Boosting Off-Policy RLVR with Data-Centric Replay

Yuxiao He[1,2] **Ziqi Wang**[1,2] **Junge Zhang**[1,2,*]

[1]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, 101408, China.

[2]National Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China.

{heyuxiao2024,wangziqi2024}@ia.ac.cn [*]Corresponding author: jgzhang@nlpr.ia.ac.cn

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) has become an important approach for improving the reasoning capabilities of large language models (LLMs). On-policy training remains the dominant RLVR paradigm, but its reliance on fresh rollouts leads to poor data efficiency. To address this bottleneck, off-policy paradigm has attracted increasing attention by reusing historical trajectories through replay buffers. Despite its efficiency advantages, off-policy training often struggles to match the performance of on-policy training. Our analysis reveals that this limitation primarily stems from data staleness introduced by replay buffers: to maintain training stability, mainstream algorithms discard a large fraction of high-staleness optimization signals, leading to substantial performance degradation. To address this issue, we propose **STAR** (**St**aleness-**A**ware **R**eplay), a data-centric replay framework decoupled from specific optimization algorithms. STAR improves off-policy RLVR by constructing low-staleness and high-quality training data. Experiments on a range of mathematical reasoning and code generation tasks show that STAR consistently improves off-policy RLVR with up to 19% relative performance gains, and enables it to surpass the on-policy baseline using only around 13% of the rollout data while achieving $5\times$–$6\times$ wall-clock speedup.

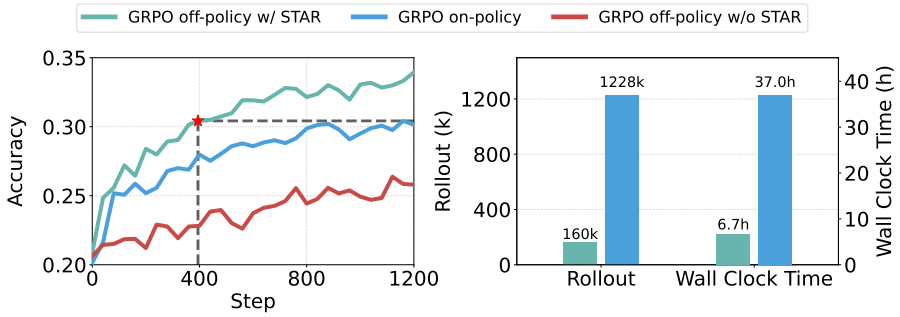


Figure 1: STAR improves the performance and efficiency of off-policy RLVR. We use GRPO as the backbone algorithm and Qwen-2.5-Math-7B as the base model. Rollout cost is measured by the number of generated trajectories. **Left**: STAR substantially improves off-policy training, reaching the converged performance of on-policy RLVR within fewer than $1/3$ of the training steps. **Right**: STAR reaches the converged performance of on-policy RLVR using only 13.0% of the rollout data, while achieving a $5.5\times$ wall-clock speedup.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Reinforcement Learning with Verifiable Rewards (RLVR) has become a pivotal methodology for enabling Large Language Models (LLMs) to solve complex problems (Jaech et al., 2024; Guo et al., 2025). RLVR empowers LLMs to attain advanced reasoning capabilities through iterative environmental interactions and feedback-driven learning. Nevertheless, the inherent trial-and-error nature of RL entails extensive trajectory rollouts, resulting in severe data inefficiency. This challenge is further exacerbated in the prevalent on-policy paradigm. In this setting, trajectories are discarded after a single gradient update, which precludes effective data reuse. To mitigate these issues, off-policy RL (Watkins, 1989; Watkins & Dayan, 1992; Mroueh et al., 2025) has gained increasing attention. This paradigm utilizes a replay buffer to reuse generated trajectories, thereby improving data efficiency (Mnih et al., 2015; Horgan et al., 2018).

While efficient, the off-policy setting has been widely observed to compromise the performance of mainstream RLVR algorithms. This performance degradation stem from the staleness of data within the replay buffer (Rahimi-Kalahroudi et al., 2023; Wan et al., 2026). To ensure training stability, algorithms employing trust-region mechanisms, such as GRPO (Shao et al., 2024), are compelled to clip a large portion of these high-staleness data. This excessive clipping significantly reduces the volume of effective learning signals, leading to a tangible performance degradation.

To address this challenge, recent research has pivoted toward algorithmic adaptations for off-policy RLVR (Xi et al., 2025; Yang et al., 2025; Zheng et al., 2025, 2026). The predominant strategy involves refining the trust region mechanism. These approaches attempt to employ more relaxed trust regions without compromising training stability. By doing so, they seek to incorporate a higher volume of optimization signals to recover lost performance.

While these improvements partially mitigate performance degradation, they still fail to close the performance gap with on-policy RLVR in practice. Our analysis of training dynamics reveals a critical phenomenon. Despite employing more relaxed trust regions, the clipping fraction in off-policy training remains significantly higher than the on-policy baseline. Attempting to further relax the trust region to retain more signals leads directly to training collapse. This suggests that existing methods have reached the limits of algorithmic optimization. We contend that the root cause of this bottleneck is not the trust region mechanism itself. Instead, the data staleness introduced by the replay buffer acts as a fundamental constraint on the performance ceiling of off-policy RLVR. We further characterize this mechanism by showing that the Total Variation (TV) divergence between the behavior and target policies controls an upper bound on the clipping-induced discard loss, which is in turn strongly associated with the observed performance degradation.

Driven by these findings, surpassing the performance ceiling of off-policy training necessitates a paradigm shift from purely algorithmic refinement toward the construction of low-staleness training data. In off-policy RL, the construction of training data is determined by the buffer sampling strategy. Yet, recent works have largely overlooked this dimension, typically defaulting to uniform random sampling. To address this gap, we propose **STAR** (**St**aleness-**A**ware **R**eplay), a data-centric framework decoupled from specific optimization algorithms. STAR constructs low-staleness and high-quality training data by performing weighted resampling based on both data staleness and gradient contribution. Furthermore, STAR incorporates a hard-sample recurrence mechanism to facilitate adaptive curriculum learning, thereby effectively mitigating the entropy collapse commonly observed in off-policy training processes.

Overall, our contributions are summarized as follows:

- We systematically analyze the data staleness introduced by replay buffers in off-policy RLVR and establish a theoretical link between staleness and optimization signal loss, revealing a key mechanism behind off-policy performance degradation.
- We propose STAR, a data-centric replay framework decoupled from specific optimization algorithms. By constructing low-staleness and high-quality training data, STAR significantly improves both the performance and efficiency of off-policy RLVR.
- Experiments on mathematical reasoning and code generation tasks show that STAR consistently improves off-policy RLVR, achieving up to 19% relative performance gains. Moreover, STAR enables off-policy RLVR to surpass the on-policy baseline using only around 13% of the rollout data while achieving $5\times$–$6\times$ wall-clock speedup.

2 Related Work

Algorithmic Refinements for Off-Policy RLVR Recent studies on off-policy RLVR primarily focus on algorithm-level refinements. These methods generally seek to improve performance by designing more fine-grained trust regions that retain more optimization signals. DCPO (Yang et al., 2025) follows the idea of DAPO (Yu et al., 2025) and introduces dynamic clipping ranges for low-probability tokens to retain more optimization signals. GSPO (Zheng et al., 2025) constructs the trust region using sequence-level importance sampling ratios instead of token-level ratios. BAPO (Xi et al., 2025) adopts a more relaxed trust region for positive samples to balance positive and negative optimization signals. M2PO (Zheng et al., 2026) controls the batch-level second moment of importance sampling ratios, discarding only extreme tokens during optimization. Although these methods partially mitigate the performance degradation of off-policy RLVR, our empirical results show that they still struggle to surpass the on-policy baseline in practice, which limits the applicability of off-policy RLVR.

Data-Centric Refinements for Off-Policy RLVR Several recent works attempt to improve off-policy RLVR from a data-centric perspective. Although some are referred to as *replay* methods, they do not reuse rollout data through replay buffers to reduce rollout cost, but instead introduce additional data for performance gains. For example, RePO (Li et al., 2025) incorporates extra off-policy samples during policy optimization, increasing training time compared with on-policy training. RLEP (Zhang et al., 2025) and LUFFY (Yan et al., 2025) further rely on samples from stronger external models, introducing additional generation overhead. While these methods improve performance, they sacrifice the data-efficiency advantage of off-policy training. Classical replay methods, such as importance resampling (IR), provide a general way to reweight off-policy data according to policy discrepancy (Schlegel et al., 2019; Xie et al., 2023). However, these methods are not tailored to modern RLVR objectives with clipping trust region, where prioritizing high-importance samples can further increase clipping and reduce optimization signals (detailed in Appendix D.3).

3 Preliminaries

3.1 Off-policy RL with Replay Buffer

In the off-policy reinforcement learning (RL) framework, the learning process is decoupled into two asynchronous phases: trajectory rollout and policy optimization (Lin, 1992; Mnih et al., 2015; Nair et al., 2015; Horgan et al., 2018). This architecture enables the reuse of historical interaction data, thereby significantly enhancing sample efficiency.

Rollout Phase: A behavior policy μ, typically a historical version of the target policy, is responsible for interacting with the environment. In the context of LLMs, for a prompt x sampled from the dataset $\mathcal{D}$, the behavior policy generates a corresponding response $y \sim \mu(\cdot|x)$. Subsequently, a feedback signal $r = R(x, y)$ is provided by the environment. The collected transitions are archived in a replay buffer $\mathcal{B} = \{(x_i, y_i, r_i)\}_{i=1}^{N}$.

Optimization Phase: Instead of relying on real-time interactions, the training process periodically retrieves data from the replay buffer $\mathcal{B}$. In each iteration, a batch of size M is sampled from the buffer distribution:

$$\mathcal{D}_{batch} = \{(x_j, y_j, r_j)\}_{j=1}^{M} \sim \mathcal{B}. \tag{1}$$

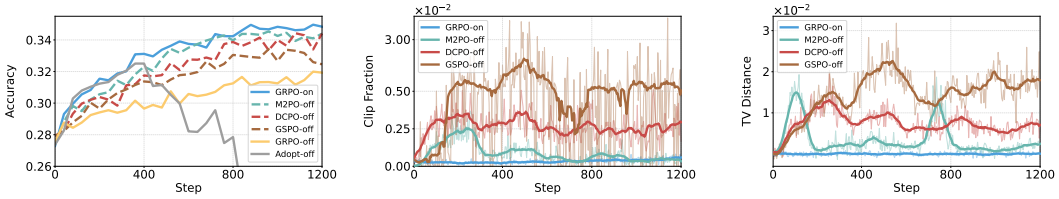
The target policy π_θ is then updated by optimizing a specific objective function $\mathcal{J}(\theta)$ using the sampled batch:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{J}(\pi_\theta, \mu, \mathcal{D}_{batch}). \tag{2}$$

3.2 Trust Region in Policy Gradient Methods

Modern RLVR algorithms predominantly utilize the first-order trust region (Schulman et al., 2015) approximation introduced by PPO (Schulman et al., 2017), which employs a clipped surrogate objective to ensure training stability:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(\rho \hat{A}_t, \text{clip}(\rho, 1-\epsilon, 1+\epsilon)\hat{A}_t) \right], \tag{3}$$



(a) Accuracy Comparison (b) Clipping Fraction (c) TV Distance

Figure 2: Analysis of training dynamics and data staleness in off-policy RLVR. Mainstream off-policy RLVR algorithms partially mitigate performance degradation but still fail to surpass the on-policy baseline, while exhibiting higher data staleness and clipping fractions during training.

where $\rho = \frac{\pi_\theta(y|x)}{\mu(y|x)}$ denotes the importance sampling (IS) ratio (Precup et al., 2000, 2001), with π_θ and μ representing the target policy and the behavior policy, respectively. Mainstream RLVR algorithms, such as GRPO (Shao et al., 2024), consistently adopt this PPO-style clipping mechanism to maintain training stability:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G}\min\left(\frac{\pi_\theta(y_i|x)}{\mu(y_i|x)}A_i, \text{clip}\left(\frac{\pi_\theta(y_i|x)}{\mu(y_i|x)}, 1-\epsilon, 1+\epsilon\right)A_i\right)\right]. \tag{4}$$

Throughout this work, the term trust region specifically refers to this PPO-style clipping mechanism.

4 Motivation

In this section, we first empirically identify the performance bottleneck of off-policy RLVR. We then provide a theoretical analysis characterizing how data staleness leads to optimization signal loss, and complement this analysis by examining the relationship between this loss and the observed accuracy gap.

Off-policy RLVR improves data efficiency by reusing trajectories through replay buffers, but the resulting data staleness often leads to performance degradation. Recent algorithmic refinements, including DCPO (Yang et al., 2025), M2PO (Zheng et al., 2026), and GSPO (Zheng et al., 2025), can partially mitigate this degradation. However, as shown in Figure 2a, they still struggle to match the on-policy GRPO (Shao et al., 2024) baseline. Examination of the training dynamics shows that these methods consistently maintain a significantly higher clipping fraction than the on-policy baseline (Figure 2b).

To further investigate this limitation, we adopt the clipping fraction of the on-policy baseline at each training step by clipping samples in descending order of staleness. Although this initially improves performance, it eventually induces training instability and collapse (see *Adopt-off* in Figure 2a). This indicates that retaining more optimization signals through algorithm-level relaxation alone is not a viable solution, motivating us to examine the data side of the problem. Specifically, we quantify data staleness using the TV distance (detailed in Appendix F.3) between the behavior and target policies. As shown in Figure 2c, off-policy training exhibits substantially higher data staleness than the on-policy baseline. This observation strongly suggests that excessive data staleness may be a key driver of the high clipping fraction, causing a large portion of optimization signals to be discarded.

Motivated by this empirical evidence, we next theoretically characterize how data staleness induces optimization signal loss, thereby explaining its impact on the performance of off-policy RLVR. In the t-th update step, let π_θ denote the target policy and μ represent the behavior policy of the training batch. Given the IS ratio $\rho = \frac{\pi_\theta(y|x)}{\mu(y|x)}$ and the advantage $A(x,y)$, the ideal surrogate objective is:

$$L_t^{\text{ideal}}(\pi_\theta;\mu) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[A(x,y)\right] = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)}\left[\rho(x,y)A(x,y)\right], \tag{5}$$

In RLVR, to maintain training stability under policy shifts, mainstream algorithms utilize a clipping trust region $[c_t^{\text{low}}, c_t^{\text{high}}]$, where $0 < c_t^{\text{low}} < 1 < c_t^{\text{high}}$. This truncation leads to the clipped surrogate objective:

$$L_t^{\text{clip}}(\pi_\theta;\mu) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)}\left[\min\left(\rho A(x,y), \text{clip}\left(\rho, c_t^{\text{low}}, c_t^{\text{high}}\right)A(x,y)\right)\right], \tag{6}$$

Since the clipping operation zeroes out the gradients of samples that trigger clipping, these samples provide no effective gradient contribution to the policy parameters θ. Therefore, from the perspective of policy gradients, L_t^{clip} is equivalent to the following masked objective (detailed in Appendix D.1):

$$L_t^{\text{mask}}(\pi_\theta; \mu) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)} \left[\rho A(x,y) m_t(x,y)\right], \tag{7}$$

where

$$m_t(x,y) = \mathbf{1}\left\{A(x,y) \geq 0,\, \rho \leq c_t^{\text{high}}\right\} + \mathbf{1}\left\{A(x,y) < 0,\, \rho \geq c_t^{\text{low}}\right\}. \tag{8}$$

Accordingly, we define the discard loss $\Delta_t^{\text{disc}}(\pi_\theta, \mu)$ as the discrepancy between the ideal and actual surrogates:

$$\Delta_t^{\text{disc}}(\pi_\theta, \mu) := L_t^{\text{ideal}}(\pi_\theta; \mu) - L_t^{\text{mask}}(\pi_\theta; \mu). \tag{9}$$

This metric Δ_t^{disc} quantifies the volume of effective optimization signals that are discarded due to the staleness of the training data. To establish the practical significance of this metric, we investigate the relationship between the discard loss Δ_t^{disc} and the actual performance degradation Δ_t^{acc}, defined as the accuracy gap relative to the on-policy baseline.

While a direct theoretical mapping from discarded optimization signals to accuracy degradation is non-trivial, we first examine their empirical association during off-policy GSPO training. As shown in Figure 3, the discard loss $\left|\Delta_t^{\text{disc}}\right|$ is strongly correlated with the accuracy gap Δ_t^{acc}, with $r = 0.985$ and $R^2 = 0.970$.

We further conduct a more rigorous validation across GSPO, DCPO, and M2PO. To account for shared training-time trends, we compute both partial correlations controlling for training step and correlations between first differences. As shown in Table 1, the partial correlations range from 0.912 to 0.961, while the first-difference correlations range from 0.728 to 0.751, with all 95% confidence intervals excluding zero. These results show that the association is not solely induced by a shared temporal trend and generalizes across different off-policy algorithms.

Together, these results provide robust empirical evidence that a larger discard loss is associated with greater accuracy degradation.

Table 1: Robustness analysis of the association between discard loss and accuracy degradation.

Algorithm	**Partial r [95% CI]**	**First-difference r [95% CI]**
GSPO	0.961 [0.897, 0.974]	0.736 [0.502, 0.856]
DCPO	0.953 [0.891, 0.976]	0.751 [0.519, 0.873]
M2PO	0.912 [0.838, 0.958]	0.728 [0.487, 0.862]

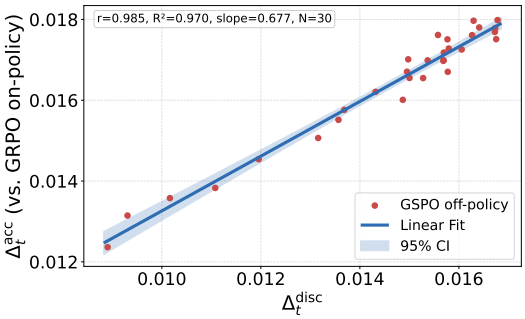


Figure 3: Empirical association between the discard loss $\left|\Delta_t^{\text{disc}}\right|$ and the accuracy gap Δ_t^{acc} during off-policy training with GSPO.

Building upon this evidence, Theorem 1 establishes that the discard loss is upper-bounded by the TV distance between the target and behavior policies.

Theorem 1 (TV-Bound on Optimization Signal Loss)**.** *For a policy gradient algorithm employing the clipped trust-region mechanism at the t-th update step, let π_θ be the target policy and μ be the behavior policy of the training batch. Given the clipping interval $[c_t^{\text{low}}, c_t^{\text{high}}]$ where $0 < c_t^{\text{low}} < 1 < c_t^{\text{high}}$, the discard loss Δ_t^{disc} satisfies:*

$$|\Delta_t^{\text{disc}}| \leq 2\epsilon_A \kappa(c_t^{\text{low}}, c_t^{\text{high}}) D_{TV}(\pi_\theta, \mu) \tag{10}$$

where ϵ_A denotes the upper bound of the advantage signal, $D_{TV}(\pi_\theta, \mu)$ is the Total Variation distance between the target policy and the behavior policy over the training data, and κ is the sensitivity coefficient defined by the clipping boundaries:

$$\kappa(c_t^{\text{low}}, c_t^{\text{high}}) = \max\left\{\frac{c_t^{\text{high}}}{c_t^{\text{high}} - 1}, \frac{c_t^{\text{low}}}{1 - c_t^{\text{low}}}\right\}. \tag{11}$$

In summary, our analysis yields the following two-stage characterization:

$$D_{\text{TV}}(\pi_\theta, \mu) \xrightarrow{\text{theoretical upper-bound control}} \left|\Delta_t^{\text{disc}}\right| \xleftrightarrow{\text{empirical association}} \Delta_t^{\text{acc}}. \tag{12}$$

Theorem 1 provides theoretical control of the discard loss through the TV distance, while our empirical analysis across multiple backbone algorithms supports a robust association between discard loss and accuracy degradation. Together, these results position TV distance as an actionable quantity for improving off-policy RLVR and motivate the construction of low-TV training data.

5 Method

In this section, we propose **STAR** (**St**aleness-**A**ware **R**eplay), a data-centric framework for improving off-policy RLVR. STAR operates on both phases (see Section 3.1) of off-policy RL. During policy optimization, it performs weighted resampling based on data staleness and sample quality, constructing low-staleness and high-quality training batches. During trajectory rollout, it introduces Hard-Sample Recurrence (HSR), which replays hard prompts near the model's capability boundary to mitigate entropy collapse and maintain exploration. We detail these two components below, with the complete algorithm provided in Appendix E.

5.1 Weighted Resampling

Based on the analysis in Section 4, reducing policy divergence in replayed data is a key ingredient for overcoming the performance bottleneck of off-policy RLVR. To this end, we investigate the TV divergence at the granularity of individual samples. It can be proved that the contribution of a single sample (x, y) to the total distance $D_{TV}(\pi_\theta, \mu)$ is given by $|\rho - 1|$, where $\rho = \frac{\pi_\theta(y|x)}{\mu(y|x)}$ represents the IS ratio between the target policy π_θ and the behavior policy μ. Detailed proofs are provided in the Appendix D.2. This suggests that to construct low-TV training data, samples with smaller $|\rho - 1|$ should be prioritized when sampling from the replay buffer.

Beyond reducing staleness, we further improve training efficiency by prioritizing high-quality samples. We measure sample quality using the second moment of the sample gradient: a larger gradient second moment indicates that a sample contributes more strongly to the expected parameter update, and is therefore more informative for policy optimization.

In practice, computing the gradient for each sample is computationally prohibitive, necessitating an efficient proxy. We demonstrate in Theorem 2 that the sample gradient second moment is positively correlated with the variance of sample accuracy.

Theorem 2 (Approximation of Gradient Second Moment)**.** *For a prompt x, under binary verifiable rewards and group-relative advantages, the per-prompt stochastic gradient second moment satisfies:*

$$\mathbb{E}\|\hat{g}_x\|^2 \approx \frac{G-1}{G^2} a(x)(1-a(x))C_x \tag{13}$$

where G is the group size and $a(x)$ is the group accuracy for prompt x. Here, $C_x = \mathrm{tr}(F_x)$, and

$$F_x = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\nabla_\theta \log \pi_\theta(y|x) \nabla_\theta \log \pi_\theta(y|x)^\top\right] \tag{14}$$

is the Fisher information matrix conditioned on prompt x.

Theorem 2 indicates that $a(x)(1-a(x))$ serves as an effective metric for assessing sample quality.

Building on the analysis of staleness and quality, we derive the final unified metric for weighted resampling. The priority score $\mathcal{P}_i$ for each sample i in the replay buffer is defined as:

$$\mathcal{P}_i = \frac{a_i(1-a_i)}{|\bar{\rho}_i - 1| + \epsilon} \tag{15}$$

where ϵ is a small constant for numerical stability and $\bar{\rho}(y|x)$ is the length-normalized IS ratio:

$$\bar{\rho}(y|x) = \exp\left(\frac{1}{L}\sum_{t=1}^{L} \log \frac{\pi_\theta(y_t|x, y_{<t})}{\mu(y_t|x, y_{<t})}\right) = \rho(y|x)^{1/L} \tag{16}$$

We apply length normalization to eliminate the bias toward response lengths, as the IS ratio is sensitive to sequence length due to its cumulative nature. The final sampling probability w_i is then obtained by normalizing these priority scores across all samples in the replay buffer $\mathcal{B}$:

$$w_i = \frac{\mathcal{P}_i}{\sum_{j \in \mathcal{B}} \mathcal{P}_j} \tag{17}$$

Since the pass rate a_i is already computed and stored during the rollout phase, the computational overhead for determining w_i primarily stems from computing $\bar{\rho}$ across the replay buffer. This additional cost is negligible compared to the rollout reduction enabled by the off-policy paradigm.

5.2 Hard-Sample Recurrence

While replay buffers improve sample efficiency, repeatedly optimizing on historical data can reinforce existing successful behaviors and reduce policy diversity, causing the policy to converge prematurely to strategies that solve only a limited subset of prompts. To mitigate this issue, STAR introduces Hard-Sample Recurrence (HSR). HSR maintains a pool of prompts whose rollout accuracy falls below a predefined threshold τ, and replays them in subsequent rollout phases by replacing a fraction λ of the original prompts. By repeatedly exposing the policy to prompts that are not yet reliably solved, HSR encourages stochastic rollouts to explore alternative strategies; once a successful trajectory is sampled, the reward signal reinforces it through policy updates. In this way, HSR helps mitigate entropy collapse, preserve exploration, and improve the performance ceiling of off-policy RLVR.

6 Experiments

6.1 Setup

Datasets and Models. We conduct our experiments across two domains: mathematical reasoning and code generation. In the math domain, we utilize DeepScaler (Luo et al., 2025) dataset to train Qwen-2.5-Math-7B (Yang et al., 2024). The evaluation covers a comprehensive set of benchmarks, including AIME 2024, AIME 2025, AMC 2023, Minerva Math (Lewkowycz et al., 2022), Olympiad Bench (He et al., 2024), and Math500 (Lightman et al., 2024). To ensure robust evaluation results, we adopt the mean@16 metric. In the code generation domain, we use Llama-3.2-Instruct-3B (Meta, 2024) as the base model, training on a mixed dataset consisting of Taco (Li et al., 2023), Codeforces (Penedo et al., 2025), CodeContests (Li et al., 2022), and APPS (Hendrycks et al., 2021). Evaluation is conducted on HumanEval (Chen et al., 2021), LeetCode (Xia et al., 2025), and LiveCodeBench (Jain et al., 2024), using the pass@1 metric. For all tasks, we perform evaluations every 40 training steps using three independent random seeds. For each seed, performance is first averaged over the final three evaluation checkpoints after convergence. We then report the mean across the three seeds for each benchmark, with the Avg. column additionally reporting the standard deviation across seeds.

Training. We implement a custom off-policy asynchronous training framework based on VERL (Sheng et al., 2024). Each trial runs 1,200 steps. For STAR, we set the hard-sample accuracy threshold to $\tau = 0.1$ and the hard-pool replacement ratio to $\lambda = 0.05$. For code generation tasks, all model outputs are verified using a locally deployed Sandbox Fusion environment. Further details are provided in the Appendix F.1 and Appendix F.2.

Baselines. We evaluate STAR on a series of mainstream RLVR algorithms: GRPO (Shao et al., 2024), GSPO (Zheng et al., 2025), M2PO (Zheng et al., 2026), DCPO (Yang et al., 2025), and BAPO (Xi et al., 2025). We conduct comparative experiments by evaluating the performance of these baselines with and without the integration of STAR in off-policy setting. To further demonstrate the efficiency advantages of off-policy training with STAR, we use on-policy GRPO as a baseline and measure the wall-clock time speedup to reach the peak performance of the on-policy baseline. For GRPO, we set the clipping threshold to $\epsilon = 0.2$, while for the other methods, we adopt the optimal hyperparameter configurations reported in their respective works.

6.2 Main Result

Table 2 and Figure 1 present the main results on mathematical reasoning. Due to space limitations, the experimental results on code generation are provided in Appendix A.1. Overall, STAR consistently improves the average performance of all five off-policy backbones, including GRPO, GSPO, M2PO,

Table 2: Performance comparison on mathematical reasoning tasks. GRPO, GSPO, M2PO, DCPO and BAPO are used as backbone algorithms, with on-policy GRPO as the baseline. Speedup denotes the wall-clock acceleration to reach the converged performance of the on-policy baseline, and – indicates failure to reach it. The best performance on each benchmark is highlighted in **bold**.

Method	AIME24	AIME25	AMC23	Minerva	Olympiad	Math	Avg.	Speedup
MODEL: QWEN-2.5-MATH-7B								
ON-POLICY								
GRPO	28.3	16.8	60.7	38.4	41.9	81.1	44.5 $\pm$0.4	1.0×
OFF-POLICY								
GRPO	26.4	11.2	55.3	30.2	38.7	74.6	39.4 $\pm$0.4	–
+STAR	31.6	19.4	63.1	40.7	45.3	83.6	47.2 $\pm$0.5 (+7.8)	**5.5**×
GSPO	27.0	13.1	58.3	35.2	40.3	76.2	41.6 $\pm$1.1	–
+STAR	31.4	16.0	**64.2**	39.2	43.9	81.3	46.0 $\pm$0.8 (+4.4)	4.9×
M2PO	29.0	16.6	58.2	38.2	41.7	80.9	44.1 $\pm$0.7	–
+STAR	**32.8**	**20.2**	61.7	**40.8**	**45.4**	84.6	**47.5** $\pm$0.6 (+3.4)	5.2×
DCPO	28.6	16.4	58.6	37.1	42.0	80.4	43.8 $\pm$0.5	–
+STAR	32.2	19.8	62.3	39.4	**45.4**	**85.0**	47.4 $\pm$0.6 (+3.6)	5.0×
BAPO	28.4	16.3	58.0	37.6	41.5	79.8	43.6 $\pm$0.8	–
+STAR	32.1	20.0	61.8	40.1	45.2	84.2	47.2 $\pm$0.5 (+3.6)	5.0×

DCPO, and BAPO, with gains ranging from 3.4 to 7.8 points. This consistency demonstrates that STAR is not tied to a specific policy optimization objective and can complement different algorithmic designs. Across all five comparisons, the performance gains substantially exceed the reported standard deviations, which range from 0.4 to 1.1 points. This provides additional evidence that the improvements are robust to random variation.

STAR not only recovers the performance degradation associated with off-policy training but also enables all five backbones to outperform the on-policy GRPO baseline. Even when combined with vanilla GRPO, which is not specifically designed for off-policy optimization, STAR achieves an average score of 47.2 ± 0.5, compared with 44.5 ± 0.4 for on-policy GRPO. STAR also improves the stronger off-policy baselines: M2PO increases from 44.1 ± 0.7 to 47.5 ± 0.6, achieving the highest average score, while DCPO reaches a comparable 47.4 ± 0.6. Without STAR, all five off-policy methods underperform the on-policy baseline in average accuracy. Together, these results support the effectiveness of constructing low-staleness and high-quality training batches for improving replay-based policy optimization.

In addition to its performance gains, STAR improves data efficiency and wall-clock training efficiency. As shown in Figure 1, GRPO+STAR reaches the converged performance of on-policy GRPO using fewer than $1/3$ of the training steps and only 13.0% of the rollout data. The corresponding end-to-end training time decreases from 37.0 hours to 6.7 hours, yielding a $5.5\times$ speedup. These results highlight the potential of data-centric replay to combine improved reasoning performance with substantially reduced rollout requirements and training time.

6.3 Analysis of Training Dynamics

STAR constructs low-staleness and high-quality training data through weighted resampling. As shown in Figure 4 (middle), vanilla off-policy GRPO exhibits a much larger TV distance than the on-policy baseline, indicating severe data staleness induced by replay buffers. In contrast, STAR substantially reduces the TV distance between the behavior policy and the target policy over the sampled training data. According to Theorem 1, a lower TV distance tightens the upper bound on clipping-induced discard loss, thereby preserving more effective optimization signals in off-policy RLVR. The analysis of clipping fraction (Figure 4 (left)) further supports this observation: STAR consistently decreases the clipping fraction during training, bringing it close to the level of the on-policy baseline. This suggests that STAR preserves more valid optimization signals by sampling less stale data. Moreover, weighted resampling not only introduces more effective optimization signals, but also improves their

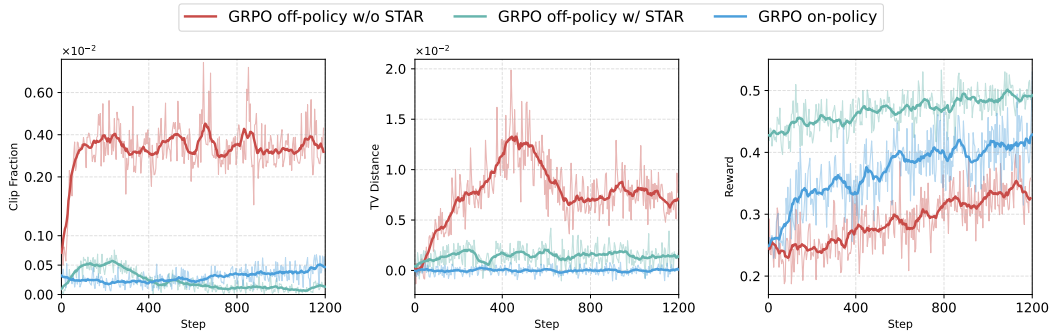


Figure 4: Training dynamics of STAR. **Left**: clipping fraction; **Middle**: TV distance; **Right**: training reward. STAR reduces the clipping fraction and TV distance while improving reward during off-policy training.

Table 3: Comparison with existing data-centric methods. STAR achieves competitive performance while providing significant efficiency advantages.

Method	Math	Code	Avg.	Rollouts	Speedup	External Model
On-Policy	44.5	34.1	39.3	1.0×	1.0×	No
LUFFY	**49.4**	**38.7**	**44.1** (+4.8)	1.1×	1.2×	Yes
RLEP	48.6	36.0	42.3 (+3.0)	1.2×	1.3×	Yes
RePO	45.1	34.2	39.7 (+0.4)	0.9×	1.5×	No
STAR (ours)	47.2	36.3	41.8 (+2.5)	**0.2**×	**5.5**×	No

quality. As shown in Figure 4 (right), STAR achieves higher reward than the on-policy baseline throughout training, indicating that it preferentially selects more informative samples that better support policy improvement. Together, these results show that STAR improves off-policy training by simultaneously reducing staleness-induced signal loss and increasing the informativeness of sampled data.

6.4 Comparison with Other Data-Centric Methods

We further compare STAR with existing data-centric methods, including LUFFY (Yan et al., 2025), RLEP (Zhang et al., 2025), and RePO (Li et al., 2025). All methods use GRPO as the backbone algorithm. For efficiency comparison, we report the relative rollout cost, i.e., the number of generated trajectories, and the wall-clock speedup required for each method to reach the on-policy baseline performance. As discussed in Section 2, although these methods are referred to as *replay* methods, they do not reuse rollout data through replay buffers to reduce rollout cost, but instead introduce additional data for performance gains. As shown in Table 3, LUFFY and RLEP achieve competitive or even higher performance than STAR, but rely on stronger external models and additional rollouts, resulting in much lower data efficiency and wall-clock speedup. RePO does not use an external model, but underperforms STAR in both performance and efficiency. These results suggest that improving performance by sacrificing data efficiency and asynchronous training undermines the motivation for adopting off-policy RLVR.

7 Conclusion

In this paper, we identify replay-buffer-induced data staleness as a key factor constraining the performance ceiling of off-policy RLVR. To address this issue, we propose STAR, a data-centric replay framework that constructs low-staleness and high-quality training data. Experiments on various tasks show that STAR consistently improves both the performance and efficiency of off-policy RLVR, offering an effective path toward fully realizing its potential.

8 Limitations

Although STAR improves performance and training efficiency in highly off-policy RLVR settings, several limitations remain.

First, our theoretical analysis focuses on policy-gradient methods with PPO-style clipping. Although this clipping mechanism is widely adopted in current RLVR algorithms, the theoretical characterization may not directly apply to optimization objectives with substantially different trust-region designs or off-policy correction mechanisms.

Second, our theoretical results rely on several simplifying assumptions, including bounded advantage signals and binary verifiable rewards. These assumptions are natural in many RLVR settings considered in this paper, especially for mathematical reasoning and code-generation tasks with verifiable outcomes. However, how the analysis extends to more open-ended tasks with non-binary, subjective, or learned reward signals remains underexplored.

Finally, while STAR improves rollout efficiency and wall-clock training speed, extending STAR to larger-scale models and broader application scenarios remains an important direction for future work.

Acknowledgments and Disclosures

Funding Statement. This work is supported by the Chinese Academy of Sciences Project for Young Scientists in Basic Research (YSBR-107), and in part by the Beijing Major Science and Technology Project under Contract no. Z251100008425009 and the Youth Innovation Promotion Association, CAS.

Competing Interests Disclosure. The authors declare no competing interests.

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL `https://arxiv.org/abs/2107.03374`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024. URL `https://aclanthology.org/2024.acl-long.211.pdf`.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. *arXiv preprint arXiv:2105.09938*, 2021. URL `https://arxiv.org/abs/2105.09938`.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018. URL `https://arxiv.org/abs/1803.00933`.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024. URL https://arxiv.org/abs/2412.16720.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL https://arxiv.org/abs/2403.07974.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. *arXiv preprint arXiv:2309.06180*, 2023. URL https://arxiv.org/abs/2309.06180.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023. URL https://arxiv.org/abs/2312.14852.

Siheng Li, Zhanhui Zhou, Wai Lam, Chao Yang, and Chaochao Lu. RePO: Replay-enhanced policy optimization. *arXiv preprint arXiv:2506.09340*, 2025. URL https://arxiv.org/abs/2506.09340.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/pdf?id=v8L0pN6EOi.

L.J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1992. ISSN 08856125. URL http://dx.doi.org/10.1023/A:1022628806385.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019. URL https://arxiv.org/abs/1711.05101.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. DeepScaleR: Surpassing O1-Preview with a 1.5B model by scaling RL, 2025. URL https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Meta. Llama 3.2 3B Instruct, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Youssef Mroueh, Nicolas Dupuis, Brian Belgodere, Apoorva Nitsure, Mattia Rigotti, Kristjan Greenewald, Jiri Navratil, Jerret Ross, and Jesus Rios. Revisiting group relative policy optimization: Insights into on-policy and off-policy training. *arXiv preprint arXiv:2505.22257*, 2025. URL https://arxiv.org/abs/2505.22257.

Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, Shane Legg, Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015. URL https://arxiv.org/abs/1507.04296.

Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. Codeforces, 2025. URL https://huggingface.co/datasets/open-r1/codeforces.

Doina Precup, Richard S Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 759–766, 2000.

Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pp. 417–424, 2001.

Ali Rahimi-Kalahroudi, Janarthanan Rajendran, Ida Momennejad, Harm van Seijen, and Sarath Chandar. Replay buffer with local forgetting for adapting to local environment changes in deep model-based reinforcement learning. In *Conference on Lifelong Learning Agents*, pp. 21–42. PMLR, 2023.

Matthew Schlegel, Wesley Chung, Daniel Graves, Jian Qian, and Martha White. Importance resampling for off-policy prediction. *arXiv preprint arXiv:1906.04328*, 2019. URL https://arxiv.org/abs/1906.04328.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. *arXiv preprint arXiv:2409.19256*, 2024. URL https://arxiv.org/abs/2409.19256.

Xu Wan, Yansheng Wang, Wenqi Huang, and Mingyang Sun. Buffer matters: Unleashing the power of off-policy reinforcement learning in large language model reasoning. *arXiv preprint arXiv:2602.20722*, 2026. URL https://arxiv.org/abs/2602.20722.

Christopher Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992. URL https://api.semanticscholar.org/CorpusID:208910339.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

Zhiheng Xi, Xin Guo, Yang Nan, Enyu Zhou, Junrui Shen, Wenxiang Chen, Jiaqi Liu, Jixuan Huang, Zhihao Zhang, Honglin Guo, et al. BAPO: Stabilizing off-policy reinforcement learning for LLMs via balanced policy optimization with adaptive clipping. *arXiv preprint arXiv:2510.18927*, 2025. URL https://arxiv.org/abs/2510.18927.

Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. LeetCodeDataset: A temporal dataset for robust evaluation and efficient training of code LLMs. *arXiv preprint arXiv:2504.14655*, 2025. URL https://arxiv.org/abs/2504.14655.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. *arXiv preprint arXiv:2302.03169*, 2023. URL https://arxiv.org/abs/2302.03169.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025. URL https://arxiv.org/abs/2504.14945.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024. URL https://arxiv.org/abs/2409.12122.

Shihui Yang, Chengfeng Dou, Peidong Guo, Kai Lu, Qiang Ju, Fei Deng, and Rihui Xin. DCPO: Dynamic clipping policy optimization. *arXiv preprint arXiv:2509.02333*, 2025. URL https://arxiv.org/abs/2509.02333.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, 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, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Hongzhi Zhang, Jia Fu, Jingyuan Zhang, Kai Fu, Qi Wang, Fuzheng Zhang, and Guorui Zhou. RLEP: Reinforcement learning with experience replay for LLM reasoning. *arXiv preprint arXiv:2507.07451*, 2025. URL https://arxiv.org/abs/2507.07451.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023. URL https://arxiv.org/abs/2304.11277.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025. URL https://arxiv.org/abs/2507.18071.

Haizhong Zheng, Jiawei Zhao, and Beidi Chen. Prosperity before collapse: How far can off-policy RL reach with stale data on LLMs? In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 82657–82679, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8590b946e4f0c2b1262cb5b047924653-Paper-Conference.pdf.

A Additional Experimental Results

A.1 Results on Code Generation

Table 4 reports the results on code generation benchmarks. Consistent with the mathematical reasoning results in the main text, STAR improves all five off-policy backbones, yielding average gains of 2.9–5.8 points. With vanilla GRPO, STAR increases the average score from 30.5 ± 0.4 to 36.3 ± 0.6, surpassing the on-policy GRPO baseline of 34.1 ± 0.5. STAR also improves GSPO, M2PO, and DCPO from 32.1, 34.0, and 34.0 to 35.8, 36.9, and 37.0, respectively. For BAPO, STAR raises the average score from 33.8 ± 0.6 to 36.8 ± 0.7, further supporting its applicability across different optimization algorithms. These gains substantially exceed the corresponding reported standard deviations of 0.4–1.2 points, providing additional evidence that the improvements are robust to random variation. DCPO+STAR achieves the highest average score of 37.0 ± 0.4, while M2PO+STAR achieves the best LiveCodeBench performance of 16.2.

In addition to performance gains, STAR preserves the efficiency advantage of off-policy training. Across all five backbones, STAR reaches the converged performance of the on-policy baseline with $5.5\times$–$6.1\times$ wall-clock speedup. DCPO+STAR achieves the largest speedup of $6.1\times$, while GRPO+STAR and BAPO+STAR both achieve $5.7\times$ acceleration. Together, these results demonstrate that the benefits of low-staleness and high-quality replay data extend beyond mathematical reasoning to code generation, improving both performance and training efficiency.

Table 4: Performance comparison on code generation tasks. GRPO, GSPO, M2PO, DCPO and BAPO are used as backbone algorithms, with on-policy GRPO as the baseline. Speedup denotes the wall-clock acceleration to reach the converged performance of the on-policy baseline, and – indicates failure to reach it. The best performance on each benchmark is highlighted in **bold**.

Method	HumanEval	LeetCode	LiveCodeBench	Avg.	Speedup
MODEL: LLAMA-3.2-INSTRUCT-3B					
ON-POLICY					
GRPO	67.1	21.0	14.3	34.1 ±0.5	1.0×
OFF-POLICY					
GRPO	61.8	17.2	12.4	30.5 ±0.4	–
+STAR	70.2	22.6	16.1	36.3 ±0.6 (+5.8)	5.7×
GSPO	64.0	18.6	13.7	32.1 ±1.2	–
+STAR	70.8	21.4	15.1	35.8 ±0.9 (+3.7)	5.5×
M2PO	66.7	21.8	13.6	34.0 ±0.6	–
+STAR	71.4	23.2	**16.2**	36.9 ±0.6 (+2.9)	5.8×
DCPO	67.2	21.3	13.6	34.0 ±0.5	–
+STAR	**72.6**	**23.8**	14.8	**37.0** ±0.4 (+3.0)	**6.1**×
BAPO	66.9	21.5	13.2	33.8 ±0.6	–
+STAR	71.8	23.5	15.1	36.8 ±0.7 (+3.0)	5.7×

A.2 Robustness under Aggressive Off-Policy Regimes

To evaluate the robustness of STAR under more aggressive off-policy regimes, we use Qwen2.5-Math-1.5B, with all other training and evaluation settings identical to those used in Section 6.2, and vary two key system parameters: the maximum policy-lag cap of replay samples and the synchronization interval between the behavior and target policies. The default setting uses a lag cap of 32 policy updates and synchronizes the behavior policy every 4 optimization mini-batches. We independently increase the lag cap from 32 to 128 and the synchronization interval from 4 to 16 while keeping the remaining settings unchanged.

As shown in Table 5, increasing the lag cap from 32 to 128 has only a minor effect on average performance, reducing it from 48.5 to 48.3. Although the larger buffer admits more stale samples, STAR assigns lower sampling probabilities to samples with large behavior-current policy mismatch

Table 5: Robustness of STAR under more aggressive off-policy regimes. The lag cap specifies the maximum permitted policy lag of replay samples, and the synchronization interval specifies the number of optimization mini-batches between behavior-policy synchronizations.

Method	Lag Cap	Sync Interval	Minerva	Olympiad	Math	Avg.
On-policy GRPO	–	–	31.4	35.4	70.1	45.6
STAR (Larger Lag Cap)	128	4	**33.8**	37.6	73.4	48.3
STAR (Rarer Synchronization)	32	16	32.7	37.4	71.9	47.3
STAR (Default)	**32**	**4**	33.6	**38.2**	**73.8**	**48.5**

through the staleness term $|\bar{\rho}_i - 1|$. Consequently, increasing the permitted policy lag does not directly cause highly stale samples to dominate the sampled optimization batches.

Increasing the synchronization interval from 4 to 16 produces a larger performance reduction, lowering the average score from 48.5 to 47.3, although STAR continues to outperform the on-policy GRPO baseline by 1.7 points. With less frequent synchronization, the behavior policy used for rollout becomes increasingly outdated, such that even newly generated samples may already exhibit substantial policy mismatch. In this regime, importance-based resampling can prioritize the freshest available data but cannot compensate for an insufficient supply of fresh samples. These results demonstrate that STAR remains effective under substantially increased policy lag, while sufficiently frequent policy synchronization remains important for maintaining a reliable supply of fresh replay data.

A.3 Curriculum-Controlled Baselines

The quality term $a(x)(1 - a(x))$ and HSR both induce a prompt curriculum, which may contribute to STAR's performance independently of replay. We conduct this study using Qwen2.5-Math-1.5B, with all other training and evaluation settings identical to those used in Section 6.2, except for the sampling strategies specified below. To separate curriculum effects from the replay-specific benefit of staleness-aware sampling, we compare the following methods:

- **On-policy GRPO.** The standard on-policy baseline without quality-based prompt selection or HSR.
- **On-policy + Quality + HSR.** An on-policy curriculum baseline using soft dynamic sampling. At each rollout phase, we freeze the current policy, generate fresh prompt groups, compute their group accuracy $a(x)$, and accept each group with probability $4a(x)(1 - a(x))$ until the training batch is full. The accepted prompt distribution is therefore proportional to $a(x)(1 - a(x))$, while all retained trajectories remain strictly on-policy. This baseline applies the same HSR prompt-recurrence rule as STAR.
- **Recency + Quality + HSR.** A replay-based baseline that retains the same replay buffer, quality term, and HSR mechanism as STAR, but replaces the IS-ratio-based staleness score with policy age. Specifically, its priority score is defined as $P_i^{\text{recency}} = \frac{a_i(1-a_i)}{d_i+1}$, where d_i is the number of policy versions elapsed since sample i was generated. This baseline requires no additional forward passes for priority computation.
- **STAR.** The complete method using the quality term, HSR, and IS-ratio-based staleness estimation.

As shown in Table 6, adding quality-based sampling and HSR improves the on-policy GRPO baseline from 45.6 to 47.4, confirming the benefit of the induced prompt curriculum. However, this baseline relies on over-generating and rejecting fresh trajectories to fill each optimization batch, increasing the rollout count from 432k to 781k. STAR achieves an average score of 48.5, outperforming the curriculum-controlled on-policy baseline by 1.1 points with gains on all three benchmarks, while requiring only 107k rollouts. These results indicate that STAR's gains are not fully explained by prompt curriculum alone.

The recency-based replay baseline achieves an average score of 47.0, which is 1.5 points below STAR despite using the same quality term, HSR mechanism, and replay buffer. Policy age provides

Table 6: Curriculum-controlled ablation on mathematical reasoning benchmarks. *Rollouts* denotes the total number of generated trajectories.

Method	Minerva	Olympiad	Math	Avg.	Rollouts
On-policy GRPO	31.4	35.4	70.1	45.6	432k
On-policy + Quality + HSR	32.8	37.1	72.2	47.4	781k
Recency + Quality + HSR	31.9	36.7	72.3	47.0	**102k**
STAR	**33.6**	**38.2**	**73.8**	**48.5**	107k

only an indirect measure of staleness because consecutive policy versions may differ substantially, whereas samples generated several versions apart may remain behaviorally close to the current policy. In contrast, STAR uses the length-normalized IS ratio $|\bar{\rho}_i - 1|$ to directly estimate the mismatch between the behavior and current policies, supporting IS-ratio-based staleness estimation over the cheaper recency heuristic.

A.4 Ablation on Hard-Sample Recurrence

We conduct an ablation study with GRPO as the backbone algorithm to evaluate the effect of the HSR component in STAR. As shown in Figure 5, removing HSR causes the policy entropy to steadily decline during training, revealing a clear entropy-collapse phenomenon. This indicates that repeatedly optimizing on historical replay data can reduce policy diversity and drive the policy toward premature convergence. In contrast, HSR enables the entropy to recover after the early training stage and remain consistently higher, suggesting that hard-prompt recurrence helps maintain exploration by exposing the policy to prompts that are not yet reliably solved. Table 7 shows that HSR also improves performance on both mathematical reasoning and code generation. Together, these results support the effectiveness of HSR in mitigating entropy collapse and improving the performance of off-policy RLVR.

Table 7: Ablation results of HSR. HSR improves the performance of off-policy RLVR.

Method	Math	Code	Avg.
STAR w/o HSR	45.6	35.7	40.7
STAR w/ HSR	**47.2**	**36.3**	**41.8**

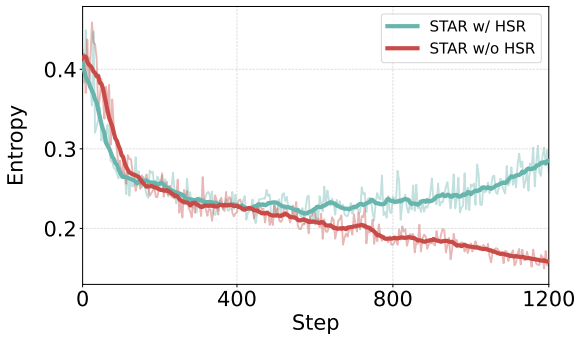


Figure 5: Ablation results of HSR. HSR maintains higher policy entropy and mitigates entropy collapse.

A.5 Ablation on HSR Hyperparameters

Figure 6 presents the sensitivity analysis of the two HSR hyperparameters: the hard-sample accuracy threshold τ and the hard-pool replacement ratio λ. We use GRPO as the backbone algorithm, train Qwen2.5-Math-7B on the mathematical reasoning task, and report the average performance across

the same evaluation benchmarks used in the main text. The results show that the threshold τ plays a critical role in HSR. The best performance is achieved around $\tau = 0.1$, while using an overly large threshold, such as $\tau = 0.4$, consistently degrades performance. This suggests that HSR is most effective when the hard-sample pool focuses on genuinely difficult prompts that the current policy cannot solve reliably. If the threshold is too large, the pool includes many relatively easy prompts, making recurrence less targeted and reducing its benefit.

In contrast, the replacement ratio λ is less sensitive within a moderate range. Across $\lambda \in \{0.025, 0.05, 0.1, 0.2\}$, HSR generally maintains strong performance when τ is properly chosen. However, overly large replacement ratios can still hurt performance, since replacing too many original prompts reduces rollout diversity and overemphasizes previously seen hard prompts. Overall, the default setting $\tau = 0.1$ and $\lambda = 0.05$ achieves the best performance, providing a good balance between hard-prompt recurrence and rollout diversity.

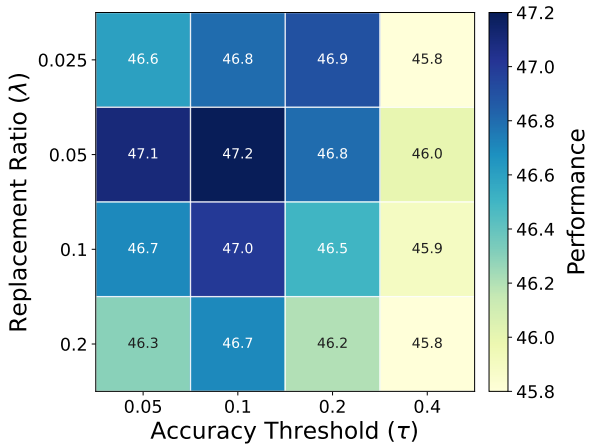


Figure 6: Sensitivity analysis of HSR hyperparameters. We vary the hard-sample accuracy threshold τ and the hard-pool replacement ratio λ, and report the average performance on mathematical reasoning benchmarks. HSR performs best with a relatively strict hard-sample threshold and a moderate replacement ratio. The default setting $(\tau = 0.1, \lambda = 0.05)$ achieves the best performance.

A.6 Ablation on Weighted Resampling Metrics

We conduct an ablation study to evaluate the contribution of the two metrics used in STAR's weighted resampling strategy. Recall that STAR assigns the priority score of each replay sample as

$$P_i = \frac{a_i(1 - a_i)}{|\bar{\rho}_i - 1| + \epsilon}, \tag{18}$$

where the numerator serves as the quality metric and the denominator serves as the staleness metric. To disentangle their effects, we compare STAR with two ablated variants.

For *STAR w/o Quality Metric*, we remove the numerator quality metric and define the priority score as

$$P_i^{\text{w/o quality}} = \frac{1}{|\bar{\rho}_i - 1| + \epsilon}. \tag{19}$$

This variant only prioritizes low-staleness samples.

For *STAR w/o Staleness Metric*, we remove the denominator staleness metric and define the priority score as

$$P_i^{\text{w/o staleness}} = a_i(1 - a_i). \tag{20}$$

This variant only prioritizes high-quality samples according to their estimated gradient contribution.

Table 9: End-to-end runtime breakdown of the STAR training pipeline. Activity fractions are measured over a steady-state training window and are not mutually exclusive because rollout and learner execution overlap asynchronously.

Runtime Metric	**Included Operations / Definition**	**Fraction**
Rollout path active	Trajectory generation, reward computation and verification, and replay-buffer insertion	86%
Learner path active	Priority and log-probability computation, advantage computation, and policy optimization	72%
Rollout-only time	Rollout execution without concurrent learner activity	22%
Coordination path active	Policy synchronization, data transfer, and execution scheduling	6%

For both variants, the final sampling probability w_i is then obtained by normalizing these priority scores across all samples in the replay buffer $\mathcal{B}$:

$$w_i = \frac{P_i}{\sum_{j \in \mathcal{B}} P_j}. \tag{21}$$

As shown in Table 8, removing either metric degrades performance, confirming that both components are useful for weighted resampling. When the Quality Metric is removed, the average performance drops from 41.8 to 40.1, indicating that prioritizing informative samples is important for improving the efficiency of policy optimization. The degradation is more substantial when the Staleness Metric is removed, where the average performance further drops to 37.3. This suggests that controlling data staleness is the dominant factor for effective off-policy RLVR, which is consistent with our analysis in Section 4 showing that high data staleness causes excessive optimization signal loss and limits the performance ceiling of off-policy RLVR. Overall, these results validate the design of STAR's weighted resampling strategy: effective replay sampling should select samples that are not only informative, but also sufficiently fresh under the current policy.

Table 8: Ablation results of the weighted resampling metrics. Both the Quality Metric and the Staleness Metric contribute to STAR, while removing the Staleness Metric leads to a larger performance drop.

Method	**Math**	**Code**	**Avg.**
STAR	**47.2**	**36.3**	**41.8**
STAR w/o Quality Metric	45.5	34.7	40.1 (-1.7)
STAR w/o Staleness Metric	42.6	32.0	37.3 (-4.5)

B Wall-Clock Accounting and Runtime Breakdown

The wall-clock time reported in this work measures the complete end-to-end training process, including trajectory generation, verification and reward computation, buffer insertion, buffer-wide rescoring, advantage computation, policy optimization, policy synchronization, data transfer, and scheduling. For example, in Figure 1, on-policy GRPO requires 37.0 hours to reach its converged performance, whereas GRPO+STAR reaches the same target performance in 6.7 hours, corresponding to an end-to-end speedup of $37.0/6.7 = 5.5\times$.

To characterize how this time is distributed, we profile the complete STAR pipeline over a steady-state training window. For each execution path, we merge all of its active time intervals and report the duration of their union divided by the total end-to-end elapsed time. Because rollout generation and policy optimization execute asynchronously, the reported fractions describe overlapping activities and therefore need not sum to 100%.

The 86% rollout-path activity includes trajectory generation, verification and reward computation, buffer insertion, and the initial buffer-filling rollouts. The 72% learner-path activity includes replay-buffer rescoring, advantage computation, and policy optimization, while the 6% coordination-path activity includes policy synchronization, data transfer, and scheduling. The 22% rollout-only fraction

denotes rollout activity without concurrent learner execution; consequently, the remaining $86\% - 22\% = 64\%$ of elapsed time contains concurrent rollout and learner activity.

All replay-specific costs, including buffer rescoring, policy synchronization, verification, data transfer, and scheduling, are therefore included in the reported wall-clock time rather than omitted from the accounting. The resulting speedup reflects the combined effects of asynchronous rollout-learner overlap, trajectory reuse that reduces the need for newly generated rollouts, and fewer optimization steps required to reach the on-policy target performance.

C Proofs of Theoretical Results

C.1 Proof of Theorem 1

Proof. For notational simplicity, we omit the dependence on t when there is no ambiguity. We assume that $\mu(y|x) > 0$ whenever $\pi_\theta(y|x) > 0$, ensuring that the importance sampling (IS) ratio $\rho(x, y) = \frac{\pi_\theta(y|x)}{\mu(y|x)}$ is well-defined. Furthermore, the advantage signal is assumed to be uniformly bounded such that $|A(x, y)| \leq \epsilon_A$.

Recall from Section 4 that the clipping mask $m_t(x, y)$ and the discard loss Δ_t^{disc} are defined as:

$$m_t(x, y) = \mathbf{1}\left\{A(x, y) \geq 0, \rho(x, y) \leq c_t^{\text{high}}\right\} + \mathbf{1}\left\{A(x, y) < 0, \rho(x, y) \geq c_t^{\text{low}}\right\} \tag{22}$$

$$\Delta_t^{\text{disc}} = L_t^{\text{ideal}}(\pi_\theta; \mu) - L_t^{\text{mask}}(\pi_\theta; \mu) \tag{23}$$

where $1 - m_t(x, y) = 1$ indicates that the sample triggers the clipping trust region and its gradient contribution is discarded. Using the definition of L_t^{ideal} and L_t^{mask}, we have:

$$\Delta_t^{\text{disc}} = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)}\left[\rho(x, y)A(x, y)(1 - m_t(x, y))\right] \tag{24}$$

By taking the absolute value and applying the uniform bound $|A(x, y)| \leq \epsilon_A$, we obtain:

$$\left|\Delta_t^{\text{disc}}\right| \leq \epsilon_A \mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)}\left[\rho(x, y)(1 - m_t(x, y))\right] \tag{25}$$

Based on the definition of $m_t(x, y)$, a sample triggers clipping ($1 - m_t(x, y) = 1$) only if it falls into one of the following two regions:

- High-ratio clipping: $A(x, y) \geq 0$ and $\rho(x, y) > c_t^{\text{high}}$.
- Low-ratio clipping: $A(x, y) < 0$ and $\rho(x, y) < c_t^{\text{low}}$.

In the high-ratio region, since $\rho > c_t^{\text{high}} > 1$, we can bound ρ as:

$$\rho = \frac{\rho}{\rho - 1}(\rho - 1) \leq \frac{c_t^{\text{high}}}{c_t^{\text{high}} - 1}|\rho - 1| \tag{26}$$

where we utilize the fact that $z/(z - 1)$ is monotonically decreasing for $z > 1$. Similarly, in the low-ratio region, since $0 \leq \rho < c_t^{\text{low}} < 1$, we have:

$$\rho = \frac{\rho}{1 - \rho}(1 - \rho) \leq \frac{c_t^{\text{low}}}{1 - c_t^{\text{low}}}|1 - \rho| \tag{27}$$

where we utilize the fact that $z/(1 - z)$ is monotonically increasing for $0 \leq z < 1$.

Combining these two cases with the sensitivity coefficient $\kappa(c_t^{\text{low}}, c_t^{\text{high}}) = \max\left\{\frac{c_t^{\text{high}}}{c_t^{\text{high}}-1}, \frac{c_t^{\text{low}}}{1-c_t^{\text{low}}}\right\}$, the inequality becomes:

$$\left|\Delta_t^{\text{disc}}\right| \leq \epsilon_A \kappa(c_t^{\text{low}}, c_t^{\text{high}}) \mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)}\left[|\rho(x, y) - 1|\right] \tag{28}$$

To establish the relationship with the Total Variation (TV) distance, we expand the expectation term by substituting $\rho(x, y) = \pi_\theta(y|x)/\mu(y|x)$:

$$\mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)}\left[|\rho(x,y)-1|\right] = \mathbb{E}_{x\sim\mathcal{D}}\left[\sum_y \mu(y|x)\left|\frac{\pi_\theta(y|x)}{\mu(y|x)}-1\right|\right] \tag{29}$$

$$= \mathbb{E}_{x\sim\mathcal{D}}\left[\sum_y |\pi_\theta(y|x)-\mu(y|x)|\right] \tag{30}$$

By the definition of Total Variation distance over the training data, $D_{TV}(\pi_\theta, \mu) = \frac{1}{2}\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_y |\pi_\theta(y|x) - \mu(y|x)|\right]$. Thus, we obtain the identity:

$$\mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)}\left[|\rho(x,y)-1|\right] = 2D_{TV}(\pi_\theta, \mu) \tag{31}$$

Substituting this identity back into the bounded inequality, we arrive at the final result:

$$\left|\Delta_t^{\text{disc}}\right| \le 2\epsilon_A \kappa(c_t^{\text{low}}, c_t^{\text{high}}) D_{TV}(\pi_\theta, \mu) \tag{32}$$

This concludes the proof. □

C.2 Proof of Theorem 2

Proof. Fix a prompt x. Let $y_1, \ldots, y_G$ be i.i.d. samples from the policy $\pi_\theta(\cdot|x)$. We define the population accuracy $a_\theta(x)$ and the binary rewards r_i as:

$$a_\theta(x) := \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[r(x,y)], \qquad r_i := r(x, y_i) \in \{0, 1\}. \tag{33}$$

The group mean reward $\bar{r}$ and the group-relative advantage A_i are defined as:

$$\bar{r} := \frac{1}{G}\sum_{j=1}^{G} r_j, \qquad A_i := r_i - \bar{r}. \tag{34}$$

Let $s_i := \nabla_\theta \log \pi_\theta(y_i|x)$ denote the score function for each sample. The per-prompt stochastic policy gradient is given by:

$$\hat{g}_x = \frac{1}{G}\sum_{i=1}^{G} A_i s_i. \tag{35}$$

Expanding the second moment of the gradient energy, we have:

$$\mathbb{E}\|\hat{g}_x\|^2 = \mathbb{E}\left\|\frac{1}{G}\sum_{i=1}^{G} A_i s_i\right\|^2 \tag{36}$$

$$= \frac{1}{G^2}\sum_{i=1}^{G}\mathbb{E}[A_i^2\|s_i\|^2] + \frac{1}{G^2}\sum_{i\neq j}\mathbb{E}[A_i A_j s_i^\top s_j]. \tag{37}$$

For binary rewards, r_i follows a Bernoulli distribution with variance $\mathrm{Var}(r_i) = a_\theta(x)(1 - a_\theta(x))$. Since $A_i = r_i - \bar{r}$, a direct calculation of the variance for centered samples yields:

$$\mathbb{E}[A_i^2] = \mathrm{Var}(r_i - \bar{r}) = \frac{G-1}{G} a_\theta(x)(1 - a_\theta(x)). \tag{38}$$

Next, we define the prompt-conditioned Fisher trace C_x as:

$$C_x := \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\|\nabla_\theta \log \pi_\theta(y|x)\|^2\right] = \mathrm{tr}(F_x). \tag{39}$$

We can decompose the second moment as:

$$\mathbb{E}\|\hat{g}_x\|^2 = \frac{G-1}{G^2} a_\theta(x)(1 - a_\theta(x)) C_x + R_x, \tag{40}$$

where the remainder term R_x is:

$$R_x = \frac{1}{G^2}\sum_{i=1}^{G}\mathrm{Cov}(A_i^2, \|s_i\|^2) + \frac{1}{G^2}\sum_{i\neq j}\mathbb{E}[A_i A_j s_i^\top s_j]. \tag{41}$$

In the context of RLVR training, we introduce two simplifying assumptions:

1. Weak Score-Reward Coupling: The magnitude of the parameter update $\|s_i\|^2$ is approximately independent of the reward-based advantage A_i^2, implying $\mathrm{Cov}(A_i^2, \|s_i\|^2) \approx 0$.

2. Weak Cross-Rollout Correlation: The interaction between the direction of the score functions $s_i^\top s_j$ and the coupled advantages $A_i A_j$ for different rollouts is negligible, i.e., $\sum_{i \neq j} \mathbb{E}[A_i A_j s_i^\top s_j] \approx 0$.

Under these assumptions, R_x is treated as a lower-order term. Consequently, the second moment is approximated by the diagonal terms:

$$\mathbb{E}\|\hat{g}_x\|^2 \approx \frac{1}{G^2} \sum_{i=1}^{G} \mathbb{E}[A_i^2]\mathbb{E}[\|s_i\|^2] = \frac{1}{G^2} \cdot G \cdot \left(\frac{G-1}{G} a_\theta(x)(1 - a_\theta(x)) \right) C_x. \tag{42}$$

Simplifying the expression, we obtain the final result:

$$\mathbb{E}\|\hat{g}_x\|^2 \approx \frac{G-1}{G^2} a_\theta(x)(1 - a_\theta(x)) C_x. \tag{43}$$

This concludes the proof. □

D Additional Theoretical Derivations

D.1 Derivation of the Clipping Mask

We provide a derivation of the clipping mask used in Section 4. Recall that the clipped surrogate objective is given by

$$L_t^{\mathrm{clip}}(\pi_\theta; \mu) = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \mu(\cdot|x)} \left[\min \left(\rho A(x,y), \mathrm{clip}\Big(\rho, c_t^{\mathrm{low}}, c_t^{\mathrm{high}}\Big) A(x,y) \right) \right], \tag{44}$$

where $\rho = \pi_\theta(y|x)/\mu(y|x)$.

The effective gradient contribution of each sample depends on whether the unclipped branch $\rho A(x,y)$ or the clipped branch $\mathrm{clip}(\rho, c_t^{\mathrm{low}}, c_t^{\mathrm{high}})A(x,y)$ is selected by the minimum operator.

When $A(x,y) \geq 0$, the objective selects

$$\min \left(\rho A(x,y), \mathrm{clip}\Big(\rho, c_t^{\mathrm{low}}, c_t^{\mathrm{high}}\Big) A(x,y) \right).$$

Since $A(x,y) \geq 0$, the minimum is determined by the smaller coefficient of $A(x,y)$. Thus, for $\rho \leq c_t^{\mathrm{high}}$, the unclipped term $\rho A(x,y)$ is selected and the sample contributes an effective policy-gradient signal. In contrast, for $\rho > c_t^{\mathrm{high}}$, the clipped term $c_t^{\mathrm{high}} A(x,y)$ is selected, which is constant with respect to ρ and therefore provides no effective gradient contribution through the importance ratio.

When $A(x,y) < 0$, multiplying by the negative advantage reverses the ordering. In this case, the clipped branch is selected when $\rho < c_t^{\mathrm{low}}$, yielding $c_t^{\mathrm{low}} A(x,y)$, which is constant with respect to ρ and provides no effective gradient contribution through the importance ratio. The unclipped branch is selected when $\rho \geq c_t^{\mathrm{low}}$, and the sample retains its effective policy-gradient contribution.

Therefore, the condition under which a sample retains an effective gradient contribution can be summarized by the following mask:

$$m_t(x,y) = \mathbf{1}\left\{ A(x,y) \geq 0,\, \rho \leq c_t^{\mathrm{high}} \right\} + \mathbf{1}\left\{ A(x,y) < 0,\, \rho \geq c_t^{\mathrm{low}} \right\}. \tag{45}$$

Using this mask, the gradient-carrying component of the clipped surrogate can be represented as

$$L_t^{\mathrm{mask}}(\pi_\theta; \mu) = \mathbb{E}_{x \sim \mathcal{D},\, y \sim \mu(\cdot|x)} \left[\rho A(x,y) m_t(x,y) \right]. \tag{46}$$

D.2 Single-Sample Contribution TV Distance

We derive the single-sample contribution to the total variation (TV) distance between the target policy π_θ and the behavior policy μ over the training data. For each prompt x, the conditional TV distance is defined as

$$D_{\mathrm{TV}}(\pi_\theta(\cdot|x), \mu(\cdot|x)) = \frac{1}{2} \sum_y |\pi_\theta(y|x) - \mu(y|x)| \,. \tag{47}$$

The TV distance over the training data is then

$$D_{\mathrm{TV}}(\pi_\theta, \mu) = \mathbb{E}_{x\sim\mathcal{D}} \left[D_{\mathrm{TV}}(\pi_\theta(\cdot|x), \mu(\cdot|x)) \right] . \tag{48}$$

Substituting the definition of conditional TV distance gives

$$D_{\mathrm{TV}}(\pi_\theta, \mu) = \frac{1}{2} \mathbb{E}_{x\sim\mathcal{D}} \left[\sum_y |\pi_\theta(y|x) - \mu(y|x)| \right] . \tag{49}$$

Assuming that $\mu(y|x) > 0$ whenever $\pi_\theta(y|x) > 0$, the importance ratio $\rho(x, y) = \pi_\theta(y|x)/\mu(y|x)$ is well-defined. We can rewrite the difference between the two policies as

$$\pi_\theta(y|x) - \mu(y|x) = \mu(y|x) \left(\rho(x, y) - 1 \right) . \tag{50}$$

Therefore,

$$\begin{aligned} D_{\mathrm{TV}}(\pi_\theta, \mu) &= \frac{1}{2} \mathbb{E}_{x\sim\mathcal{D}} \left[\sum_y \mu(y|x) \left| \rho(x, y) - 1 \right| \right] \\ &= \frac{1}{2} \mathbb{E}_{x\sim\mathcal{D},\, y\sim\mu(\cdot|x)} \left[\left| \rho(x, y) - 1 \right| \right] . \end{aligned} \tag{51}$$

This shows that, up to the constant factor $1/2$ in the definition of TV distance, the contribution of a single sample (x, y) to the total TV distance is given by

$$\left| \rho(x, y) - 1 \right| . \tag{52}$$

In the main text, we therefore use $|\rho - 1|$ as a sample-level staleness measure, where ρ is used as shorthand for $\rho(x, y)$.

D.3 A Mechanistic Mismatch between IR and Trust-Region Clipping

Importance resampling (IR) provides a classical mechanism for selecting off-policy data by assigning larger sampling probabilities to samples with larger importance weights. In the off-policy RLVR setting, for a replay-buffer sample (x, y) generated by the behavior policy μ, the importance ratio with respect to the current target policy π_θ is

$$\rho(x, y) = \frac{\pi_\theta(y|x)}{\mu(y|x)}. \tag{53}$$

An IR replay strategy would increase the probability of sampling data with larger $\rho(x, y)$, since such samples are considered more important under the target-policy distribution.

However, this principle is misaligned with modern RLVR objectives based on PPO-style trust-region clipping. For a sample with positive advantage $A(x, y) > 0$, the clipped surrogate objective discards its effective gradient contribution once the importance ratio exceeds the upper clipping boundary:

$$\rho(x, y) > 1 + \epsilon. \tag{54}$$

That is, high-ratio positive samples are precisely the samples that are more likely to trigger clipping. Consequently, prioritizing samples with large importance ratios increases the probability of selecting samples whose gradients are clipped, thereby reducing the amount of effective optimization signal used for policy improvement.

This issue can be seen more directly from the masked-objective view in Section 4. Let $m(x, y)$ denote whether a sample provides an unclipped gradient contribution. For positive-advantage samples, we have

$$m(x, y) = \mathbf{1}\{\rho(x, y) \le 1 + \epsilon\}. \tag{55}$$

Under a uniform replay sampler, the expected amount of effective positive-advantage signal is proportional to

$$\mathbb{E}_{(x,y)\sim\mathcal{B}}\left[\rho(x,y)A(x,y)\mathbf{1}\{\rho(x,y)\leq 1+\epsilon\}\right]. \tag{56}$$

In contrast, an IR sampler assigns larger sampling weights to samples with larger $\rho(x,y)$. This shifts the sampled batch toward the high-ratio region, where $\mathbf{1}\{\rho(x,y)\leq 1+\epsilon\}$ is more likely to be zero. As a result, although IR increases the likelihood of sampling high-importance data, a larger fraction of these samples do not contribute effective gradients after trust-region clipping.

Therefore, the objective of IR replay is not aligned with the optimization mechanism of modern RLVR. Rather than prioritizing samples solely according to importance ratios, replay sampling should account for whether the selected data can still provide valid gradient signals under the trust region. This motivates STAR to explicitly penalize stale samples with large deviations from the behavior policy and to prioritize samples that are both low-staleness and informative.

E Algorithmic Details

E.1 Overall Pipeline

STAR is designed as a data-centric replay framework for off-policy RLVR. It operates on the two standard phases of off-policy reinforcement learning: trajectory rollout and policy optimization. The trajectory rollout phase determines which prompts are used to generate new responses and how the generated data are stored in the replay buffer. The policy optimization phase determines which samples are selected from the replay buffer to update the target policy.

At a high level, STAR modifies both phases. During trajectory rollout, STAR uses Hard-Sample Recurrence (HSR) to mix newly sampled prompts from the training set with previously identified hard prompts. This encourages the policy to revisit prompts near its current capability boundary and helps mitigate premature entropy collapse. During policy optimization, STAR uses Weighted Resampling to construct optimization batches from the replay buffer. The sampling priority jointly considers data staleness and sample quality, so that the policy is preferentially updated using low-staleness and informative samples.

The complete rollout and optimization procedures are shown in Algorithms 1 and 2, respectively. These two phases are executed asynchronously in the off-policy training pipeline.

E.2 Trajectory Rollout with Hard-Sample Recurrence

Algorithm 1 describes the trajectory rollout phase of STAR. The goal of this phase is to generate new response trajectories while maintaining a pool of hard prompts that should be revisited in future rollout phases.

At the beginning of each rollout phase, STAR constructs a prompt batch $\mathcal{X}$ of size N. A fraction $1-\lambda$ of the prompts are sampled from the original training prompt set $\mathcal{D}$, forming the base prompt set $\mathcal{X}_{\text{base}}$. The remaining fraction λ is sampled from the hard-sample pool $\mathcal{H}$, forming $\mathcal{X}_{\text{hard}}$. The final rollout prompt batch is then given by

$$\mathcal{X}=\mathcal{X}_{\text{base}}\cup\mathcal{X}_{\text{hard}}.$$

The recurrence ratio λ controls how strongly STAR revisits hard prompts. When $\lambda=0$, the rollout phase degenerates to ordinary sampling from the training set. A larger λ increases the probability of revisiting hard prompts, thereby encouraging the policy to repeatedly explore prompts that are not yet reliably solved.

For each prompt $x\in\mathcal{X}$, the current policy π_θ generates a group of G responses:

$$\{y_g\}_{g=1}^{G}\sim\pi_\theta(\cdot|x).$$

The use of grouped responses follows the standard RLVR setting, where multiple candidate responses are sampled for the same prompt and evaluated by a verifiable reward function. Each response y_g is evaluated by the environment, which provides a binary verifiable reward:

$$r_g=R(x,y_g).$$

In our setting, r_g indicates whether the generated response satisfies the verifiable criterion, such as producing the correct final answer for a mathematical reasoning problem or passing the corresponding tests for a code generation problem.

After evaluating the G responses, STAR computes the group accuracy of prompt x:

$$a(x) = \frac{1}{G} \sum_{g=1}^{G} r_g.$$

This quantity measures how reliably the current policy can solve prompt x. A low value of $a(x)$ indicates that the prompt remains difficult for the current policy, while a high value indicates that the prompt is already mostly solved. The group accuracy serves two roles in STAR. First, it is used to maintain the hard-sample pool $\mathcal{H}$. Second, it is stored in the replay buffer and later used as a sample-quality signal during weighted resampling.

For each generated response y_g, STAR stores the tuple

$$(x, y_g, r_g, \log \pi_\theta(y_g|x), a(x))$$

into the replay buffer $\mathcal{B}$. The rollout log-probability $\log \pi_\theta(y_g|x)$ records the probability assigned to the response by the policy that generated the sample. When the sample is later reused from the replay buffer, this rollout policy is treated as the behavior policy μ, and the stored log-probability becomes

$$\log \mu(y_g|x).$$

This allows the optimization phase to compute the importance sampling ratio between the current target policy and the behavior policy without requiring access to the historical model parameters.

Finally, STAR updates the hard-sample pool $\mathcal{H}$ according to the group accuracy $a(x)$. If

$$a(x) < \tau,$$

then prompt x is considered hard and is added to or updated in $\mathcal{H}$. If x was already in $\mathcal{H}$, its stored information can be refreshed using the latest rollout statistics. If $a(x) \geq \tau$ and $x \in \mathcal{H}$, the prompt is removed from the hard-sample pool, since the current policy can now solve it with sufficiently high reliability. In this way, $\mathcal{H}$ dynamically tracks prompts near the policy's capability boundary, rather than keeping a fixed set of hard examples throughout training.

Algorithm 1 STAR: Trajectory Rollout Phase with Hard-Sample Recurrence

Require: Current policy π_θ, training prompt set $\mathcal{D}$, replay buffer $\mathcal{B}$, hard-sample pool $\mathcal{H}$, rollout batch size N, group size G, hard threshold τ, recurrence ratio λ
Ensure: Updated replay buffer $\mathcal{B}$ and hard-sample pool $\mathcal{H}$

Sample $(1-\lambda)N$ prompts $\mathcal{X}_{\text{base}}$ from $\mathcal{D}$
Sample λN hard prompts $\mathcal{X}_{\text{hard}}$ from $\mathcal{H}$
Construct rollout prompt batch $\mathcal{X} \leftarrow \mathcal{X}_{\text{base}} \cup \mathcal{X}_{\text{hard}}$
for each prompt $x \in \mathcal{X}$ **do**
  Generate G responses $\{y_g\}_{g=1}^{G} \sim \pi_\theta(\cdot \mid x)$
  Evaluate verifiable rewards $\{r_g\}_{g=1}^{G}$ using reward function $R(x, y_g)$
  Compute group accuracy $a(x) \leftarrow \frac{1}{G} \sum_{g=1}^{G} r_g$
  Store rollout data $\{(x, y_g, r_g, \log \pi_\theta(y_g \mid x), a(x))\}_{g=1}^{G}$ into $\mathcal{B}$
  if $a(x) < \tau$ **then**
    Add or update x in the hard-sample pool $\mathcal{H}$
  else if $x \in \mathcal{H}$ **then**
    Remove x from the hard-sample pool $\mathcal{H}$
  end if
end for
return $\mathcal{B}, \mathcal{H}$

E.3 Policy Optimization with Weighted Resampling

Algorithm 2 describes the policy optimization phase of STAR. The goal of this phase is to construct an optimization batch from the replay buffer that is both low-staleness and informative for policy improvement.

For each sample

$$i = (x_i, y_i, r_i, \log \mu_i(y_i|x_i), a_i) \in \mathcal{B},$$

STAR first computes the current log-probability of the response under the target policy:

$$\log \pi_\theta(y_i|x_i).$$

Here, μ_i denotes the behavior policy that generated sample i. In implementation, $\log \mu_i(y_i|x_i)$ is exactly the rollout log-probability stored when the sample was inserted into the replay buffer.

Then the length-normalized IS ratio can be computed by:

$$\bar{\rho}_i = \exp\left(\frac{1}{L_i}\left(\log \pi_\theta(y_i|x_i) - \log \mu_i(y_i|x_i)\right)\right),$$

where L_i is the response length of y_i.

The objective of weighted resampling is to construct low-TV training data. Appendix D.2 shows that the sample-level contribution to TV distance is proportional to $|\rho_i - 1|$. Therefore, STAR uses the length-normalized quantity

$$|\bar{\rho}_i - 1|$$

as a practical sampling indicator for sample staleness, without introducing a preference over response length.

In addition to staleness, STAR also considers sample quality. For each sample i, the group accuracy a_i inherited from its prompt is used to compute

$$a_i(1 - a_i).$$

This term is maximized when a_i is around 0.5, corresponding to prompts that are neither too easy nor completely unsolved. Such prompts typically lie near the policy's capability boundary and provide more informative learning signals. In contrast, prompts with $a_i \approx 1$ are already mostly solved, while prompts with $a_i \approx 0$ may provide limited successful trajectories for reinforcement. Thus, $a_i(1 - a_i)$ acts as a lightweight proxy for sample informativeness.

Combining the staleness and quality terms, STAR assigns each replay sample the priority score

$$P_i = \frac{a_i(1 - a_i)}{|\bar{\rho}_i - 1| + \epsilon},$$

where ϵ is a small positive constant for numerical stability.

After computing P_i for all samples in the replay buffer, STAR normalizes the priority scores into sampling probabilities:

$$w_i = \frac{P_i}{\sum_{j \in \mathcal{B}} P_j}.$$

An optimization batch $\mathcal{D}_{\text{batch}}$ of size M is then sampled from the replay buffer according to the categorical distribution defined by $\{w_i\}_{i \in \mathcal{B}}$. The target policy π_θ is finally updated using the sampled batch and the underlying policy optimization objective, such as GRPO.

Algorithm 2 STAR: Policy Optimization Phase with Weighted Resampling

Require: Current policy π_θ, replay buffer $\mathcal{B}$, optimization batch size M, small constant ϵ
Ensure: Updated policy π_θ

1: **for** each sample $i = (x_i, y_i, r_i, \log \mu_i(y_i \mid x_i), a_i) \in \mathcal{B}$ **do**
2: Compute current log-probability $\log \pi_\theta(y_i \mid x_i)$
3: Compute length-normalized IS ratio

$$\bar{\rho}_i = \exp\left(\frac{1}{L_i}\left(\log \pi_\theta(y_i \mid x_i) - \log \mu_i(y_i \mid x_i)\right)\right)$$

4: Compute priority score

$$P_i = \frac{a_i(1-a_i)}{|\bar{\rho}_i - 1| + \epsilon}$$

5: **end for**
6: Normalize priority scores into sampling probabilities

$$w_i = \frac{P_i}{\sum_{j \in \mathcal{B}} P_j}$$

7: Sample an optimization batch $\mathcal{D}_{\text{batch}}$ of size M from $\mathcal{B}$ according to $\{w_i\}_{i \in \mathcal{B}}$
8: Update π_θ on $\mathcal{D}_{\text{batch}}$
9: **return** π_θ

F Implementation and Technical Details

F.1 Implementation Details of Asynchronous Training

We implement STAR in a customized asynchronous training framework built upon VERL (Sheng et al., 2024). The framework decouples trajectory rollout from policy optimization, allowing generated trajectories to be reused through a replay buffer.

To control data staleness, we restrict the maximum policy lag between the behavior policy of each sample and the current target policy. Specifically, the replay buffer only retains samples whose behavior policy is within 32 target-policy update steps of the current target policy. Samples exceeding this staleness threshold are discarded from the buffer. This constraint prevents extremely stale data from dominating replay and keeps off-policy optimization within a controlled policy-drift range.

To maintain data diversity during replay, we impose a minimum buffer size before policy optimization starts. The replay buffer is required to contain at least 16 rollout batches. If the number of available samples in the buffer is below this threshold, the optimization process is blocked until additional rollout data are generated. This warm-up requirement prevents the optimizer from repeatedly sampling from an overly small buffer, which could reduce data diversity and amplify sampling bias.

For trajectory generation, we use a rollout batch size of 128 prompts. Experience generation is performed using the vLLM (Kwon et al., 2023) engine. During rollout, we set the sampling temperature to $T = 1.0$ and generate $G = 16$ responses per prompt. The generated responses are evaluated by verifiable task-specific feedback and then inserted into the replay buffer together with their rewards, rollout log-probabilities, and group-level statistics.

For efficiency, the behavior policy is synchronized with the target policy every 4 optimization mini-batches. This synchronization frequency balances training throughput and data freshness: less frequent synchronization improves rollout efficiency but increases policy lag, while more frequent synchronization reduces staleness at the cost of additional synchronization overhead.

For Hard-Sample Recurrence, we set the hard-sample accuracy threshold to $\tau = 0.1$ and the hard-pool replacement ratio to $\lambda = 0.05$. This setting introduces a small but persistent recurrence of hard prompts, improving exploration around the policy's capability boundary without overwhelming the rollout batch with previously seen prompts.

F.2 Training Details

For mathematical reasoning experiments, we use Qwen2.5-Math-7B as the base model and set the maximum context length to 4,096 tokens. For code generation experiments, we use Llama-3.2-Instruct-3B and set the maximum context length to 8,192 tokens.

All experiments are conducted on 16 NVIDIA A100 GPUs. To accelerate training via data parallelism, we use Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023). Each training trial is run for 1,200 optimization steps.

Following Yu et al. (2025), all methods are trained without a KL regularization term. We optimize the models using AdamW (Loshchilov & Hutter, 2019) with a learning rate of 1×10^{-6}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay 0.01.

F.3 Estimation of TV Proxy for Sequence Models

In the theoretical analysis, $D_{\mathrm{TV}}(\pi_\theta, \mu)$ denotes the standard total variation distance between the target policy π_θ and the behavior policy μ. For sequence models, exact computation of the sequence-level TV distance is intractable, since it requires summing over the entire response space. Moreover, a direct sequence-level importance ratio,

$$\rho(y|x) = \frac{\pi_\theta(y|x)}{\mu(y|x)} = \exp\left(\sum_{\ell=1}^{L} \log \frac{\pi_\theta(y_\ell|x, y_{<\ell})}{\mu(y_\ell|x, y_{<\ell})}\right),$$

can have high variance due to multiplicative accumulation across tokens.

Therefore, in our empirical analysis, we use a token-level sampled TV proxy to quantify policy staleness. For each sampled token y_ℓ, we compute the token-level importance ratio

$$r_\ell = \exp\left(\log \pi_\theta(y_\ell|x, y_{<\ell}) - \log \mu(y_\ell|x, y_{<\ell})\right).$$

The reported TV proxy is computed as

$$\widehat{D}_{\mathrm{TV}}^{\mathrm{tok}}(\pi_\theta, \mu) = \frac{1}{2} \operatorname{Avg}_{(x,y),\, \ell:\, m_\ell = 1} \left[|r_\ell - 1|\right],$$

where m_ℓ denotes the response mask indicating valid response tokens, and the average is taken over sampled response tokens in the mini-batch.

This quantity is not an exact sequence-level TV distance. Instead, it is a sample-based token-level proxy that measures the average deviation between the target and behavior policies on the tokens used for optimization. It is suitable for empirical diagnostics because it directly reflects token-level policy drift on replayed data, avoids the exponential length amplification of sequence-level ratios, and has substantially lower variance than sequence-level estimators.

For variable-length responses, the masked token average provides token-level normalization: longer responses contribute more valid tokens to the estimate, but the metric does not multiply or accumulate ratios across the whole sequence. To further reduce the effect of extreme ratios, we clamp the token log-ratio to $[-20, 20]$ before exponentiation. The metric is then aggregated by ordinary averaging across mini-batches, without additional exponential moving average smoothing.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 8 discusses the scope of the clipping-based theory, reward assumptions, and empirical evaluation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The theoretical statements specify their assumptions, and complete proofs are provided in Appendices C.1 and C.2.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper describes the evaluated models, datasets, tasks, metrics, baselines, training steps, framework in the experiment and appendix sections.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The paper uses public models and datasets. The complete training code is being organized for public release; the manuscript does not yet provide access to it with full reproduction instructions.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the main experimental settings. Additional implementation and technical details are provided in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The experimental setup describes evaluation with three independent random seeds. For each seed, performance is averaged over the final three evaluation checkpoints after convergence. Tables 2 and 4 report benchmark means across seeds, with the Avg. columns additionally reporting the standard deviation across seeds, rather than the standard error or a confidence interval.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides detailed experimental settings and compute resources in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, the research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper focuses on foundational optimization methods for RLVR rather than a specific deployment, and does not include a dedicated discussion of positive and negative societal impacts.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new high-risk pretrained model, image generator, scraped dataset, or other asset requiring a dedicated responsible-release safeguard.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper cites the existing models, datasets, benchmarks, and software frameworks used in the experiments, but the current manuscript does not explicitly list the licenses and terms of use for each asset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce a new dataset, benchmark, or released model asset.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments, human-subject studies, participant instructions, or participant compensation.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: The paper does not involve human-subject research or crowdsourcing, so IRB approval or equivalent review is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Beyond the policy models described in the paper, our use of LLM tools was limited to partial writing assistance and code implementation assistance.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.