# ENDOGENOUS FORGETTING IN RL POST-TRAINING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

A central challenge in scaling reinforcement learning (RL) for LLM post-training is how models continue to learn new behaviors while preserving previously acquired ones. We study *endogenous forgetting*—where a model loses previously accessible behaviors due to non-stationarity that arises because the evolving model determines the effective data distribution—as a potential barrier to scaling on-policy RL. We first isolate endogenous forgetting via a controlled experiment: we collect successful traces from an RL run and show that supervised learning on these traces in temporal order leads to substantially more forgetting and lower solution coverage than training with a shuffled order. We then investigate successful-trajectory replay as a way to mitigate endogenous forgetting. Across multiple reasoning tasks and model families, replay reduces forgetting and prompt-level performance oscillations, achieves comparable or higher final pass@1, and substantially improves solution coverage relative to standard RL.

## 1 INTRODUCTION

Reinforcement learning (RL) has become a key approach for post-training to improve the reasoning capabilities of large language models. At the same time, recent evidence shows that RL exhibits several undesirable behaviors. Reasoning coverage can shrink, with large-$k$ pass@$k$ failing to exceed that of the base model, and prolonged training can suffer from entropy collapse or premature convergence (Yue et al., 2025; Nguyen et al., 2025; Liu et al., 2026; Team et al., 2025).

Degradation in coverage is notably reminiscent of catastrophic forgetting, which has been studied extensively within the context of continual learning (Parisi et al., 2019). This literature considers a model trained on tasks in sequence, such that optimization on the current dataset deteriorates the model's capability on preceding tasks (McCloskey & Cohen, 1989). This behavior is not ideal, but it is also not unexpected, as SGD's guarantees are only with respect to the current training distribution. As such, catastrophic forgetting is an *exogenous* phenomenon, in that it is a consequence of fitting parameters on a user-specified task sequence that deviates from the stationary setting for which the underlying optimization machinery was designed.

This paper considers a more subtle version of forgetting in language model post-training, occurring even when RLVR is carried out on a fixed training dataset, with prompts sampled i.i.d. at each optimization step. Even in this benign setting, the on-policy nature of RL induces non-stationarity, both in the distribution of prompts receiving a training signal and in the distribution of responses being reinforced. This induces an endogenous curriculum, skewing the effective training distribution in a way that can degrade model capabilities.

non-stationarity induced by on-policy RL is considerably problematic in algorithms like GRPO, where prompts that the model reliably solves yield zero group-relative advantage and hence contribute zero gradient. These prompts yield no training signal, but can suffer due to interference from updates driven by active, gradient-contributing prompts, with no counteracting force to restore them. This can have two negative consequences. First, the policy may drift slowly from previously mastered prompts, which oscillate between being learned and unlearned, damaging post-training sample efficiency. Alternatively, and more severely, the policy might rapidly move so far away that it is unable to produce rewarding sequences even occasionally. Because recovery would require sampling a correct rollout, which the updated policy cannot, the model does not regain the capability.

We refer to these phenomena as **endogenous forgetting**. **Specifically, we define endogenous forgetting as the erosion of previously acquired capabilities caused by drift that the learning algo-**



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.

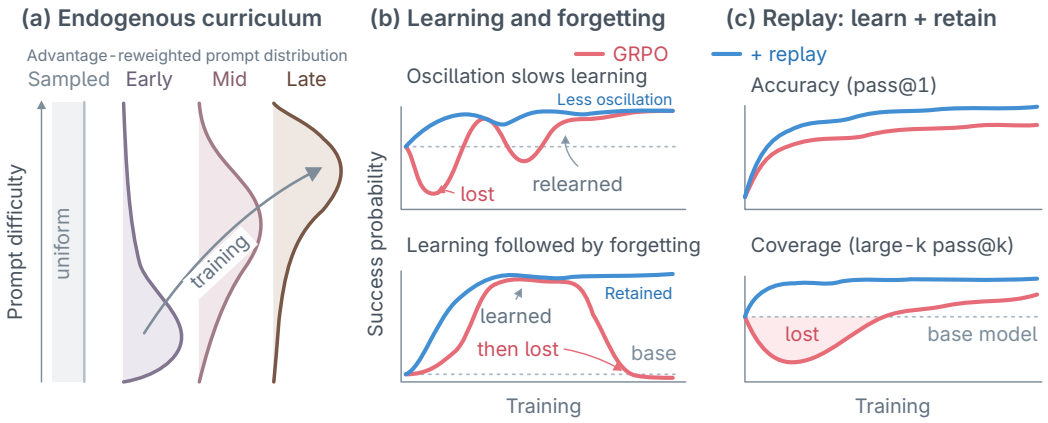


Figure 1: **Endogenous forgetting in RL post-training. (a)** Despite uniform sampling from a fixed dataset, the evolving policy changes which prompts provide nonzero advantages, inducing an endogenous curriculum. **(b)** This non-stationarity can cause learning–forgetting oscillations and loss of previously acquired behavior; replaying historical successes can mitigate both. **(c)** As a result, replay can improve test pass@1 while preserving solution coverage (large-$k$ pass@$k$).

**rithm itself induces on the effective training distribution, even though the underlying dataset is held fixed.** When using group-mean-subtraction as an advantage, like in GRPO and variants, this is especially severe, as mastered prompts yield zero group-relative advantage and provide no learning signal whatsoever. In particular, this work

1. presents motivational SFT experiments that isolate the effect of the curriculum induced in GRPO, training on successful trajectories collected during RL in their original temporal order and comparing against a shuffled, stationary skyline. We show that the curriculum induced by RL is prone to endogenous forgetting, resulting in a reduction in learning efficiency, coverage (in terms of pass@$k$), and overall performance in comparison to a model that has the privilege of learning from stationary data.
2. shows, in this same SFT setting, that a simple replay buffer remedies the disparity between a model trained under the RL-induced curriculum and one that is not.
3. experiments, finally, with including this replay buffer in GRPO and introduces a win-rate variant of prioritized experience replay that works especially well. Like in the SFT regime, the explicit term designed to remediate endogenous forgetting results in more stable and efficient learning, improving performance in comparison to standard GRPO.

Our results demonstrate that endogenous forgetting is an obstacle in prolonged RL post-training, overwriting capabilities the policy has already acquired. Still, we show that it is easily remedied, suggesting that an intervention designed to resolve endogenous forgetting, such as the replay buffer described in this work, is worth considering as a default component of extended RL deployments.

## 2 Endogenous Non-stationarity and Forgetting in RL

In this section, we first discuss the sources of endogenous non-stationarity in RL post-training and define how we quantify learning and forgetting during training. We then use a controlled supervised learning experiment to show that this non-stationarity can induce endogenous forgetting and that replaying previously discovered successful trajectories helps mitigate it.

### 2.1 Preliminaries

**On-policy RL setting.** We consider on-policy RL post-training. Let $\mathcal{D}_{\text{tr}}$ denote a fixed distribution over training prompts. We optimize a policy $\pi_\theta$ to maximize the expected reward $\mathbb{E}_{q\sim\mathcal{D}_{\text{tr}},\, o\sim\pi_\theta(\cdot|q)}[R(q,o)]$, where $R(q,o)$ is the reward assigned by a fixed verifier to response $o$ for prompt $q$. Denote $\pi_t = \pi_{\theta_t}$ as the policy at the start of training step $t$. Our main experiments focus on GRPO (Shao et al., 2024), a widely used RL algorithm for LLM post-training with verifi-

able rewards. Appx. B.1 discusses extensions to other RL algorithms. We omit group-level reward standard-deviation normalization and the KL penalty from the GRPO objective, as in Dr. GRPO (Liu et al., 2025). At each RL step $t$, for each sampled prompt $q \sim \mathcal{D}_{\text{tr}}$, we generate a group of $G$ responses $\{o_i\}_{i=1}^{G}$ from the current behavior policy $\pi_t$. For each $o_i$, the group-relative advantage is $\hat{A}_i = R(q, o_i) - \frac{1}{G}\sum_{j=1}^{G} R(q, o_j)$. We maximize the clipped surrogate objective

$$\mathcal{J}_t(\theta) = \mathbb{E}_{\substack{q\sim\mathcal{D}_{\text{tr}},\\ o_{1:G}\sim\pi_t(\cdot|q)}}\left[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{\ell=1}^{|o_i|}\min\left(\rho_{i,\ell}(\theta)\hat{A}_i,\ \text{clip}(\rho_{i,\ell}(\theta), 1-\epsilon, 1+\epsilon)\,\hat{A}_i\right)\right], \tag{1}$$

where $\rho_{i,\ell}(\theta) = \frac{\pi_\theta(o_{i,\ell}|q,o_{i,<\ell})}{\pi_t(o_{i,\ell}|q,o_{i,<\ell})}$ and the behavior policy $\pi_t$ is held fixed while optimizing $\theta$.

**Endogenous non-stationarity in RL post-training.** Under Equation 1, groups with zero reward variance have zero advantage and contribute no gradient. For $\mathbf{o} = (o_1, \ldots, o_G)$, the distribution of prompt-response groups, conditioned on nonzero reward variance, can be written as $\mu_t(q, \mathbf{o}) \propto \mathcal{D}_{\text{tr}}(q)\prod_{i=1}^{G}\pi_t(o_i \mid q)\,\mathbf{1}[\text{Var}_i(R(q, o_i)) > 0]$. This reveals two forms of non-stationarity[1]. First, the effective prompt distribution changes as the policy's probability of generating a non-zero advantage group evolves, which induces an implicit curriculum. For example, under binary rewards, all-correct and all-incorrect groups provide no policy-gradient signal, as $\text{Var}_i(R(q, o_i)) = 0$. Second, the conditional response distribution $\pi_t(o \mid q)$ changes over the course of training, even for a fixed prompt. Both the effective prompt distribution and rollout distribution can change, despite an unchanging training dataset.

**Measuring learning and forgetting.** Aggregate performance measures do not distinguish newly attained gains from the loss of previously attained performance, which can offset each other throughout the course of model fitting. To disentangle these dynamics, we introduce *the attained gain* and *unrecovered loss*. Let $\mathcal{Q}_{\text{eval}}$ denote the fixed evaluation set, let $\mathcal{D}_{\text{eval}}$ denote its prompt distribution, and let $m(q, t)$ denote the true pass@1 on $q$ under $\pi_t$ at checkpoint $t$[2]. Let $m^\star(q, t) = \max_{0\le\tau\le t} m(q, \tau)$ denote the best performance attained from initialization 0 up to $t$, where $\tau$ ranges over evaluated checkpoints. We measure *attained gain* as the increase from initial to best performance up to $t$ and *unrecovered loss* as the shortfall from that best performance at $t$, averaged over evaluation prompts:

$$\mathcal{G}^{\text{att}}(t) = \mathbb{E}_{q\sim\mathcal{D}_{\text{eval}}}[m^\star(q, t) - m(q, 0)], \qquad \mathcal{F}^{\text{unrec}}(t) \quad = \mathbb{E}_{q\sim\mathcal{D}_{\text{eval}}}[m^\star(q, t) - m(q, t)].$$

At each checkpoint, these quantities decompose the overall performance change as

$$\bar{m}(t) - \bar{m}(0) = \mathcal{G}^{\text{att}}(t) - \mathcal{F}^{\text{unrec}}(t),$$

where $\bar{m}(t) = \mathbb{E}_{q\sim\mathcal{D}_{\text{eval}}}[m(q, t)]$ is the average pass@1 over the fixed evaluation set at checkpoint $t$. We use attained gain and unrecovered loss as our primary *learning* and *forgetting* metrics, respectively. When the time argument is omitted, $m^\star(q)$, $\mathcal{G}^{\text{att}}$, and $\mathcal{F}^{\text{unrec}}$ denote their values at the final checkpoint $\pi_T$. Appendix E.1.1 relates them to cumulative increases and decreases, separating out recovered losses. Our decomposition generalizes the correctness-based temporal forgetting metrics of Li et al. (2026), as discussed in Appendix C.

### 2.2 Effect of Endogenous Non-stationarity

In agreement with prior work, our experiments (Section 3) reveal undesirable dynamics during RL training, including degradation in pass@$k$ and substantial oscillations in prompt-level performance. We hypothesize that these behaviors arise, at least in part, from the endogenous curriculum induced by the RL post-training procedure. To isolate the effect of this non-stationary data distribution from other aspects of RL optimization, we conduct a controlled offline SFT experiment.

We first collect all verified-correct trajectories generated throughout the course of RL post-training, and note the step at which the generation was produced. We discard the resulting model, and treat

[1] The actual gradient contributions also depend on advantage magnitudes, importance ratios, and clipping, which are not represented in $\mu_t$.

[2] Under independent sampling, true pass@$k$ is $1 - (1 - m(q, t))^k$, strictly increasing in $m$ for any fixed $k \ge 1$, so the signs of true prompt-level changes agree for all $k$, though their magnitudes and aggregate trends can differ.

Table 1: **Ordered SFT incurs greater unrecovered loss and lower pass@64 across all six settings, suggesting the endogenous curriculum is harmful.** Despite higher attained gains in every setting, Ordered SFT achieves lower pass@1 in four of six settings. It also yields a smaller fraction of prompts with monotonic improvement in all settings.

| Domain | Model | SFT type | Pass@1 ↑ | Pass@64 ↑ | $\mathcal{G}^{\text{att}}$ ↑ | $\mathcal{F}^{\text{unrec}}$ ↓ | Monotonic improv.(%) ↑ |
|---|---|---|---|---|---|---|---|
| KK | Qwen2.5-3B | Ordered | **50.8** | 81.0 | **53.5** | 9.2 | 19.4 |
| | | Shuffled | 47.6 (-3.2) | **83.3** (+2.2) | 48.2 (-5.3) | **7.1** (-2.1) | **21.0** (+1.6) |
| Math | Qwen2.5-3B | Ordered | 23.2 | 52.3 | **16.3** | 8.6 | 9.0 |
| | | Shuffled | **26.0** (+2.9) | **60.4** (+8.1) | 14.9 (-1.4) | **4.3** (-4.3) | **16.7** (+7.7) |
| | Llama3.2-3B-Instruct | Ordered | 21.3 | 46.1 | **18.3** | 9.3 | 3.7 |
| | | Shuffled | **22.8** (+1.5) | **51.7** (+5.7) | 16.3 (-2.0) | **5.8** (-3.5) | **9.5** (+5.8) |
| Coding | Qwen2.5-3B | Ordered | **76.4** | 92.9 | **32.4** | 5.3 | 35.2 |
| | | Shuffled | 75.0 (-1.4) | **94.1** (+1.2) | 29.4 (-3.0) | **3.8** (-1.6) | **41.8** (+6.6) |
| | Llama3.2-3B-Instruct | Ordered | 65.9 | 85.5 | **47.5** | 11.2 | 42.3 |
| | | Shuffled | **69.5** (+3.6) | **89.5** (+4.0) | 44.2 (-3.4) | **4.2** (-7.0) | **56.3** (+14.0) |
| | Qwen3-1.7B-Base | Ordered | 69.6 | 87.7 | **46.8** | 14.6 | 47.9 |
| | | Shuffled | **72.3** (+2.7) | **89.8** (+2.1) | 44.3 (-2.5) | **9.3** (-5.3) | **56.1** (+8.2) |

this large collection of trajectories as a dataset for SFT. Starting from the same initial model as was used in the first step, we train two SFT variants with identical hyperparameters, each making a single pass through these data:

1. *Ordered SFT*, where trajectories are presented in the order they were generated during RL.
2. *Shuffled SFT*, where the same trajectories are randomly permuted before training.

Ordered SFT preserves the endogenous curriculum of RL, whereas shuffling removes it. Here, ordered SFT can be thought of as an analogue for conventional RL post-training, relegated to a corresponding endogenous curriculum, while the shuffled alternative acts as a skyline, with privileged, i.i.d. access to the same rewarding data but *without* the curriculum. Both variants train on exactly the same trajectories, so any difference in learning and forgetting can be attributed to data order alone.

**Experimental setup.** We consider three reasoning domains and measure performance on in-distribution validation data. (1) `KK`: we adopt the Knights and Knaves task, which allows explicit control of difficulty by varying the number of inhabitants in the puzzle (Xie et al., 2025a;b). Each instance is a logic puzzle in which every inhabitant is either a knight, who always tells the truth, or a knave, who always lies, and the model must infer the identity of each inhabitant from their statements. We consider puzzles with 3–7 inhabitants and use the balanced training set[3] with 1,000 instances at each difficulty level. For evaluation, we use the disjoint test set with 100 examples per difficulty level. (2) `Math`: we train on the Polaris-53K (An et al., 2025) training dataset and evaluate on a uniformly-sampled held-out set of 600 examples. (3) `Coding`: we train on the deduplicated MBPP training set (Austin et al., 2021) and evaluate on MBPP+ (Liu et al., 2023). More details and prompt templates are provided in Appendix D.1. For all experiments, we use Qwen2.5-3B, Llama3.2-3B-Instruct, and Qwen3-1.7B-Base. For SFT, we collect correct trajectories from RL runs and fine-tune the base models for 1 epoch. At each evaluation checkpoint, we sample multiple responses per prompt and estimate pass@1 and pass@$k$ using the unbiased estimator (Chen et al., 2021). Appx. D.2 contains further experimental details.

**Endogenous non-stationarity can induce forgetting.** As shown in Table 1, Ordered SFT reaches higher final attained gains in all settings but incurs greater unrecovered loss, meaning it forgets more over the course of training than shuffled SFT. In Fig. 10 (Appx. D.3), we visualize the learning and forgetting curves throughout training for these settings. While attained gain keeps increasing over training, Ordered SFT accumulates substantially greater unrecovered loss than Shuffled SFT.

**Endogenous forgetting can hurt pass@1 and reduce solution coverage.** Despite higher attained gain, Ordered SFT achieves lower final pass@1 in most settings due to greater unrecovered loss. It also shows greater degradation in solution coverage, with lower final pass@64 than Shuffled SFT

[3] `https://huggingface.co/datasets/K-and-K/knights-and-knaves`

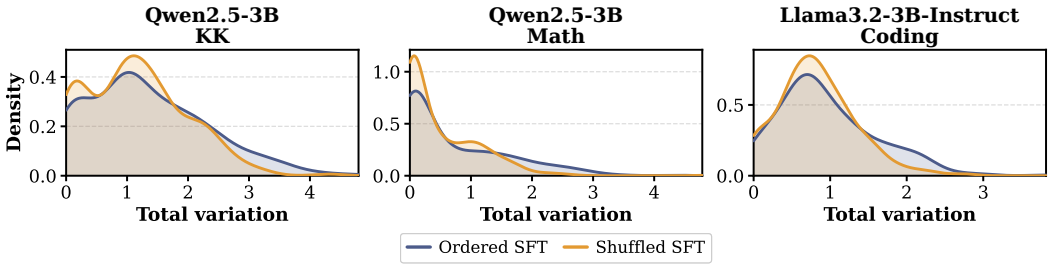


Figure 2: **Ordered SFT shows more prompt-level learning-forgetting oscillations.** Distributions of per-prompt total variation $V(q)$ on `KK` and `Math` with Qwen2.5-3B and on `Coding` (MBPP+) with Llama3.2-3B-Instruct. Ordered SFT has a longer right tail, which shuffling reduces.

across all six settings. Since large-$k$ pass@$k$ is more sensitive to changes on low-success prompts, this suggests greater coverage loss on prompts the model solves only rarely. Such losses are undesirable for RL training, where further improvements should not come at the cost of model coverage.

**Ordered SFT induces more prompt-level learning-forgetting oscillations.** To understand the effect on prompt-level learning dynamics, we quantify cumulative changes in each prompt's success probability using total variation, $V(q) = \sum_{t=1}^{T} |m(q,t) - m(q,t-1)|$, where $t$ indexes consecutive evaluation checkpoints[4]. For a prompt whose performance changes monotonically, $V(q) = |m(q,T) - m(q,0)| \leq 1$, so values above one necessarily involve oscillations between learning and forgetting. Fig. 2 shows a longer right tail under Ordered SFT than under Shuffled SFT, indicating larger accumulated oscillations in estimated prompt-level success probabilities. Ordered SFT also yields a smaller fraction of prompts with monotonic improvement in all six settings (Table 1), defined as prompts with at least one detected improvement and no detected declines, with the formal definition in Appx. E.1. Therefore, the endogenous curriculum induced by the RL policy can yield more unstable prompt-level learning, where prompts oscillate between being learned and unlearned rather than improving steadily.

**Ordered SFT exhibits forgetting of earlier rollouts.** We next examine forgetting directly at the level of training rollouts, by tracking how the model's loss changes on solutions encountered earlier in the SFT stream. Let $\mathcal{B}_\tau$ denote the batch presented at step $\tau$ of the Ordered SFT run. At checkpoint $t$, we define the *old* rollouts as all batches seen previously, $\bigcup_{\tau<t} \mathcal{B}_\tau$, and the *new* rollouts as the current batch $\mathcal{B}_t$. For both SFT variants, we evaluate the model *before* the step-$t$ update on these same two reference sets. Fig. 3 shows a comparison for Llama3.2-3B-Instruct on `Coding`. Here, the two data arrangements achieve similar loss on new rollouts, indicating comparable fit to the solutions currently arriving in the stream. In contrast, later in training, Ordered SFT exhibits higher loss on old rollouts than Shuffled SFT, despite having previously trained on these examples.

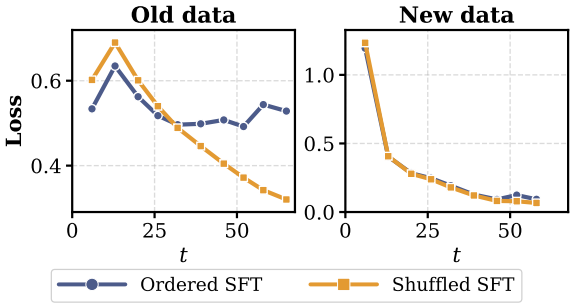


Figure 3: **Loss on old and new rollouts for Llama3.2-3B-Instruct on `Coding`.**

**Takeaway.** The non-stationary data distribution induced by the RL policy can cause endogenous forgetting: it increases unrecovered loss, amplifies learning-forgetting oscillations, weakens retention of earlier successful rollouts, and ultimately hurts downstream performance and solution coverage. Importantly, these effects persist even when RL optimization itself is removed, suggesting that the endogenous curriculum alone is sufficient to drive forgetting. Conversely, removing its temporal ordering through shuffled SFT substantially reduces forgetting and preserves greater solution coverage, motivating interventions that might quell this non-stationarity during RL training.

[4]Evaluating at every training step is computationally costly, so we evaluate at a fixed stride during training.

### 2.3 MITIGATING ENDOGENOUS FORGETTING: REPLAY IN ORDERED SFT

While Shuffled SFT reduces forgetting, it requires access to rollouts from future steps and is therefore infeasible during online training. Replay offers a natural way to retain access to earlier successes by mixing historical rollouts with current ones. We therefore first test whether replay can mitigate the endogenous forgetting in Ordered SFT, using the same experimental setup as in Section 2.2, with additional replay samples.

Specifically, we use the standard SFT loss, denoted by $\mathcal{L}_{\mathrm{SFT}}(\theta;\mathcal{B}) = -\frac{1}{|\mathcal{B}|}\sum_{(q,o)\in\mathcal{B}}\log\pi_\theta(o \mid q)$ for a minibatch $\mathcal{B}$. At SFT step $t$, in addition to the current minibatch $\mathcal{B}_t$, we maintain a replay buffer $\mathcal{M}_t \subseteq \bigcup_{s<t}\mathcal{B}_s$ of successful trajectories from earlier minibatches. We sample a replay minibatch $\mathcal{R}_t$ from this buffer according to a sampling distribution $\nu_t$ and compute the same SFT loss on it. By default, we sample a prompt uniformly from the prompts represented in the buffer, then sample a stored rollout uniformly for that prompt. We minimize the combined loss

$$\mathcal{L}_{\mathrm{SFT+Rep}}(\theta,t) = \mathcal{L}_{\mathrm{SFT}}(\theta;\mathcal{B}_t) + \lambda\,\mathcal{L}_{\mathrm{SFT}}(\theta;\mathcal{R}_t),$$

where $\lambda \geq 0$ controls the replay strength. The replay term is omitted when the buffer is empty. We set $\lambda = 1$ and use the same batch size for the current and replay minibatch.

**Replay reduces forgetting and improves solution coverage.** Replay reduces unrecovered loss by 0.8–6.9 percentage points and consistently improves pass@64 (Table 2).

**The retention benefit extends to other sampling distributions.** We also test *hard replay*, which samples prompts in proportion to $1 - p_t(q)$, where $p_t(q)$ is the estimated success probability, then uniformly selects a stored successful rollout for that prompt. *Cross-entropy (CE) replay* instead samples stored rollouts in proportion to their SFT loss. Both variants reduce unrecovered loss across all six settings and improve final pass@64 (Appx. D.4 Table 5 and Fig. 11).

Table 2: **Replay in Ordered SFT can reduce unrecovered loss.** Replay minus Ordered SFT (percentage points) at the final shared checkpoint.

| Model | $\Delta$P@1 $\uparrow$ | $\Delta$P@64 $\uparrow$ | $\Delta\mathcal{F}^{\mathrm{unrec}}$ $\downarrow$ |
|---|---|---|---|
| *Domain: KK* | | | |
| Qwen2.5-3B | $-2.6$ | $\mathbf{+0.7}$ | $\mathbf{-0.8}$ |
| *Domain: Math* | | | |
| Qwen2.5-3B | $\mathbf{+1.4}$ | $\mathbf{+7.2}$ | $\mathbf{-1.7}$ |
| Llama3.2-3B-Instruct | $\mathbf{+1.2}$ | $\mathbf{+3.9}$ | $\mathbf{-2.4}$ |
| *Domain: Coding* | | | |
| Qwen2.5-3B | $-0.8$ | $\mathbf{+0.7}$ | $\mathbf{-1.1}$ |
| Llama3.2-3B-Instruct | $-1.1$ | $\mathbf{+2.0}$ | $\mathbf{-2.0}$ |
| Qwen3-1.7B-Base | $\mathbf{+1.6}$ | $\mathbf{+1.6}$ | $\mathbf{-6.9}$ |

## 3 MITIGATING ENDOGENOUS FORGETTING IN ON-POLICY TRAINING

Building on the success of the SFT experiments, we now investigate whether *successful-trajectory replay* can also reduce forgetting and improve generalization in on-policy RL.

### 3.1 METHODOLOGY

To apply replay in on-policy RL training, at step $t$, let $\nu_t$ denote a distribution over entries in $\mathcal{M}_t$. Motivated by self-imitation learning (Oh et al., 2018), we define the replay objective as the expected token-normalized log-likelihood of trajectories sampled from this distribution: $\mathcal{J}_{\mathrm{rep},t}(\theta) = \mathbb{E}_{(q,o^+)\sim\nu_t}\left[\frac{1}{|o^+|}\sum_{\ell=1}^{|o^+|}\log\pi_\theta\left(o^+_\ell \mid q, o^+_{<\ell}\right)\right]$. At each step $t$, we optimize $\mathcal{J}^{\mathrm{RR}}_t(\theta) = \mathcal{J}_t(\theta) + \lambda\mathcal{J}_{\mathrm{rep},t}(\theta)$, where $\mathcal{J}_t(\theta)$ denotes the objective in Equation 1, evaluated on the current on-policy batch at step $t$, and the fixed weight $\lambda \geq 0$ controls the relative contribution of the replay term. The first term learns from trajectories generated by the current policy, whereas the second maintains the likelihood of previously discovered correct trajectories. Equivalently, the replay term can be viewed as an auxiliary SFT loss on historical verified solutions.

**Prioritized replay.** Intuitively, successful rollouts are collected more frequently from high-pass-rate prompts, so the replay buffer can overrepresent behaviors the policy already performs reliably. We therefore bias replay toward prompts whose successes are rarer and potentially more difficult to retain. We factorize the replay distribution over a stored prompt-trajectory pair as $\nu_t(q,o^+) = \nu_t^{\mathrm{prompt}}(q)\nu_t^{\mathrm{traj}}(o^+ \mid q)$. Motivated by prioritized replay (Schaul et al., 2015; Oh et al., 2018), we prioritize buffered prompts by the advantage score $1 - p_t(q)$, where $p_t(q) = \mathbb{E}_{o\sim\pi_t(\cdot|q)}[R(q,o)]$

Table 3: **Replay reduces forgetting and improves final performance.** Across all seven settings, replay reduces unrecovered loss and increases the fraction of prompts with monotonic improvement relative to standard GRPO. Replay in RL achieves comparable or higher pass@1 and higher pass@64, approaching base-model solution coverage on `Math` and `Coding` and exceeding it on `KK`. The full table, including base-model performance, appears in Appx. D.5 (Table 6).

| Domain | Model | Method | Pass@1 ↑ | Pass@64 ↑ | $\mathcal{G}^{\text{att}}$ ↑ | $\mathcal{F}^{\text{unrec}}$ ↓ | Monotonic imp.(%) ↑ |
|---|---|---|---|---|---|---|---|
| `KK` | Qwen2.5-3B | Standard | 93.4 | 98.0 | 92.6 | 5.8 | 14.6 |
| | | Replay | **96.6** (+3.2) | **100.0** (+2.0) | **92.8** (+0.1) | **2.7** (-3.1) | **22.4** (+7.8) |
| `Math` | Qwen2.5-3B | Standard | 23.1 | 54.0 | **17.8** | 10.2 | 4.3 |
| | | Replay | **25.5** (+2.4) | **62.7** (+8.7) | 14.5 (-3.3) | **4.5** (-5.7) | **16.8** (+12.5) |
| | Llama3.2-3B-Instruct | Standard | 19.7 | 42.7 | **15.6** | 8.2 | 5.8 |
| | | Replay | **19.9** (+0.2) | **57.8** (+15.1) | 10.4 (-5.2) | **2.8** (-5.4) | **14.8** (+9.0) |
| | Qwen3-8B-Base | Standard | 34.4 | 68.8 | **28.9** | 18.2 | 6.5 |
| | | Replay | **41.3** (+6.9) | **75.5** (+6.7) | 23.4 (-5.4) | **5.9** (-12.3) | **24.3** (+17.8) |
| `Coding` | Qwen3-1.7B-Base | Standard | 71.2 | 80.6 | 42.7 | 8.8 | 57.7 |
| | | Replay | **76.3** (+5.1) | **93.2** (+12.6) | **43.0** (+0.4) | **4.2** (-4.6) | **74.3** (+16.7) |
| | Qwen2.5-3B | Standard | 76.6 | 90.0 | **33.4** | 6.2 | 58.2 |
| | | Replay | **78.3** (+1.7) | **93.0** (+3.0) | 32.5 (-0.9) | **3.6** (-2.6) | **73.0** (+14.8) |
| | Llama3.2-3B-Instruct | Standard | **69.8** | 80.1 | **49.2** | 8.9 | 48.1 |
| | | Replay | 69.7 (-0.1) | **88.0** (+7.9) | 46.7 (-2.4) | **6.6** (-2.4) | **62.4** (+14.3) |

denotes the current success probability. This choice is also consistent with our empirical observation that unrecovered loss is particularly pronounced among prompts with low initial pass rates (Fig. 16).

Unfortunately, computing the current success probability $p_t(q)$ for every buffered prompt would require generating online rollouts, which is prohibitively expensive. In practice, we instead replace it with the most recently observed group-level estimate of the pass rate, $\widetilde{p}_t(q) = G^{-1}\sum_{i=1}^{G} R(q, o_i)$. Because $\widetilde{p}_t(q)$ is only updated when $q$ is sampled on-policy, it can become stale after replay. Consequently, a prompt with a low stored pass rate may be selected repeatedly. To reduce repeated selection and encourage diversity in replay, we use a simple cooldown mechanism. Let $\tau^{\text{rep}}(q)$ denote the most recent step at which prompt $q$ was replayed, and let $H$ be the number of RL steps in one training epoch. We assign the prompt-level weight $c_t(q) = 1$ if $q$ has never been replayed, and $c_t(q) = \min\{1, (t - \tau^{\text{rep}}(q))/H\}$ otherwise. Immediately after a prompt is replayed, its weight is reduced to 0 and then recovers linearly to 1 over the following $H$ steps. Combining the priority with the cooldown weight, prompts are sampled according to $\nu_t^{\text{prompt}}(q) = \frac{(1-\widetilde{p}_t(q))c_t(q)}{\sum_{q' \in \mathcal{Q}(\mathcal{M}_t)}(1-\widetilde{p}_t(q'))c_t(q')}$. Conditional on selecting prompt $q$, we sample uniformly among its stored successful trajectories. The complete procedure is summarized in Algorithm 1 (Appx. E).

### 3.2 Experiments

**Setup.** We use the same tasks, models, and evaluation protocols described in Section 2.2. We additionally evaluate Qwen3-8B-Base on `Math`. For each model–domain setting, we train standard RL and RL with replay for the same number of gradient steps. For replay, we set $\lambda = 0.1$ across all settings and replay batch sizes are specified in Appx. D.

**Replay reduces unrecovered loss during RL training.** Table 3 shows that replay consistently reduces unrecovered loss across all seven model-domain settings. In Fig. 4, we visualize two representative settings, which show that unrecovered loss continues to accumulate under standard RL, while replay sharply reduces forgetting. Additional results are provided in Appx. D.5 (Fig. 12).

**Replay reduces prompt-level oscillations and increases the fraction of monotonically improving prompts during RL training.** Fig. 5 compares the prompt-level total variation $V(q)$ (as defined in Section 2.2) across all three `Coding` models. Standard RL has a substantially longer right tail, indicating larger cumulative oscillations in prompt-level success probabilities. Replay reduces this tail and increases the fraction of prompts with monotonic improvement in all seven settings, as shown in

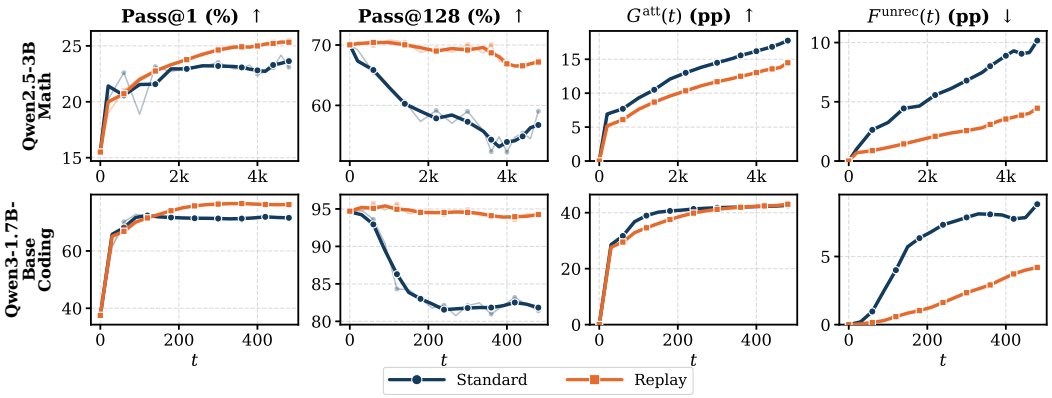


Figure 4: **Replay reduces unrecovered loss and improves final performance throughout training.** Under standard RL, pass@1 improves while pass@128 substantially declines, with unrecovered forgetting accumulating throughout training. Replay sharply reduces $\mathcal{F}^{\text{unrec}}(t)$, preserving substantially more solution coverage and yielding better final pass@1. While standard RL plateaus relatively early, replay continues to improve later in training.

Table 3. Fig. 15 (Appx. D.5) provides qualitative examples of these dynamics on individual prompts. These results suggest that using replay in RL effectively mitigates endogenous forgetting.

**RL with replay improves pass@1, preserves solution coverage, and sustains gains over longer training.** As shown in Table 3, replay achieves comparable attained gains to standard RL on `KK` and `Coding`, while substantially reducing unrecovered loss. On `Math`, replay attains smaller gains, but the reduction in unrecovered loss more than compensates for this difference, resulting in higher or comparable final pass@1. For example, with Qwen3-8B-Base on `Math`, replay reduces unrecovered loss by 12.3 points and improves final pass@1 by 6.9 points. Replay also achieves consistently higher large-$k$ pass@$k$ across all seven settings, with substantially less degradation relative to base-model coverage. Results for additional values of $k$ are provided in Appx. D.5 (Fig. 13).

Fig. 4 further reveals how these differences emerge over training. Although attained gains continue to accumulate, pass@1 under standard RL begins to plateau early as growing unrecovered loss increasingly offsets these gains. Replay substantially suppresses this accumulation of forgetting, allowing pass@1 to continue improving while preserving higher solution coverage. The same pattern is reflected in the training-reward curves in Fig. 14, where replay continues to make progress after standard RL has largely saturated. Together, these results suggest that mitigating forgetting can help prolonged RL training and sustain performance gains.

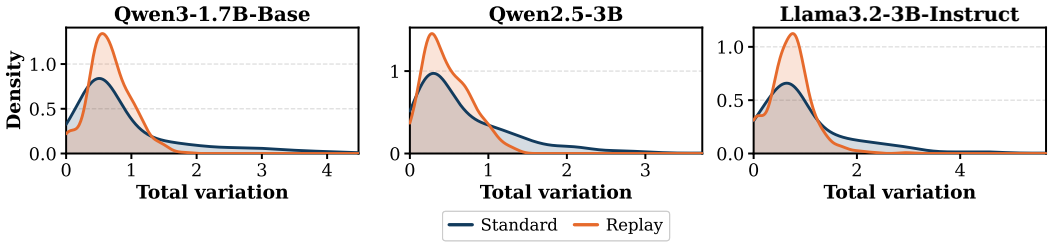


Figure 5: **Standard RL exhibits substantial prompt-level oscillations which replay reduces.** Distributions of per-prompt total variation $V(q)$ on `Coding` (MBPP+) for three models. Replay significantly reduces the high-variation tail relative to standard RL.

**Replay also accelerates learning on `KK`.** Beyond its benefits in mitigating forgetting and improving final performance, replay improves pass@1 and pass@64 more rapidly during early training on `KK` (Fig. 6). Using the standard RL's final performance as the target, replay reaches its pass@1 level about 3.3k gradient steps earlier and its pass@64 level about 5.5k steps earlier. We hypothesize that the base model initially solves relatively few examples for `KK`, with an initial pass@1 of only 6.5%. Successful trajectories are therefore scarce early in training, and replay may accelerate learning by reusing these rare positive examples. We leave to future work a systematic investigation of when replay also accelerates learning and how this depends on the base model's prior.

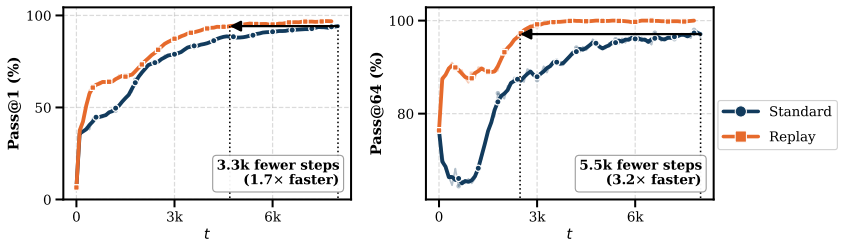


Figure 6: **Replay accelerates learning on KK.** Using replay improves pass@1 and pass@64 more rapidly during training in the KK task. Horizontal arrows indicate that replay reaches standard RL final pass@1 and pass@64 approximately 3.3k and 5.5k gradient steps earlier, respectively.

## 4 Related Work

**Temporal forgetting and non-stationarity in LLM post-training.** Prior work studies whether on-policy RL adaptation to new tasks or domains causes catastrophic forgetting of general pretrained capabilities (Shenfeld et al., 2026; Chen et al., 2025b; Phan et al., 2026). We instead study *endogenous forgetting* induced by the changing policy itself. Recent work studies related forms of temporal instability during RL, including forgetting of previously solved prompts and correct-to-incorrect transitions over training (Nguyen et al., 2025; Qin et al., 2026; Li et al., 2026). We discuss concrete distinctions in the extended related work (Appendix C). These phenomena also relate to non-stationarity and interference in deep RL, which can impair generalization in stationary environments (Igl et al., 2020; Fedus et al., 2020; Kaplanis et al., 2019). Our work unifies forms of endogenous forgetting, connects it to the endogenous curriculum induced by on-policy training through controlled experiments, and studies successful-trajectory replay as a training-time mitigation.

**Reasoning coverage and the limits of RL.** A closely related line of work asks whether RL expands reasoning capabilities beyond those inherited from pretraining or primarily increases the probability of already accessible solutions (Yue et al., 2025; Tajwar et al., 2026). Improvements in aggregate pass@1 can coexist with declines in pass@$k$ at large $k$, indicating that higher single-sample accuracy need not imply broader solution coverage (Yue et al., 2025; Nguyen et al., 2025). Beyond evaluation, preserving coverage matters for continued learning: under finite on-policy sampling, declining success probabilities make previously accessible solutions less likely to be observed and reinforced. Our work focuses on mitigating endogenous forgetting throughout RL training, which helps preserve the base model's solution coverage.

**Experience replay in RL post-training.** Experience replay has long been used in RL (Lin, 1992; Mnih et al., 2015; Schaul et al., 2015; Oh et al., 2018; Rolnick et al., 2019) but it has only recently been adopted for post-training. Most commonly, the goal is to reduce generation cost through better sample efficiency (Arnal et al., 2026; Sun et al., 2026; Zhang et al., 2025; Ma et al., 2026). GEM (Anthony et al.) preserves past successes with emphasis on rare-success retention in sparse-reward settings. Although we show improved sample efficiency in some settings, our goal is to study the impact of replay on endogenous forgetting. Our work uses previously discovered successful trajectories as auxiliary historical supervision alongside the on-policy objective to mitigate the effects of endogenous non-stationarity and maintain solution coverage during prolonged training.

## 5 Conclusion

We study endogenous forgetting in RL post-training, where the policy loses previously acquired capabilities even when training on a fixed dataset. By decomposing changes in performance into attained gain and unrecovered loss, we show that forgetting can offset gains from learning and reduce solution coverage. Through motivational SFT experiments, we show that training on successful RL trajectories in their original temporal order leads to more forgetting than training on the same trajectories in shuffled order, suggesting that the non-stationarity of the endogenous curriculum contributes to this forgetting. Across multiple models and reasoning tasks, successful-trajectory replay reduces unrecovered loss and learning-forgetting oscillations, improves final pass@1, and maintains solution coverage. These findings suggest that preserving previously acquired capabilities is important for sustaining progress in prolonged RL post-training.

### AI USE STATEMENT

We used generative AI tools to support writing and editing, primarily for identifying and correcting typographical and grammatical errors. We also used AI assistance for code implementation and debugging. All AI-assisted content was closely guided, reviewed, and verified by the authors, who take full responsibility for the final manuscript and code.

### ETHICS STATEMENT

This paper studies the endogenous forgetting phenomenon in on-policy RL post-training. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

### REPRODUCIBILITY STATEMENT

Our code repository is available at `https://anonymous.4open.science/r/iclr2027_submission_endogenous_forgetting-0664`. In Section 2.2 and Appendix D, we provide detailed descriptions of experimental setup, including dataset, models, training and evaluation configurations.

## REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.662. URL https://aclanthology.org/2024.acl-long.662/.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, et al. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models, 2025. *URL https://hkunlp. github. io/blog/2025/Polaris*, 2, 2025.

GX-Chen Anthony, Stephanie Milani, Jatin Prakash, Sumit Chopra, Rob Fergus, and Rajesh Ranganath. Good experience maximization. In *Decision-Making from Offline Datasets to Online Adaptation: Black-Box Optimization to Reinforcement Learning*.

Charles Arnal, Vivien Cabannes, Taco Cohen, Julia Kempe, and Remi Munos. Efficient rl training for llms with experience replay. *arXiv preprint arXiv:2604.08706*, 2026.

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

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025a.

Howard Chen, Noam Razin, Karthik Narasimhan, and Danqi Chen. Retaining by doing: The role of on-policy data in mitigating forgetting. *arXiv preprint arXiv:2510.18874*, 2025b.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

William Fedus, Dibya Ghosh, John D Martin, Marc G Bellemare, Yoshua Bengio, and Hugo Larochelle. On catastrophic interference in atari 2600 games. *arXiv preprint arXiv:2002.12499*, 2020.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. *arXiv preprint arXiv:1902.00255*, 2019.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Venkata Sai Surya Subramanyam Duvvuri, Manzil Zaheer, Inderjit Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms. In *International Conference on Learning Representations*, volume 2026, pp. 72438–72467, 2026.

Yuetai Li, Zhangchen Xu, Fengqing Jiang, Bhaskar Ramasubramanian, Luyao Niu, Bill Yuchen Lin, Xiang Yue, and Radha Poovendran. Temporal sampling for forgotten reasoning in llms. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28309–28327, 2026.

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

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *Advances in Neural Information Processing Systems*, 38:17998–18031, 2026.

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.

Weiyu Ma, Yongcheng Zeng, Yan Song, Xinyu Cui, Jian Zhao, Xuhui Liu, and Mohamed Elhoseiny. Freshness-aware prioritized experience replay for llm/vlm reinforcement learning. *arXiv preprint arXiv:2604.16918*, 2026.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

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.

Phuc Minh Nguyen, Chinh D La, Duy MH Nguyen, Nitesh V Chawla, Binh T Nguyen, and Khoa D Doan. The reasoning boundary paradox: How reinforcement learning constrains language models. *arXiv preprint arXiv:2510.02230*, 2025.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

Hoang Phan, Xianjun Yang, Yuanshun Yao, Jingyu Zhang, Shengjie Bi, Xiaocheng Tang, Madian Khabsa, Lijuan Liu, and Deren Lei. Beyond reasoning gains: Mitigating general-capability forgetting in large reasoning models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 34368–34398, 2026.

Chuanyu Qin, Chenxu Yang, Qingyi Si, Naibin Gu, Peng Fu, and Zheng Lin. Learning to solve, forgetting to retain: Correct-set turnover in rlvr. *arXiv preprint arXiv:2606.03087*, 2026.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

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

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.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 59839–59864, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/618c95f4557c15b253fb0e6f548ea0c0-Paper-Conference.pdf.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, Mingyuan Zhou, and Huan Zhang. Improving data efficiency for llm reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay. *Advances in Neural Information Processing Systems*, 38: 173449–173476, 2026.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. Maximum likelihood reinforcement learning. *arXiv preprint arXiv:2602.02710*, 2026.

Microsoft AI Team et al. Mai-thinking-1: Building a hill-climbing machine. Technical report, Technical report, Microsoft AI, 2026. https://microsoft. ai/pdf/mai-thinking . . . , 2025.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 2742–2785, 2025a.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025b.

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.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 57654–57689. Curran Associates, Inc., 2025. doi: 10.52202/085713-1933. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/537d5aa768c2d534016a4d06f87bc8fb-Paper-Conference.pdf.

Yuheng Zhang, Wenlin Yao, Changlong Yu, Yao Liu, Qingyu Yin, Bing Yin, Hyokun Yun, and Lihong Li. Improving sampling efficiency in rlvr through adaptive rollout and response reuse. *arXiv preprint arXiv:2509.25808*, 2025.

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, 2025. URL https://arxiv.org/abs/2507.18071.

# A LIMITATIONS

We show that one mitigation, successful-trajectory replay, can reduce endogenous forgetting, but it is not necessarily the best one. Other approaches may also be effective and complementary to replay. For example, prioritizing prompts by how much they have been forgotten, such as their prompt-level unrecovered loss, could target endogenous forgetting more directly, but identifying these prompts during training requires further work, since the success probabilities of buffered prompts are only updated when they are sampled on-policy and can become stale. We leave the exploration of other mitigation methods to future work.

# B DISCUSSION

## B.1 HOW BROADLY DOES ENDOGENOUS NON-STATIONARITY APPLY?

The endogenous shift described in Section 2 arises whenever training responses are sampled from an evolving policy, even with a fixed prompt distribution and reward function. Its effect on the availability of learning signals depends on the algorithm's reward and advantage construction.

**Group-relative advantages.** GRPO, Dr. GRPO, DAPO, GSPO, and CISPO retain a group-wise average-reward baseline despite differences in normalization, importance weighting, and clipping (Shao et al., 2024; Liu et al., 2025; Yu et al., 2026; Zheng et al., 2025; Chen et al., 2025a; Khatri et al., 2026). RLOO (Ahmadian et al., 2024) uses the average reward of the other responses in the same group as the baseline for each response. Consequently, equal rewards within a group yield zero reward-based advantages under these constructions. Under binary reward, all-correct and all-incorrect groups will provide zero advantages. In practice, commonly used dynamic sampling methods make this selection explicit by retaining only response groups with nonzero reward variance when constructing update batches (Yu et al., 2026; Khatri et al., 2026; Team et al., 2025). This will induce the endogenous curriculum. The equal-reward condition also applies to continuous rewards.

**Critic-based advantages.** Critic-based PPO (Schulman et al., 2017) estimates advantages using a learned value function, so equal terminal rewards need not yield zero advantages. Nevertheless, critic-based learning remains subject to endogenous non-stationarity. As described by Igl et al. (2020), policy updates change both the distribution of visited states $d_{\pi_\theta}(s)$ and the true value function $V^{\pi_\theta}(s)$ that the critic approximates. Moreover, when the critic learns from targets that incorporate its own predictions of future rewards, these targets change as the critic is updated, even under a fixed policy. This further changes advantage estimates and policy-gradient updates. Thus, avoiding the group-relative zero-advantage mechanism does not remove non-stationarity in either the sampled trajectories or their supervision.

Beyond changes in effective prompt exposure, the response distribution $\pi_t(o \mid q)$ also evolves, changing which reasoning strategies and solution trajectories are sampled for the same prompt. This response-level non-stationarity persists across these algorithms, regardless of whether advantages are constructed from group-relative rewards or a learned critic.

# C EXTENDED RELATED WORK

**Temporal forgetting and non-stationarity in LLM post-training.** Prior work studies whether on-policy RL adaptation to new tasks or domains causes catastrophic forgetting of general pretrained capabilities (Shenfeld et al., 2026; Chen et al., 2025b; Phan et al., 2026). We instead study *endogenous forgetting*: the policy becomes less likely to solve prompts that the base model or an earlier RL policy could solve, even as training continues within the same domain. Recent work identifies related temporal instability in RL. Nguyen et al. (2025) study RL-induced coverage loss and propose SELF, which filters out greedily solved prompts and regularizes the learned policy toward the base policy using forward KL. However, the forward-KL objective regularizes toward the full base-model distribution rather than selectively preserving previously successful trajectories, and its fixed base-policy reference does not explicitly protect successful behaviors acquired later during RL. In contrast, our replay mechanism continually preserves successful trajectories generated throughout training. Qin et al. (2026) study correct-set forgetting: they only focus on prompts that previously produced all-correct rollout groups and revisit them with on-policy rollouts, rather than a replay

buffer. Li et al. (2026) track correct-to-incorrect transitions in greedy predictions, capturing forgetting of both base-model and RL-acquired solutions, and recover lost capabilities at inference time by sampling across historical checkpoints. However, this requires storing and loading multiple checkpoints at inference time and does not prevent forgetting in the final trained policy itself. These observations also relate to non-stationarity and interference in deep RL, which can impair generalization even in stationary environments (Igl et al., 2020; Fedus et al., 2020; Kaplanis et al., 2019). Our work unifies and defines endogenous forgetting through a decomposition of prompt-level performance changes into attained gain and unrecovered loss. We investigate the underlying mechanism through controlled experiments, connecting forgetting to the implicit endogenous curriculum induced by on-policy training, and study successful-trajectory replay as a training-time mitigation.

**Relation to temporal forgetting metrics.** Li et al. (2026) define the *Ever Correct Score* $P_{\mathrm{ECS}}$ as the fraction of prompts answered correctly at least once across checkpoints, and the *Temporal Forgetting Score* $P_{\mathrm{TFS}}$ as the fraction answered correctly at some checkpoint but incorrectly at the final checkpoint. These binary metrics fit within our formulation as special cases. Specifically, let $m(q,t) \in \{0,1\}$ denote answer correctness, use the same uniformly weighted evaluation set and checkpoint set including initialization, and express all scores as fractions rather than percentages. Then

$$\mathcal{F}^{\mathrm{unrec}} = P_{\mathrm{TFS}}, \qquad \mathcal{G}^{\mathrm{att}} = P_{\mathrm{ECS}} - P_{\mathrm{Base}},$$

where $P_{\mathrm{Base}} = \bar{m}(0)$ is initial accuracy. These identities follow from $P_{\mathrm{ECS}} = \mathbb{E}_{q\sim\mathcal{D}_{\mathrm{eval}}}[m^\star(q)]$ and $P_{\mathrm{TFS}} = P_{\mathrm{ECS}} - \bar{m}(T)$. Our formulation extends this binary view to prompt-level success probabilities and provides a unified decomposition of cumulative learning and forgetting into attained gain, recovered loss, and unrecovered loss.

# D Implementation Details

## D.1 Datasets

**KK.** We adopt the Knights and Knaves (KK) task from Xie et al. (2025a) as a controlled reasoning environment. Each instance is a logic puzzle in which every inhabitant is either a knight, who always tells the truth, or a knave, who always lies, and the model must infer the identity of each inhabitant from their statements. We control task difficulty by varying the number of inhabitants, for which the space of possible identity assignments grows exponentially with group size. Fig. 7 shows an example prompt. We consider puzzles with 3–7 inhabitants and use the balanced training set with 1,000 instances at each difficulty level[5]. For evaluation, we use a disjoint KK test set with 100 examples per difficulty level (3–7).

**Math.** We train on the Polaris-53K (An et al., 2025) training dataset and evaluate on a uniformly sampled held-out set of 600 examples. Fig. 8 shows an example prompt.

**Coding.** We train on the deduplicated MBPP training set (Austin et al., 2021) which contains 320 questions and evaluate on MBPP+[6] benchmark (Liu et al., 2023). Fig. 9 shows an example prompt.

## D.2 Training and Evaluation Configurations

We implement RL training using the `verl` framework. Binary reward is assigned based on answer correctness. Table 4 reports the RL training and validation configurations. For SFT experiments, we train with batch size 256 and learning rate $10^{-5}$. For a training batch of $B$ prompts and group size $G$, we sample $BG/2$ successful prompt-response pairs from the replay buffer.

**Response truncation.** We observe that models sometimes generate nonsensical content after producing the final answer, e.g., following `\boxed{}` or `</answer>`. In all training runs, we therefore truncate any content generated after the model has produced the expected answer template.

[5] https://huggingface.co/datasets/K-and-K/knights-and-knaves

[6] https://huggingface.co/datasets/evalplus/mbppplus

You first think about the reasoning process in your mind and then provide the answer. The reasoning process and answer are enclosed within `<think> </think>` and `<answer> </answer>` tags, respectively, i.e., `<think> reasoning process here </think><answer> answer here </answer>`. List the identity of each person one by one, for example, `<answer> (1) Zoey is a knight (2) Oliver is a knight (3) ...</answer>`.

A very special island is inhabited only by knights and knaves. Knights always tell the truth, and knaves always lie. You meet 3 inhabitants: Michael, Zoey, and Ethan. Michael was heard saying, "Ethan is a knight if and only if Michael is a knight." "Zoey is a knight or Ethan is a knight," Zoey mentioned. Ethan asserted: "Michael is a knave if and only if Zoey is a knave." So who is a knight and who is a knave?

Figure 7: Example prompt for the `KK` task.

**[system]**
Please reason step by step and put the final answer in `\boxed{}`.

**[user]**
Let $x$, $y$, and $z$ be positive real numbers. Find the maximum value of

$$\frac{xyz}{(1+5x)(4x+3y)(5y+6z)(z+18)}.$$

The answer is in the form $\frac{m}{n}$, where $\gcd(m,n)=1$. Please provide the value of $m+n$. Let's think step by step and output the final answer within `\boxed{}`.

Figure 8: Example prompt for the `Math` task.

**[system]**
Please reason step by step inside a `<think> </think>` block, then put your final solution in a single `python` code block.

**[user]**
Write a function to find the longest chain which can be formed from the given set of pairs. Your implementation should pass tests like the following (more tests will be used for evaluation):

```
assert max_chain_length([Pair(5, 24), Pair(15, 25),
                         Pair(27, 40), Pair(50, 60)], 4) ==
3
```

Figure 9: Example prompt for the `Coding` task.

Table 4: **RL training and validation configurations.**

| Configuration | KK | Math (3B) | Math (8B) | Coding |
|---|---|---|---|---|
| Learning rate | $3 \times 10^{-7}$ | $10^{-6}$ | $10^{-6}$ | $10^{-6}$ |
| Training batch size | 256 | 64 | 64 | 64 |
| PPO mini-batch size | 64 | 64 | 64 | 64 |
| Group size | 16 | 8 | 16 | 8 |
| Training temperature | 1.0 | 1.0 | 1.0 | 1.0 |
| Validation temperature | 0.8 | 0.6 | 0.6 | 0.6 |
| Training top-$p$ | 1.0 | 1.0 | 1.0 | 1.0 |
| Validation top-$p$ | 0.95 | 0.95 | 0.95 | 0.95 |
| Maximum response length | 2048 | 3072 | 3072 | 3072 |
| Clipping parameter $\epsilon$ | 0.2 | 0.2 | 0.2 | 0.2 |
| Replay sampled batch size | 2048 | 256 | 512 | 256 |

### D.3 Additional SFT Training Dynamics

Fig. 10 presents pass@1, pass@64, attained gain, and unrecovered loss for the model-domain settings in Table 1.

### D.4 SFT Replay Sampling Strategies

For the Ordered SFT replay experiments in Section 2.3, the buffer $\mathcal{M}_t \subseteq \bigcup_{s<t} \mathcal{B}_s$ contains successful prompt-response pairs from earlier minibatches. Let $\mathcal{Q}_t = \{q : \exists o^+, (q, o^+) \in \mathcal{M}_t\}$ be the set of represented prompts and let $n_t(q)$ be the number of stored rollouts for prompt $q$. Our default uniform replay samples $q$ uniformly from $\mathcal{Q}_t$, then samples one of its $n_t(q)$ stored rollouts uniformly, giving $\nu_t^{\text{uniform}}(q, o^+) = 1/(|\mathcal{Q}_t| n_t(q))$. Thus, prompts receive equal probability even if they have different numbers of stored rollouts. Hard replay instead samples prompts with weights $1 - p_t(q)$, where $p_t(q) \in [0, 1]$ is the estimated success probability, and still samples a rollout uniformly within the selected prompt. CE replay directly weights individual stored pairs by their SFT loss $\ell_t(q, o^+) \geq 0$. These choices can be written as pair-level sampling weights:

$$w_t^{\text{uniform}}(q, o^+) = \frac{1}{n_t(q)}, \qquad w_t^{\text{hard}}(q, o^+) = \frac{1 - p_t(q)}{n_t(q)}, \qquad w_t^{\text{CE}}(q, o^+) = \ell_t(q, o^+).$$

For a buffered pair $e = (q, o^+)$ in a nonempty buffer with positive total weight, the normalized distribution for strategy $a \in \{\text{uniform}, \text{hard}, \text{CE}\}$ is

$$\nu_t^a(e) = \frac{w_t^a(e)}{\sum_{e' \in \mathcal{M}_t} w_t^a(e')}, \qquad e \in \mathcal{M}_t.$$

Hard replay gives greater total probability to prompts with lower estimated success probabilities, whereas CE replay gives greater probability to stored solutions with higher SFT loss. These priorities capture different properties: a prompt's success probability concerns generating any correct response, while the SFT loss measures how well the model fits a particular historical solution. All three variants use $\lambda = 1$ and a replay minibatch of the same size as the current minibatch; only the replay sampling distribution changes. Fig. 11 compares the three strategies using Qwen2.5-3B on `Math` and `Coding`. Table 5 reports attained gain and unrecovered loss for all six settings. All three strategies reduce unrecovered loss relative to Ordered SFT, but no strategy consistently achieves the lowest loss.

### D.5 Additional RL Training Dynamics

Table 6 provides the full version of Table 3, including base-model performance for all seven settings.

Figs. 12 and 13 complement Fig. 4 with learning, forgetting, and performance trajectories for four additional model-domain settings. Fig. 14 reports training reward curves for all six `Math` and

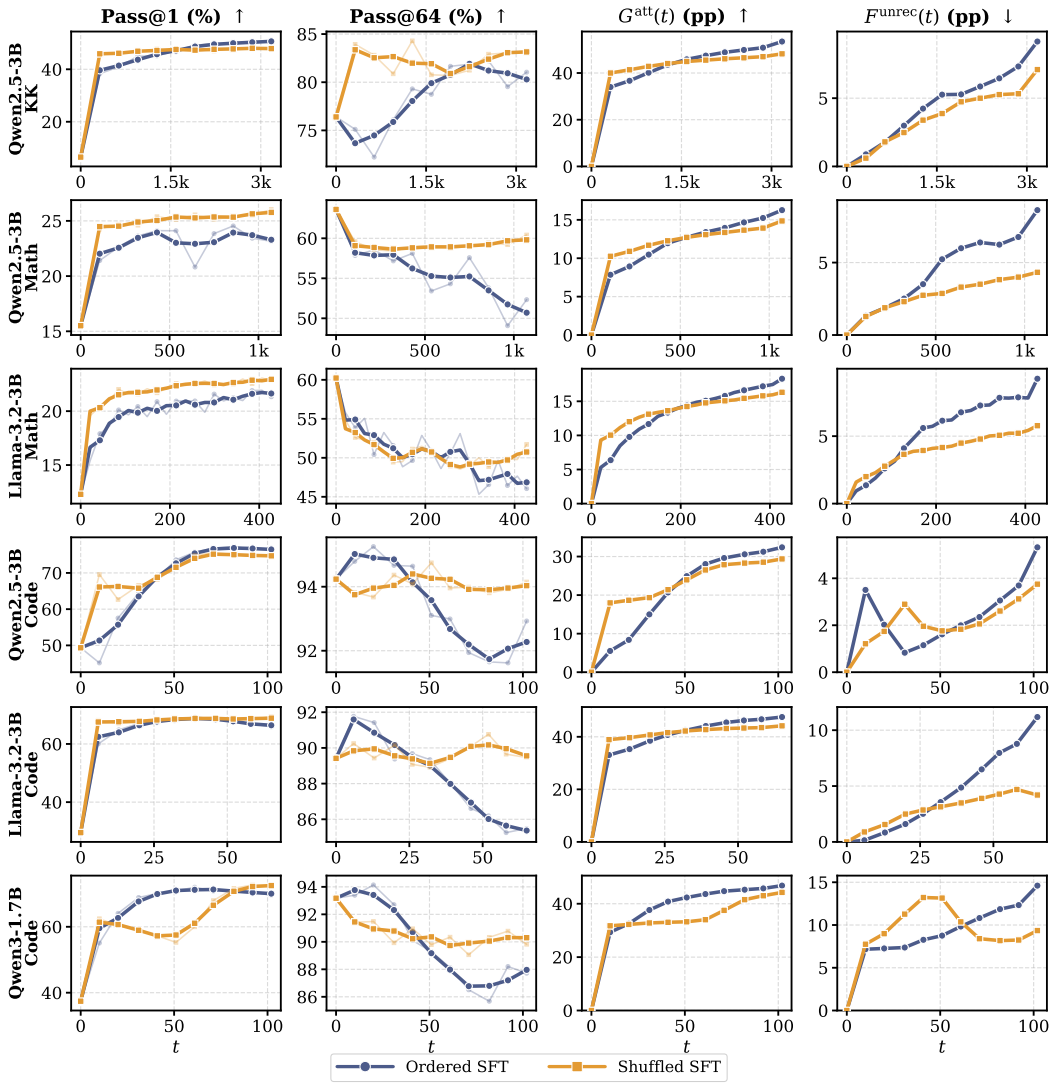


Figure 10: **Shuffling reduces final unrecovered loss and improves pass@64 across all settings.** Columns show pass@1, pass@64, attained gain $\mathcal{G}^{\text{att}}(t)$, and unrecovered loss $\mathcal{F}^{\text{unrec}}(t)$ over SFT steps $t$. Ordered SFT consistently shows higher unrecovered loss and lower coverage at the end of training.

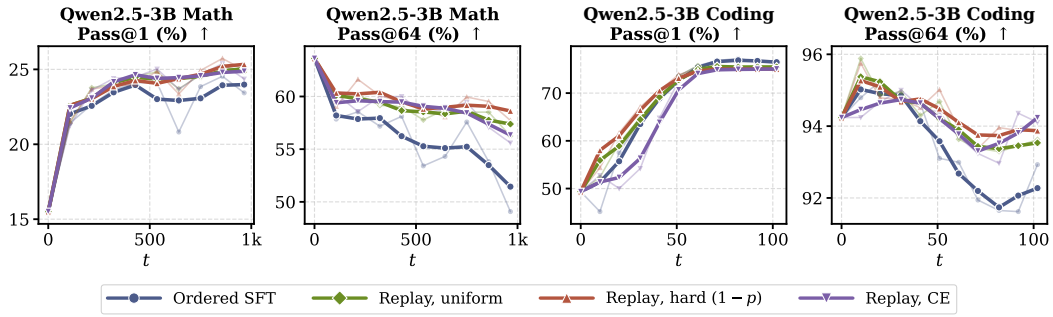


Figure 11: **Replay improves solution coverage in Ordered SFT across sampling strategies.** Pass@1 and pass@64 for Qwen2.5-3B on `Math` (left) and `Coding` (right). All three replay variants (uniform, hard, and cross-entropy) achieve higher final pass@64 than Ordered SFT in both domains, with higher final pass@1 on `Math` and slightly lower final pass@1 on `Coding`.

Table 5: **Replay reduces unrecovered loss across sampling strategies.** Attained gain $\mathcal{G}^{\text{att}}$ and unrecovered loss $\mathcal{F}^{\text{unrec}}$ are reported in percentage points. These running-peak quantities are evaluated using only the checkpoints shared by all settings in each comparison. Because the replay ablations run for fewer steps, the shared evaluation window is shorter than in Table 1 (for `KK`, $t \leq 1588$ here versus $t \leq 3175$ there). This explains why the Ordered SFT values can differ between the two tables; values should be compared within each table. Bold marks the lowest unrecovered loss in each setting, including ties.

| Domain | Model | Ordered SFT | | + Uniform | | + Hard | | + CE | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\mathcal{G}^{\text{att}}$ | $\mathcal{F}^{\text{unrec}}$ | $\mathcal{G}^{\text{att}}$ | $\mathcal{F}^{\text{unrec}}$ | $\mathcal{G}^{\text{att}}$ | $\mathcal{F}^{\text{unrec}}$ | $\mathcal{G}^{\text{att}}$ | $\mathcal{F}^{\text{unrec}}$ |
| `KK` | Qwen2.5-3B | 47.3 | 6.7 | 43.9 | **5.9** | 44.7 | 6.0 | 44.3 | 6.1 |
| `Math` | Qwen2.5-3B | 16.2 | 8.3 | 15.9 | **6.6** | 16.0 | **6.6** | 15.8 | 6.9 |
| | Llama3.2-3B-Instruct | 18.3 | 9.3 | 17.2 | **6.9** | 17.2 | 7.1 | 16.3 | **6.9** |
| `Coding` | Qwen2.5-3B | 32.4 | 5.3 | 30.5 | 4.2 | 30.3 | 4.5 | 29.9 | **4.1** |
| | Llama3.2-3B-Instruct | 47.5 | 11.2 | 44.4 | 9.1 | 44.1 | **8.5** | 44.0 | 9.0 |
| | Qwen3-1.7B-Base | 46.8 | 14.6 | 41.5 | 7.7 | 41.1 | 7.7 | 40.2 | **6.9** |

`Coding` settings. Fig. 15 complements these aggregate results with selected prompt-level trajectories on `Math` and `KK`. Fig. 16 further examines unrecovered loss under standard RL across prompts grouped by their initial pass@1.

# E Algorithms

## E.1 Learning and Forgetting Metrics

### E.1.1 Cumulative Changes and Recovered Loss

Using the evaluation distribution and checkpoint sequence defined in Section 2, let $\Delta(q, t_j) = m(q, t_j) - m(q, t_{j-1})$ denote the change in prompt-level pass@1 between consecutive checkpoints. The cumulative increases and decreases, averaged over evaluation prompts, are

$$\mathcal{G}(t_J) = \mathbb{E}_{q\sim\mathcal{D}_{\text{eval}}}\left[\sum_{j=1}^{J}[\Delta(q,t_j)]_+\right], \qquad \mathcal{F}(t_J) = \mathbb{E}_{q\sim\mathcal{D}_{\text{eval}}}\left[\sum_{j=1}^{J}[-\Delta(q,t_j)]_+\right],$$

where $[z]_+ = \max\{z, 0\}$. The cumulative increase counts both gains beyond a prompt's previous best performance and recovery after earlier declines. Only the former contribute to $\mathcal{G}^{\text{att}}$, so the

Table 6: **Full RL results, including base-model performance.** Standard denotes GRPO without replay, and Replay adds successful-trajectory replay. Colored parentheses report Replay minus Standard in percentage points, and bold compares these two methods.

| Domain | Model | Method | Pass@1 ↑ | Pass@64 ↑ | $\mathcal{G}^{\text{att}}$ ↑ | $\mathcal{F}^{\text{unrec}}$ ↓ | Monotonic imp.(%) ↑ |
|---|---|---|---|---|---|---|---|
| KK | Qwen2.5-3B | Base | 6.5 | 76.4 | – | – | – |
| | | Standard | 93.4 | 98.0 | 92.6 | 5.8 | 14.6 |
| | | Replay | **96.6** (+3.2) | **100.0** (+2.0) | **92.8** (+0.1) | **2.7** (-3.1) | **22.4** (+7.8) |
| Math | Qwen2.5-3B | Base | 15.5 | 63.6 | – | – | – |
| | | Standard | 23.1 | 54.0 | **17.8** | 10.2 | 4.3 |
| | | Replay | **25.5** (+2.4) | **62.7** (+8.7) | 14.5 (-3.3) | **4.5** (-5.7) | **16.8** (+12.5) |
| | Llama3.2-3B-Instruct | Base | 12.3 | 60.2 | – | – | – |
| | | Standard | 19.7 | 42.7 | **15.6** | 8.2 | 5.8 |
| | | Replay | **19.9** (+0.2) | **57.8** (+15.1) | 10.4 (-5.2) | **2.8** (-5.4) | **14.8** (+9.0) |
| | Qwen3-8B-Base | Base | 23.7 | 77.3 | – | – | – |
| | | Standard | 34.4 | 68.8 | **28.9** | 18.2 | 6.5 |
| | | Replay | **41.3** (+6.9) | **75.5** (+6.7) | 23.4 (-5.4) | **5.9** (-12.3) | **24.3** (+17.8) |
| Coding | Qwen3-1.7B-Base | Base | 37.5 | 93.3 | – | – | – |
| | | Standard | 71.2 | 80.6 | 42.7 | 8.8 | 57.7 |
| | | Replay | **76.3** (+5.1) | **93.2** (+12.6) | **43.0** (+0.4) | **4.2** (-4.6) | **74.3** (+16.7) |
| | Qwen2.5-3B | Base | 49.4 | 94.2 | – | – | – |
| | | Standard | 76.6 | 90.0 | **33.4** | 6.2 | 58.2 |
| | | Replay | **78.3** (+1.7) | **93.0** (+3.0) | 32.5 (-0.9) | **3.6** (-2.6) | **73.0** (+14.8) |
| | Llama3.2-3B-Instruct | Base | 29.6 | 89.6 | – | – | – |
| | | Standard | **69.8** | 80.1 | **49.2** | 8.9 | 48.1 |
| | | Replay | 69.7 (-0.1) | **88.0** (+7.9) | 46.7 (-2.4) | **6.6** (-2.4) | **62.4** (+14.3) |

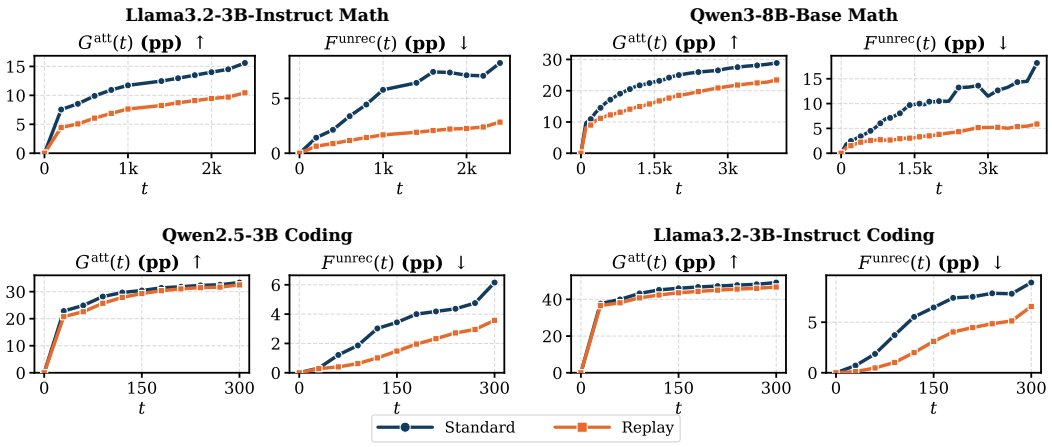


Figure 12: **Replay reduces unrecovered loss across additional `Math` and `Coding` settings.** Paired panels show attained gain $\mathcal{G}^{\text{att}}(t)$ and unrecovered loss $\mathcal{F}^{\text{unrec}}(t)$ over training steps $t$, in percentage points, for Llama3.2-3B-Instruct and Qwen3-8B-Base on Math (top), and Qwen2.5-3B and Llama3.2-3B-Instruct on Coding (bottom). Circles denote GRPO; squares denote GRPO with replay.

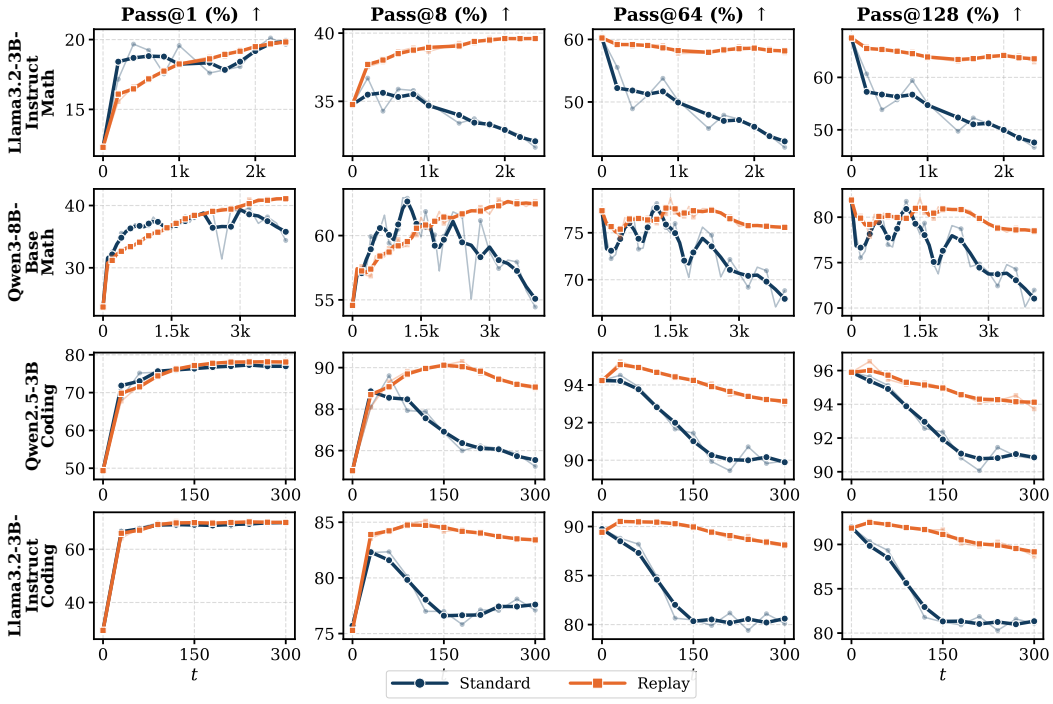


Figure 13: **Replay better preserves large-$k$ solution coverage during RL training.** Columns show pass@1, pass@8, pass@64, and pass@128 (%) over training steps $t$ for the four additional model-domain settings in Fig. 12.

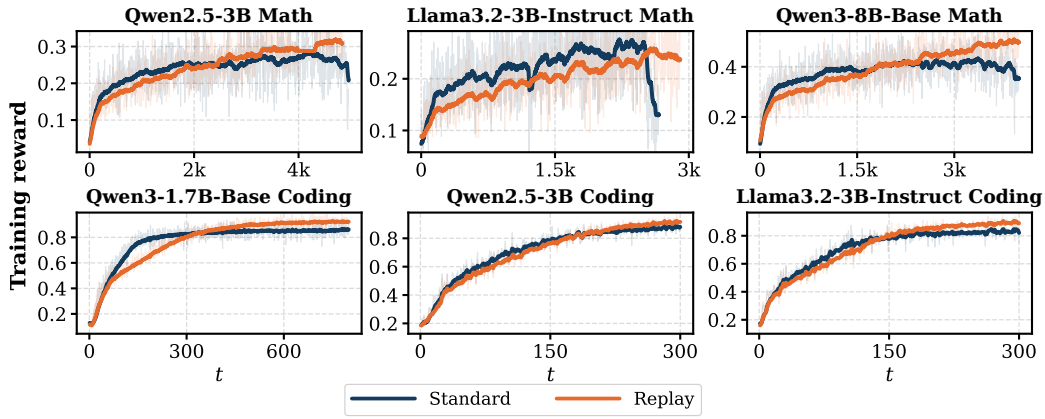


Figure 14: **Replay achieves higher final training rewards in the six plotted settings.** Training reward over steps $t$ for GRPO and GRPO with replay on `Math` (top) and `Coding` (bottom).

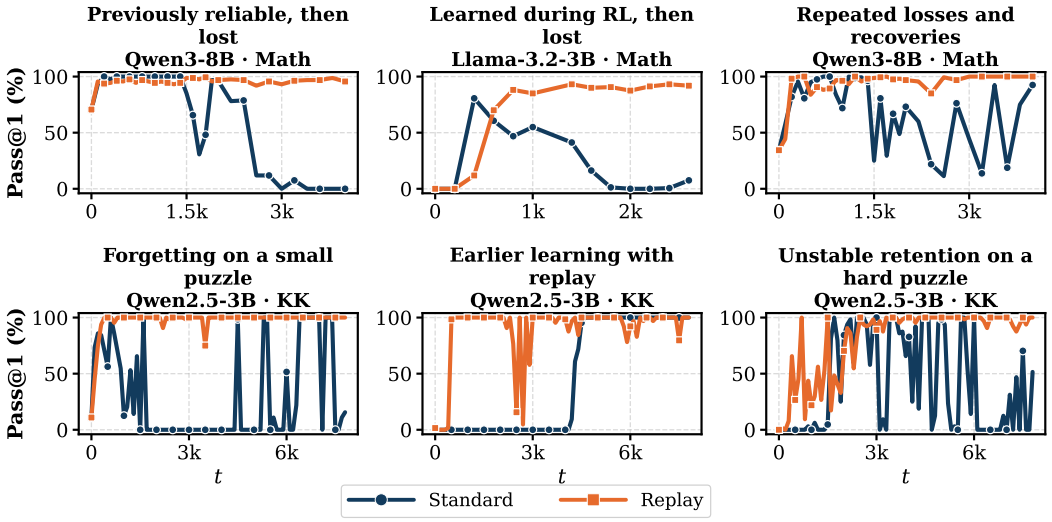


Figure 15: **Qualitative examples of prompt-level learning and forgetting.** Each panel tracks estimated pass@1 for one evaluation prompt over training steps $t$ under standard RL and RL with replay. Top: `Math` examples illustrate loss of previously reliable performance, loss of gains acquired during RL, and repeated forgetting and recovery. Bottom: `KK` examples illustrate forgetting on a small puzzle, earlier learning with replay, and unstable retention on a harder puzzle. In these selected examples, replay better retains earlier successes or reaches high success probability sooner.

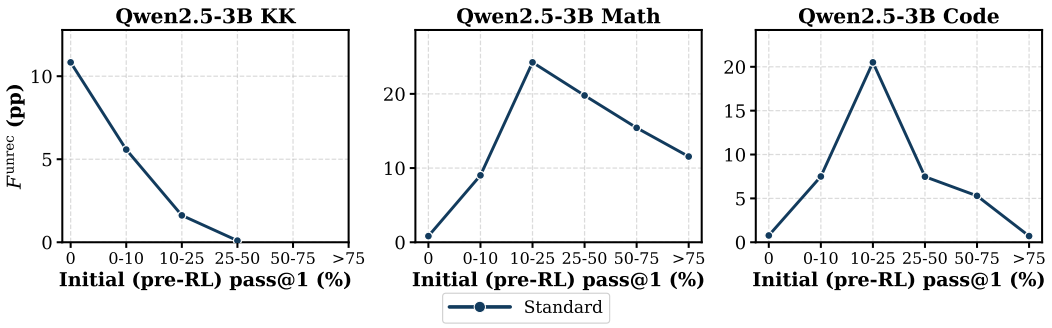


Figure 16: **Unrecovered loss stratified by initial prompt success rate.** Unrecovered loss $\mathcal{F}^{\text{unrec}}$ (percentage points) under standard RL for Qwen2.5-3B on `KK`, `Math`, and `Coding`, with prompts grouped by their pre-RL pass@1. The largest losses occur in the zero-success bin on `KK` and the 10–25% bin on `Math` and `Coding`.

**Algorithm 1** On-policy RL with successful-trajectory replay

**Require:** Training distribution $\mathcal{D}_{\mathrm{tr}}$, replay buffer $\mathcal{M}$, and fixed replay weight $\lambda \geq 0$
Initialize policy $\pi_0$ and $\mathcal{M} \leftarrow \varnothing$
**for** $t = 0, \ldots, T-1$ **do**
  Sample prompts and generate on-policy rollout groups from $\pi_t$
  Compute rewards and the on-policy objective $\mathcal{J}_t(\theta)$
  Update $\widetilde{p}_t(q)$ for sampled prompts
  **if** $\mathcal{M} \neq \varnothing$ **then**
    Sample replay prompts using $\nu_t^{\mathrm{prompt}}$ (uniformly if total priority is zero)
    Sample stored successful trajectories uniformly for each prompt
    Compute the replay objective $\mathcal{J}_{\mathrm{rep},t}(\theta)$
  **else**
    Set $\mathcal{J}_{\mathrm{rep},t}(\theta) \leftarrow 0$
  **end if**
  Update $\theta_t$ to $\theta_{t+1}$ by maximizing $\mathcal{J}_t(\theta) + \lambda \mathcal{J}_{\mathrm{rep},t}(\theta)$
  Record $\tau^{\mathrm{rep}}(q)$ for replayed prompts
  Add newly generated successful trajectories to $\mathcal{M}$
**end for**

remaining component $\mathcal{C} = \mathcal{G}(t_J) - \mathcal{G}^{\mathrm{att}}$ is the *recovered loss*: performance regained after an earlier decrease, up to the previous best. Since consecutive changes telescope, $\mathcal{G}(t_J) - \mathcal{F}(t_J) = \bar{m}(t_J) - \bar{m}(t_0) = \mathcal{G}^{\mathrm{att}} - \mathcal{F}^{\mathrm{unrec}}$, giving

$$\mathcal{G}(t_J) = \mathcal{G}^{\mathrm{att}} + \mathcal{C}, \qquad \mathcal{F}(t_J) = \mathcal{F}^{\mathrm{unrec}} + \mathcal{C}.$$

Thus, recovered losses contribute equally to cumulative increases and decreases and cancel in the overall performance change.

#### E.1.2 Prompt Groups

Inspired by the example-level view of learning dynamics (Toneva et al., 2018), we call $\Delta(q, t_j) > 0$ a *learning event* and $\Delta(q, t_j) < 0$ a *forgetting event*. We define prompts with *monotonic improvement* (at least one increase and no decreases) as:

$$\mathcal{Q}_{\text{gain-only}} = \{q \in \mathcal{Q}_{\mathrm{eval}} : \exists j, \Delta(q, t_j) > 0 \text{ and } \forall j, \Delta(q, t_j) \geq 0\}.$$

#### E.1.3 Threshold Calibration

At each checkpoint $t_j$, we draw $n$ independent responses for every evaluation prompt $q$. Let $c_q(t_j)$ denote the number of correct responses and estimate pass@1 as $\hat{m}(q, t_j) = c_q(t_j)/n$. The observed checkpoint-to-checkpoint change is $\hat{\Delta}(q, t_j) = \hat{m}(q, t_j) - \hat{m}(q, t_{j-1})$. Finite-rollout variation can produce apparent positive or negative changes even when the underlying policy performance is stationary, so we calibrate these changes under an exact conditional null.

For each prompt and adjacent checkpoint pair, let $T_q(t_j) = c_q(t_{j-1}) + c_q(t_j)$. Under the null hypothesis that the single-response success probability is unchanged across the interval, conditioning on $T_q(t_j)$ eliminates the unknown success probability, and $c_q^*(t_j) \mid T_q(t_j) \sim \mathrm{Hypergeometric}(2n, T_q(t_j), n)$, with $c_q^*(t_{j-1}) = T_q(t_j) - c_q^*(t_j)$. We use this distribution to compute the exact conditional null distribution of $\hat{\Delta}^*(q, t_j)$.

We define $\epsilon(q, t_j)$ as the smallest value satisfying $\Pr_0(|\hat{\Delta}^*(q, t_j)| \leq \epsilon(q, t_j) \mid T_q(t_j)) \geq 1 - \alpha$, with the two-sided nominal level $\alpha = 0.05$. We classify a prompt as improving when $\hat{\Delta}(q, t_j) > \epsilon(q, t_j)$ and degrading when $\hat{\Delta}(q, t_j) < -\epsilon(q, t_j)$. Under the symmetric conditional null, each directional false-positive rate is at most $\alpha/2 = 0.025$. Because the null distribution is discrete, the achieved rates can be lower.

### E.2 RL with Replay Algorithm

Algorithm 1 shows the full RL replay algorithm.

*Zero-priority fallback.* Let $\mathcal{Q}(\mathcal{M}_t)$ denote the set of prompts represented in the replay buffer. If the total prompt priority is zero for a nonempty buffer, we use $\nu_t^{\text{prompt}}(q) = 1/|\mathcal{Q}(\mathcal{M}_t)|$ for $q \in \mathcal{Q}(\mathcal{M}_t)$. We then sample a stored successful trajectory uniformly within each selected prompt, as in the positive-priority case. The replay term is omitted when the buffer is empty.