

RLoop: An Self-Improving Framework for Reinforcement Learning with Iterative Policy Initialization

Zhiyuan Zeng^{1 2} Jiashuo Liu² Zhangyue Yin¹ Ge Zhang^{† 2} Wenhao Huang^{† 2} Xipeng Qiu^{† 1 3}
¹Fudan University ²M-A-P ³Shanghai Innovation Institute
cengzy23@m.fudan.edu.cn gezhang@umich.edu rubio8741@gmail.com xpqiu@fudan.edu.cn

Abstract

While Reinforcement Learning for Verifiable Rewards (RLVR) is powerful for training large reasoning models, its training dynamics harbor a critical challenge: “RL overfitting,” where models gain training rewards but lose generalization. Our analysis reveals this is driven by policy overspecialization and catastrophic forgetting of diverse solutions generated during training. Standard optimization discards this valuable inter-step policy diversity. To address this, we introduce **RLoop**, a self-improving framework built on iterative policy initialization. RLoop transforms the standard training process into a virtuous cycle: it first uses RL to explore the solution space from a given policy, then filters the successful trajectories to create an expert dataset. This dataset is used via Rejection-sampling Fine-Tuning (RFT) to refine the initial policy, creating a superior starting point for the next iteration. This loop of exploration and exploitation via iterative re-initialization effectively converts transient policy variations into robust performance gains. Our experiments show RLoop mitigates forgetting and substantially improves generalization, boosting average accuracy by 9% and pass@32 by over 15% compared to vanilla RL.

1. Introduction

Reinforcement Learning (RL), particularly through policy gradient methods such as PPO and its variants, has emerged as a cornerstone for aligning Large Language Models (LLMs) with complex human objectives. By enabling optimization for non-differentiable reward signals, RL has catalyzed significant advancements in diverse domains, including instruction following (Ouyang et al., 2022) and mathematical reasoning (DeepSeek-AI et al., 2025).

However, our investigation into the application of RL for

complex reasoning tasks reveals a critical, yet previously under-explored, challenge: a phenomenon we term **RL overfitting**, analogous to its supervised learning counterpart. As illustrated in Figure 1, we observe a stark divergence between the training objective and true generalization performance. While the in-distribution reward signal exhibits a steady increase throughout training (e.g., for over 700 steps), the model’s generalization capabilities—measured by out-of-distribution test accuracy and pass@k metrics—stagnate or even degrade much earlier (e.g., around 140 steps). This divergence strongly suggests that the RL agent becomes overly specialized in exploiting high-reward trajectories within its known distribution, leading to a model that is confident yet brittle when confronted with unseen problems.

To dissect the underlying dynamics of this overfitting phenomenon, we conducted a deeper empirical analysis of the RL training process. As shown in Figure 2b, we found that standard RL training suffers from **catastrophic forgetting**, especially in the later stages, where the model discards approximately 30% of the knowledge acquired during early training. This finding indicates that policies at different training steps are substantially distinct. Such inter-step policy diversity represents a valuable asset for exploration, yet it is typically discarded in conventional training paradigms. While prior work has acknowledged the importance of trajectory diversity (Cui et al., 2025; Wang et al., 2025a), it has primarily focused on the diversity generated by a single policy at a fixed step, overlooking the rich diversity across different training checkpoints.

Inspired by the potential of harnessing this inter-step diversity, we propose **RLoop**, a self-improving framework centered around iterative policy initialization. Instead of a single, monolithic training run, RLoop recasts the process as a virtuous cycle where the policy is progressively refined and re-initialized. Each iteration consists of two phases:

1. **Exploration Phase:** Starting from the current policy θ_i , we run a standard RL process not to find a single optimal policy, but to generate a diverse pool of solution trajectories. The significant policy shifts across

[†]Corresponding authors

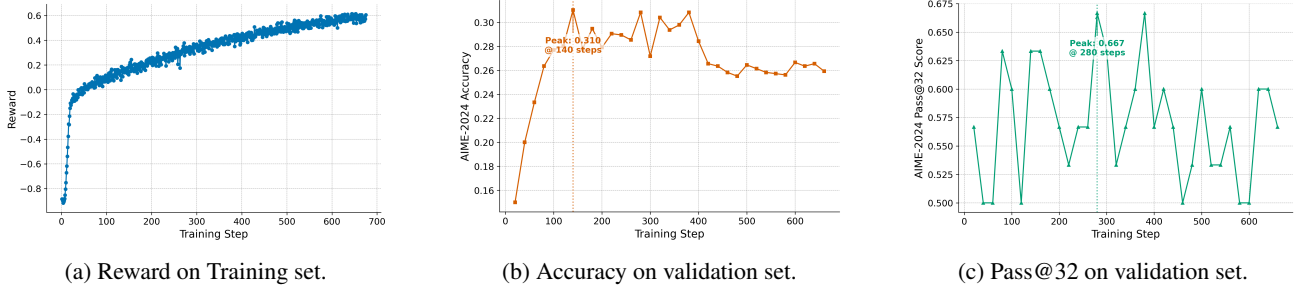


Figure 1: The reward, accuracy and pass@32 score of Qwen-2.5-math-7b trained with the DAPO algorithm evaluated on AIME-2024.

RL steps act as a powerful, built-in exploration mechanism.

2. **Exploitation Phase:** We curate the trajectories generated during exploration by filtering for successful outcomes. This “expert” dataset, D_{expert}^i , is then used to refine the initial policy θ_i via Supervised Fine-Tuning. The resulting improved policy, θ_{i+1} , serves as a superior starting point for the subsequent exploration phase.

The crucial step is that this consolidated policy θ_{i+1} is not the final output, but serves as a superior initial policy for the next Exploration Phase. RLoop thus establishes a self-contained improvement loop: RL explores possibilities from a stable base, and RFT consolidates the findings into a better base. This process of iterative policy initialization allows the model to systematically accumulate knowledge, turning the transient diversity from RL into robust, generalizable capabilities. Unlike prior works that rely on external expert data to bridge RL and supervised learning (Yan et al., 2025; Zhang et al., 2025b; Chen et al., 2025), RLoop bootstraps its own progress. To further stabilize this self-improvement, we incorporate an active learning strategy to ensure the model continually focuses on challenging problems.

Our main contributions are summarized as follows:

- We empirically identify and characterize the “RL overfitting” phenomenon in LLMs, demonstrating that reward improvements do not necessarily translate to enhanced generalization.
- We reveal that this overfitting is linked to catastrophic forgetting and highlight the untapped potential of inter-step policy diversity, a valuable resource discarded by standard RL.
- We propose **RLoop**, an iterative self-improvement framework that effectively balances exploration and exploitation by alternating between RL for diverse solution generation and RFT for knowledge consolidation.

- Our experiments demonstrate that RLoop significantly outperforms vanilla RL on challenging math reasoning benchmarks, particularly in pass@k metrics. We further show that RLoop mitigates forgetting and make the RL training more stable.

2. Related Works

Generalization of RLVR. The generalization capabilities of RLVR are a subject of active research with conflicting findings. Several theoretical and empirical studies suggest that RL can lead to strong generalization (Chu et al., 2025; Zhang et al., 2025a; Anonymous, 2025), with some work even demonstrating its effectiveness with a single question-answer pair (Wang et al., 2025b). However, this optimistic view is challenged by other research. Yue et al. (2025) empirically found that while RLVR improves greedy-decoding accuracy, it can degrade pass@k performance, especially for large k . This suggests that standard RL may merely improve test-time efficiency rather than enhancing the model’s core ability to solve novel problems. Conversely, other studies (Liu et al., 2025b) report that RLVR can indeed improve pass@k scores on certain tasks. Our work contributes to this debate by identifying a key dynamic: both accuracy and pass@k can degrade during training due to an overfitting-like phenomenon, which we aim to resolve.

Efforts to improve RLVR generalization can be categorized into three main perspectives:

1. **Data-centric Approaches:** These methods focus on enriching the training data. For instance, Li et al. (2025a) and Liang et al. (2025) propose augmenting the question set to expose the model to a wider range of states. Others focus on curriculum learning; Prakash & Buvanesh (2025) found that mixing simple and hard questions facilitates knowledge transfer, while Li et al. (2025b) advocate for increasing the rollout budget for more challenging problems, based on the finding that performance correlates with the number of unique problems solved.

2. **Algorithm-centric Approaches:** These methods modify the RL algorithm itself. A prominent line of work explores the relationship between performance and policy entropy (Cui et al., 2025; Wang et al., 2025a). Wang et al. (2025a) propose masking gradients from low-entropy tokens to focus learning on more uncertain parts of the reasoning process. Similarly, Cui et al. (2025) introduce methods like Clip-Cov and KL-Cov to constrain gradients from high-variance tokens, thereby enhancing exploration.
3. **Initialization-centric Approaches:** Recognizing that a strong starting point is crucial for RL, these approaches focus on creating a superior initial policy. Works like Wang et al. (2025c) and Guha et al. (2025) achieve this by pre-training or fine-tuning models on large, high-quality, reasoning-intensive corpora before applying RL.

Our proposed RLoop framework aligns with the initialization-centric perspective but with a critical distinction: it operates as a **self-improving**, iterative re-initialization loop. Unlike methods requiring extensive human effort for data curation, RLoop synthesizes its own "expert" data for re-initialization directly from the trajectories generated during the RL phase, creating a fully autonomous improvement cycle.

Combining RL and SFT. The RLoop framework, which iterates between RL and Rejection-sampling Fine-Tuning (RFT, a form of SFT), belongs to a broader class of methods that combine SFT and RL. The most common paradigm is a simple pipeline where SFT provides the initial policy for a subsequent RL phase (Ouyang et al., 2022).

More intricate integrations have also been explored. Some methods aim to incorporate SFT data directly into the RL process, for instance, through off-policy learning (Yan et al., 2025). Others attempt to merge the SFT and RL objectives into a single loss function for joint training (Zhang et al., 2025b; Yan et al., 2025; Chen et al., 2025). A different approach, exemplified by Ma et al. (2025a), involves interleaving SFT steps within the RL training loop, using high-quality solutions discovered during RL to reinforce the policy. Addressing the catastrophic forgetting issues observed in such methods, Yuan et al. (2025) proposed constraining the SFT updates. Another line of work, including Chen et al. (2024) and Zhong et al. (2025), formulates the problem as a latent variable model akin to a Variational Autoencoder (VAE) (Kingma & Welling, 2022), where SFT updates a reward model and RL improves the policy in an alternating fashion.

Our RLoop framework is distinct from these prior works in two fundamental ways. First, it employs a cyclical, macro-level iteration between distinct RFT and RL phases, rather

than a fine-grained interleaving or joint loss. Second, and more importantly, RLoop is entirely self-contained, bootstrapping its SFT data from the RL agent’s own successful explorations. This eliminates the need for any external expert data, setting it apart from most hybrid SFT-RL methods.

3. Preliminary Study: Characterizing Policy Dynamics in RLVR

As established in the introduction and illustrated in Figure 1, standard RLVR exhibits an overfitting-like behavior, where training rewards diverge from validation performance. To delve into the mechanisms behind this phenomenon, we conduct a preliminary study analyzing three key metrics: the learning rate, the forgetting rate, and trajectory similarity. The learning and forgetting rates quantify the model’s ability to acquire new problem-solving capabilities and its tendency to lose previously acquired ones, respectively. Trajectory similarity measures the distributional shift in generated solutions across different training steps.

3.1. Learning and Forgetting Dynamics

To understand the trade-offs during training, we analyze the policy’s ability to both learn new problems and forget old ones. We define the learning rate from a checkpoint at step i to a later checkpoint at step j ($i < j$) as the proportion of validation problems that the policy at step j can solve, but the policy at step i could not. Symmetrically, the forgetting rate is the proportion of problems that the policy at step i could solve, but the policy at step j fails to solve.

The Learning Matrix (Figure 2a) reveals that the model continuously learns to solve new problems throughout training. The non-zero values in the upper triangle indicate that later policies acquire capabilities that earlier policies lacked. This demonstrates that continued training is not merely fitting to noise; the agent is genuinely expanding its problem-solving repertoire.

However, this learning comes at a cost, as shown by the Forgetting Matrix (Figure 2b). The matrix reveals a significant level of forgetting, with rates frequently exceeding 10% and reaching as high as 35%, particularly between distant checkpoints. This observation empirically confirms that the RLVR is not only acquiring new skills but is also simultaneously discarding previously learned ones.

The interplay between learning and forgetting explains the performance stagnation observed in RLVR. In the early stages of training, the learning rate surpasses the forgetting rate, leading to a net increase in validation performance. As training progresses, the forgetting rate begins to catch up with or even exceed the learning rate. Therefore, the model’s performance on the validation set oscillates or de-

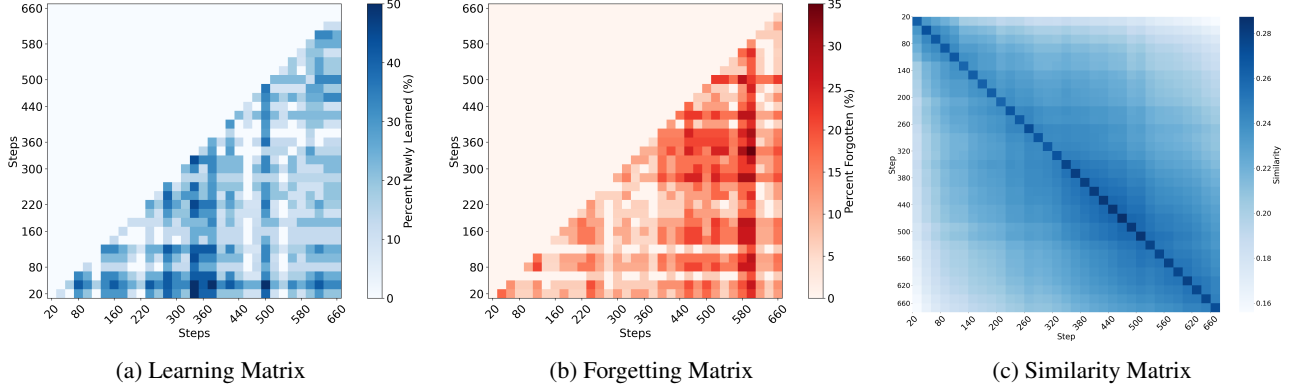


Figure 2: The value at (i, j) in the Learning Matrix represents the percentage of validation problems that the policy at step j can solve but the policy at step i cannot. Conversely, the value in the Forgetting Matrix represents problems solvable at step i but not at step j . The value at (i, j) in the Similarity Matrix indicates the average n-gram similarity of trajectories between policies from step i and step j . For all analyses, we sample 32 solutions for each question in the validation set.

creases. This dynamic provides a compelling explanation for why prolonged training does not necessarily lead to better generalization.

3.2. Similarity Matrix

To complement the performance-based analysis of the forgetting matrix, we analyze the lexical similarity of the generated trajectories. This metric quantifies the textual consistency of solutions generated by policies at two distinct training steps, i and j .

The core idea is to measure the overlap of n-grams (specifically, bi-grams) between the sets of solutions generated at these two steps. The process involves calculating the pairwise similarity for all solution pairs using the Jaccard index, and then aggregating these scores into a single, robust metric. The detailed mathematical formulation for this calculation is deferred to Appendix A.

The resulting Similarity Matrix (Figure 2c) shows that the intra-step similarity (diagonal entries, typically >0.26) is substantially higher than the inter-step similarity (off-diagonal entries, typically <0.2). Moreover, the similarity systematically decreases as the distance between steps increases.

Taken together, the Learning, Forgetting, and Similarity Matrices provide compelling evidence from two different perspectives—performance and solution form—that policies at different RL training steps are remarkably diverse. This motivates our core idea: to explicitly collect and consolidate this valuable, yet typically discarded, diversity for more robust generalization.

4. Methodology

4.1. RLoop Framework

To counteract the overfitting and instability identified in Figure 1, we introduce RLoop, an iterative training framework designed to harness the inter-step policy diversity. RLoop transforms the standard linear training process into a cyclical one, explicitly alternating between an RL-based exploration phase and an RFT-based exploitation phase.

The framework operates as an iterative loop, as detailed in Algorithm 1:

1. **Exploration Phase (RL):** Starting from a base policy π_{θ_i} , we execute a standard RL training process for a fixed number of steps. The primary goal of this phase is not to train the policy to convergence, but to leverage it as a powerful search algorithm. The inherent stochasticity and policy drift across steps drive the model to explore diverse modes of the solution space. We collect trajectories from multiple intermediate checkpoints within this phase to create a rich and varied dataset, $D_{\text{RL}}^i = \{\tau_1, \dots, \tau_N\}$.
2. **Exploitation Phase (RFT):** In this phase, we distill and consolidate the knowledge discovered during exploration. First, we filter the collected trajectories using the reward signal, retaining only successful solutions to form an “expert” dataset: $D_{\text{expert}}^i = \{\tau \in D_{\text{RL}}^i \mid R(\tau) > 0\}$. We then use this curated dataset to fine-tune the initial policy π_{θ_i} via Supervised Fine-Tuning (SFT). The resulting improved policy, $\pi_{\theta_{i+1}}$, becomes the starting point for the next iteration of the loop, thus creating a self-improving cycle.

The overall process is summarized in Algorithm 1.

Algorithm 1 Iterative Policy Initialization

Initialize: Start with a base policy π_{θ_0} (e.g., Qwen-2.5-7b-math).
for $i = 0$ **to** $I - 1$ **do**
 // — *Exploration Step* —
 Initialize RL policy from π_{θ_i} .
 Run RL for N_{RL} steps to generate a set of trajectories D_{RL}^i .
 // — *Exploitation Step* —
 Filter for successful trajectories: $D_{\text{expert}}^i = \{\tau \in D_{RL}^i \mid R(\tau) > 0\}$.
 Initialize a new policy from the same starting point π_{θ_i} .
 Perform Supervised Fine-Tuning on this policy using D_{expert}^i to obtain $\pi_{\theta_{i+1}}$.
 $\theta_{i+1} = \arg \max_{\theta} \sum_{\tau \in D_{\text{expert}}^i} \log \pi_{\theta}(\tau)$
end for
Return: The final refined policy π_{θ_I} .

4.2. Active Learning for Focused Exploitation

The efficacy of the RLoop framework arises from the complementary strengths of RL and RFT. A closer look at their optimization dynamics reveals why RFT provides stable exploitation and, crucially, why this stability benefits from an active learning strategy.

The policy gradients for RL (e.g., REINFORCE) and RFT differ fundamentally in their weighting schemes:

$$\nabla_{\theta} J_{RL}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta}} [A(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau)] \quad (1)$$

$$\nabla_{\theta} J_{RFT}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{RL}}} [R(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau)] \quad (2)$$

- **RL’s Relative Weighting:** The advantage function $A(\tau)$ measures a trajectory’s quality *relative* to the policy’s average performance. This makes RL effective at differentiating good from better but provides a vanishing learning signal when all sampled trajectories are already successful (i.e., their advantages are near zero).
- **RFT’s Absolute Weighting:** In contrast, RFT uses the *absolute* reward $R(\tau)$ as a weight (effectively 1 for success and 0 for failure). This provides a stable, low-variance learning signal that reinforces every successful trajectory, regardless of the batch’s overall performance.

While RFT’s stability is a key advantage, it introduces a potential inefficiency: RFT may over-invest capacity on problems the model already solves consistently. Since every successful trajectory receives an equal weight of 1, the model might spend excessive effort reinforcing its knowledge of “easy” problems.

This observation directly motivates our use of active learning. To make the exploitation phase more efficient and targeted, we apply a filter before the RFT step. We identify a subset of problems that the current policy finds “hard” (e.g., those with a low success rate across generated samples). The RFT update is then performed exclusively on

successful trajectories from this hard subset. This active learning strategy ensures that the model’s capacity is focused on expanding its capabilities at the frontier of its knowledge, preventing redundant updates on mastered tasks and optimizing computational resource usage.

4.3. Theoretical Grounding: RFT as Importance-Weighted MLE

The RFT phase is not merely a heuristic; it can be theoretically grounded as a form of policy improvement derived from Maximum Likelihood Estimation (MLE) with importance sampling.

Ideally, we would want to train our policy π_{θ} to match an unknown “expert” distribution $p^*(\tau)$ that produces correct and generalizable solutions. This corresponds to maximizing the log-likelihood:

$$\mathcal{L}_{MLE}(\theta) = \mathbb{E}_{\tau \sim p^*} [\log \pi_{\theta}(\tau)]. \quad (3)$$

We cannot sample directly from p^* , but we can sample from the RL policy which can be seen as an approximation of p^* . Therefore, we use importance sampling to re-express this objective using trajectories sampled from our RL policy, $\pi_{\theta_{RL}}$:

$$\mathcal{L}_{MLE}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{RL}}} \left[\frac{p^*(\tau)}{\pi_{\theta_{RL}}(\tau)} \log \pi_{\theta}(\tau) \right]. \quad (4)$$

The key challenge is the unknown importance weight $w(\tau) = p^*(\tau)/\pi_{\theta_{RL}}(\tau)$. However, we can approximate this weight using the reward signal $R(\tau)$. Intuitively, a trajectory τ with a high reward is more likely to belong to the expert distribution p^* than a trajectory with a low reward. For binary rewards ($R(\tau) \in \{0, 1\}$), this leads to a simple and powerful approximation:

$$w(\tau) = \frac{p^*(\tau)}{\pi_{\theta_{RL}}(\tau)} \propto R(\tau). \quad (5)$$

Table 1: The performance comparison between the base model (Qwen-2.5-7b-Math), RL and RLoop.

Dataset	Method	Avg@32	Pass@8	Pass@16	Pass@32
AIME	<i>Base</i>	6.00	43.65	46.09	46.66
	<i>RL</i>	31.04	50.96	56.97	63.33
	<i>RLoop</i>	37.60	58.77	66.69	73.33
MinervaMath	<i>Base</i>	8.49	25.78	29.36	31.61
	<i>RL</i>	18.37	27.64	29.27	29.63
	<i>RLoop</i>	19.88	29.95	31.58	32.59
Omini-Math	<i>Base</i>	8.00	26.38	31.58	36.20
	<i>RL</i>	19.81	27.70	29.23	30.00
	<i>RLoop</i>	21.00	31.74	34.58	37.00
MATH	<i>Base</i>	24.71	67.32	73.19	76.00
	<i>RL</i>	56.78	66.54	68.64	70.20
	<i>RLoop</i>	58.93	75.50	78.08	80.00
Avg	<i>Base</i>	16.80	40.78	45.06	47.62
	<i>RL</i>	31.50	43.21	46.03	48.29
	<i>RLoop</i>	34.35	49.00	52.73	55.73

By substituting this approximation into the importance-sampled objective (Equation 4), we arrive at the RFT objective:

$$\mathcal{L}_{\text{RFT}}(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_{\text{RL}}}} [R(\tau) \log \pi_{\theta}(\tau)]. \quad (6)$$

This objective is precisely the SFT loss applied to the reject-sampling dataset D_{expert} , thus providing a principled justification for our method.

5. Experiments

5.1. Experiment Setting

Datasets and Evaluation. For training, we employ the DAPO-17k dataset (Yu et al., 2025), which consists of 17,000 challenging mathematical problems. To ensure a comprehensive assessment of generalization, we evaluate our models on a suite of widely recognized benchmarks: AIME 2024, MinervaMath (Lewkowycz et al., 2022), OminiMath (Gao et al., 2025), and the MATH-500 test set (Hendrycks et al., 2021).

RL Setup. Our base model for all reinforcement learning experiments is Qwen-2.5-7b-Math. In line with the approach of R1-Zero (DeepSeek-AI et al., 2025), we apply RL directly to this base model with rule-based reward. We employ the DAPO algorithm (Yu et al., 2025) with a group size of 16 and a maximum generation length of 2048 tokens. For checkpoint selection, we use AIME 2024 as our validation set, selecting the model that achieves the best performance for final evaluation.

RLoop Setup For our proposed RLoop framework, the RFT phase utilizes trajectories cached during the preceding RL exploration phase. We implement an active learning strategy by filtering this data, retaining only successful trajectories from “hard” problems—defined as those where the model’s success rate during the RL phase was below 10%. Each full iteration of the RLoop cycle consists of 200 RL training steps followed by one epoch of RFT on the curated dataset.

5.2. Main Results

We conduct a comprehensive comparison between our proposed RLoop framework and a standard vanilla RL baseline (DAPO). The results are presented in Table 1. To ensure a fair comparison of computational costs, the vanilla RL baseline was trained for 600 steps, while RLoop was run for three iterations, with each iteration comprising 200 RL steps. The computational overhead of the RFT phase is negligible relative to the RL phase, making the total training budgets for both methods comparable.

As evidenced by the results, RLoop consistently and substantially outperforms the vanilla RL baseline across all evaluation benchmarks, in terms of both accuracy (Average@32) and Pass@k. The most significant gains are observed in the Pass@k scores, highlighting RLoop’s ability to generate a more diverse set of correct solutions. As expected, the performance improvement on AIME 2024 is particularly pronounced, as it was used as the validation set for checkpoint selection.

A crucial observation is the detrimental effect of vanilla

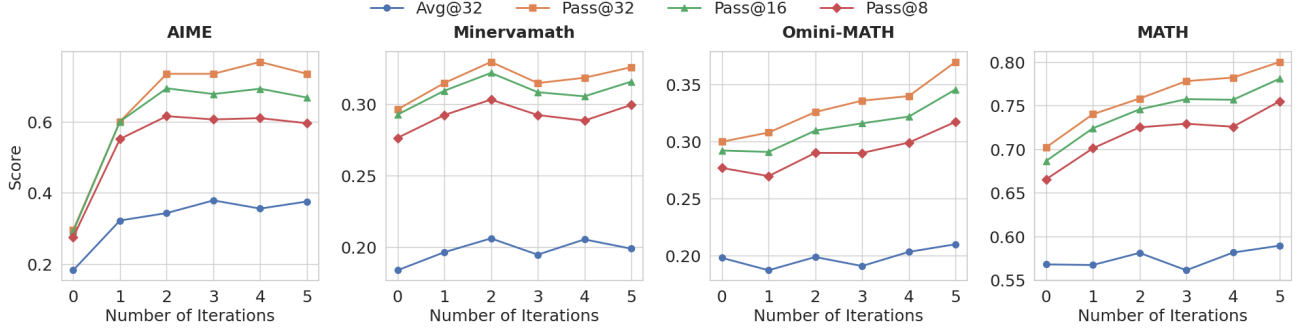


Figure 3: The performance of Qwen-2.5-7b-Math trained with RLoop in different number of iterations, in terms of accuracy and pass@k score.

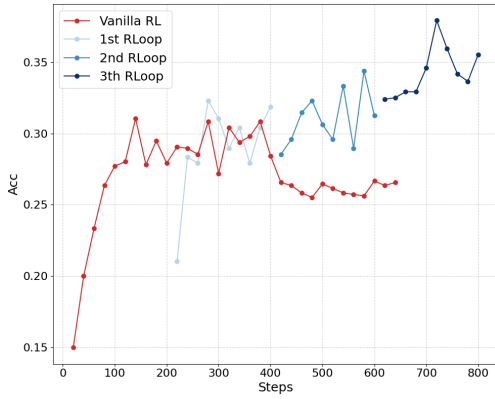


Figure 4: Compare the accuracy of vanilla RL and RLoop at different training steps.

RL on the model’s pass@k performance. On three out of four benchmarks (MinervaMath, Omni-Math, and MATH), the pass@k scores of the RL-trained model are worse than those of the original base model, especially at larger values of k like $k = 32$. This aligns with findings from prior work (Yue et al., 2025), which posited that standard RL might fail to genuinely enhance the intrinsic reasoning capabilities of LLMs. However, our RLoop framework not only reverses this degradation but surpasses the base model’s performance by a significant margin. This suggests that the performance drop is not an inherent flaw of using RL for reasoning, but rather a byproduct of the standard, continuous training paradigm that leads to overfitting.

Interestingly, vanilla RL shows a performance gain on AIME 2024, in contrast to its degradation on other benchmarks. We hypothesize that this is due to a closer distributional similarity between the AIME 2024 dataset and the DAPO-17k training set. This further supports our claim that vanilla RL tends to overfit to the training distribution, leading to diminished performance on out-of-distribution (OOD) tasks. RLoop, by cyclically exploring and consolidat-

ing knowledge, effectively mitigates this issue and achieves superior generalization.

5.3. Scalability Analysis

In this section, we investigate the scalability of RLoop by analyzing its performance over an extended number of iterations and comparing its learning dynamic against vanilla RL.

Scaling with More Iterations Figure 3 illustrates the performance of RLoop as a function of the number of iterations. The results demonstrate a clear positive scaling trend: performance on both accuracy (Avg@32) and pass@k metrics improves with additional iterations. This trend is particularly evident on the Omni-Math and MATH benchmarks. Notably, the improvement in pass@k scores is more pronounced than the gains in accuracy, suggesting that continued iterations primarily enhance the model’s ability to generate a diverse set of correct solutions.

Contrasting Scalability Dynamics To understand how RLoop utilizes computational budget differently from vanilla RL, we plot its performance against vanilla RL on a continuous training step axis. As shown in Figure 4, each 200-step RL phase of a RLoop iteration is juxtaposed with the corresponding training window of the vanilla RL baseline.

The comparison reveals a stark contrast. Vanilla RL (the red curve) exhibits classic overfitting: its performance on the validation set peaks around 300 steps and then steadily degrades, indicating that further training is detrimental. In contrast, RLoop effectively leverages the additional computational budget. While vanilla RL’s performance deteriorates, RLoop continues to achieve new performance heights with each subsequent iteration.

A closer examination reveals a fascinating dynamic within each RLoop iteration. The performance often rises before

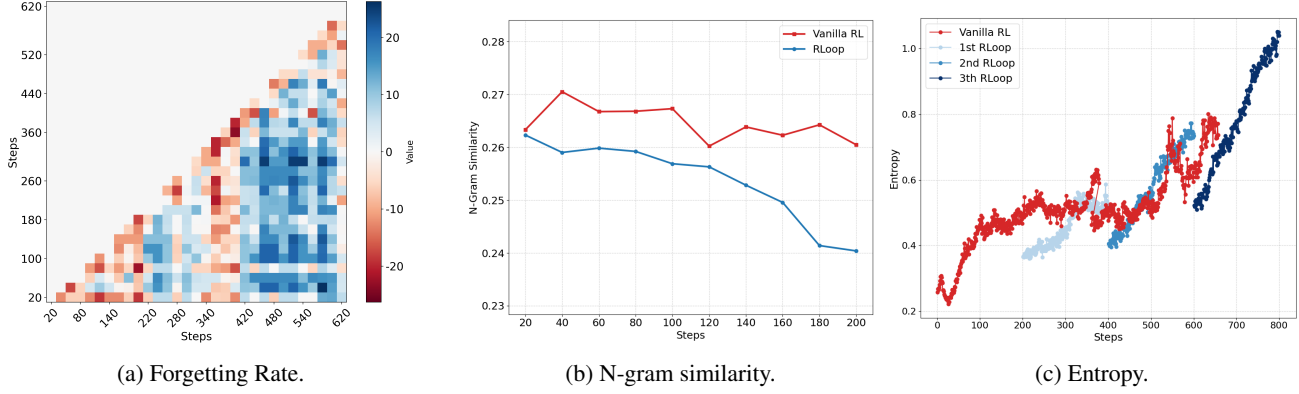


Figure 5: (a): Analysis of RLoop’s mechanisms compared to vanilla RL. (a) Differential forgetting matrix (Vanilla RL Forgetting - RLoop Forgetting). Blue indicates RLoop forgets less. (b) N-gram similarity comparison, where lower values imply higher diversity. (c) Token-level policy entropy over training steps.

plateauing or slightly fluctuating, mirroring the overfitting pattern of vanilla RL on a micro-scale. However, the crucial difference is that each new iteration begins from a superior starting point established by the RFT phase. This cyclical process allows RLoop to progressively climb to higher performance levels, escaping the terminal decline that plagues standard RL.

5.4. Why can RLoop Improve Generalization of RL?

Having established RLoop’s superior performance in Table 1 and Figure 3, we now dissect the underlying mechanisms responsible for its improved generalization. Our analysis focuses on three key areas: **catastrophic forgetting**, **trajectory diversity**, and **policy entropy**. To facilitate a direct comparison, we adopt the experimental setup from Section 5.3, aligning the i -th iteration of RLoop with the corresponding training window of the vanilla RL baseline (steps $200i$ to $200(i + 1)$).

Less Forgetting To quantify the difference in knowledge retention, we compute a differential forgetting matrix, defined as the forgetting rate of vanilla RL minus that of RLoop. As shown in Figure 5a, blue cells indicate that RLoop forgets less (a positive outcome), while red cells indicate the opposite. The matrix is predominantly blue, providing strong visual evidence that RLoop generally suffers from less catastrophic forgetting than the standard RL baseline.

A deeper analysis reveals a crucial distinction between *intra-iteration* (within an RL phase) and *inter-iteration* (across RFT resets) forgetting. The forgetting rates within each 200-step RL phase of RLoop (the block-diagonal regions) are comparable to those of vanilla RL, exhibiting a similar level of instability. However, the forgetting between iterations (the off-diagonal blocks) is substantially lower

for RLoop. This indicates that the RFT phase is highly effective at consolidating knowledge and serving as a stable anchor, preventing the long-term, catastrophic forgetting that plagues uninterrupted RL training.

Better Trajectory Diversity We next examine trajectory diversity by comparing the n-gram similarity of generated solutions, a metric inversely proportional to diversity. The process of estimating n-gram similarity is shown in Appendix A. Figure 5b shows that RLoop consistently maintains a lower average n-gram similarity than vanilla RL throughout the training process. Since lower similarity corresponds to higher diversity, this result demonstrates that RLoop promotes a more diverse set of generated solutions. This heightened diversity is a key factor contributing to RLoop’s superior generalization and, in particular, its significantly improved pass@k performance.

High Entropy Policy entropy is widely regarded as a proxy for exploration in RL (Wang et al., 2025a; Cui et al., 2025; Liu et al., 2025b). We therefore compare the token-level entropy of policies trained with RLoop and vanilla RL. As shown in Figure 5c, the entropy for both methods generally increases over time, and crucially, RLoop maintains a policy entropy comparable to that of vanilla RL. It suggests that RLoop’s benefits of reduced forgetting and increased diversity are achieved without sacrificing policy exploration.

5.5. RLoop Improves Training Stability

A well-documented challenge in prolonged RL fine-tuning of LLMs is training instability, often manifesting as gradient explosion and catastrophic training collapse (Yao et al., 2025; He & Lab, 2025; Ma et al., 2025b; Liu et al., 2025a). Our experiments with vanilla RL confirm this issue; we

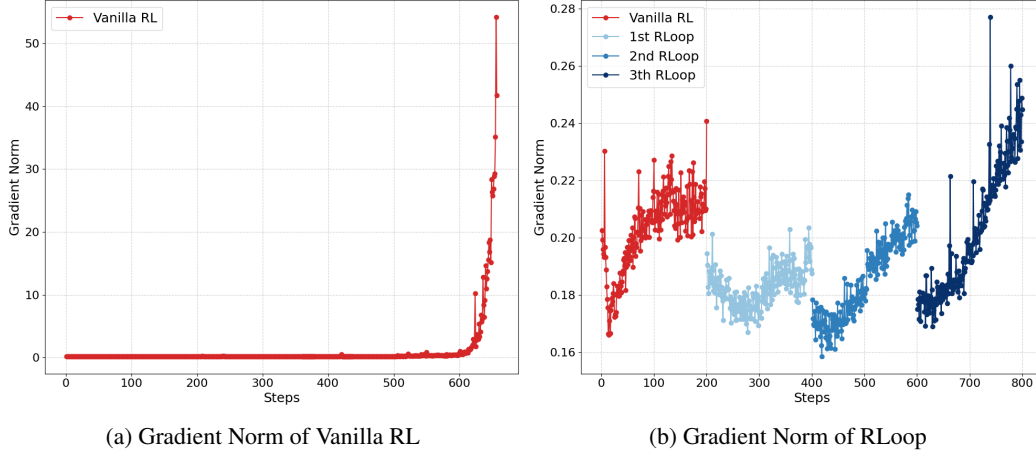


Figure 6: Comparison of gradient norm stability. (a) Vanilla RL exhibits explosive gradient growth after 600 steps, leading to training collapse. (b) RLoop maintains a stable, low gradient norm throughout its iterative training process.

observed a training collapse around 750 steps, preceded by an uncontrolled surge in the gradient norm. We find that our RLoop framework inherently mitigates this instability.

Figure 6 provides a clear illustration of this effect. The gradient norm for the vanilla RL baseline (Figure 6a) remains manageable for approximately 600 steps before experiencing explosive growth, quickly exceeding 50.0 and causing the training process to fail. In stark contrast, the gradient norm for RLoop (Figure 6b) remains remarkably stable and bounded. Even after three full iterations, corresponding to a total of 800 RL steps, the norm stays below 0.3, demonstrating the framework’s robustness against the instabilities that plague standard RL.

The source of this stability lies in RLoop’s cyclical “reset” mechanism. Instead of a single, prolonged optimization trajectory, RLoop performs a series of shorter, bounded RL explorations. Crucially, each exploration phase begins from a “refreshed” policy. This policy is not the potentially unstable endpoint of the previous RL phase, but rather a new model created by fine-tuning the original, stable base model on a small, high-quality dataset of expert trajectories. This periodic re-anchoring to a stable base prevents the policy from drifting into volatile regions of the parameter space. In contrast, the vanilla RL process, after 750 steps (equivalent to approximately 45 epochs over the training data), is likely over-optimizing on a fixed dataset, leading to the observed gradient explosion.

6. Conclusion

In this work, we identified “RL overfitting” as a critical challenge in RLVR, linking it to catastrophic forgetting and the under-utilization of inter-step policy diversity. To address this, we proposed RLoop, an iterative framework

that transforms RL’s instability into a strength by cyclically alternating between an RL exploration phase to generate diverse solutions and an RFT exploitation phase to consolidate knowledge. Our experiments demonstrate that RLoop significantly outperforms vanilla RL, particularly in pass@k metrics, by mitigating long-term forgetting, enhancing solution diversity, and ensuring training stability. By reframing training instability as a valuable source of exploration, our work offers a robust and principled solution to current RL challenges and paves the way for more stable, generalizable, and powerful reasoning models.

References

- Anonymous. Generalization of RLVR using causal reasoning as a testbed. In *Submitted to The Fourteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=DZjbL9BuHs>. under review.
- Chen, H., Feng, Y., Liu, Z., Yao, W., Prabhakar, A., Heinecke, S., Ho, R., Mui, P., Savarese, S., Xiong, C., and Wang, H. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding. *CoRR*, abs/2411.04282, 2024. doi: 10.48550/ARXIV.2411.04282. URL <https://doi.org/10.48550/arXiv.2411.04282>.
- Chen, L., Han, X., Shen, L., Bai, J., and Wong, K. Beyond two-stage training: Cooperative SFT and RL for LLM reasoning. *CoRR*, abs/2509.06948, 2025. doi: 10.48550/ARXIV.2509.06948. URL <https://doi.org/10.48550/arXiv.2509.06948>.
- Chu, T., Zhai, Y., Yang, J., Tong, S., Xie, S., Schuurmans, D., Le, Q. V., Levine, S., and Ma, Y. SFT

- memorizes, RL generalizes: A comparative study of foundation model post-training. *CoRR*, abs/2501.17161, 2025. doi: 10.48550/ARXIV.2501.17161. URL <https://doi.org/10.48550/arXiv.2501.17161>.
- Cui, G., Zhang, Y., Chen, J., Yuan, L., Wang, Z., Zuo, Y., Li, H., Fan, Y., Chen, H., Chen, W., Liu, Z., Peng, H., Bai, L., Ouyang, W., Cheng, Y., Zhou, B., and Ding, N. The entropy mechanism of reinforcement learning for reasoning language models. *CoRR*, abs/2505.22617, 2025. doi: 10.48550/ARXIV.2505.22617. URL <https://doi.org/10.48550/arXiv.2505.22617>.
- DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., and Li, S. S. Deepseek-rl: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL <https://doi.org/10.48550/arXiv.2501.12948>.
- Gao, B., Song, F., Yang, Z., Cai, Z., Miao, Y., Dong, Q., Li, L., Ma, C., Chen, L., Xu, R., Tang, Z., Wang, B., Zan, D., Quan, S., Zhang, G., Sha, L., Zhang, Y., Ren, X., Liu, T., and Chang, B. Omni-math: A universal olympiad level mathematic benchmark for large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL <https://openreview.net/forum?id=yaqPf0KAlN>.
- Guha, E. K., Marten, R., Keh, S., Raoof, N., Smyrnis, G., Bansal, H., Nezhurina, M., Mercat, J., Vu, T., Sprague, Z., Suvarna, A., Feuer, B., Chen, L., Khan, Z., Frankel, E., Grover, S., Choi, C., Muennighoff, N., Su, S., Zhao, W., Yang, J., Pimpalgaonkar, S., Sharma, K., Ji, C. C., Deng, Y., Pratt, S. M., Ramanujan, V., Saad-Falcon, J., Li, J., Dave, A., Albalak, A., Arora, K., Wulfe, B., Hegde, C., Durrett, G., Oh, S., Bansal, M., Gabriel, S., Grover, A., Chang, K., Shankar, V., Gokaslan, A., Merrill, M. A., Hashimoto, T., Choi, Y., Jitsev, J., Heckel, R., Sathiamoorthy, M., Dimakis, A. G., and Schmidt, L. Openthoughts: Data recipes for reasoning models. *CoRR*, abs/2506.04178, 2025. doi: 10.48550/ARXIV.2506.04178. URL <https://doi.org/10.48550/arXiv.2506.04178>.
- He, H. and Lab, T. M. Defeating nondeterminism in llm inference. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250910. <https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/>.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In Vanschoren, J. and Yeung, S. (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual, 2021*. URL <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html>.
- Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2022. URL <https://arxiv.org/abs/1312.6114>.
- Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V. V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., Wu, Y., Neyshabur, B., Gur-Ari, G., and Misra, V. Solving quantitative reasoning problems with language models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *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, 2022*. URL http://papers.nips.cc/paper_files/paper/2022/hash/18abbeef8cfe9203fdf9053c9c4fe191-Abstract-Conference.html.
- Li, J., Lu, H., Wen, K., Yang, Z., Gao, J., Lin, H., Wu, Y., and Zhang, J. Questa: Expanding reasoning capacity in llms via question augmentation. *CoRR*, abs/2507.13266, 2025a. doi: 10.48550/ARXIV.2507.13266. URL <https://doi.org/10.48550/arXiv.2507.13266>.
- Li, Z., Chen, C., Yang, T., Ding, T., Sun, R., Zhang, G., Huang, W., and Luo, Z.-Q. Knapsack rl: Unlocking exploration of llms via optimizing budget allocation, 2025b. URL <https://arxiv.org/abs/2509.25849>.
- Liang, X., Li, Z., Gong, Y., Shen, Y., Wu, Y. N., Guo, Z., and Chen, W. Beyond pass@1: Self-play with variational problem synthesis sustains RLVR.

- CoRR*, abs/2508.14029, 2025. doi: 10.48550/ARXIV.2508.14029. URL <https://doi.org/10.48550/arXiv.2508.14029>.
- Liu, J., Li, Y., Fu, Y., Wang, J., Liu, Q., and Shen, Y. When speed kills stability: Demystifying rl collapse from the inference-training mismatch, 2025a. URL <https://yingru.notion.site/When-Speed-Kills-Stability-Demystifying-RL-Inference-Training-Mismatch-27>.
- Liu, M., Diao, S., Lu, X., Hu, J., Dong, X., Choi, Y., Kautz, J., and Dong, Y. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *CoRR*, abs/2505.24864, 2025b. doi: 10.48550/ARXIV.2505.24864. URL <https://doi.org/10.48550/arXiv.2505.24864>.
- Ma, L., Liang, H., Qiang, M., Tang, L., Ma, X., Wong, Z. H., Niu, J., Shen, C., He, R., Cui, B., and Zhang, W. Learning what reinforcement learning can’t: Interleaved online fine-tuning for hardest questions. *CoRR*, abs/2506.07527, 2025a. doi: 10.48550/ARXIV.2506.07527. URL <https://doi.org/10.48550/arXiv.2506.07527>.
- Ma, W., Zhang, H., Zhao, L., Song, Y., Wang, Y., Sui, Z., and Luo, F. Stabilizing moe reinforcement learning by aligning training and inference routers, 2025b. URL <https://arxiv.org/abs/2510.11370>.
- Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *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*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.
- Prakash, J. and Buvanesh, A. What can you do when you have zero rewards during rl? <https://spiffy-airbus-472.notion.site/What-Can-You-Do-When-You-Have-Zero-Rewards-During-RL-250601939>, 2025. Notion Blog.
- Wang, S., Yu, L., Gao, C., Zheng, C., Liu, S., Lu, R., Dang, K., Chen, X., Yang, J., Zhang, Z., Liu, Y., Yang, A., Zhao, A., Yue, Y., Song, S., Yu, B., Huang, G., and Lin, J. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. *CoRR*, abs/2506.01939, 2025a. doi: 10.48550/ARXIV.2506.01939. URL <https://doi.org/10.48550/arXiv.2506.01939>.
- Wang, Y., Yang, Q., Zeng, Z., Ren, L., Liu, L., Peng, B., Cheng, H., He, X., Wang, K., Gao, J., Chen, W., Wang, S., Du, S. S., and Shen, Y. Reinforcement learning for reasoning in large language models with one training example. *CoRR*, abs/2504.20571, 2025b. doi: 10.48550/ARXIV.2504.20571. URL <https://doi.org/10.48550/arXiv.2504.20571>.
- Wang, Z., Zhou, F., Li, X., and Liu, P. Octothinker: Mid-training incentivizes reinforcement learning scaling. *CoRR*, abs/2506.20512, 2025c. doi: 10.48550/ARXIV.2506.20512. URL <https://doi.org/10.48550/arXiv.2506.20512>.
- Yan, J., Li, Y., Hu, Z., Wang, Z., Cui, G., Qu, X., Cheng, Y., and Zhang, Y. Learning to reason under off-policy guidance. *CoRR*, abs/2504.14945, 2025. doi: 10.48550/ARXIV.2504.14945. URL <https://doi.org/10.48550/arXiv.2504.14945>.
- Yao, F., Liu, L., Zhang, D., Dong, C., Shang, J., and Gao, J. Your efficient rl framework secretly brings you off-policy rl training, August 2025. URL <https://fengyao.notion.site/off-policy-rl>.
- Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Fan, T., Liu, G., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Dai, W., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W., Zhang, Y., Yan, L., Qiao, M., Wu, Y., and Wang, M. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025. doi: 10.48550/ARXIV.2503.14476. URL <https://doi.org/10.48550/arXiv.2503.14476>.
- Yuan, X., Chen, X., Yu, T., Shi, D., Jin, C., Lee, W., and Mitra, S. Mitigating forgetting between supervised and reinforcement learning yields stronger reasoners, 2025. URL <https://arxiv.org/abs/2510.04454>.
- Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Yue, Y., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *CoRR*, abs/2504.13837, 2025. doi: 10.48550/ARXIV.2504.13837. URL <https://doi.org/10.48550/arXiv.2504.13837>.
- Zhang, S., Liu, Q., Qin, G., Naumann, T., and Poon, H. Med-rlvr: Emerging medical reasoning from a 3b base model via reinforcement learning. *CoRR*, abs/2502.19655, 2025a. doi: 10.48550/ARXIV.2502.19655. URL <https://doi.org/10.48550/arXiv.2502.19655>.

Zhang, W., Xie, Y., Sun, Y., Chen, Y., Wang, G., Li, Y., Ding, B., and Zhou, J. On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting. *CoRR*, abs/2508.11408, 2025b. doi: 10.48550/ARXIV.2508.11408. URL <https://doi.org/10.48550/arXiv.2508.11408>.

Zhong, H., Yin, Y., Zhang, S., Xu, X., Liu, Y., Zuo, Y., Liu, Z., Liu, B., Zheng, S., Guo, H., Wang, L., Hong, M., and Wang, Z. Brite: Bootstrapping reinforced thinking process to enhance language model reasoning. *CoRR*, abs/2501.18858, 2025. doi: 10.48550/ARXIV.2501.18858. URL <https://doi.org/10.48550/arXiv.2501.18858>.

A. Detailed Calculation of Trajectory Similarity

This appendix provides the detailed methodology for computing the n-gram-based trajectory similarity score, as referenced in Section 3.

To quantify the lexical consistency between different reasoning steps, we compute an n-gram-based similarity score. This metric evaluates the textual overlap between the set of solutions generated at two distinct steps, denoted as step i and step j . For each input prompt, our model generates N unique solutions. Consequently, for a single prompt, we have two sets of solutions: $S_i = \{s_{i,1}, s_{i,2}, \dots, s_{i,N}\}$ for step i , and $S_j = \{s_{j,1}, s_{j,2}, \dots, s_{j,N}\}$ for step j .

The calculation proceeds in two main stages:

1. Pairwise Solution Similarity via Jaccard Index First, we define the similarity between any pair of individual solutions, one from each step ($s_{i,a} \in S_i$ and $s_{j,b} \in S_j$). This is based on their shared n-grams (we use bigrams, $n = 2$, in our implementation). For each solution s , we generate a set of its n-grams, denoted as $G_n(s)$. The similarity between two solutions is then calculated using the Jaccard similarity coefficient:

$$J(s_{i,a}, s_{j,b}) = \frac{|G_n(s_{i,a}) \cap G_n(s_{j,b})|}{|G_n(s_{i,a}) \cup G_n(s_{j,b})|}$$

This value measures the proportion of shared n-grams relative to the total unique n-grams across both solutions.

2. Overall Step Similarity via Averaging To obtain a single similarity score for a given prompt, we compute the Jaccard similarity for all $N \times N$ possible pairs of solutions between step i and step j . The final similarity for that prompt, $\text{Sim}_{\text{prompt}}(S_i, S_j)$, is the arithmetic mean of these N^2 pairwise scores:

$$\text{Sim}_{\text{prompt}}(S_i, S_j) = \frac{1}{N^2} \sum_{a=1}^N \sum_{b=1}^N J(s_{i,a}, s_{j,b})$$

This aggregation provides a robust measure of the overall similarity between the two sets of solutions. The final reported similarity between step i and step j is the average of these $\text{Sim}_{\text{prompt}}$ scores across all prompts in the dataset.