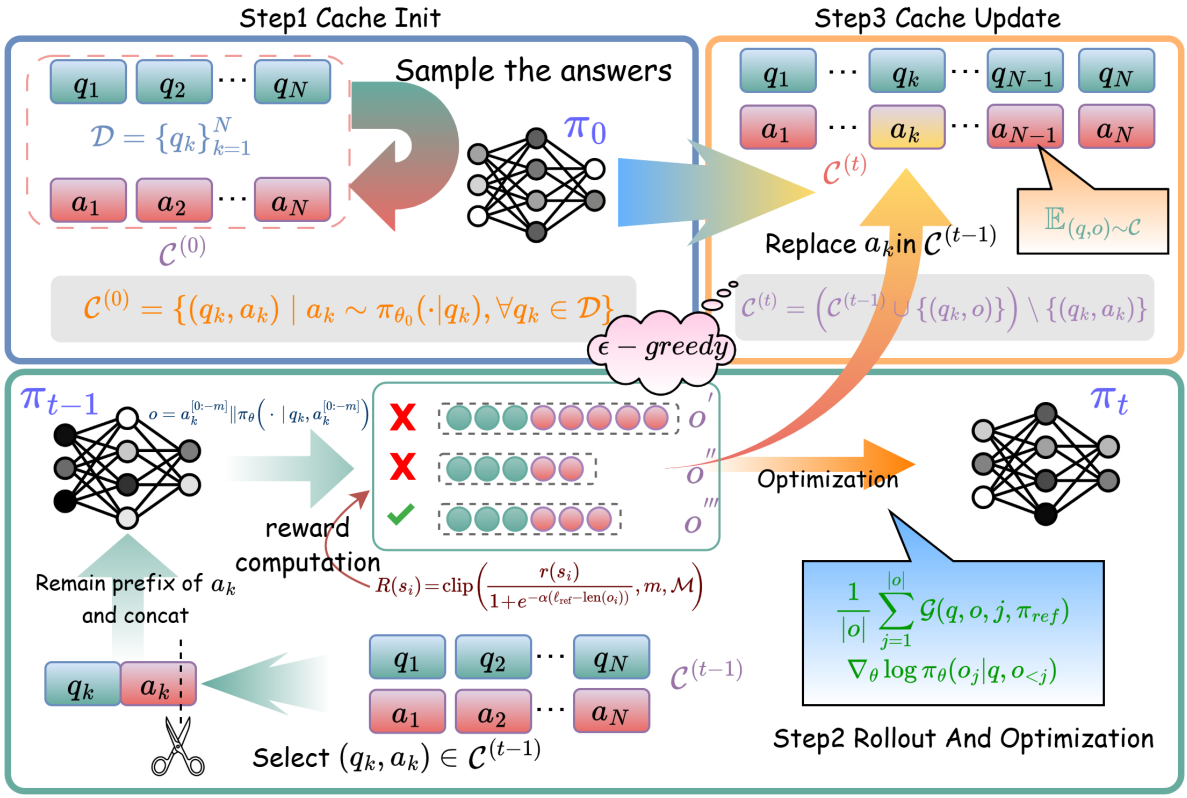

Figure 2: Overview of the RPO framework. The entire training process is described as follows: Cached answer fragments are used by the model to generate new responses; either the best or a random response is selected based on the reward system for optimization; and the cache is continuously updated to improve training efficiency and stability.

unaltered. As a result, RPO exhibits a plug-and-play nature and can be easily integrated into other reinforcement fine-tuning algorithms.

Compared with traditional full-path reasoning optimization, RPO introduces previously sampled historical response trajectories as constraints during subsequent sampling. In this way, the policy space π_θ explored during training is restricted. This constraint regularizes the gradient descent space during learning, which can be expressed as:

$$Var(\|\nabla_\theta \mathcal{J}_{\text{RPO}}\|_2) \leq Var(\|\nabla_\theta \mathcal{J}_{\text{ALL}}\|_2). \quad (9)$$

where,$\nabla_\theta \mathcal{J}_{\text{ALL}}$ represents the gradient of the full-path reasoning optimization. Theoretically, our method enables a more stable training process. Detailed mathematical proofs are provided in Appendix A.

3.3 Cache Update

After each gradient update, we adopt the ε-greedy algorithm to update the experience cache by selecting the highest-reward response from the current inference results. Specifically, when the random variable $u \sim \mathcal{U}(0,1)$ satisfies $u \leq \varepsilon$, we select the response with the highest group reward; otherwise, we randomly select a suboptimal response. We formalize the update process as:

$$\mathcal{C}^{(t)} = \left(\mathcal{C}^{(t-1)} \cup \{(q_k, o)\}\right) \setminus \{(q_k, a_k)\}, \quad (10)$$

where $o = o_{\text{argmax}\{R_i\}_{i=1}^G}$ if $u \leq \varepsilon$, and $o = o_{g'}$ otherwise. Here, $o_{\text{argmax}\{R_i\}_{i=1}^G}$ denotes the highest-reward response in the group, and $o_{g'}$ is another randomly selected candidate response.

3.4 Length-Aware Reward Shaping

However, since the same response prefix is shared during the sampling phase, the diversity of responses within the group is reduced compared to the full-path reasoning optimization algorithm. This lower diversity results in more similar reward signals, thereby diminishing the effectiveness of policy gradient estimation. To ensure meaningful gradients, reasonable reward differences should be maintained within the group, even when all responses are correct.

A **Length-Aware Reward Shaping** method is proposed to address this issue. This method is

based on the assumption: *For the same question, a reasoning path that reaches the correct answer more concisely should be rewarded with a higher value.* Specifically, for each response o_i in the group, its length-aware reward $R(s_i)$ is computed as:

$$R(s_i) = \text{clip}\left(\frac{r(s_i)}{1+e^{-\alpha(\ell_{\text{ref}}-\text{len}(o_i))}}, m, \mathcal{M}\right), \quad (11)$$

where, $r(s_i)$ is the original reward, ℓ_{ref} is the average length of group G, defined as $\ell_{\text{ref}} = \frac{1}{|G|}\sum_{i=1}^{|G|}\text{len}(o_i)$. The parameter $\alpha > 0$ controls the sensitivity of the reward to length differences. m and $\mathcal{M}$ are the lower and upper bounds for reward clipping to avoid extremely large or small values. $\text{clip}(\cdot, m, \mathcal{M})$ denotes restricting a value within the interval $[m, \mathcal{M}]$.

We then iterate the above steps in Sections 3.2 and 3.3 until a predefined stopping step T is reached.

Through mathematical derivation, we demonstrate that length-aware rewards are better suited for the RPO algorithm; the two can complement each other, and when the guiding path is within a certain threshold, they can enable the model to achieve greater performance gains. In contrast, traditional GRPO or DAPO algorithms, which use the full reasoning path, lack an initial fixed guiding path. This results in high variance for length-aware rewards, making it difficult to accurately estimate the true effective policy gradient, and thus they are not suitable for using length-aware rewards. Detailed proofs are provided in Appendix B.

4 Experiments

4.1 Experimental Setup

Base Models. To demonstrate the effectiveness and generality of RPO, we evaluate it on two open-source inference models with 1.5B and 7B parameters, namely **Deepseek-r1-qwen-distill-1.5b** and **Deepseek-r1-qwen-distill-7b** (DeepSeek-AI et al., 2025; Bai et al., 2023). Notably, we skip the supervised fine-tuning (SFT) phase, which is usually a prerequisite for reinforcement learning to enhance performance (Chu et al., 2025), as the selected models have already undergone this stage (DeepSeek-AI et al., 2025).

Datasets. We evaluate the models on six standard reasoning evaluation datasets: `aime25`(math ai, b), `aime24`(math ai, a), `math500`(Hendrycks et al., 2021), `amc23`(math ai, c), `minerva`(Lewkowycz et al., 2022) and `olympicbench`(He et al., 2024). To ensure fairness, all evaluations use the `lighteval`(Habib et al., 2023) toolkit.

Implementation Details. During training, we use 7k samples from the `open-rs` dataset (Dang and Ngo, 2025) with a global batch size of 576 for 4 epochs. Experiments are run on a single H20 machine with 8×H20 96G GPUs. We generate 6 samples per prompt, set the temperature to 0.7, and fix the maximum generation length at 4096. For the length-aware reward, we use $m = 0.5$, $M = 1$, and $\alpha = 0.01$. All models are fully fine-tuned. Due to time constraints, only zero-shot performance is averaged over three runs; all other ablation experiments are run once.

4.2 Zero-shot performance

We set the maximum truncation length L for each group to half of the minimum response length ℓ. Then, we train the DeepSeek-R1-Qwen-1.5B and DeepSeek-R1-Qwen-7B models for four epochs using the original GRPO algorithm, the DAPO algorithm, as well as their RPO variants, with a batch size of 576 and a maximum generation length of 4096 tokens. To ensure that the experimental results are not caused by randomness, we repeat the training three times for each experiment, then compare their mean accuracy on the designated evaluation datasets.

As shown in Table 1, as mentioned in the Method section, length-aware rewards complement the RPO algorithm. Incorporating group-wise length-aware rewards enables RPO to achieve higher accuracy on test benchmarks than GRPO and DAPO for both the 1.5B and 7B model sizes. Without group-wise length-aware rewards, RPO may experience some performance degradation; therefore, when using RPO for accelerated training, it is recommended to include group-wise length-aware rewards to enhance performance.

4.3 Training Time Overhead

To investigate the training time overhead of GRPO and RPO, each experiment is conducted on a machine with 8 H20 GPUs, using only a single GPU for sampling during the training phase. It should be noted that RPO introduces additional inference overhead during the cache initialization phase, where parallel inference is performed across all GPUs using the `vllm` framework. When the dataset

Table 1: Performance of the RPO algorithm on test datasets. w/ R means length-aware reward is used, w/o R means length-aware reward is not used. +RPO shows the effect of applying the RPO algorithm on top of each method.

Model	AIME25	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
1.5B Models							
DeepSeek-R1-Qwen-1.5B	16.7	28.8	82.2	62.9	26.5	43.3	43.4
+ GRPO (w/o R)	24.4	31.1	85.7	72.5	29.8	51.3	49.1
+RPO	24.4 0	25.6 -5.5	84.3 -1.4	69.2 -3.3	29.5 -0.3	51.7 +0.4	47.5 -1.6
+ GRPO (w/ R)	22.2	32.2	83.8	70.8	27.5	50.5	47.8
+RPO	24.4 +2.2	35.6 +3.4	85.3 +1.5	83.3 +12.5	29.8 +2.3	51.8 +1.3	51.7 +3.9
+ DAPO (w/o R)	30.0	24.4	86.2	84.2	29.7	52.7	51.2
+RPO	28.9 -1.1	24.4 0	86.0 -0.2	84.5 +0.3	29.3 -0.4	52.1 -0.6	50.9 -0.3
+ DAPO (w/ R)	26.7	30.0	85.0	84.1	29.7	51.1	50.2
+RPO	32.2 +5.5	30.0 0	86.2 +1.2	86.1 +2	29.1 -0.6	52.3 +1.2	52.7 +2.5
7B Models							
DeepSeek-R1-Qwen-7B	43.3	55.5	92.8	90.0	44.5	67.4	65.6
+ GRPO (w/o R)	43.3	53.3	95.0	90.0	44.5	67.2	65.6
+RPO	43.3 0	46.6 -6.7	92.5 -2.5	89.2 -0.8	42.3 -2.2	67.7 +0.5	63.6 -2
+ GRPO (w/ R)	40.0	48.9	95.0	88.3	43.5	66.0	63.6
+RPO	50.0 +10	61.1 +12.2	94.2 -0.8	90.8 +2.5	43.7 +0.2	67.3 +1.3	67.8 +4.2
+ DAPO (w/o R)	43.3	53.3	94.6	90.2	45.1	67.7	65.7
+RPO	46.7 +3.4	52.2 -1.1	94.2 -0.4	91.2 +1	42.7 -2.4	64.9 -2.8	65.3 -0.4
+ DAPO (w/ R)	42.2	56.7	93.2	91.8	44.6	64.5	65.5
+RPO	46.7 +4.5	54.5 -2.2	94.8 +1.6	95.2 +3.4	43.1 -1.5	64.5 0	66.5 +1

Table 2: The average number of tokens generated per sample with the RPO method.

L	1.5B Model	7B Model
300	145.88	147.06
500	158.41	168.17
800	382.20	397.89
GRPO	2689.51	2457.91

Table 3: Training time of RPO and GRPO under 4 epochs with $L = 800$, h represents hours.

Method	1.5B Model	7B Model
GRPO	77.28 h	84.53 h
+RPO	8.37 h	23.50 h

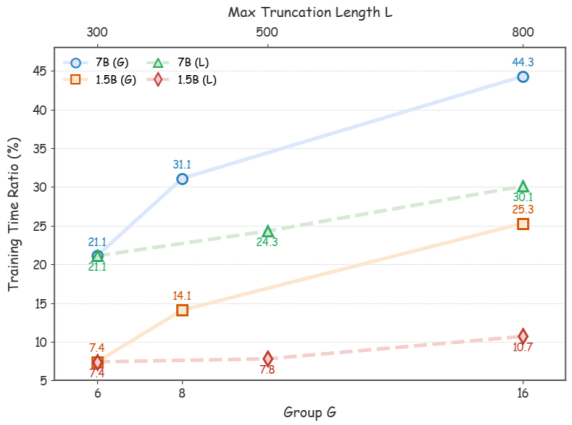

Figure 3: The impact of maximum truncation length L and group size G on the acceleration ratio of POER under the 1.5B and 7B model settings.

size is 7k and the parallel batch size is 256, this phase takes approximately 20 minutes. Our experiments reveal that the primary factors affecting the relative training speed between RPO and GRPO are the number of group samples G and the maximum truncation length L, while the impact of batch size is relatively minor.

We study training speed for both 1.5B and 7B models. With maximum truncation length fixed at $L = 300$, we set per-GPU batch sizes 2 (1.5B) and 1 (7B), and evaluate group sizes $G = 6, 8, 16$. As shown in Figure 3, smaller G yields greater acceleration for RPO, reducing training time to 7.4% of GRPO for 1.5B and 21.1% for 7B. We also study the effect of L with $G = 6$.

The actual training speed is affected by many factors, so we propose a fairer comparison: using the average tokens generated per sample. Since prefill is much faster than decoding, more tokens in prefill lead to shorter decoding time. As shown in Table 2, under the original GRPO algorithm, each sample requires an average of 2689.51 tokens and 2457.91 tokens for the 1.5B and 7B models, respectively.

In contrast, with the RPO algorithm, the number can be reduced to as low as 145.88 tokens and 147.06 tokens. From the perspective of the decode stage, the time overhead of RPO is only about 5% of that of GRPO. Table 3 presents the detailed

Table 4: Performance of GRPO and RPO on Evaluation Datasets in Multi-Step Iteration Scenarios.

Model	AIME25	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
DeepSeek-R1-Qwen-7B	43.3	55.5	92.8	90.0	44.5	67.4	65.6
+ GRPO(w/o R)	40.0	50.0	94.2	90.0	41.2	66.7	63.7
+ RPO	46.6 +6.6	56.7 +6.7	92.8 -1.4	90.0 0	41.4 +0.2	66.1 -0.6	65.6 +1.9
DeepSeek-R1-Qwen-1.5B	16.7	28.8	82.2	62.9	26.5	43.3	43.4
+ GRPO(w/o R)	10.0	10.0	67.0	45.0	20.6	31.4	34.8
+ RPO	20.0 +10	36.7 +26.7	82.8 +15.8	72.5 +27.5	29.4 +8.8	51.5 +20.1	48.8 +14

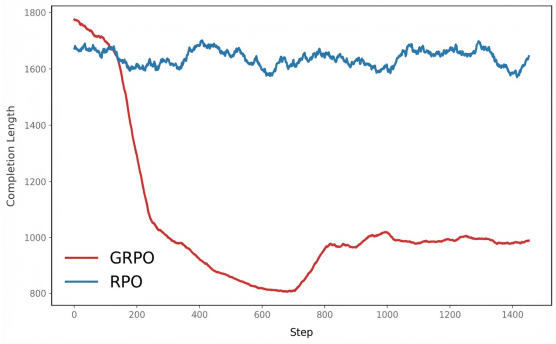

Figure 4: Response length of RPO and GRPO with full-trajectory optimization on 1.5B model across training steps.

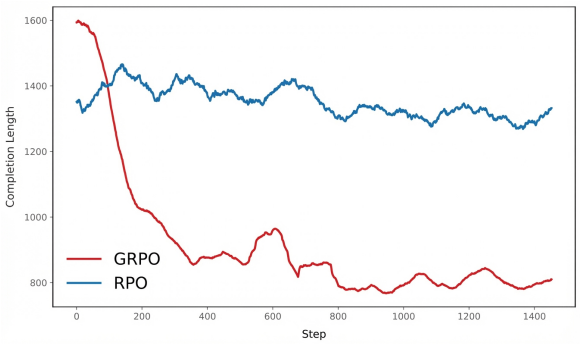

Figure 5: Response length of RPO and GRPO with full-trajectory optimization on 7B model across training steps.

training time overhead of the original GRPO and RPO algorithms over 4 epochs.

It is worth noting that the average number of tokens generated by the GRPO algorithm for the 1.5B model is higher than that for the 7B model. However, when constrained by RPO, the number of tokens generated is lower. This is because the RPO algorithm preserves shorter correct answers, and the exploration capability of the 1.5B model, once guided, is weaker compared to that of the 7B model, leading to this phenomenon.

4.4 Stability Analysis

Conventional reinforcement learning methods, such as GRPO and PPO, are prone to instability in multi-step training, with performance and response length often deteriorating as iterations increase. Accordingly, GRPO fine-tuning typically limits iteration numbers, using accuracy and response length to measure model degradation(DeepSeek-AI et al., 2025). RPO addresses this by using a cache pool mechanism, and we conduct comparative experiments to quantify its improved training stability over GRPO.

During the training process, we use a batch size of 18 to train the 7B and 1.5B models for four epochs, and monitor changes in response length and model performance, as shown in Figure 4 and 5. In this multi-step iterative training setup, GRPO experiences a collapse in response length around the 200th iteration, while RPO maintains stable response lengths throughout the process. On the other hand, as shown in Table 4, the model performance after training with GRPO deteriorated, especially for the 1.5B model, where accuracy dropped by 8.6%. In contrast, RPO results in a 5.4% improvement in accuracy.

Beyond the instability from reward sparsity, GRPO suffers from strong locality due to its inter-group comparison strategy, limiting performance improvements. RPO mitigates this by introducing an experience cache, using an external cached policy $\pi_{\mathcal{C}}$ to approximate the main policy π_θ during updates. This provides a global context, enhances training stability, and allows RPO to maintain consistent performance over long iterations.

5 More Analysis

Impact of Maximum Truncation Length and α. Intuitively, the maximum truncation length L and α are not independent factors. To study their effect on training, we train the model with combinations of $L = 300, 0.5\ell, \ell$ and $\alpha = 0, 0.01, 0.1, 1$. Notably, when $\alpha = 0$, the intra-group length-aware reward is disabled, so all correct reasoning paths receive the same reward.

The DeepSeek-R1-Qwen-1.5B model is trained for two epochs with a batch size of 336 to amplify differences in training outcomes for easier observation. The evaluation results are shown in Figure 6: under a fixed L, performance first improves as α increases and then declines.

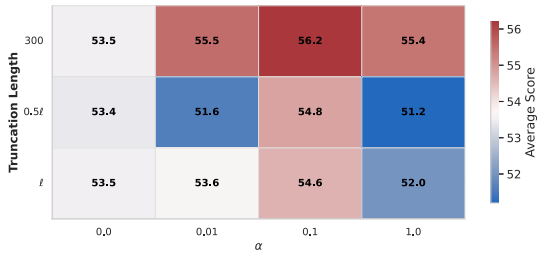

Figure 6: Heatmap of the Impact of L and α on Model Accuracy.

Effect of max_length on Exploration Capability. To investigate the impact of `max_length` settings on the model's initial exploration ability, we set the maximum truncate length of RPO to 300 and examined the performance of the 1.5B and 7B models on the AIME24 dataset under two settings: `max_length = 2048` and `max_length = 4096`. As shown in Figure 7, RPO demonstrates lower exploration ability compared to GRPO, and the gap between the two methods gradually widens as `max_length` increases. This result also indicates that RPO exhibits a certain disadvantage in exploration ability during the early iterations. However, this result compares exploration capabilities without updating the cache pool. In this scenario, RPO's exploration ability is clearly inferior to GRPO, which adopts a full-path optimization strategy.

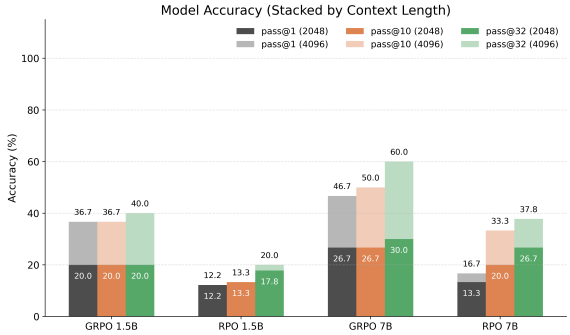

Figure 7: With the maximum truncation length set to 300, the Pass@N performance of the RPO algorithm and the full-path optimization GRPO algorithm on the AIME24 dataset is shown. The opaque bars correspond to the case with a maximum generation length of 2048, while the transparent bars correspond to the case with a maximum generation length of 4096.

Impact of Cache Pool Update Strategy on Model's Pass@N Performance. To study the effect of training epochs on exploration capability, we evaluate the 1.5B and 7B models on the AIME24 dataset with `max_length` set to 4096 under `epoch = 1,4`. As shown in Figure 8, as the number of training epochs increases and the cache pool is updated, the model is able to explore more diverse and higher-quality solution paths, with performance steadily improving to eventually match or become comparable to the full-path optimization GRPO.

Overall, although RPO exhibits reduced raw exploration ability in the early stages, the experience cache and the epsilon-greedy strategy guide the model toward higher-quality paths, enabling RPO's exploration capability in later stages to be essentially on par with that of full-path GRPO. As a trade-off strategy, RPO sacrifices some early exploration capacity but achieves significant training acceleration while ultimately attaining comparable final performance, making this trade-off clearly worthwhile.

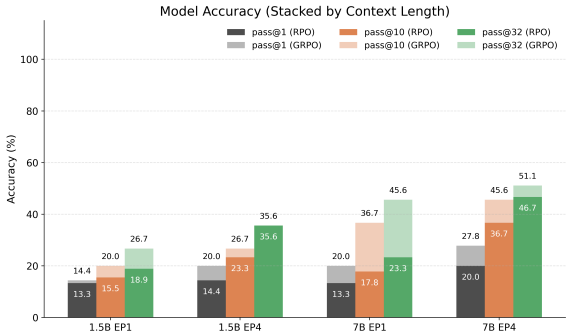

Figure 8: With the maximum truncation length set to 300 and the maximum generation length set to 4096, we investigate the Pass@N performance of the RPO algorithm and the full-path optimization GRPO algorithm on the AIME24 dataset under training epochs set to 1 and 4. The transparent bars represent GRPO, while the opaque bars represent RPO.

6 Conclusion

In this paper, we propose RPO, a plug-and-play algorithm for optimizing reinforcement fine-tuning of large language models. RPO introduces an experience replay mechanism that enables the model to leverage previously collected high-quality responses during generation. This approach substantially reduces training time while improving model performance and enhancing stability during the reinforcement fine-tuning process.

Limitations

The main limitation of the RPO algorithm is that it sacrifices response diversity for training speed. By sharing historical reasoning prefixes, the generated results within a group tend to converge, which reduces the variance of reward signals and necessitates a reliance on additional "length-aware reward shaping" to maintain performance.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *Preprint*, arXiv:2501.17161.

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. *Preprint*, arXiv:2503.16219.

DeepSeek-AI, Daya Guo, Dejian Yang, and et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *Preprint*, arXiv:2501.04519.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. 2025. Segment policy optimization: Effective segment-level credit assignment in rl for large language models. *Preprint*, arXiv:2505.23564.

Nathan Habib, Clémentine Fourrier, Hynek Kydlíček, Thomas Wolf, and Lewis Tunstall. 2023. Lighteval: A lightweight framework for llm evaluation.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *Preprint*, arXiv:2402.14008.

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

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.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, Zheng Zhang, Wei Shen, Qian Liu, Chenghua Lin, Jian Yang, Ge Zhang, and Wenhao Huang. 2025. Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *Preprint*, arXiv:2508.17445.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 3.

math ai. a. AIME24, year = 2024, url = https://huggingface.co/datasets/math-ai/aime24.

math ai. b. AIME25, year = 2025, url = https://huggingface.co/datasets/math-ai/aime25.

math ai. c. AMC23, year = 2023, url = https://huggingface.co/datasets/math-ai/ amc23.

OpenAI, , Aaron Jaech, and Adam Kalai et al. 2024a. Openai o1 system card. *Preprint*, arXiv:2412.16720.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, and et al. 2024b. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

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

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Xinming Wang, Jian Xu, Aslan H Feng, Yi Chen, Haiyang Guo, Fei Zhu, Yuanqi Shao, Minsi Ren, Hongzhu Yi, Sheng Lian, Hongming Yang, Tailin Wu, Han Hu, Shiming Xiang, Xu-Yao Zhang, and Cheng-Lin Liu. 2025a. The hitchhiker's guide to autonomous research: A survey of scientific agents.

Xinming Wang, Jian Xu, Aslan H Feng, Yi Chen, Haiyang Guo, Fei Zhu, Yuanqi Shao, Minsi Ren, Hongzhu Yi, Sheng Lian, and 1 others. 2025b. The hitchhiker's guide to autonomous research: A survey of scientific agents. *Authorea Preprints*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

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. *Preprint*, arXiv:2503.14476.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. Glm-130b: An open bilingual pre-trained model. *Preprint*, arXiv:2210.02414.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *Preprint*, arXiv:2503.18892.

Xuechen Zhang, Zijian Huang, Yingcong Li, Chenshun Ni, Jiasi Chen, and Samet Oymak. 2025. Bread: Branched rollouts from expert anchors bridge sft rl for reasoning. *Preprint*, arXiv:2506.17211.

A Proof of RPO Gradient Stability

Policy gradient estimation. The reasoning process of traditional full-path reasoning optimization and *Replay-based Policy Optimization*(RPO) can be expressed as $\pi(\cdot|q_k)$ and $\pi(\cdot|(q_k, a_k))$, where $q_k \in C_{raw}$, is a sample in raw training dataset C_{raw} and corresponding $(q_k, a_k) \in C$, is one example in training replay buffer C, which updates during the training process.

For any sample q_k, it holds that $(q_k) \subset (q_k, a_k)$. Hence, for the response space of an arbitrary policy model, the total variance can give

$$\mathrm{Var}(\cdot \mid q_k) = \mathbb{E}_{(q_k,a_k)|q_k}\Big[\mathrm{Var}\big(\cdot \mid (q_k, a_k)\big)\Big] + \mathrm{Var}_{(q_k,a_k)|q_k}\Big(\mathbb{E}\big[\cdot \mid (q_k, a_k)\big]\Big). \tag{B-1}$$

Because the second term on the right-hand side is non-negative, i.e. $\mathrm{Var}_{(q_k,a_k)|q_k}\left(\mathbb{E}[\cdot|(q_k, a_k)]\right) \geq 0$, we obtain

$$\mathrm{Var}(\cdot \mid q_k) \;\geq\; \mathbb{E}_{(q_k,a_k)|q_k}\Big[\mathrm{Var}\big(\cdot \mid (q_k, a_k)\big)\Big]. \tag{B-2}$$

Treating (q_k, a_k) as an augmentation of q_k allows this inequality to simplify to

$$\mathrm{Var}(\cdot \mid q_k) \;\geq\; \mathrm{Var}\big(\cdot \mid (q_k, a_k)\big). \tag{B-3}$$

In the policy space, (B-3) becomes

$$\mathrm{Var}\big(\pi_\theta(a_g \mid q_k)\big) \;\geq\; \mathrm{Var}\big(\pi_\theta(a_g \mid (q_k, a_k))\big). \tag{B-4}$$

Assume there exists a parameter vector θ_0 such that the policy can be locally approximated by the first-order expansion

$$\pi_\theta = \pi_{\theta_0} + \nabla\pi^\top(\theta_0)\big(\theta - \theta_0\big). \tag{B-5}$$

The variance of π_θ in a neighbourhood of θ_0 can then be estimated as

$$\sigma^2\big(\pi_\theta\big) \approx \nabla\pi_{\theta_0}^\top \Sigma_\theta \, \nabla\pi_{\theta_0}, \tag{B-6}$$

where Σ_θ denotes the covariance matrix of the parameter estimates.

Throughout training, the realizations of π_θ can be treated as i.i.d. random variables. As the sample size $n \to \infty$, the empirical mean and variance converge to

$$\mu\big(\pi_\theta\big) = \mathbb{E}\big[\pi_\theta\big] = \frac{1}{n}\sum_{i=1}^{n}(\pi_\theta)_i, \qquad \mathrm{Var}\big(\pi_\theta\big) = \mathbb{E}\big[\pi_\theta^2\big] - \mu\big(\pi_\theta\big)^2 \approx \frac{n}{n-1}\,\nabla\pi_{\theta_0}^\top \Sigma_\theta \, \nabla\pi_{\theta_0}.$$

On account of $\mathbb{E}[\pi_\theta^2] = \mu(\pi_\theta)^2$, while $\Sigma_\theta = I$ also holds, then

$$\sigma^2\big(\pi_\theta\big) = \mathrm{Var}\big(\pi_\theta\big) = \nabla\pi_{\theta_0}^\top \nabla\pi_{\theta_0} = \big\|\nabla\pi_{\theta_0}\big\|_2^2. \tag{B-7}$$

Combining the above with (B-3) yields the policy space gradient estimation.

$$\Big\|\nabla\pi_\theta\big(a_g \mid q_k\big)\Big\|_2 \;\geq\; \Big\|\nabla\pi_\theta\big(a_g \mid (q_k, a_k)\big)\Big\|_2. \tag{B-8}$$

This establishes that conditioning on the augmented information (q_k, a_k) strictly reduces—or at worst preserves—the magnitude of the policy-gradient variance.

Let's review the GRPO update policy. At training step t, the optimisation target of *Generative Reinforcement Policy Optimisation* (GRPO) can be written as

$$\begin{aligned}\mathcal{J}_{\mathrm{GRPO}}(\theta_t) = \mathbb{E}_{(q,a)\sim\mathcal{C}^{(t-1)}(Q,A),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{t-1}}(\cdot|q))} \\ \left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{j=1}^{|o_i|}\min\left(r_{i,j}(\theta_t)\hat{A}_{i,j}, \mathrm{clip}\Big(r_{i,j}(\theta_t), 1-\epsilon, 1+\epsilon\Big)\hat{A}_{i,j} - \beta D_{\mathrm{KL}}(\pi_{\theta_t}\|\pi_{\mathrm{ref}})\right)\right]\end{aligned} \tag{B-9}$$

Here

$$r_{i,j}(\theta_t) = \frac{\pi_{\theta_t}(o_{i,j} \mid q)}{\pi_{\theta_{t-1}}(o_{i,j} \mid q)}, \qquad \hat{A}_{i,j} = \frac{R_i - \text{mean}\big(\{R_i\}_{i=1}^{G}\big)}{\text{std}\big(\{R_i\}_{i=1}^{G}\big)}.$$

Relative to GRPO, RPO only changes the policy ratio by conditioning on the prefix $o_{i,<j}$:

$$r_{i,j}^{\text{RPO}}(\theta_t) = \frac{\pi_{\theta_t}(o_{i,j} \mid q,\, o_{i,<j})}{\pi_{\theta_{t-1}}(o_{i,j} \mid q,\, o_{i,<j})}. \tag{B-10}$$

Because the `clip` operation truncates high-error updates, both algorithms behave identically whenever clipping is activated.

Gradient of the optimization policy. For the plain GRPO ratio one obtains

$$\nabla_\theta\, r_{i,j}(\theta_t) = \frac{\nabla_\theta \pi_{\theta_t}(o_{i,j} \mid q)}{\pi_{\theta_{t-1}}(o_{i,j} \mid q)} = \frac{\pi_{\theta_t}(o_{i,j} \mid q)}{\pi_{\theta_{t-1}}(o_{i,j} \mid q)}\, \nabla_\theta \log \pi_{\theta_t}(o_{i,j} \mid q). \tag{B-11}$$

The Kullback–Leibler divergence with respect to a frozen reference policy π_{ref} satisfies

$$\begin{aligned} \nabla_\theta D_{\text{KL}}\big(\pi_{\theta_t} \,\|\, \pi_{\text{ref}}\big) &= \nabla_\theta \mathbb{E}_{\pi_{\theta_t}}\Big[\log \pi_{\theta_t} - \log \pi_{\text{ref}}\Big] \\ &= \mathbb{E}_{\pi_{\theta_t}}\Big[\nabla_\theta \log \pi_{\theta_t} + \big(\log \pi_{\theta_t} - \log \pi_{\text{ref}}\big)\nabla_\theta \log \pi_{\theta_t}\Big] \\ &= \mathbb{E}_{\pi_{\theta_t}}\Big[\nabla_\theta \log \pi_{\theta_t}\, \big(\log \tfrac{\pi_{\theta_t}}{\pi_{\text{ref}}} + 1\big)\Big] \end{aligned} \tag{B-12}$$

As $\mathbb{E}_{\pi_{\theta_t}}[\nabla_\theta \log \pi_{\theta_t}] = 0$ by normalisation, the expression simplifies to

$$\begin{aligned} \nabla_\theta D_{\text{KL}} =& \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(o|s) \cdot \log \pi_\theta(o|s)\right] \\ =& \sum_{\pi_\theta} \pi_\theta \nabla_\theta \log \pi_\theta(o|s) \cdot \log \pi_\theta(o|s) \\ =& \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \pi_\theta \nabla_\theta \log \pi_\theta(o_i|s) \cdot \log \pi_\theta(o_i|s) \end{aligned} \tag{B-13}$$

Resulting policy gradient. Aggregating the intra-group updates yields the estimator

$$\begin{aligned} \nabla_\theta \mathcal{J}_{\text{ALL}} &= \mathbb{E}_{q,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{j=1}^{|o_i|}\Big(\tfrac{\hat{A}_{i,j}}{\pi_{\theta_{t-1}}(o|q)} - \beta\, \log \pi_{\theta_t}(o \mid q)\Big)\, \pi_{\theta_t}(o \mid q)\, \nabla_\theta \log \pi_{\theta_t}(o \mid q)\right] \\ &= \mathbb{E}_{q,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{j=1}^{|o_i|}\Big(\tfrac{\hat{A}_{i,j}}{\pi_{\theta_{t-1}}(o|q)} - \beta\, \log \pi_{\theta_t}(o \mid q)\Big)\, \nabla_\theta \pi_{\theta_t}(o \mid q)\right]. \end{aligned} \tag{B-14}$$

The second line follows by noting that $\pi_{\theta_t} \nabla_\theta \log \pi_{\theta_t} = \nabla_\theta \pi_{\theta_t}$. Equation above provides the final form of the GRPO gradient used for parameter updates at step t.

Without consideration of KL divergence. If the KL–divergence term is temporarily ignored, the GRPO gradient estimator reduces to

$$\nabla_\theta \mathcal{J}_{\text{ALL}} = \mathbb{E}_{q,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\big(\tfrac{\hat{A}_{i,t}}{\pi_{\theta_{t-1}}}\big)\, \nabla \pi_\theta(a_g \mid q_k)\right]. \tag{B-15}$$

Because of equation (B-8),

$$\|\nabla_\theta \mathcal{J}_{\text{ALL}}\|_2 \;\geq\; \|\nabla_\theta \mathcal{J}_{\text{RPO}}\|_2. \tag{B-16}$$

Including the KL divergence. At the initial step ($t = 0$) both algorithms share the same reference policy, hence

$$\nabla_\theta \mathcal{J}_{\mathrm{ALL}} = \nabla_\theta \mathcal{J}_{\mathrm{RPO}}. \tag{B-17}$$

For the first update ($t = 1$) (B-8) implies

$$\|\nabla \pi_{\theta_1}(a_g \mid q_k)\|_2 \;\geq\; \|\nabla \pi_{\theta_1}(a_g \mid q_k, a_k)\|_2. \tag{B-18}$$

Here we record $\nabla \pi_{\theta_1}(a_g \mid q_k, a_k)$ as $\nabla \pi'_{\theta_1}$. Consequently, the difference of the two policy gradients becomes

$$\begin{aligned}\Delta_1 \;&=\; \|\nabla_\theta \mathcal{J}_{\mathrm{ALL}}\|_2 \;-\; \|\nabla_\theta \mathcal{J}_{\mathrm{RPO}}\|_2 \\ &=\; \mathbb{E}_{q,\{o_i\}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\Big(\|(\tfrac{\hat{A}_{i,1}}{\pi_{\theta_0}} - \beta \log \pi_{\theta_1})\nabla \pi_{\theta_1}\|_2 - \|(\tfrac{\hat{A}'_{i,1}}{\pi_{\theta_0}} - \beta \log \pi'_{\theta_1})\nabla \pi'_{\theta_1}\|_2\Big)\right].\end{aligned} \tag{B-19}$$

Let $\delta := \nabla_\theta \pi_{\theta_1}(i) - \nabla_\theta \pi'_{\theta_1}(i) \geq 0$. For one random dimension the mean-value theorem yields

$$\pi_{\theta_1}(i)\log \pi_{\theta_1}(i) - \pi'_{\theta_1}(i)\log \pi'_{\theta_1}(i) = \frac{\partial(\pi \log \pi)}{\partial \pi}\Big|_{\pi=\zeta}(\pi_{\theta_1}(i) - \pi'_{\theta_1}(i)), \quad \zeta \in [\pi'_{\theta_1}, \pi_{\theta_1}] \subset [0, 1). \tag{B-20}$$

Taking the directional derivative with respect to θ gives

$$\begin{aligned}(1 + \log \pi_{\theta_1}(i))\nabla \pi_{\theta_1}(i) - (1 + \log \pi'_{\theta_1}(i))\nabla \pi'_{\theta_1}(i) =& \frac{\partial \pi_\theta \log \pi_\theta}{\partial\, \pi_\theta}|_{\pi_\theta=\zeta}(\nabla \pi_{\theta_1}(i) - \nabla \pi'_{\theta_1}(i)) \\ =& \frac{\partial \pi_\theta \log \pi_\theta}{\partial\, \pi_\theta}|_{\pi_\theta=\zeta}\delta\end{aligned} \tag{B-21}$$

Hence

$$\log \pi_{\theta_1}(i)\,\nabla \pi_{\theta_1}(i) - \log \pi'_{\theta_1}(i)\,\nabla \pi'_{\theta_1}(i) = \big(\tfrac{\partial(\pi \log \pi)}{\partial \pi}\big|_{\pi=\zeta} - 1\big)\delta = (\log \zeta)\,\delta \;\leq\; 0, \tag{B-22}$$

because $\log \zeta < 0$.

Extending this argument component-wise to the full parameter vector shows

$$\log \pi_{\theta_1}\,\nabla \pi_{\theta_1} \preceq \log \pi'_{\theta_1}\,\nabla \pi'_{\theta_1}, \tag{B-23}$$

and therefore

$$\| - \beta \log \pi_{\theta_1}\,\nabla \pi_{\theta_1}\|_2 \geq \| - \beta \log \pi'_{\theta_1}\,\nabla \pi'_{\theta_1}\|_2, \tag{B-24}$$

We have thus established

$$\|\nabla_{\theta_1} \mathcal{J}_{\mathrm{ALL}}\|_2 \;\geq\; \|\nabla_{\theta_1} \mathcal{J}_{\mathrm{RPO}}\|_2. \tag{B-25}$$

Let the generic update rule be

$$\theta_i = \theta_{i-1} + \eta\, \nabla_\theta \mathcal{J}. \tag{B-26}$$

Then

$$\frac{\nabla \pi_{\theta_i}}{\pi_{\theta_{i-1}}} = \frac{\nabla\big(\pi_{\theta_{i-1}} + \eta\, \nabla \pi_{\theta_{i-1}} \nabla_{\theta_{i-1}} \mathcal{J}\big)}{\pi_{\theta_{i-1}}} \;\geq\; \frac{\nabla \pi_{\theta_{i-1}}}{\pi_{\theta_{i-1}}} = \nabla \log \pi_{\theta_{i-1}}, \tag{B-27}$$

i.e. each step re-enters the original policy-gradient (PG) regime. Using (B-25) one obtains for every $i \geq 1$

$$\frac{\nabla \pi_{\theta_i}}{\pi_{\theta_{i-1}}} - \frac{\nabla \pi'_{\theta_i}}{\pi'_{\theta_{i-1}}} \;\geq\; \frac{\nabla \pi_{\theta_{i-1}}}{\pi_{\theta_{i-1}}} - \frac{\nabla \pi'_{\theta_{i-1}}}{\pi'_{\theta_{i-1}}} \;\geq\; \nabla \log \frac{\pi_{\theta_{i-1}}}{\pi'_{\theta_{i-1}}}. \tag{B-28}$$

By induction this yields the general relation

$$\|\nabla_\theta \mathcal{J}_{\mathrm{ALL}}\|_2 \;\geq\; \|\nabla_\theta \mathcal{J}_{\mathrm{RPO}}\|_2 \qquad \text{for all optimisation steps.} \tag{B-29}$$

Equation (B-29) completes the proof that, under identical hyper-parameters, GRPO provides gradient updates at least as large as those of RPO, both without and with the KL divergence. Meanwhile, since both of them follow a normal distribution with zero mean, it follows that:

$$Var(\|\nabla_\theta \mathcal{J}_{\mathrm{RPO}}\|_2) \leq Var(\|\nabla_\theta \mathcal{J}_{\mathrm{ALL}}\|_2) \tag{B-30}$$

B theorem

Preliminary. Let q denote the prompt, o a sampled response with length $\ell = \mathrm{len}(o)$, and let the group-wise reference length be $\ell_{\mathrm{ref}} = \frac{1}{G}\sum_{i=1}^{G} \mathrm{len}(o_i)$. Write $\Delta\ell = \ell - \ell_{\mathrm{ref}}$ and fix a window $|\Delta\ell| \le \tau$. For a sensitivity parameter $\alpha > 0$ define the length weight $s_\alpha(\ell) = \sigma\big(\alpha(\ell_{\mathrm{ref}} - \ell)\big) = (1 + e^{-\alpha(\ell_{\mathrm{ref}} - \ell)})^{-1} \in (0, 1)$ and the shaped reward $R_\alpha = \mathrm{clip}\big(s_\alpha(\ell)\, r,\ m,\ \mathcal{M}\big)$ with clipping bounds $m < \mathcal{M}$, where r is the original per-sample reward.

Consider RPO with replay distribution μ and current policy π_θ, truncated importance ratio $\rho = \min\big(c, \frac{\pi_\theta(o|q)}{\mu(o|q)}\big)$ for a constant $c \ge 1$, token-averaged score function $\nabla_\theta \log \pi_\theta(o \mid q) = \frac{1}{|o|}\sum_{j=1}^{|o|} \nabla_\theta \log \pi_\theta(o_j \mid o_{<j}, q)$, and a centered advantage $A'_\alpha = R_\alpha - b_\alpha$ with group baseline $b_\alpha = \mathbb{E}[R_\alpha \mid q, \mathrm{group}]$. The single-sample gradient contribution is

$$g_\alpha \;=\; \rho\,\Big(A'_\alpha - \beta \log \pi_\theta(o \mid q)\Big)\,\nabla_\theta \log \pi_\theta(o \mid q). \tag{C-1}$$

Assume $\|\nabla_\theta \log \pi_\theta(o \mid q)\| \le L$, $\mathbb{E}[r^2] < \infty$, and that

1. the conditional variance $\sigma_A^2(\ell) := \mathrm{Var}(A \mid \ell)$ of the unshaped advantage A is nondecreasing in ℓ,
2. the tail probability $\mathbb{P}\big(\frac{\pi_\theta}{\mu} > c \mid \ell\big)$ is nondecreasing in ℓ.

If $\alpha\tau \le 1$, then there exists $\alpha^\star > 0$ such that for all $0 < \alpha \le \alpha^\star$ the mean-squared error $\mathrm{MSE}_\alpha := \mathrm{Var}(g_\alpha) + \|\mathbb{E}[g_\alpha] - \nabla_\theta J\|^2$ of the RPO gradient estimator with length-aware shaping satisfies

$$\mathrm{MSE}_\alpha \;<\; \min\Big\{\mathrm{MSE}_0^{\mathrm{RPO}},\ \mathrm{MSE}_0^{\mathrm{ALL}},\ \inf_{\tilde\alpha > 0} \mathrm{MSE}_{\tilde\alpha}^{\mathrm{ALL}}\Big\}, \tag{C-2}$$

that is, it strictly improves upon both the unshaped RPO baseline and the ALL baselines in a nontrivial neighborhood of $\alpha = 0$.

Proof. The proof makes explicit the first-order behavior in α of both the variance and the bias terms. Throughout the window $|\Delta\ell| \le \tau$ the sigmoid admits the uniform Taylor expansion

$$s_\alpha(\ell) \;=\; \frac{1}{2} \;-\; \frac{\alpha}{4}\,\Delta\ell \;+\; R_2(\alpha, \ell), \qquad |R_2(\alpha, \ell)| \;\le\; C_2\,\alpha^2\tau^2, \tag{C-3}$$

for some constant C_2 independent of α and ℓ. Writing $R_\alpha = s_\alpha(\ell)\, r$ on the non-clipping region and absorbing the clipping into the moment bounds later, the centered advantage becomes

$$A'_\alpha \;=\; \Big(\tfrac{1}{2}\,r - \mathbb{E}[\tfrac{1}{2}\,r]\Big) \;-\; \frac{\alpha}{4}\Big(\Delta\ell\, r - \mathbb{E}[\Delta\ell\, r]\Big) \;+\; \underbrace{R_2(\alpha, \ell)\, r - \mathbb{E}[R_2(\alpha, \ell)\, r]}_{=: E_2(\alpha)}. \tag{C-4}$$

Substituting (C-4) into (C-1) and taking expectations yields

$$\mathbb{E}[g_\alpha] - \mathbb{E}[g_0] \;=\; -\frac{\alpha}{4}\,\mathbb{E}\Big[\rho\,\big(\Delta\ell\, r - \mathbb{E}[\Delta\ell\, r]\big)\,\nabla_\theta \log \pi_\theta(o \mid q)\Big] \;+\; \mathbb{E}\Big[\rho\, E_2(\alpha)\,\nabla_\theta \log \pi_\theta(o \mid q)\Big]. \tag{C-5}$$

By Cauchy–Schwarz and the bounds on ρ and the score function, the norm of the first term on the right-hand side satisfies

$$\Big\|\mathbb{E}\Big[\rho\,\big(\Delta\ell\, r - \mathbb{E}[\Delta\ell\, r]\big)\,\nabla_\theta \log \pi_\theta\Big]\Big\| \;\le\; c\,L\,\Big(\mathbb{E}\big[(\Delta\ell\, r - \mathbb{E}[\Delta\ell\, r])^2\big]\Big)^{1/2} \;\le\; c\,L\,\tau\,\big(\mathbb{E}[r^2]\big)^{1/2}, \tag{C-6}$$

hence $\|\mathbb{E}[g_\alpha] - \mathbb{E}[g_0]\| \le \frac{\alpha}{4}\,c\,L\,\tau\,(\mathbb{E}[r^2])^{1/2} + c\,L\,\mathbb{E}[|E_2(\alpha)|]$. Using $|E_2(\alpha)| \le 2C_2\alpha^2\tau^2|r|$ and $\mathbb{E}[r^2] < \infty$ gives the bias bound

$$\|\mathbb{E}[g_\alpha] - \mathbb{E}[g_0]\| \;\le\; C_b\,\alpha\,\tau \;+\; C'_b\,\alpha^2\tau^2, \tag{C-7}$$

for constants C_b, C_b' depending only on $(c, L, \mathbb{E}[r^2], C_2)$. Consequently the squared-bias contribution to MSE_α is $O(\alpha^2\tau^2)$.

For the variance term, expand the second moment as

$$\mathbb{E}\big[\|g_\alpha\|^2\] \;\le\; c^2\,\mathbb{E}\Big[\big(A'_\alpha - \beta\log\pi_\theta\big)^2\,\big\|\nabla_\theta\log\pi_\theta\big\|^2\Big] \;\le\; c^2L^2\,\mathbb{E}\Big[\big(A'_\alpha - \beta\log\pi_\theta\big)^2\Big]. \tag{C-8}$$

The cross terms between A'_α and $\beta\log\pi_\theta$ are uniformly bounded in α by Jensen and the finite second moments of r and $\log\pi_\theta$. The α-dependent leading component arises from $\mathbb{E}[{A'_\alpha}^2]$. Within the non-clipping region and after centering, the contribution that depends on length is proportional to

$$\mathbb{E}\big[s_\alpha(\ell)^2\,\sigma_A^2(\ell)\big] \;=\; \mathbb{E}[s_\alpha(\ell)^2]\,\mathbb{E}[\sigma_A^2(\ell)] \;-\; \mathrm{Cov}\big(s_\alpha(\ell)^2,\;\sigma_A^2(\ell)\big). \tag{C-9}$$

Since $s_\alpha(\ell)$ is nonincreasing in ℓ while $\sigma_A^2(\ell)$ is nondecreasing in ℓ by assumption, the reverse Chebyshev inequality ensures that the covariance in (C-9) is nonpositive and is strictly negative unless $s_\alpha(\ell)^2$ and $\sigma_A^2(\ell)$ are almost surely constant. Differentiating $\mathbb{E}[s_\alpha(\ell)^2]$ at $\alpha = 0$ and using (C-3) yields $\mathbb{E}[s_\alpha(\ell)^2] = \frac{1}{4} + O(\alpha^2\tau^2)$, while differentiating the covariance at $\alpha = 0$ gives a strictly negative slope whenever the variance $\sigma_A^2(\ell)$ is not degenerate. Therefore there exists $\eta > 0$ such that

$$\mathrm{Var}(g_\alpha) \;\le\; c^2L^2\Big(\tfrac{1}{4}\,\overline{\sigma_A^2} \;-\; \eta\,\alpha \;+\; O(\alpha^2\tau^2)\Big) \;+\; C_\beta, \tag{C-10}$$

where $\overline{\sigma_A^2} = \mathbb{E}[\sigma_A^2(\ell)]$ and C_β collects the β-dependent but α-independent finite terms.

The RPO-specific truncation bias can be written as the deviation between the untruncated importance-weight estimator and the truncated one. Let $w = \frac{\pi_\theta}{\mu}$ and $X_\alpha = (A'_\alpha - \beta\log\pi_\theta)\,\nabla_\theta\log\pi_\theta$. The bias vector equals

$$b_{\mathrm{clip}}(\alpha) \;=\; \mathbb{E}\big[(w-\rho)\,X_\alpha\big] \;=\; \mathbb{E}\big[(w-c)^+\,X_\alpha\big], \tag{C-11}$$

so that $\|b_{\mathrm{clip}}(\alpha)\| \le \mathbb{E}\big[(w-c)^+\,\|X_\alpha\|\big] \le \mathbb{E}\big[(w-c)^+\,(|A'_\alpha| + |\beta|\,|\log\pi_\theta|)\,L\big]$.

Assumption 2 implies that the event $\{w > c\}$ is more likely at larger ℓ, whereas $|A'_\alpha|$ is reduced at larger ℓ because $s_\alpha(\ell)$ decreases with ℓ and the clipping of R_α further upper-bounds its magnitude.

Consequently the mapping $\alpha \mapsto \|b_{\mathrm{clip}}(\alpha)\|$ is nonincreasing for small α, and in particular $\|b_{\mathrm{clip}}(\alpha)\| \le \|b_{\mathrm{clip}}(0)\|$. Since MSE_α contains $\|b_{\mathrm{clip}}(\alpha)\|^2$, this term does not increase with α near zero.

Combining (C-7) and (C-10) and adding the nonincreasing truncation-bias square gives

$$\mathrm{MSE}_\alpha \;=\; \mathrm{Var}(g_\alpha) + \big\|\mathbb{E}[g_\alpha] - \nabla_\theta J\big\|^2 \;\le\; c^2L^2\Big(\tfrac{1}{4}\,\overline{\sigma_A^2} - \eta\,\alpha + O(\alpha^2\tau^2)\Big) + \|b_{\mathrm{clip}}(\alpha)\|^2 + O(\alpha^2\tau^2). \tag{C-12}$$

Choosing $\alpha^\star > 0$ sufficiently small so that the linear decrease $-\eta\,\alpha$ dominates the aggregated $O(\alpha^2\tau^2)$ remainders ensures that $\mathrm{MSE}_\alpha < \mathrm{MSE}_0^{\mathrm{RPO}}$ for all $0 < \alpha \le \alpha^\star$ with $\alpha\tau \le 1$.

Since GRPO coincides with the on-policy case without any truncation channel, its α-dependence shares the same variance reduction mechanism but lacks the nonincreasing truncation-bias term $\|b_{\mathrm{clip}}(\alpha)\|^2$; therefore the same choice of α also yields $\mathrm{MSE}_\alpha^{\mathrm{RPO}} < \min\{\mathrm{MSE}_0^{\mathrm{ALL}}, \inf_{\tilde\alpha>0}\mathrm{MSE}_{\tilde\alpha}^{\mathrm{ALL}}\}$ whenever $\|b_{\mathrm{clip}}(0)\| > 0$, which holds generically under assumption (ii). This proves the stated improvement.

Remark. The token-wise averaging in GRPO, $\frac{1}{|o|}\sum_{j=1}^{|o|}$, multiplies the effective per-sample weight by $|o|^{-1}$ and thus accentuates the negative covariance in (C-9), because $|o|^{-1}$ is also nonincreasing in ℓ. The group baseline b_α used to define A'_α guarantees that the constant component of $s_\alpha(\ell)$ is removed, while the window condition $\alpha\tau \le 1$ keeps $s_\alpha(\ell)$ within the near-linear regime where (C-3) is valid and the remainder terms are uniformly controlled.

C Using a large model's cache pool to guide small model training

We design the following experiment to explore whether introducing a more powerful model for question sampling during the cache pool initialization phase can influence the resulting cache policy, thereby allowing the original model to indirectly benefit from the distillation of the stronger model's reasoning capabilities.

Specifically, we use the cache pool initialized by deepseek-r1-qwen-7b as the initial cache pool for deepseek-r1-qwen-1.5b. Then, following the original experimental setup, we train for two epochs and evaluate the final performance. As shown in Table 5, when trained using the cache pool generated by the 7B model, the 1.5B model did not significantly improve performance.

Table 5: The Performance of a 7B Model's Cache Pool on a 1.5B Model.

Model	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
DS-R1-Qwen-1.5B	23.3	84.8	75.0	28.7	53.5	53.1

D Cost Overhead

In this section, we present the cost overhead of several additional open-source models with the same parameters, as well as that of the series of models based on our RPO algorithm.

Table 6: Comparison of data usage and computational costs with 1.5B models.

	DeepScaleR-1.5B	Still-3-1.5B	RPO
Base Model	DeepSeek-R1-Distill-Qwen-1.5B		
Hardware	8× A100 80GB	1×8 A100 80GB	1×8 A100 80GB
Time	240h	150h	3h
Cost Est.	$3629	$2268	$24

Table 7: Comparison of data usage and computational costs with 7B models.

	rStar-Math-7B(Guan et al., 2025)	Eurus-2-7B-PRIME
Base Model	Qwen2.5-Math-7B	
Hardware	10×8 H100 80GB, 15×4 A100 40GB	1×8 A100 80GB
Time	–	72h
Cost Est.	–	$1088
	Qwen2.5-SimpleRL(Zeng et al., 2025)	**RPO**
Base Model	Qwen2.5-Math-7B	DS-R1-Distill-1.5B
Hardware	4×6 A100 80GB	1×8 A100 80GB
Time	36h	7h
Cost Est.	$1633	$56

E More Analysis

The impact of cache pool update strategies. To investigate the impact of different cache pool update strategies on model performance, we set ϵ to 0, 0.1, 0.5, and 1 during training. In addition, we also evaluate the model with cache pool updates completely disabled. As shown in Table 8, the performance of the 1.5B model exhibits a trend of first improving and then declining as ϵ increases. The best performance is achieved when $\epsilon = 0.1$, with an average accuracy of 52.6%.

Table 8: The impact of ϵ on model zero-shot performance. In the table, no update denotes the case where the cache pool is not updated, which serves as a baseline for comparison.

ϵ	AIME25	AIME24	MATH500	AMC23	Minerva	OlyB	Avg
0	23.3	30.0	84.8	75.0	28.3	52.4	50.0
0.1	23.3	36.7	85.4	87.5	29.4	53.0	52.6
0.5	30.0	26.7	83.8	72.5	28.3	52.4	49.0
0.9	26.7	36.7	84.4	70.0	29.8	53.0	50.1
no update	26.7	30.0	82.8	75.0	29.4	51.2	49.2

F Time Overhead for Cache Pool Initialization

This section reports whether RPO can still achieve significant training acceleration and performance improvement under extreme conditions, such as when the number of epochs is only 1.

Table 9: Cache pool initialization time (minutes) for 1.5B and 7B models under different GPU types, dataset sizes, and GPU counts.

1.5B Model

Dataset	GPU	1	4	8
7k	H20	34.78	24.61	15.10
	A100	32.11	21.45	13.98
70k	H20	347.91	249.14	160.87
	A100	327.19	214.78	135.89

7B Model

Dataset	GPU	1	4	8
7k	H20	58.57	26.44	17.95
	A100	55.43	22.56	16.19
70k	H20	582.95	261.28	179.13
	A100	566.49	238.91	167.57

Table 9 shows the model initialization time for the 1.5B and 7B models under different GPU count

configurations. Table 10 shows the training time for one epoch on an 8-card H20 machine and an 8-card A100 machine, including the computational overhead of cache initialization. As seen from the table, even in extreme cases with only a single epoch of training, RPO can still provide significant acceleration.

Table 10: Training time comparison (in hours) of DeepSeek-R1-Qwen models on H20 and A100 GPUs.

Model	H20 (hours)	A100 (hours)
DS-R1-Qwen-1.5B		
RPO	14.45	12.41
GRPO	40.98	37.35
DeepSeek-R1-Qwen-7B		
RPO	39.64	37.38
GRPO	114.50	105.70

G Pseudo Code for RPO Training Process

The pseudo code of *Replay-based Policy Optimization*(RPO) in the training process is as follows.

Algorithm 1: Compact RPO Training

Input: Dataset $\mathcal{D}$, model π_{θ_0}, cache $\mathcal{C}^{(0)}$, params η, β, ϵ, G
Output: Model π_{θ_T}, cache $\mathcal{C}^{(T)}$
Initialize Cache:
$\mathcal{C}^{(0)} = \emptyset$
for $q_k \in \mathcal{D}$ **do**
  $\mathcal{C}^{(0)} \leftarrow \mathcal{C}^{(0)} \cup \{(q_k, \pi_{\theta_0}(\cdot|q_k))\}$
for $t \leftarrow 1$ **to** T **do**
  Rollout:
  for $q_k \in \mathcal{D}$ **do**
    $a_k \leftarrow \{a \mid (q_k, a) \in \mathcal{C}^{(t-1)}\}$
    for $g \leftarrow 1$ **to** G **do**
      $\tilde{a}_i^{(g)} \leftarrow$ Concat$(a_k^{[0:-m]}, \pi_{\theta_{t-1}}(\cdot|q_k, a_k^{[0:-m]}))$
      $R_i^{(g)} \leftarrow r(q_k, \tilde{a}_i^{(g)})$
  Optimize:
  $\theta_t \leftarrow \theta_{t-1} + \eta \nabla_\theta J_{\mathrm{GRPO-Cache}}$
  Update Cache:
  for $q_k \in \mathcal{D}$ **do**
    if $u \leq \epsilon$ **then**
      $\mathcal{C}^{(t-1)} \cup \left\{(q_k, o_{\text{argmax}\{R_i\}_{i=1}^G})\right\} \setminus \{(q_k, a_k)\}$
    else
      $\mathcal{C}^{(t)} \leftarrow \mathcal{C}^{(t-1)} \cup \{(q_k, o_{g'})\} \setminus (q_k, a_k)$; // $g' \sim \mathcal{U}(1, G)$