IMPLAY: IMPORTANCE-WEIGHTED EXPERIENCE REPLAY FOR EFFICIENT POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Group Relative Policy Optimization (GRPO) has become a standard algorithm for post-training large language models (LLMs). However, each update depends on many on-policy rollouts, making generation the dominant training cost. Experience replay can reduce this cost by reusing past rollouts, but GRPO's group-relative advantages become stale as the policy evolves. We first analyze off-policiness in GRPO with experience replay and find that advantages can become substantially stale after only 10–20 optimizer steps, while sample age is only weakly indicative of this staleness.

Motivated by this analysis, we propose **IMPLAY**, an experience replay method for GRPO that introduces two mechanisms to address policy drift: importance-weighted advantage re-estimation, which uses importance sampling to re-estimate replayed trajectories' group-relative advantages under the current policy; and drift-aware eviction, which uses the same signal to identify and discard trajectories that have become too off-policy for reliable reuse. Evaluated across six mathematical reasoning benchmarks, **IMPLAY** matches GRPO's performance under the same training setup while requiring up to 47% fewer generated rollouts and 12% faster training. At a matched generation budget, **IMPLAY** outperforms ExGRPO, the state-of-the-art experience-replay method for GRPO, by up to 3.0 percentage points on the more challenging benchmarks. Ablations show that drift-aware eviction maintains a replay buffer in which importance-weighted advantage re-estimation remains effective.

1 INTRODUCTION

Post-training has become a crucial step for eliciting reasoning abilities in large language models (LLMs) with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) becoming a standard algorithm for reasoning post-training. GRPO avoids learning a critic by estimating advantages relative to a group of rollouts sampled for the same problem. While effective, this on-policy, group-based design is computationally expensive: every update requires generating multiple fresh rollouts, making generation—rather than gradient computation—dominate the training cost (Kim et al., 2026). Moreover, each rollout is typically used for only a single update before being discarded, leaving much of the compute invested in its generation underutilized.

Classical reinforcement learning (RL) addresses the inefficiency of fully on-policy training through experience replay (Lin, 1992; Schaul et al., 2015), which stores past rollouts in a replay buffer and reuses them for subsequent updates. Recent work has adapted experience replay to LLM post-training (Li et al., 2025; Zhan et al., 2026; Yoo et al., 2026; Mao et al., 2026), showing that past rollouts can improve performance without proportionally increasing the on-policy rollout budget. However, replay introduces a central challenge for GRPO: *how should the advantage of a replayed rollout be estimated as the policy changes?*

Some methods (Li et al., 2025; Yoo et al., 2026; Mao et al., 2026) reuse the advantage estimated when a rollout[1] was originally generated, but that advantage is relative to a group sampled from an earlier policy. As the policy evolves, the stored advantage becomes stale and provides a biased estimate of the rollout's advantage under the current policy. These methods therefore rely on tight

[1] In this work, we use rollout and trajectory interchangeably.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

recency control to limit this drift. In contrast, methods that re-estimate the advantage at replay time (Zhan et al., 2026; Guo et al., 2026) obtain a more up-to-date estimate but require fresh on-policy rollouts to construct a new comparison group, offsetting the computational savings that motivate experience replay.

We first analyze how policy drift affects replayed samples in GRPO (§4) and find that group-relative advantages computed at generation time can become stale after only 10–20 optimizer steps. Importantly, this staleness is highly sample-dependent, and sample age alone is a poor indicator of whether a stored advantage remains reliable. Motivated by this observation, we introduce **IMPLAY**, an experience replay algorithm for GRPO that addresses policy drift with two new mechanisms (§5). First, **importance-weighted advantage re-estimation** recomputes a replayed trajectory's advantage under the current policy. It reweights rewards in the trajectory's original reference group by their current-to-behavior importance ratios and uses the resulting weighted mean reward to compute the advantage. Second, **drift-aware eviction** uses the same importance ratio signal to remove entries in the replay buffer that have become too off-policy for reliable advantage re-estimation.

We evaluate **IMPLAY** against fully on-policy GRPO under the same optimization setup, as well as existing experience-replay methods, across six mathematical reasoning benchmarks and two model families (§7). **IMPLAY** matches the performance of GRPO while requiring up to 47% fewer generated rollouts and has 12% faster training. Under the same generation budget, it outperforms Ex-GRPO, the state-of-the-art (SOTA) experience replay method for GRPO by up to 3.0 percentage points on harder benchmarks. To understand the interaction between **IMPLAY**'s components, we ablate importance-weighted advantage re-estimation and drift-aware eviction, showing that both components are necessary, with drift-aware eviction preventing degenerate importance weights and maintaining reliable advantage estimates during replay. Overall, our results show that experience replay can substantially improve the efficiency of GRPO when replay staleness is explicitly controlled.

2 PRELIMINARIES

Reinforcement Learning with Verifiable Rewards (RLVR). RLVR trains a policy using a reward $r(q, o)$ produced by a verifier typically with a binary success/failure reward. It is commonly used for mathematical reasoning and code generation, where the response's final answer can be checked programmatically against a ground-truth solution, e.g., by matching the extracted answer for math or by executing unit tests for code.

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) is a widely adopted critic-free policy gradient method. Instead of training a value network to serve as a baseline, it samples a group of responses $\mathcal{G}_q = \{o_i\}_{i=1}^{G}$ per prompt and uses reward statistics of this group to compute the advantage of each response following

$$\widehat{A}_i = \frac{r(q, o_i) - \mu_{\mathcal{G}_q}}{\sigma_{\mathcal{G}_q}} \tag{1}$$

where $\mu_{\mathcal{G}_q} = \frac{1}{G}\sum_{j=1}^{G} r(q, o_j)$ and $\sigma_{\mathcal{G}_q}$ are the mean and standard deviation of rewards in the group $\mathcal{G}_q$. The on-policy objective maximizes a clipped surrogate function:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{q,\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|q)} \left[\frac{1}{\sum_{i=1}^{G}|o_i|} \sum_{i=1}^{G}\sum_{t=1}^{|o_i|} \text{CLIP}\left(\rho_{i,t}(\theta, \theta_{\text{old}}), \widehat{A}_i\right) \right], \tag{2}$$

with

$$\begin{aligned} \text{CLIP}(\rho,\, \widehat{A}) &= \min\Big(\rho\, \widehat{A},\ \text{clip}\big(\rho,\, 1-\epsilon,\, 1+\epsilon\big)\, \widehat{A}\Big), \\ \text{clip}\big(\rho,\, 1-\epsilon,\, 1+\epsilon\big) &= \min\big(\max(\rho,\, 1-\epsilon),\ 1+\epsilon\big). \end{aligned} \tag{3}$$

$\rho_{i,t}(\theta, \theta_{\text{old}})$ is the importance sampling ratio explained below.

Following Dr. GRPO (Liu et al., 2025) and DAPO (Yu et al., 2026), we **omit the standard deviation normalization** in Eq. 1 and use the **token mean aggregation** in Eq. 2.

Importance Sampling. Importance sampling reweights samples drawn from one distribution to estimate expectations under another: given rollouts collected under a behavior policy $\pi_{\theta_{\text{beh}}}$, it corrects for the mismatch when evaluating the current policy π_θ.

In the RLVR setup with autoregressive language models, a rollout is a token sequence, so the ratio factorizes over tokens. The token-level importance sampling ratio is

$$\rho_{i,t}(\theta, \theta_{\text{beh}}) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{beh}}}(o_{i,t} \mid q, o_{i,<t})}, \tag{4}$$

and we define the length-normalized sequence-level importance ratio as

$$\rho_i(\theta, \theta_{\text{beh}}) = \left(\frac{\pi_\theta(o_i \mid q)}{\pi_{\theta_{\text{beh}}}(o_i \mid q)}\right)^{\frac{1}{|o_i|}} = \exp\left(\frac{1}{|o_i|}\sum_{t=1}^{|o_i|} \log \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{beh}}}(o_{i,t} \mid q, o_{i,<t})}\right). \tag{5}$$

Experience Replay. Experience replay retains the policy's rollouts generated during training in a replay buffer and reuses them in later optimization steps, rather than discarding them after a single update. In classical RL, it improves data efficiency by reusing past experience for multiple updates (Lin, 1992).

Given a replay ratio α, each training batch of problems $\mathcal{B}$ is partitioned into an on-policy subset $\mathcal{B}_{\text{on}}$, whose rollouts are freshly sampled from the current policy, and a replay subset $\mathcal{B}_{\text{replay}}$, whose rollouts are reused from the buffer. We define $\alpha = |\mathcal{B}_{\text{replay}}|/|\mathcal{B}|$ as the fraction of problems drawn from replay.

The training objective is then

$$\begin{aligned}\mathcal{J}_{\text{ER}}\left(\theta;\, \mathcal{M}_{\text{fresh}}, \mathcal{M}_{\text{replay}}\right) = \frac{1}{N_{\text{fresh}} + N_{\text{replay}}}\Bigg[&\sum_{o \in \mathcal{M}_{\text{fresh}}} \sum_{t=1}^{|o|} \text{CLIP}\left(\rho_{o,t}(\theta, \theta_{\text{old}}), \widehat{A}_o\right) \\ &+ \sum_{o \in \mathcal{M}_{\text{replay}}} \sum_{t=1}^{|o|} \text{CLIP}\left(\rho_{o,t}(\theta, \theta_{\text{beh}}(o)), \widehat{A}_o^{\text{replay}}\right)\Bigg],\end{aligned} \tag{6}$$

where $\mathcal{M}_{\text{fresh}}$ and $\mathcal{M}_{\text{replay}}$ are the fresh and replayed trajectories in the batch respectively, and $N_{\text{fresh}} = \sum_{o \in \mathcal{M}_{\text{fresh}}} |o|$, and $N_{\text{replay}} = \sum_{o \in \mathcal{M}_{\text{replay}}} |o|$. Here, $\widehat{A}_o^{\text{replay}}$ denotes the advantage assigned to a replayed rollout. Accordingly, the on-policy term takes the importance ratio $\rho_{i,t}$ with respect to $\pi_{\theta_{\text{old}}}$, whereas the replay term uses the behavior policy $\pi_{\theta_{\text{beh}}}$ that generated the rollout.

3 Related Work

Experience replay is a standard mechanism in off-policy reinforcement learning that reuses previously collected trajectories and thus improves sample efficiency by reducing the need for new environment interactions (Lin, 1992). Recent works have adapted experience replay to LLM RL post-training, especially with GRPO. These works differ in how they control the distribution shift of replayed trajectories and estimate their advantages. One group (Li et al., 2025; Yoo et al., 2026; Liang et al., 2025; Mao et al., 2026) reuses the advantage estimated when the replay rollout was generated. Among these, Yoo et al. (2026) prioritizes replaying samples with higher advantage magnitude, and restricts replay to recent trajectories to avoid stale advantages. **ReMix** (Liang et al., 2025) and **POPO** (Mao et al., 2026) both use current-to-behavior importance ratios to shape the clipping interval of the GRPO objective function.

The other group (Zhan et al., 2026; Guo et al., 2026) recomputes the advantage of the replay rollouts by combining them with newly generated on-policy trajectories to form a new reference group. **ExGRPO** (Zhan et al., 2026) identifies samples with medium-difficulty as most informative to replay and prioritizes those. **NexGRPO** (Guo et al., 2026) leverages the contrast between previous failed and successful rollouts enabling the policy to learn from its past mistakes. While both of these works refresh the advantage estimate of the replayed samples, they require fresh on-policy rollouts for every replayed trajectory and therefore substantially reduce the rollout savings from experience replay.

Overall, these methods handle off-policy replay either by using the advantage estimated when the replay rollout was generated (birth advantage) paired with strict age-based eviction to avoid staleness, or by regenerating fresh on-policy rollouts to obtain updated group-relative advantage estimates. In this work, we show that birth advantages can become stale rapidly and that age is a poor criterion for their staleness. These observations motivate **IMPLAY**, which uses importance sampling for re-estimating the advantage of the replay rollouts—without generating new on-policy ones—and for evicting stored rollouts that have gone too far off-policy.

4 ADVANTAGE STALENESS ANALYSIS

Replay uses trajectories collected under an earlier policy, whereas their learning signal should reflect the current policy. In this section, we quantify how the group-relative advantage of a stored trajectory changes between collection ("birth") and its later replay time.

Setup. We train Qwen3-4B-Base (Team, 2025) with GRPO on DeepScaleR-Preview (Luo et al., 2025). Each ordinary on-policy update uses a batch of 16 prompts and $G = 16$ rollouts per prompt. From each on-policy prompt group, we retain at most one successful trajectory in a replay buffer. We use this buffer only to *simulate* replay; no stored trajectory is ever inserted into an optimization batch. Thus, the GRPO updates remain entirely on-policy throughout the experiment. The buffer serves solely as a source of trajectories for asking the counterfactual question: *under the **current policy**, what advantage would this stored trajectory receive if it were replayed?*

To answer this question with low measurement noise, whenever a successful trajectory is retained, we generate an additional independent group of $1{,}024$ rollouts for its prompt under the collection-time policy. The mean reward of this diagnostic group defines its birth-time pass-rate estimate, $\hat{p}_{\mathrm{birth}}$. Beginning at training step 20, we uniformly sample eight stored trajectories from the buffer at each step. For every selected trajectory, we generate a new independent group of $1{,}024$ rollouts for the same prompt under the current policy, yielding its replay-time pass-rate estimate, $\hat{p}_{\mathrm{replay}}$. The large independent $1{,}024$-groups reduce noise enough to study trajectory-level changes in the underlying reward distribution rather than variation from the original $G = 16$ GRPO group.

For a selected stored trajectory with reward r, we define its birth- and replay-time group-relative advantages as

$$\hat{A}_{\mathrm{birth}} = r - \hat{p}_{\mathrm{birth}}, \qquad \hat{A}_{\mathrm{replay}} = r - \hat{p}_{\mathrm{replay}}, \qquad \Delta A = \hat{A}_{\mathrm{replay}} - \hat{A}_{\mathrm{birth}}.$$

A negative ΔA therefore means that the prompt has become easier for the current policy and that the birth-time advantage overestimates the trajectory's replay-time advantage.

Neither the selected stored trajectories nor the diagnostic rollouts contribute to backpropagation. The diagnostic rollouts are discarded after reward computation. This procedure produces 800 shadow-replay observations from training steps 21–120, spanning 543 distinct prompts. Further experimental details and the compute cost for this analysis are provided in Appendix B.1.

Birth-time advantages rapidly become stale as the policy improves. Figure 1(a) compares the advantage assigned to each buffered successful trajectory at generation time, $\hat{A}_{\mathrm{birth}}$, with the advantage it would receive under the current policy, $\hat{A}_{\mathrm{replay}}$ at replay time. Because every retained trajectory is successful, a negative $\Delta A = \hat{A}_{\mathrm{replay}} - \hat{A}_{\mathrm{birth}}$ means that the underlying prompt has become easier: its birth-time advantage overestimates the learning signal available when the trajectory is replayed. This discrepancy appears within only a few optimizer updates and grows steadily.

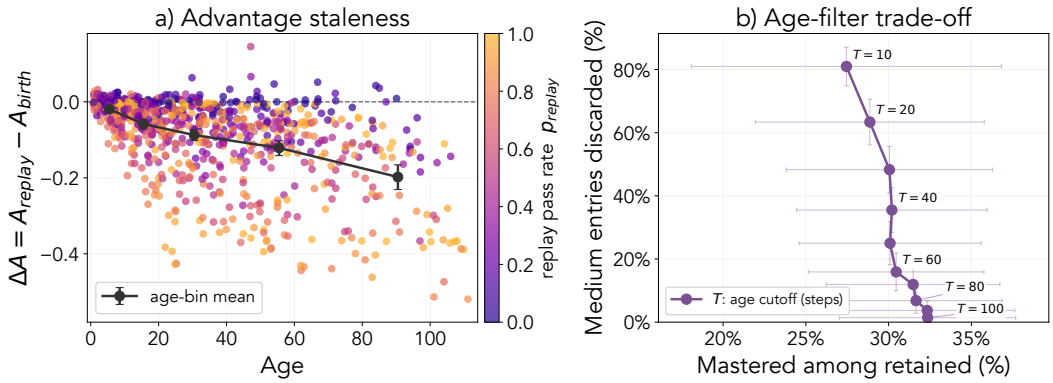


Figure 1: **Advantage staleness and the age-filter trade-off under simulated replay for Qwen3-4B-Base trained on DeepScaleR. (a)** Each point is a buffered successful trajectories selected for replay. The solid curve shows mean ΔA in age bins $[1, 10]$, $[11, 20]$, $[21, 40]$, $[41, 70]$, and $[71, 110]$. Point color denotes the replay-time pass rate, estimated from $1{,}024$ rollouts; **(b)** Trade-off induced by an age cutoff T. The horizontal axis reports the fraction of retained entries that are mastered, and the vertical axis reports the fraction of medium-difficulty entries discarded by the cutoff.

For trajectories replayed 11–20 steps after collection, the mean value of ΔA is -0.058 with the central 80% of changes ranging from -0.159 to 0.0. The mean drops to -0.087 by ages 21–40 with individual advantage changes ranging from -0.217 to -0.001. For trajectories older than 70 steps, ΔA is on average -0.198 and ranges from -0.377 to -0.043. Thus, birth-time advantages become stale within 11–20 optimizer updates and, on average, increasingly overestimate replay-time advantages as trajectories age. The error bars in Figure 1(a) show 95% prompt-clustered bootstrap confidence intervals.[2] Their narrow, entirely negative intervals show that the mean decrease is not driven by a few unusually stale prompts, but reflects a broad shift across the sampled prompt population. More uncertainty analyses are included in Appendix B.2.

Sample age is a poor criterion for replay rollout staleness. A strict age cutoff can reduce advantage staleness. For rollouts aged at most ten optimizer steps, the mean advantage change is $\Delta A = -0.019$ and the central 80% of advantage changes fall within -0.060 to 0.013. However, among all 800 data points, age only explains 24.4% of the variations in ΔA. The range of ΔA rapidly grows to $[-0.159, 0.0]$ by age 11–20 and to $[-0.311, -0.004]$ by 41–70. Thus, samples of comparable age can range from nearly unchanged to strongly stale, making age an unreliable criterion for selecting individual replay trajectories.

Moreover, a useful eviction rule should remove stale samples while keeping those that remain informative for the model. Prior work (Zhan et al., 2026) identifies medium-difficulty examples as particularly informative for replay. We define medium-difficulty prompts by their replay-time pass rate, $p_{\text{replay}} \in [0.2, 0.8]$, and mastered prompts as those with $p_{\text{replay}} > 0.8$. Figure 1 (b) shows the trade-off induced by an age cutoff T, which retains entries of age at most T. At $T = 10$, the cutoff discards 81.0% of medium-difficulty entries, while 27.5% of retained entries are mastered. Relaxing the cutoff to $T = 100$ raises the mastered fraction by only 4.9 percentage points, to 32.4%, but retains nearly all medium-difficulty entries in the buffer. Thus, strict age filtering cannot selectively remove stale or mastered samples, as it also discards the large majority of trajectories whose prompts remain at a potentially useful difficulty level.

Together, these results show that birth-time advantages quickly become stale and overestimate replayed rollouts' advantages under the current policy. Moreover, this staleness is rollout-specific, and age alone cannot reliably determine whether an individual replay rollout's stored advantage remains reliable. This motivates re-estimating advantages at replay time, and an eviction system that complements sample age, which we discuss next.

[2] See Appendix B.2 for details.

5 IMPLAY: IMPORTANCE WEIGHTED REPLAY ADVANTAGE ESTIMATION

5.1 EXPERIENCE BUFFER MANAGEMENT

We store the full group of successful *and* failed rollouts for groups that contain at least one successful rollout in the replay buffer $\mathcal{E}$. Each entry stores the prompt–response pair (q, o), its generation step (the birth time), the reward $r(q, o)$, the birth GRPO advantage $\widehat{A}_{birth}$, and the per-token behavior-policy log-probabilities $\log \pi_{\theta_{\text{beh}}}(o_t \mid q, o_{<t})_{t=1}^{|o|}$.

5.2 DRIFT-AWARE EVICTION

Replay methods typically bound staleness by a rollout's *age* (the number of gradient steps since it was generated) evicting via a maximum-age cap, an implicit proxy for how far the policy has moved since the rollout was produced (Yoo et al., 2026; Mao et al., 2026). In section 4 we showed that age filtering discards useful samples without removing advantage staleness.

We propose drift-aware eviction, which evicts samples based on their current-to-behavior importance sampling ratios. Let $\rho_i(\theta, \theta_{\text{beh}})$ be the sequence-level importance ratio from Eq. 5. Each time a rollout is replayed, we score it under the current policy and evict it when its length-normalized log-ratio exceeds a threshold:

$$|\log \rho_i(\theta, \theta_{\text{beh}})| > \tau. \tag{7}$$

Equivalently, we evict the rollout when $\rho_i(\theta, \theta_{\text{beh}})$ leaves $[e^{-\tau}, e^{\tau}]$. Because drift is observed only when a rollout is replayed, we additionally retain a loose maximum age cap to evict rollouts that are never sampled.

To prevent the replay buffer from collapsing to a low number of rollouts, we consider an eviction guard and do not evict a sample if the buffer size drops below a predefined minimum size (which we set to 1,000 in our experiments). We nevertheless keep a maximum tolerated drift τ_{high}. Rollouts with $|\log \rho_i(\theta, \theta_{\text{beh}})| > \tau_{high}$ are evicted regardless of the buffer size.

5.3 IMPORTANCE-WEIGHTED REPLAY ADVANTAGE ESTIMATION

To account for the drift between the behavior policy that generated the cohort and the current policy, we re-estimate the mean reward of the replay rollout's cohort using self-normalized importance sampling (SNIS). Let $\rho_i(\theta, \theta_{\text{beh}})$ be the sequence-level importance ratio from Eq. 5. This per-token average can be numerically small, making the resulting SNIS weights nearly uniform and leaving the re-estimated baseline close to the baseline computed when the rollout was generated. We therefore multiply the mean per-token log-ratio by a fixed coefficient:

$$s_i(\theta) = c \log \rho_i(\theta, \theta_{\text{beh}}), \qquad w_i(\theta) = \exp(s_i(\theta)) = \rho_i(\theta, \theta_{\text{beh}})^c. \tag{8}$$

$$s_i(\theta) = \frac{c}{|o_i|} \sum_{t=1}^{|o_i|} \log \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{\text{beh}}}(o_{i,t} \mid q, o_{i,<t})}. \tag{9}$$

Here, c is a fixed scaling hyperparameter. It controls the sharpness of the SNIS weights. Larger values amplify differences in average per-token drift, allowing importance weighting to have a larger effect on the re-estimated baseline; smaller values produce weights closer to uniform.

To prevent a single trajectory from dominating the estimate, we apply cohort-relative clipping in log-weight space:

$$\widetilde{s}_i(\theta) = \max\left(s_i(\theta), \max_{j \in \{1,\dots,G\}} s_j(\theta) - C\right), \qquad \widetilde{w}_i(\theta) = \exp(\widetilde{s}_i(\theta)), \tag{10}$$

where $C \geq 0$ is a clipping hyperparameter. It specifies the largest allowed difference between the highest and lowest log-weights within a cohort. After clipping, every trajectory has an unnormalized

weight at least $\exp(-C)$ times that of the highest-weight trajectory. The drift-corrected cohort baseline is

$$\widehat{\mu}_{\mathcal{G}_q}^{\mathrm{SNIS}}(\theta) = \frac{\sum_{i=1}^{G} \widetilde{w}_i(\theta) r(q, o_i)}{\sum_{i=1}^{G} \widetilde{w}_i(\theta)}. \tag{11}$$

We then assign each replayed trajectory the advantage

$$\widehat{A}_i^{\mathrm{replay}} = r(q, o_i) - \widehat{\mu}_{\mathcal{G}_q}^{\mathrm{SNIS}}(\theta). \tag{12}$$

Note that only **successful** rollouts are replayed in the policy-gradient objective. However, for each replayed successful rollout, we retain and score every member of its original generation group, including failed rollouts, using Eq. 10, in order to re-estimate the current-policy group baseline using Eq. 11. The failed rollouts in the reference-group do not themselves contribute replay policy-gradient terms. Eq. 12 replaces the stale behavior-policy advantage with an estimate targeted at the current policy, without generating additional on-policy sibling rollouts, unlike ExGRPO's fresh-sibling estimator (Zhan et al., 2026).

6 Experimental Setup

Data and Evaluation. We use DeepScaleR-Preview (Luo et al., 2025) as our training dataset. For evaluation we use six math reasoning benchmarks with varying difficulty levels: AIME24/25/26, MATH-500 (Hendrycks et al., 2021), MinervaMath (Lewkowycz et al., 2022) and OlympiadBench (He et al., 2024). We report `avg@32` over 32 independent runs for all benchmarks.

Models and Baselines. We use Qwen3-4B-Base (Team, 2025), and Olmo-3-7B-Instruct-SFT (Olmo et al., 2025) in our experiments. We compare **ImPlay** against GRPO without experience replay using problem batch sizes of 8 and 16, as well as ExGRPO (Zhan et al., 2026). For both **ImPlay** and ExGRPO, we use a problem batch size of 16, with half of each batch allocated to on-policy updates and the other half to experience replay. For the same number of optimizer steps, GRPO with batch size 16 and ExGRPO generate around $2\times$ and $1.94\times$ as many rollouts as **ImPlay**, respectively. We therefore treat these methods as upper bounds in terms of rollout budget, while GRPO with batch size 8 serves as a lower bound for **ImPlay**. For a controlled comparison under the same experimental setup, we reimplement ExGRPO with its difficulty-based question selection, entropy-based rollout selection, policy shaping, and eviction of mastered items from the replay buffer, and report results from this implementation.

Training Details. All models are trained on 8 H100 GPUs. We only replay successful rollouts, use a replay factor of $\alpha = 0.5$, and sample from the replay buffer uniformly at random. Unless otherwise specified, the age cutoff is 100 optimizer steps, $\tau = 0.001$, and $\tau_{high} = 0.003$ in Eq. 7. For the replay advantage estimation, we use $c = 512$ in Eq. 8, and $C = 2$ in Eq. 10.[3] We consider a warmup period of 20 steps where no sample is replayed, but successful samples are collected in the buffer.

More experimental details are included in Appendix A.

7 Results and Analysis

7.1 Main Results: RLVR Performance and Efficiency

We compare **ImPlay** with baselines both at a near-matched generated-rollout budget and at a fixed 500-step training, separating rollout efficiency from final accuracy.

ImPlay can match GRPO's performance at only 48% of the generation cost, while reducing training time by 12%. Figure 2 reports the average test accuracy on AIME benchmarks as a function of training steps, and cumulative generated rollouts. On Qwen3-4B-Base, **ImPlay** closely tracks and, slightly exceeds, GRPO with batch size 16 while using substantially fewer rollouts. At step 500, **ImPlay** achieves 20.7% average AIME accuracy using 66.6K rollouts, compared with 19.1% for GRPO with batch size 16 using 128.0K rollouts. **ImPlay** also matches ExGRPO at the final

[3] Appendix C.3 includes a hyperparameter sensitivity analysis for c and τ.

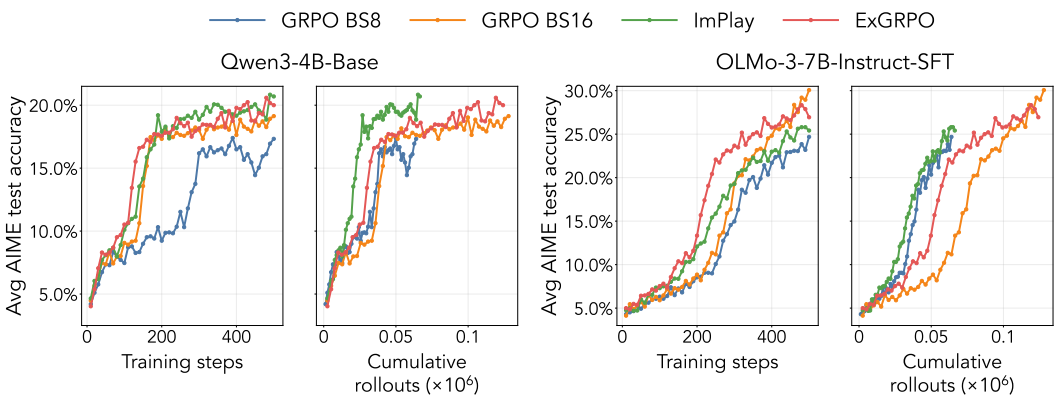


Figure 2: **Average test accuracy for AIME benchmarks across training steps, and generation budget** for Qwen3-4B-Base Olmo-3-7B-Instruct-SFT trained on DeepScaleR.

Table 1: **Final test accuracy (%) for Qwen3-4B-Base and Olmo-3-7B-Instruct-SFT trained on DeepScaleR for 500 steps.** Except for the base models, we report the average performance over the last three checkpoints. Avg. is the macro-average over the six benchmarks. The best and second-best results are shown in **bold** and underlined, respectively.

Method	AIME24	AIME25	AIME26	MATH	Minerva	Olympiad	Avg.
Qwen3-4B-Base							
Qwen3-4B-Base	4.79	1.98	4.17	38.47	7.77	13.84	11.84
GRPO BS=8	18.06	17.19	14.76	78.26	34.96	30.41	32.27
GRPO BS=16	20.80	17.99	16.01	79.55	**36.67**	32.59	33.93
ExGRPO	20.97	**20.87**	16.11	**80.66**	35.97	**32.92**	**34.58**
ImPlay	**21.81**	18.72	**18.19**	79.45	36.19	32.44	34.47
Olmo-3-7B-Instruct-SFT							
Olmo-3-7B-Instruct-SFT	5.21	2.92	2.92	55.79	17.00	18.39	17.04
GRPO BS=8	23.02	21.56	16.77	75.90	27.36	31.99	32.77
GRPO BS=16	27.50	24.20	**22.53**	78.43	28.00	33.82	35.75
ExGRPO	**28.02**	**26.70**	21.67	**79.67**	**28.93**	**35.06**	**36.68**
ImPlay	25.38	23.47	19.44	77.46	27.52	33.56	34.47

step (20.7% versus 20.0%), while using 46% fewer rollouts (66.6K versus 124.2K), and outperforms it by 2.15 percentage points under the same generation budget of 66K rollouts.

Olmo-3-7B-Instruct-SFT exhibits a different accuracy–efficiency trade-off from Qwen3-4B-Base. GRPO with batch size 8 is already highly rollout-efficient, and first reaches 20.0% average AIME accuracy after generating 44.8K rollouts, whereas GRPO with batch size 16 requires 79.4K rollouts to reach the same threshold. Thus, GRPO with batch size 8 uses 44% fewer rollouts, leaving less room for **ImPlay** to improve rollout efficiency relative to this baseline. Here, **ImPlay** tracks GRPO with batch size 16 only until approximately step 330, after which it plateaus. ExGRPO shows a similar plateau but later near step 480. At step 500, GRPO with batch size 16 achieves the highest accuracy (30.1%), followed by ExGRPO (26.9%), **ImPlay** (25.4%), and GRPO with batch size 8 (24.7%). **ImPlay** remains more rollout-efficient than ExGRPO: at the same generation budget of 66.3K rollouts, it outperforms ExGRPO by 3.05 percentage points.

Table 1 reports the final test accuracy for Qwen3-4B-Base and Olmo-3-7B-Instruct-SFT. On both models, **ImPlay** outperforms GRPO with batch size 8 on every benchmark. Averaged across benchmarks, **ImPlay** improves over GRPO by 2.20 and 1.70 percentage points (pp) on Qwen3-4B-Base and Olmo-3-7B-Instruct-SFT, respectively; gains are larger on the more challenging AIME benchmarks, reaching 2.90 and 2.31 points. On Qwen3-4B-Base, **ImPlay** matches the performance of GRPO with batch size 16 and ExGRPO while using roughly half as many on-policy rollouts.

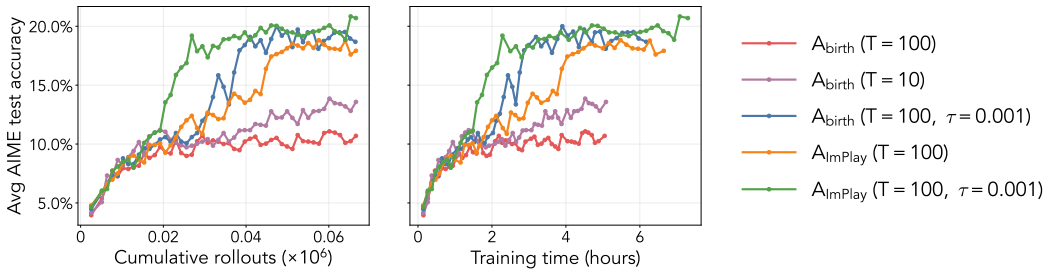


Figure 3: **Ablation of IMPLAY's components.** We report the average accuracy for AIME benchmarks across generation budget, and wall clock time for Qwen3-4B-Base trained on DeepScaleR with batch size 16. A_{birth} refers to using birth advantage, and $A_{\textbf{IMPLAY}}$ refers to using **IMPLAY**'s importance-weighted advantage. T and τ refer to age-based and drift-ware eviction thresholds.

Overall, IMPLAY's benefit depends on how the batch size shapes GRPO's accuracy–efficiency trade-off. On Olmo-3-7B-Instruct-SFT, GRPO with batch size 8 has a large rollout-efficiency advantage over batch size 16, leaving less room for experience replay to improve efficiency beyond the GRPO with batch size 8 baseline without sacrificing final accuracy. In contrast, on Qwen3-4B-Base, GRPO with batch sizes 8 and 16 closely track one another at a matched rollout budget, allowing **IMPLAY** to leverage replay to match, and slightly exceed, the performance of GRPO with batch size 16 while using approximately 48% fewer rollouts and 12% less training time. Next, we study how each **IMPLAY** component contributes to this efficiency gain.

7.2 ABLATION

To isolate the effects of replay advantage estimation and eviction, we first compare birth advantage, A_{birth}, with our importance-weighted replay advantage, $A_{\textbf{IMPLAY}}$, under age-based eviction with maximum age $T = 100$. Then we consider a stricter age limit of $T = 10$, and drift-aware eviction. Figure 3 reports average AIME accuracy as a function of cumulative generated rollouts and wall-clock training time. At step 500, A_{birth} with standard age-based eviction ($T = 100$) reaches only 10.7%. Tightening the age threshold to $T = 10$ improves accuracy to 13.6% (+2.9 pp), but still remains below GRPO with batch size 8 (17.3%) by 3.8 pp. In contrast, drift-aware eviction raises the performance to 18.7% (+8.0 pp).[4] These results are aligned with our findings in section 4: age alone is a poor criterion for replay rollout staleness. They also show that our proposed drift-aware eviction is substantially more effective than imposing a strict age threshold.

Independently, replacing A_{birth} with $A_{\textbf{IMPLAY}}$ under standard age-based eviction ($T = 100$) raises accuracy from 10.7% to 17.9% (+7.2 pp), demonstrating the benefit of importance-weighted advantage re-estimation. Appendix C.2 (Figure 5) shows that $A_{\textbf{IMPLAY}} - A_{birth}$ ranges between -0.013 and -0.312, which is aligned with our findings in section 4. Combining $A_{\textbf{IMPLAY}}$ with drift-aware eviction yields the best result, 20.7%, improving by 2.8 pp over $A_{\textbf{IMPLAY}}$ with age-based eviction and by 2.0 pp over A_{birth} with drift-aware eviction. These gains indicate that advantage re-estimation and drift-aware eviction are complementary. In Appendix C.1, we further show that drift-aware eviction maintains less degenerate SNIS weights in the buffer, reducing the need for corrective clipping.

8 CONCLUSION

In this work, we study experience replay in GRPO and show that replay advantages can become substantially stale after only a small number of optimizer steps, and that sample age alone is a poor proxy for this drift. Motivated by these findings, we introduce **IMPLAY**, an experience replay algorithm for GRPO that reduces the cost of on-policy generation by up to 47%. Our results show that carefully controlling replay staleness can make experience replay an effective tool for reducing the computational cost of GRPO post-training. We discuss limitations and directions for future work in Appendix D.

[4] Appendix C.2 (Figure 5) shows an average eviction rate of 36.2%.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods, finding the existing literature, providing feedback on research methodology and experiments, and editing the manuscript to improve readability. We have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in writing of proofs, supporting qualitative and thematic data analysis, and interpreting results, and generating synthetic datasets, assisting with translation, and cleaning and reformatting datasets are not applicable to this work. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness, claims and citations in LLM-generated literature search were checked for hallucinations, and LLM-generated re-writing suggestions were carefully verified to match the experimental results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on foundational research in LLM post-training and is not tied to any specific deployed application. We do not release new models or datasets as part of this work, limiting the direct deployment risks associated with our contributions. All models and datasets used in our experiments are publicly available under their respective Apache 2.0, and MIT licenses.

REPRODUCIBILITY STATEMENT

We provide the models and datasets used for training and evaluation in section 6. Additional experimental details including training and evaluation hyper-parameters, and compute resources can be found in Appendix A.

REFERENCES

Jialiang Guo, Fucheng Xiong, Xu He, Haodong Zhao, Ke Zeng, Xunliang Cai, et al. Turning failures into value: Negative experience replay for rlvr via confidence gating and boundary failure sampling. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36316–36334, 2026.

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.

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

Woojeong Kim, Ziyi Yang, Jing Nathan Yan, and Jialu Liu. Spend your rollouts where it counts: Rollout allocation for group-based rl post-training. *arXiv preprint arXiv:2605.26606*, 2026.

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.

Siheng Li, Zhanhui Zhou, Wai Lam, Chao Yang, and Chaochao Lu. Repo: Replay-enhanced policy optimization. *arXiv preprint arXiv:2506.09340*, 2025.

Jing Liang, Hongyao Tang, Yi Ma, Jinyi Liu, Yan Zheng, Shuyue Hu, Lei Bai, and Jianye Hao. Squeeze the soaked sponge: Efficient off-policy reinforcement finetuning for large language model. *International Conference on Learning Representations*, 2025.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Yixiu Mao, Yun Qu, Qi Wang, Heming Zou, and Xiangyang Ji. Rlvr without ineffective samples: Group prioritized off-policy optimization for llm reasoning. *arXiv preprint arXiv:2606.01281*, 2026.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *International Conference on Learning Representations (ICLR)*, 2015.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Qwen Team. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Gyeongtae Yoo, Sanghyeok Park, Soohyuk Jang, Ik-hwan Kim, and Sungroh Yoon. Rollout-level advantage-prioritized experience replay for grpo. *arXiv preprint arXiv:2606.04560*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek Wong, and Yu Cheng. Exgrpo: Learning to reason from experience. In *International Conference on Learning Representations*, volume 2026, pp. 12810–12846, 2026.

A Additional Experimental Details

Table 2 summarizes the hyperparameters used for training and evaluation.

Table 2: RLVR training and evaluation hyperparameters used in our experiments. All sequence lengths are measured using each model's own tokenizer.

Module	Parameter	Value	Description
Data	`train_batch_size`	16/8	Unique prompts sampled per optimization step.
	`max_prompt_length`	2048	Maximum input length.
	`max_response_length`	8192	Maximum response length.
Actor	`optim.lr`	1×10^{-6}	Learning rate for the actor optimizer.
	`ppo_mini_batch_size`	16/8	Actor batch size for one gradient step.
	`ppo_micro_batch_size_per_gpu`	4	Micro-batch size per GPU.
	`ppo_epochs`	1	PPO update epochs per batch.
	`clip_ratio`	0.2	Symmetric PPO clipping ratio ϵ.
	`entropy_coeff`	0.0	Entropy regularization coefficient.
Algorithm	`norm_adv_by_std_in_grpo`	False	Disable std-normalization of advantages.
	`kl_ctrl.kl_coef`	0.0	KL penalty coefficient (no KL).
Rollout	`temperature`	1.0	Sampling temperature during rollouts.
	`top_p`	1.0	Nucleus sampling threshold.
	`n`	16	Number of rollouts per prompt (group size G).
Trainer	`total_training_steps`	500	Number of training steps.
	`test_freq`	10	Validation frequency (steps).
	`save_freq`	100	Checkpoint frequency (steps).
Evaluation	`val_kwargs.n`	32	Samples per prompt (Mean@32 / Pass@32).
	`val_kwargs.temperature`	1.0	Sampling temperature during validation.
	`val_kwargs.top_p`	1.0	Nucleus threshold during validation.

B Details on Advantage Staleness Analysis

B.1 Experimental Setup and Compute Cost

This analysis is deliberately a high-compute diagnostic. For a training batch of $B = 16$ prompts with GRPO group size $G = 16$, ordinary training generates $B \times G = 256$ rollouts per optimizer step. The diagnostic additionally generates $B \times 1{,}024 = 16{,}384$ rollouts at collection time and, after warmup, $0.5B \times 1{,}024 = 8{,}192$ other rollouts. Thus, after warmup, the analysis adds $96\times$ the number of ordinary training rollouts, for $97\times$ as many generated rollouts in total; probe outputs are discarded before the GRPO loss and incur no additional backward pass.

In terms of number of generated tokens, the experiment consumed 2.850B response tokens compared with 32.8M tokens for their ordinary on-policy rollouts. Consequently, diagnostic generation accounts for 98.9% of the 6.90B response tokens generated across the two analysis runs. This cost

buys low-variance and is incurred only for this controlled advantage staleness analysis, not when applying replay-time advantage re-estimation.

The run was executed on 8 H100 GPUs for 120 steps and took 118.87 hours overall. Other training details follow Table 2.

B.2 Estimating uncertainty

To assess whether the observed staleness is driven either by the particular trajectories selected from the buffer or by finite-rollout noise in estimating their group-relative advantages, we conduct several complementary uncertainty analyses.

Uncertainty of age-binned advantage changes. The error bars in Figure 1(a) quantify uncertainty from observing a finite set of prompts in the replay diagnostic. For each age bin, we compute the mean advantage change, $\Delta A = \hat{A}_{\text{replay}} - \hat{A}_{\text{birth}}$, over all diagnostic observations in that bin. Because the same prompt can be sampled for shadow replay more than once, we use a prompt-cluster bootstrap rather than resampling individual observations.

Specifically, for an age bin containing N distinct prompts, each bootstrap replicate draws N prompts with replacement from the observed prompts. Whenever a prompt is drawn, we include all of its observations in that age bin; prompts drawn multiple times contribute their observations multiple times. We compute the mean ΔA of the resulting resampled set and repeat this procedure $10{,}000$ times. The plotted point is the mean computed from the original observations, and the error bar spans the 2.5th to 97.5th percentiles of the $10{,}000$ bootstrap means.

Thus, narrow intervals indicate that the age-binned mean advantage change would remain similar if the diagnostic had sampled a different collection of prompts. This procedure does not correct a systematically biased sampling strategy; its interpretation relies on the shadow-replay entries being sampled uniformly from the eligible replay buffer.

Finite-rollout uncertainty in individual advantage changes. For a retained successful rollout with reward $r = 1$, we estimate its birth- and replay-time advantages from independent groups of $n = 1{,}024$ rollouts. Let $\hat{p}_{\text{birth}}$ and $\hat{p}_{\text{replay}}$ denote the corresponding empirical pass rates. The estimated advantages are

$$\hat{A}_{\text{birth}} = 1 - \hat{p}_{\text{birth}}, \qquad \hat{A}_{\text{replay}} = 1 - \hat{p}_{\text{replay}},$$

and therefore

$$\Delta A = \hat{A}_{\text{replay}} - \hat{A}_{\text{birth}} = \hat{p}_{\text{birth}} - \hat{p}_{\text{replay}}.$$

Treating the two probe groups as independent Bernoulli samples, we estimate the standard error of this difference using the plug-in binomial estimator,

$$\text{SE}(\Delta A) = \sqrt{\frac{\hat{p}_{\text{birth}}(1-\hat{p}_{\text{birth}})}{n} + \frac{\hat{p}_{\text{replay}}(1-\hat{p}_{\text{replay}})}{n}}.$$

We count a negative advantage change as larger than its probe-estimation uncertainty when

$$\Delta A < -1.96\,\text{SE}(\Delta A).$$

Equivalently, the two-sided 95% normal-approximation interval for the change excludes zero in the negative direction. Under no true change, this directional criterion has a 2.5% false-positive probability. At pass rates near 0.5, its threshold is approximately 4.3 percentage points for two groups of $1{,}024$ rollouts.

The negative advantage change exceeds this margin for 57% of observations aged 11–20 steps and 68% of those aged 21–40 steps. Thus, for a substantial fraction of individual trajectories, the observed staleness is larger than can be attributed to the finite number of rollouts used to estimate their pass rates.

Uncertainty in the age-filter trade-off. For each age cutoff T, we compute two quantities from the original replay observations: the fraction of retained entries that are mastered,

$$x_T = \frac{\#\{\text{mastered entries with age } \le T\}}{\#\{\text{entries with age } \le T\}},$$

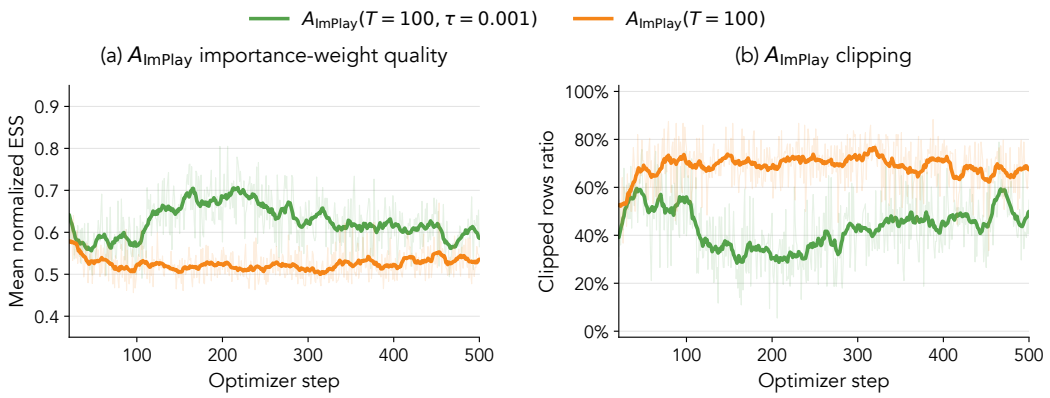


Figure 4: **Impact of drift-aware eviction on policy-ratio weights for replay advantage re-estimation.** (a) Mean normalized effective sample size (ESS) of the clipped SNIS policy-ratio weights within each replayed cohort; higher values indicate less concentrated weights. (b) Fraction of scored reference-group rollout rows whose log-weights are modified by the clipping rule in Eq. 10; lower values indicate less frequent clipping intervention. We compare **ImPlay** with ($A_{\textbf{ImPlay}}/\tau = 0.001$) and without ($A_{\textbf{ImPlay}}$) drift-aware eviction using Qwen3-4B-Base trained on DeepScaleR.

and the fraction of medium-difficulty entries discarded by the cutoff,

$$y_T = \frac{\#\{\text{medium-difficulty entries with age } > T\}}{\#\{\text{medium-difficulty entries}\}}.$$

The solid curve in Figure 1(b) shows these points computed from the original diagnostic observations.

We estimate uncertainty using the same prompt-cluster bootstrap as for the age-binned advantage changes. Specifically, we draw the same number of distinct prompts as observed in the full diagnostic set, with replacement, and retain all observations associated with every drawn prompt. Each bootstrap replicate therefore produces a complete trade-off curve, (x_T, y_T), for all displayed cutoffs. We repeat this procedure $10{,}000$ times.

Because the plotted cutoffs are evaluated on the same observations, their uncertainties are correlated. We therefore construct 95% simultaneous bands separately for the horizontal and vertical coordinates: the bands are calibrated so that, in approximately 95% of bootstrap replicates, the full set of values $\{x_T\}_T$ (or, separately, $\{y_T\}_T$) lies within its corresponding error bars. The horizontal and vertical error bars in the figure show these simultaneous bands. Unlike the error bars for the age-binned advantage means, which provide a separate 95% interval for each bin, the trade-off error bars are simultaneous across all displayed cutoffs. They therefore quantify uncertainty in the *entire* trade-off curve and account for considering multiple age cutoffs at once.

The large overlap of the horizontal intervals confirms that relaxing the cutoff produces only a small change in the mastered fraction among retained entries, whereas the vertical axis shows that strict cutoffs discard a large and reliably estimated fraction of medium-difficulty entries.

C Additional Analyses

C.1 Impact of drift-aware eviction

In this section, we study whether drift-aware eviction preserves well-conditioned policy-ratio weights for replay advantage re-estimation. For each re-estimated replay cohort G_q, let $\widetilde{w}_i(\theta)$ denote the clipped SNIS weight of rollout i (see Eq. 10). We measure the normalized effective sample

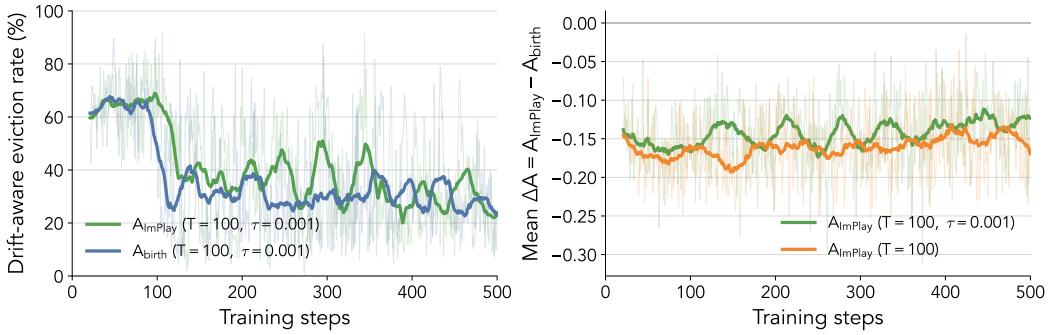


Figure 5: **Left:** Ratio of the evicted rollouts when using $A_{\textbf{IMPLAY}}$ vs A_{birth}. **Right:** Mean difference between $A_{\textbf{IMPLAY}}$ and A_{birth} when using only age-based eviction vs adding drift-aware eviction. All runs correspond to Qwen3-4B-Base trained on DeepScaleR with batch size 16.

size (ESS) as

$$\mathrm{ESS}_{G_q}^{\mathrm{norm}}(\theta) = \frac{1}{G}\frac{\left(\sum_{i=1}^{G}\widetilde{w}_i(\theta)\right)^2}{\sum_{i=1}^{G}\widetilde{w}_i(\theta)^2}, \tag{13}$$

where G is the size of the original reference group. This quantity equals one for uniform weights and decreases as the weight mass concentrates on a small number of rollouts.

Figure 4(a) compares the mean normalized ESS for $A_{\textbf{IMPLAY}}$ with and without drift-aware eviction. Both conditions use the same age-100 eviction; the drift-aware condition additionally evicts replay items whose length normalized sequence-level importance sampling ratio (Eq. 5) exceeds $\tau = 0.001$. Across the optimizer steps, drift-aware eviction has a mean normalized ESS of 0.627, compared with 0.525 for age-only eviction. This indicates that drift-aware eviction maintains a higher quality of replay cohorts in the buffer.

Figure 4(b) reports the fraction of scored reference-group rollout rows for which the cohort-relative clipping mechanism in Eq. 10 is active. Without drift-aware eviction, 68.9% of scored rollout rows are clipped on average. Drift-aware eviction reduces this to 43.2%.

Together, these results show that drift-aware eviction keeps the replay buffer in a regime where SNIS advantage re-estimation uses more diffuse weights with higher effective sample size policy-ratio and requires substantially less corrective clipping.

C.2 Eviction and replay advantage analyses

Figure 5 (left) shows the eviction rate for $A_{\textbf{IMPLAY}}$ and A_{birth} with drift-aware eviction Before the age cap becomes active at step 100, roughly 65% of sampled replay items in both runs exceed the log-ratio threshold. Once samples can also age out, this log-ratio-triggered fraction falls to 35.9% for $A_{\textbf{IMPLAY}}$ and 30.5% for A_{birth}, averaged over steps 100–500.

Figure 5 (right) shows the mean advantage change, $\Delta A = A_{\textbf{IMPLAY}} - A_{\mathrm{birth}}$. Per-step estimates range from -0.312 to -0.013 across the two methods with ratio-aware eviction having lower ΔA overall. This is expected as it evicts more drifted samples than age-only eviction. As a result, replayed samples have smaller advantage change.

C.3 Hyperparameter sensitivity

We study the coefficient c in Eq. 8 and the drift-eviction threshold τ for **IMPLAY** on Qwen3-4B-Base, with $C = 2$, $\tau_{\mathrm{high}} = 0.003$, and a maximum age of 100 steps. AIME denotes the macro-average over AIME24, AIME25, and AIME26.

The results show that $c = 256$ consistently underperforms the larger reference horizons, suggesting that this setting produces reweighting that is too weak. The largest tested value, $c = 1024$, achieves

Table 3: AIME avg@32 (%) at three training checkpoints for different reference-horizon coefficients c and drift-eviction thresholds τ.

c	τ	AIME@300 ↑	AIME@400 ↑	AIME@500 ↑
1024	0.00100	**20.38**	**19.65**	18.44
512	0.00075	17.88	17.67	17.67
512	0.00150	18.68	18.89	**19.51**
256	0.00100	11.28	12.53	11.81

the strongest early AIME accuracy but does not maintain this advantage through step 500. Among the evaluated configurations, $c = 512$ with $\tau = 0.0015$ achieves the strongest final AIME accuracy.

At $c = 512$, relaxing the eviction threshold from $\tau = 0.00075$ to $\tau = 0.0015$ improves AIME at every reported checkpoint. These results therefore support an intermediate coefficient c and a moderately permissive drift threshold.

D Limitations and Future Work

Evaluation scope. Our study focuses on mathematical reasoning with text-only models. Whether **ImPlay** yields similar accuracy–efficiency trade-offs in other RLVR domains, such as code generation, and in multimodal settings remains an open question. We also evaluate models with up to 7B parameters; the behavior of replay, policy drift, and the resulting efficiency gains at larger scales has yet to be established.

We focus on problem batch sizes of 8 and 16 because experience replay is most practically motivated when rollout generation is compute-constrained: it can fill part of an optimization batch with previously generated trajectories, reducing the number of newly generated rollouts. Because GRPO's accuracy–generation trade-off varies with batch size, **ImPlay**'s relative benefit may also differ in larger-batch regimes. Evaluating a broader range of batch sizes, particularly settings enabled by greater generation capacity, is needed to characterize this dependence.

Because post-training is computationally expensive, each training configuration is evaluated with a single random seed. Averaging the final three checkpoints reduces sensitivity to checkpoint-level variation, but it does not measure variation across independent training runs. Future work should therefore evaluate **ImPlay** across multiple seeds and report uncertainty for the accuracy–generation trade-off.

Engineering design choices. **ImPlay** uses several fixed design choices, including the reference-horizon coefficient c that scales the length-normalized log-likelihood ratio in Eq. 8, cohort clipping threshold C in Eq. 10, drift threshold τ, buffer-size guard, and maximum-age cap. We use the same settings across the two evaluated model families, and perform a hyperparameter sensitivity analysis over c and τ, but do not characterize their sensitivity across datasets, scales, or training regimes. These choices are empirically motivated, and we do not provide theoretical guarantees that the resulting clipped, length-normalized weighting scheme recovers the current-policy group baseline. Developing automatic or more theoretically grounded choices for these components is an important direction for future work.

System costs. Although **ImPlay** reduces the number of generated rollouts, it stores behavior-policy log-probabilities for complete reference groups and re-scores those groups at replay time. The resulting memory, forward-pass, and wall-clock trade-offs may vary with group size, response length, buffer size, model scale, and training system. Our end-to-end timing results on a single hardware configuration do not fully characterize these system-level trade-offs.