FRESHNESS-AWARE PRIORITIZED EXPERIENCE REPLAY FOR LLM REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) for large language models (LLMs) is expanding from mathematical reasoning to agentic tasks that require costly, multi-turn environment interactions. On-policy training repeatedly collects fresh trajectories and typically does not reuse them across rollout iterations. Experience replay can reuse these interactions, but replay priorities alone may not reflect their relevance as the policy evolves. We introduce FreshPER, a freshness-aware prioritized experience replay method for LLM training. FreshPER combines a base priority with exponential decay in trajectory age to balance priority and recency in the replay distribution. It augments fresh-rollout updates with historical replay while preserving the underlying RL objective and the current-to-behavior policy-ratio correction. Across Sokoban experiments with PPO, GRPO, and a ROLL REINFORCE++ variant, FreshPER improves over priority replay: under PPO, it reaches 43.0% validation success, compared with 33.6% for priority replay and 3.1% for on-policy training. Further results on NQ Search, ScienceWorld, and visual tasks support freshness-aware replay as a practical and effective way to learn more from costly interactions across diverse settings.

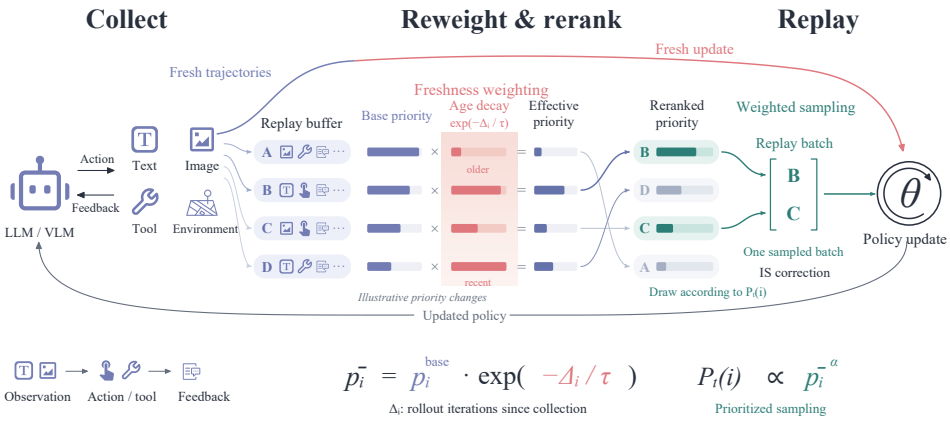


Figure 1: Fresh-plus-replay training. Each rollout iteration collects and optimizes a fresh batch, stores its replay units, and performs additional optimization on up to K eligible historical batches sampled using age-adjusted priorities.

1 INTRODUCTION

Reinforcement learning (RL) for large language models (LLMs) now spans mathematical reasoning (Shao et al., 2024; DeepSeek-AI, 2025) and agents that interact with external environments (Jin et al., 2025; Wang et al., 2025c; Zhai et al., 2024). For multi-turn agents, a trajectory contains repeated generation, tool calls, and environment feedback. Acquiring another training batch therefore requires running these interactions again, making data collection a substantial part of the training process.

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.

On-policy training with Proximal Policy Optimization (PPO) (Schulman et al., 2017), REINFORCE++ (Hu et al., 2025), or Group Relative Policy Optimization (GRPO) (Shao et al., 2024) may optimize a fresh batch for several passes, but typically does not retain it across rollout iterations. This leaves potentially useful interactions unavailable to later updates. Experience replay extends their use beyond the iteration in which they were collected, allowing further optimization without repeating the associated environment execution.

Effective replay must account for both learning value and policy mismatch: a useful trajectory can become less suitable for optimization as the policy evolves. We study whether replay improves task performance at a fixed budget of newly collected trajectories, which we call *fresh-trajectory efficiency*. We evaluate this acquisition budget alongside actor-update and wall-clock budgets to assess how effectively replay uses additional training.

Prioritized experience replay (PER) favors samples with high estimated learning value (Schaul et al., 2016). In LLM post-training, RLEP retains verified successful trajectories (Zhang et al., 2025), while RAPER combines advantage-based priorities with an age limit (Yoo et al., 2026). However, reward or advantage signals need not track the mismatch between the collection and current policies. A hard age cutoff determines which samples remain eligible, but by itself provides no graded preference for newer samples within that set.

FreshPER multiplies each trajectory or prompt group's base priority by an exponential factor based on its collection age. Among units with equal base priorities, this favors newer experience. We use rollout age as an inexpensive proxy for policy drift and tune the decay timescale empirically; Section D discusses the connection to effective sample size. The resulting sampler supplies replay batches alongside fresh updates, using stored behavior-policy log-probabilities for replay optimization (Fig. 1).

We implement FreshPER in ROLL (Wang et al., 2025b) and evaluate it with PPO, GRPO, and a ROLL REINFORCE++ variant. Matched Sokoban experiments show the largest gains over Priority under PPO and REINFORCE++, with smaller gains under GRPO. Our contributions are:

1. **Age-aware replay selection.** We combine base priorities with exponential collection-age weighting, providing a relative preference for recent experience within the eligible replay set (Section 3).
2. **Replay integration across policy optimizers.** Our implementation supports trajectory and prompt-group replay, retains behavior-policy log-probabilities, and recomputes values and GAE for PPO replay (Section A).
3. **Evaluation across acquisition and training budgets.** Matched Sokoban comparisons assess age weighting across algorithms; NQ Search, ScienceWorld, and visual tasks examine broader applicability. We complement trajectory-budget results with actor-update and elapsed-time comparisons, together with decay sensitivity and task-dependent behavior (Section 4).

2 Related Work

Policy Optimization for Language Models. PPO (Schulman et al., 2017), critic-free methods such as ReMax, RLOO, and REINFORCE++ (Li et al., 2023; Ahmadian et al., 2024; Hu et al., 2025), and GRPO (Shao et al., 2024) differ in their advantage estimators and optimization rules. DAPO and Dr. GRPO further examine clipping, sampling, and objective normalization (Yu et al., 2025; Liu et al., 2025a). We study the selection of historical experience for additional updates within these training procedures.

Reinforcement Learning for Interactive Agents. ReAct and Toolformer developed reasoning–action and tool-use paradigms through prompting and self-supervision, respectively (Yao et al., 2023; Schick et al., 2023). Search-R1, R1-Searcher, and ReSearch train retrieval decisions through RL (Jin et al., 2025; Song et al., 2025; Chen et al., 2025). Related work studies web agents (Qi et al., 2024; Wei et al., 2025b), software engineering (Wei et al., 2025a), multi-turn training stability (Wang et al., 2025c), and visual decision-making (Zhai et al., 2024; Wang et al., 2025a). Our evaluation draws on retrieval, interactive planning, and visual tasks to examine the reuse of trajectories from these settings.

Rollout Efficiency and Experience Replay. AReaL and ROLL Flash improve throughput by overlapping rollout collection and optimization (Fu et al., 2025; Lu et al., 2025). Experience replay instead obtains further updates from stored interactions (Lin, 1992; Mnih et al., 2015), with performance affected by prioritized sampling (Schaul et al., 2016; Horgan et al., 2018), buffer capacity, and replay ratio (Fedus et al., 2020). In LLM training, RLEP selects verified successes (Zhang et al., 2025), RAPER prioritizes rollouts by group-relative advantage magnitudes (Yoo et al., 2026), and CRPS replays suffix states from successful trajectories (Zhang & Yang, 2026). Other approaches couple replay with difficulty-targeted acquisition (Sun et al., 2025) or model resets (Liu et al., 2025b). Problem prioritization acts on which questions receive new rollouts (Fatemi, 2026), whereas FreshPER weights experience already in the buffer.

Policy mismatch also shapes how historical data are used. Tapered Off-Policy REINFORCE applies importance weighting (Le Roux et al., 2025); RAPER restricts rollout age; and FPER discounts priorities by replay count (Ma et al., 2022). These mechanisms act on policy ratios, replay eligibility, and past usage, respectively. FreshPER adds a relative preference based on collection age among eligible replay units, while retaining behavior-policy information for optimization.

3 METHOD: FRESHPER

3.1 FRESH ROLLOUTS AND HISTORICAL REPLAY

At outer iteration t, behavior policy π_{μ_t} collects a fresh trajectory batch $\mathcal{F}_t$. We optimize on this batch, insert its replay units into the buffer, and then optimize on up to K eligible historical batches. A replay unit is either a trajectory or a complete prompt group; GRPO preserves complete groups throughout replay. Each trajectory stores its prompt, assistant turns, observations, reward, and behavior log-probabilities. FreshPER retains the underlying algorithm's per-sample loss while changing the selection and weighting of historical data (Fig. 2).

Stored behavior log-probabilities must therefore correspond to the policy that generated each trajectory, before any fresh or replay optimization. They are retained when a trajectory is reused. For PPO, values and GAE are instead recomputed with the current critic; GRPO preserves the original prompt groups used to form its relative learning signal.

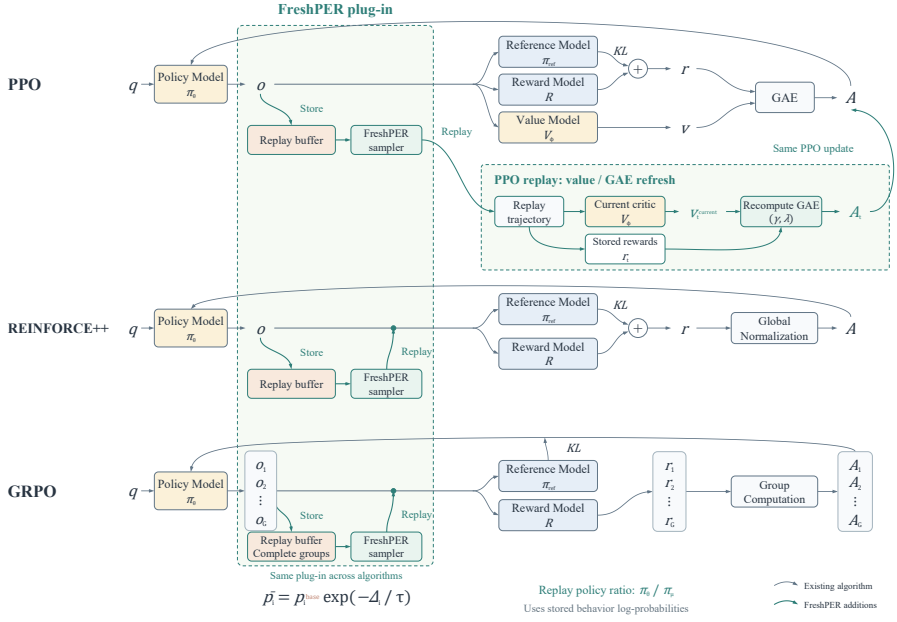


Figure 2: Replay integration with PPO, REINFORCE++, and GRPO. Teal paths show buffer storage and freshness-aware sampling. GRPO preserves prompt groups; PPO recomputes values and GAE with the current critic. All replay updates use stored behavior-policy log-probabilities.

3.2 FRESHNESS-AWARE PRIORITIZED SAMPLING

We use *priority staleness* to describe replay units that retain high base priorities despite growing mismatch between their collection policy and the current policy. A reward- or advantage-based priority need not track this mismatch. FreshPER combines the base priority used by PER (Schaul et al., 2016) with collection-age weighting.

Let $\mathcal{B}_t$ be the replay buffer and $\mathcal{I}_t \subseteq \mathcal{B}_t$ its units satisfying the age and signal filters. Unit i, collected at iteration c_i, has base priority $p_i^{\text{base}}(t) \geq 0$ and age $\Delta_i(t) = t - c_i$. For $0 < \alpha \leq 1$ and positive total priority, define

$$\bar{p}_i(t) = p_i^{\text{base}}(t) \exp\left(-\frac{\Delta_i(t)}{\tau}\right), \qquad P_t(i) = \frac{\bar{p}_i(t)^\alpha}{\sum_{j \in \mathcal{I}_t} \bar{p}_j(t)^\alpha}, \quad i \in \mathcal{I}_t. \tag{1}$$

Here $\tau > 0$ controls age decay and α controls prioritization. The resulting age factor in the sampling distribution is $\exp[-\alpha\Delta_i(t)/\tau]$. We define $\alpha = 0$ as uniform sampling over $\mathcal{I}_t$; $\tau \to \infty$ recovers PER with the same eligibility filters. Base priorities depend on the training algorithm, including reward-based scores for trajectories and reward standard deviation for GRPO groups (Section F).

Age weighting separates replay eligibility from sampling preference. Capacity constraints and signal or age filters determine which units are available; the age factor adjusts their relative sampling probabilities within that set. An older unit with a sufficiently high base priority can still be selected in preference to newer experience.

Motivation for exponential age weighting. For the corresponding importance-weighted population, normalized ESS satisfies $\text{ESS}/n \leq \exp(-D_{\text{KL}})$ (Section D). Increasing policy divergence thus lowers an exponential upper bound on effective sample size. Approximating divergence as locally proportional to rollout age motivates the exponential factor in Eq. (1). We use this relationship to choose a weighting family and select its timescale empirically.

Collection age is available for every replay unit without estimating trajectory-distribution divergence for each historical policy. Its relationship to policy drift depends on the training schedule: an outer iteration can contain several replay updates of different magnitudes. Accordingly, τ is task-specific, and age is measured from collection rather than the most recent priority write-back. Updating the estimated learning value of a trajectory does not make its behavior policy more recent.

3.3 REPLAY OPTIMIZATION AND IMPLEMENTATION

Replay uses stored behavior log-probabilities in the current-to-behavior policy ratio and the underlying algorithm's clipping rule. Separately, optional PER weights adjust for nonuniform replay sampling. Let $N_t = |\mathcal{I}_t|$. When all eligible units have positive sampling probability, the normalized weights are

$$u_i(t) = \frac{(N_t P_t(i))^{-\beta}}{\max_{j \in \mathcal{I}_t} (N_t P_t(j))^{-\beta}}, \qquad \beta \in [0, 1]. \tag{2}$$

For mean aggregation, the replay loss is $\mathcal{L}_{\text{replay}} = |\mathcal{S}|^{-1} \sum_{i \in \mathcal{S}} u_i(t)\, \ell_i^{\text{off}}$, where $\mathcal{S}$ is the replay batch and ℓ_i^{off} uses the stored behavior log-probabilities. A complete group shares one sampling weight across its trajectories; other reductions follow the underlying algorithm. Setting $\beta = 0$ disables PER weighting while retaining the current-to-behavior policy ratio. Group sampling is without replacement, so weights based on the initial draw probabilities provide only an approximate correction (Section A.2).

The CPU-side buffer refreshes effective priorities before sampling and updates base priorities when configured. Collection ages remain unchanged after replay. Warmup and eligibility checks determine when replay begins; Algorithm 1 and Section A give the full procedure.

4 EXPERIMENTS

We evaluate whether freshness-aware replay improves learning at a fixed acquisition budget and whether the gains persist after accounting for its additional optimization. Matched Sokoban experiments isolate age weighting; NQ Search and ScienceWorld compare complete configurations

in multi-turn environments. We then examine update and time budgets, parameter sensitivity, and visual tasks.

4.1 Experimental Setup

Comparisons. **On-policy** uses fresh batches without cross-iteration reuse; **Priority** adds replay without age decay; and **FreshPER** adds freshness weighting. We implement these strategies in ROLL (Wang et al., 2025b) with a REINFORCE++ normalization variant, GRPO, and PPO replay with recomputed values and GAE (Section F). Models range from 0.5B parameters on Sokoban to 7B on NQ Search and 8B on ScienceWorld; visual tasks use a 3B VLM (Section E).

Evaluation. Our primary budget is the number of newly collected trajectories, $B_{\text{fresh}}(T) = \sum_{t=1}^{T} |\mathcal{F}_t|$. It counts each member of a prompt group and includes RLEP's preparatory acquisition; replay adds no fresh trajectories. Update-count and wall-clock comparisons complement this acquisition budget by accounting for additional replay optimization. We report each task's validation success, reward, exact match (EM), or development-set score. Each curve represents one training run; captions identify any smoothing. A training step denotes an outer rollout iteration. Sections C.1 and E provide accounting and evaluation details.

4.2 Fresh-Trajectory Efficiency Across RL Algorithms

We compare On-policy, Priority, and FreshPER separately under REINFORCE++, GRPO, and PPO on Sokoban. Each run collects approximately 32,000 fresh trajectories. Within each algorithm, Priority and FreshPER share the model, base priority, replay budget, and optimization settings; only FreshPER applies age decay (Section F).

Each replay batch contains as many trajectories as a fresh batch. REINFORCE++ and PPO process up to two replay batches per rollout iteration, whereas GRPO processes up to one. Validation uses 128 fixed-seed Sokoban boards every ten rollout iterations, with the same decoding settings across training strategies (Sections E and F).

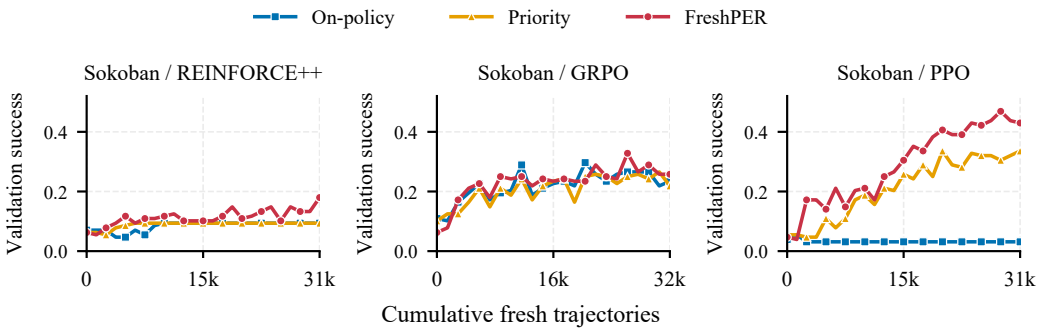


Figure 3: Sokoban validation success versus cumulative fresh trajectories under REINFORCE++, GRPO, and PPO. Curves show raw validation measurements for the three training strategies within each algorithm.

At 30,720 fresh trajectories, PPO validation success reaches 43.0% with FreshPER, compared with 33.6% for Priority and 3.1% for On-policy (Fig. 3). Under the ROLL REINFORCE++ variant, the corresponding scores are 18.0%, 9.4%, and 9.4%. Thus, FreshPER exceeds Priority by 9.4 and 8.6 percentage points in these two runs. The GRPO curves overlap more closely, showing a smaller benefit from age weighting.

PPO benefits substantially from replay even without age weighting, while FreshPER yields a further improvement over Priority. Under REINFORCE++, Priority ends at the same score as On-policy, whereas FreshPER continues to improve. These differences show why replay frequency alone is insufficient to characterize its benefit: the choice of historical data matters, and its effect depends on the policy optimizer.

4.3 Replay in Multi-Turn Environments

On NQ Search (Jin et al., 2025), agents alternate reasoning and retrieval before producing a final answer. We compare On-policy, Priority, and FreshPER under REINFORCE++. Additional on-policy PPO and GRPO runs use different model or framework settings and provide contextual references (Section E).

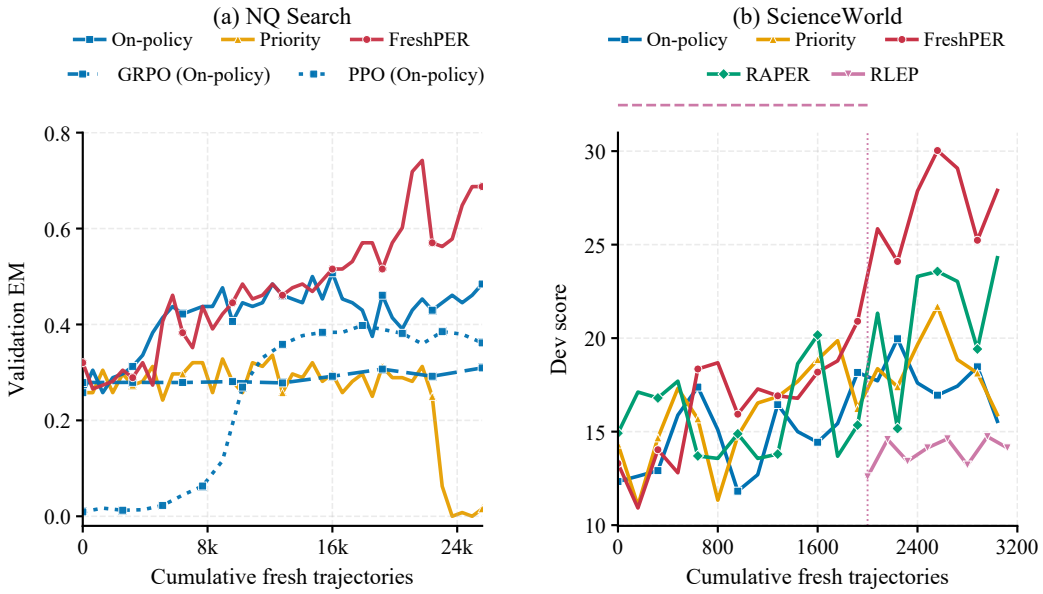


Figure 4: Validation performance versus fresh trajectories. (a) NQ Search: solid curves compare REINFORCE++ strategies; dashed GRPO and dotted PPO curves provide on-policy references. (b) ScienceWorld: five GRPO-based configurations. RLEP's bracket marks 800 preparatory and 1,200 pool-collection trajectories. Curves are unsmoothed; Sections C.1 and F.3 give protocols.

On-policy EM levels off while FreshPER continues to improve; Priority deteriorates late in training (Fig. 4(a)). At 25,600 fresh trajectories, validation EM is 0.688 for FreshPER, 0.484 for On-policy, and 0.016 for Priority. Historical reuse therefore has markedly different outcomes under the two replay configurations. The methods use their task-specific buffer capacities, base priorities, and learning-rate schedules listed in Section F.

ScienceWorld. On ScienceWorld (Wang et al., 2022), we compare RAPER (Yoo et al., 2026) and RLEP (Zhang et al., 2025) with On-policy, Priority, and FreshPER. Episodes allow up to 30 environment commands across 30 task types, and evaluation uses 60 fixed task–variation pairs. All five GRPO-based configurations use the same environment, development set, and acquisition budget of 3,200 fresh trajectories. Evaluation uses ScienceWorld's native Dev score. Each method uses its corresponding optimization settings, including learning-rate and PER-weight schedules (Section F.3).

FreshPER obtains higher Dev scores than On-policy, Priority, and RAPER at the final recorded evaluations (Fig. 4(b)). RLEP spends 800 trajectories on preparatory training and 1,200 on success-pool collection before resetting the policy and collecting another 1,200 for main training. All three stages count toward its acquisition budget. It retrieves matching successes for only six of 75 main-training groups, totaling 12 replayed trajectories. This retrieval count characterizes the success-pool coverage available to RLEP in the comparison.

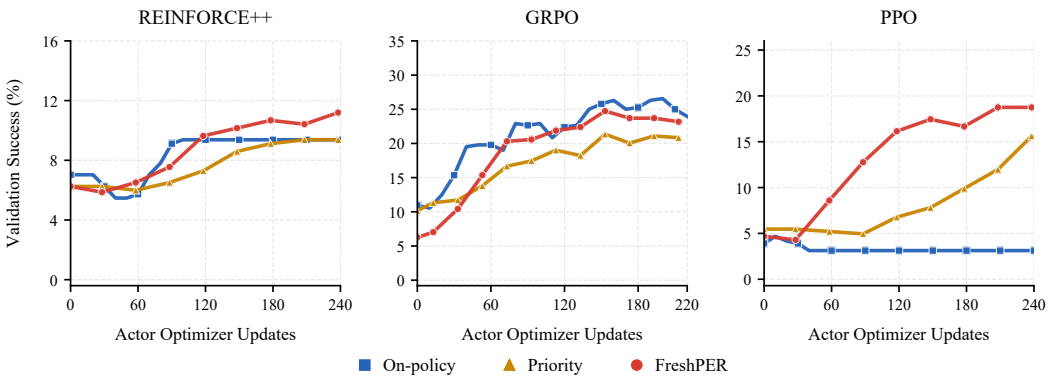
4.4 Performance Under Time and Update Budgets

Replay performs additional gradient updates between rollout iterations. Higher performance at a fixed acquisition budget could therefore reflect a larger optimization budget. To examine this possibility, we compare Sokoban at comparable actor-update counts and NQ Search at common elapsed

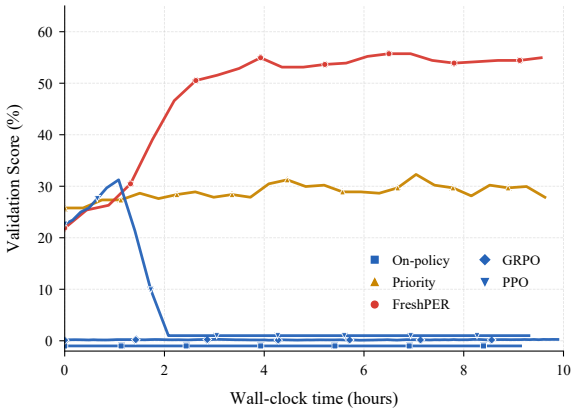
training times (Fig. 5). The update-count view compares performance at comparable cumulative actor-update counts, while elapsed time includes the cost of performing replay alongside data collection. Together, they test whether the acquisition-budget gains persist under budgets that account for the additional training.

Optimizer Updates on Sokoban. Using equal-sized fresh and replay batches within each algorithm, we compare performance against cumulative actor updates, counting both fresh and replay optimization (Fig. 5a). FreshPER outperforms both controls under PPO, achieves higher late-training success under REINFORCE++, and improves on Priority under GRPO. At comparable actor-update budgets, replay draws part of its training data from stored trajectories, reducing fresh-rollout requirements relative to on-policy training.

Wall-Clock Time on NQ Search. We compare the REINFORCE++ On-policy, Priority, and FreshPER configurations using Qwen2.5-7B-Instruct on 8 NVIDIA H200 GPUs. At common elapsed times within the runs' observed overlap, FreshPER overtakes Priority and maintains higher validation performance later in training (Fig. 5(b)). This comparison includes time spent on rollout collection and the additional off-policy replay updates, showing that the improvement over Priority persists under a comparable training-time budget. The PPO and GRPO curves provide additional references with the model and framework settings in Section E.



(a) Sokoban: actor-update budget.



(b) NQ Search: wall-clock time budget.

Figure 5: Performance under actor-update and wall-clock budgets. (a) Sokoban validation success versus actor optimizer updates under REINFORCE++, GRPO, and PPO, counting both fresh and replay updates. (b) NQ Search validation score versus elapsed training time; the main REINFORCE++ comparison uses Qwen2.5-7B-Instruct on eight H200 GPUs, with PPO and GRPO as additional references. Curves use trailing means of up to three evaluations. Section C.1 details the accounting and display conventions.

4.5 Effects of Freshness and Sampling Correction

We compare three finite decay timescales and separately toggle PER sampling correction at fixed τ (Section B). These experiments use separate Sokoban Simple and text-FrozenLake runs with the ROLL REINFORCE++ variant.

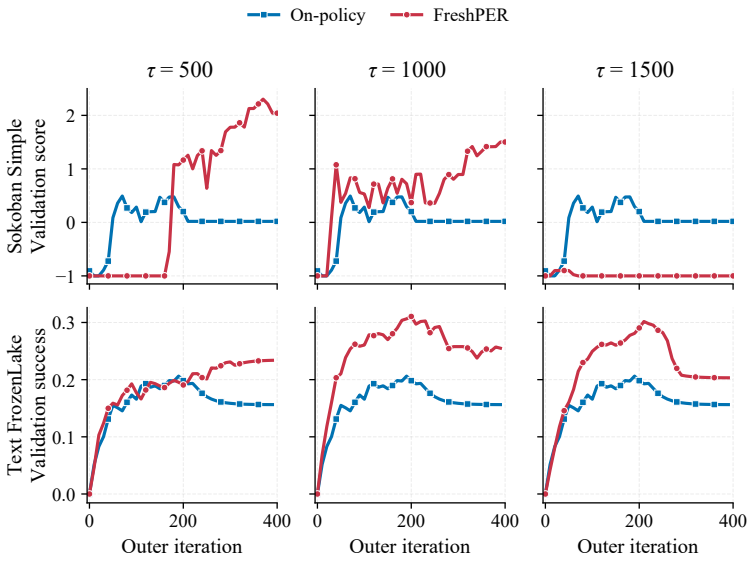


Figure 6: Decay-timescale sensitivity on Sokoban Simple (top) and text FrozenLake (bottom), with 128 fresh trajectories per outer iteration. Columns compare finite values of τ against the same no-replay On-policy reference. FrozenLake curves use EMA smoothing with $\alpha_{\text{EMA}} = 0.3$.

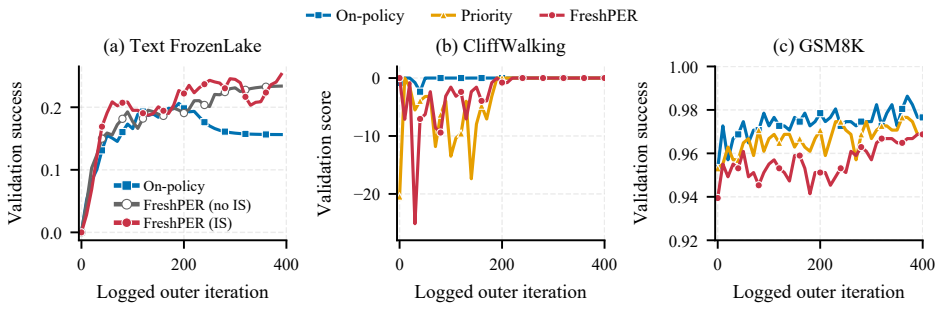


Figure 7: (a) PER sampling correction on text FrozenLake at $\tau = 500$: disabled versus $\beta = 0.4$. (b,c) CliffWalking and GSM8K show limited additional benefit under the evaluated interaction protocols; Sections B and E give details.

The decay-timescale comparison identifies different preferred settings for the two tasks (Fig. 6). On Sokoban Simple, $\tau = 500$ gives the strongest late-training performance, followed by $\tau = 1000$, while $\tau = 1500$ remains near its initial score. Text FrozenLake reaches its highest late-training success with $\tau = 1000$. These comparisons guide the choice of the freshness timescale; the matched Priority experiments separately measure the effect of introducing age weighting.

At fixed $\tau = 500$, PER sampling correction improves text-FrozenLake performance during substantial portions of training; the weighted and unweighted replay curves cross near the end (Fig. 7(a)). Both replay variants finish above On-policy. CliffWalking reaches the same final score with all three strategies, while GSM8K starts near saturation under the evaluated answer-attempt protocol (Section B). These settings complement the interactive tasks by testing replay when the baseline already attains high task performance.

4.6 Applicability to Vision-Language Tasks

We evaluate visual FrozenLake, GeoQA (Chen et al., 2021), and MathVista (Lu et al., 2023) with Qwen2.5-VL-3B-Instruct and the ROLL REINFORCE++ variant. Priority and FreshPER share reward-based priorities and task-specific buffer capacities; FreshPER uses $\tau = 1000$ (Sections E and F).

Visual FrozenLake requires actions from rendered navigation states, whereas GeoQA and MathVista require image-grounded answers with correctness feedback.

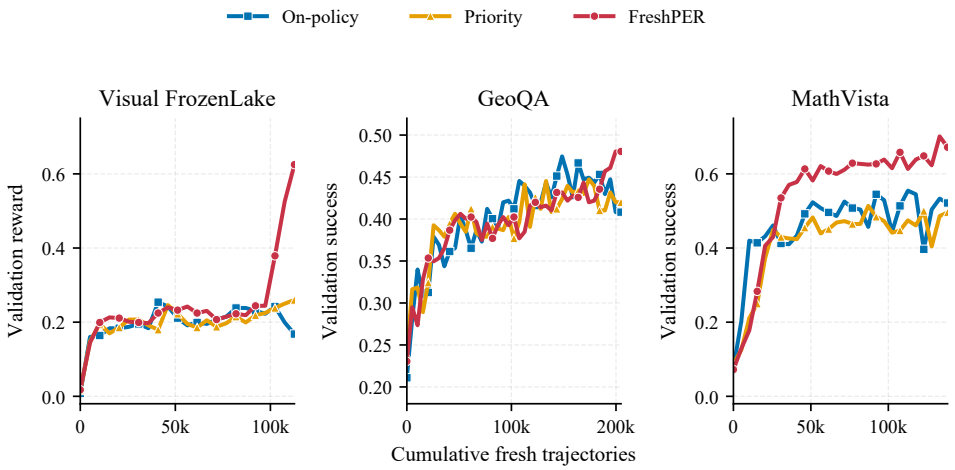


Figure 8: Raw VLM validation curves: reward on visual FrozenLake and success rate on GeoQA and MathVista. The displayed fresh-trajectory budgets reach 112,640, 204,800, and 138,240, respectively.

On MathVista, FreshPER establishes an advantage early in training and maintains it over most of the observed acquisition budget (Fig. 8). Visual FrozenLake shows a later transition: the methods track each other through much of training before FreshPER improves sharply near the end. GeoQA exhibits substantial overlap between methods. These distinct learning curves extend the evaluation to both visual reasoning and navigation using the same reward-based priority signal and decay timescale.

5 Conclusion

FreshPER combines base priorities with exponential collection-age weighting to reuse costly interactions in language model training. The same sampling rule integrates with PPO, GRPO, and the ROLL REINFORCE++ variant. Matched Sokoban experiments demonstrate improved learning from a fixed budget of fresh trajectories, while actor-update comparisons and the NQ Search wall-clock study show gains after accounting for replay's additional optimization. Experiments on ScienceWorld and visual tasks extend this evidence to multi-turn and multimodal settings. Together, these results identify collection age as a useful complement to replay priority as the policy evolves. Freshness-aware selection offers a practical way to extract more learning from acquired experience and reduce the need for repeated data collection in agent training.

Limitations and Future Work. FreshPER currently uses a fixed decay timescale for each task. An important extension is to adapt freshness weights to the observed rate of policy change. Jointly choosing the decay timescale, replay frequency, and rollout budget could further improve the allocation of training time. Future work will examine these choices across larger models and longer-horizon agent tasks.

Reproducibility Statement

Implementation details, experimental protocols, model and environment settings, and replay hyperparameters are provided in Sections A, C.1, E and F.

AI Use Statement

Generative AI tools were used for language editing and manuscript clarity. All technical content, claims, and results were reviewed and verified by the authors.

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. *arXiv preprint arXiv:2402.14740*, 2024.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Jiaqi Chen, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric Xing, and Liang Lin. GeoQA: A geometric question answering benchmark towards multimodal numerical reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 513–523. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.46. URL https://aclanthology.org/2021.findings-acl.46/.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. ReSearch: Learning to reason with search for LLMs via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025. URL https://arxiv.org/abs/2503.19470.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Mehdi Fatemi. Prioritized replay for RL post-training. *arXiv preprint arXiv:2601.02648*, 2026.

William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. *arXiv preprint arXiv:2007.06700*, 2020.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. AReaL: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025.

Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *ICLR*, 2018.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. REINFORCE++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Augustine Kong. A note on importance sampling using standardized weights. Technical Report 348, Department of Statistics, University of Chicago, 1992. URL https://d3qi0qp55mx5f5.cloudfront.net/stat/docs/tech-rpts/tr348.pdf.

Nicolas Le Roux, Marc G. Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, et al. Tapered off-policy REINFORCE: Stable and efficient reinforcement learning for LLMs. *arXiv preprint arXiv:2503.14286*, 2025.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. ReMax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):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*, 2025a.

Zichuan Liu, Jinyu Wang, Lei Song, and Jiang Bian. Sample-efficient LLM optimization with reset replay. *arXiv preprint arXiv:2508.06412*, 2025b.

Han Lu, Zichen Liu, Shaopan Xiong, Yancheng He, Wei Gao, Yanan Wu, Weixun Wang, Jiashun Liu, Yang Li, Haizhou Zhao, et al. Part II: ROLL Flash – accelerating RLVR and agentic training with asynchrony. *arXiv preprint arXiv:2510.11345*, 2025.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023. URL https://arxiv.org/abs/2310.02255.

Jue Ma, Dejun Ning, Chengyi Zhang, and Shipeng Liu. Fresher experience plays a more important role in prioritized experience replay. *Applied Sciences*, 12(23):12489, 2022.

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.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. *arXiv preprint arXiv:2411.02337*, 2024.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2016.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 2023.

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, Y.K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025. URL https://arxiv.org/abs/2503.05592.

Yifan Sun, Jingyan Shen, Yibin Wang, Tianyu Chen, Zhendong Wang, et al. Improving data efficiency for LLM reinforcement fine-tuning through difficulty-targeted online data selection and rollout replay. *arXiv preprint arXiv:2506.05316*, 2025.

Tim van Erven and Peter Harremoës. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. VAGEN: Reinforcing world model reasoning for multi-turn VLM agents. *arXiv preprint arXiv:2510.16907*, 2025a.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, Zichen Liu, Haizhou Zhao, et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025b.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025c.

Yuxiang Wei, Olivier Duchenne, Jade Copet, et al. SWE-RL: Advancing LLM reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025a.

Zhepei Wei, Wenlin Yao, Yao Liu, et al. WebAgent-R1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *ICLR*, 2023. URL https://arxiv.org/abs/2210.03629.

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. URL https://arxiv.org/abs/2606.04560.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Yuexiang Zhai, Hao Bai, Zipeng Lin, Jiayi Pan, Shengbang Tong, Yifei Zhou, Alane Suhr, Saining Xie, Yann LeCun, Yi Ma, and Sergey Levine. Fine-tuning large vision-language models as decision-making agents via reinforcement learning. *arXiv preprint arXiv:2405.10292*, 2024.

Hongzhi Zhang, Jia Fu, Jingyuan Zhang, et al. RLEP: Reinforcement learning with experience replay for LLM reasoning. *arXiv preprint arXiv:2507.07451*, 2025.

Zijing Zhang and Xiajie Yang. CRPS: Curriculum replay via progressive suffixes from successful trajectories for long-horizon LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13891–13904. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.680. URL https://aclanthology.org/2026.findings-acl.680/.

A System Design Details

A.1 End-to-End Training Procedure

Algorithm 1 specifies the fresh-plus-replay schedule. Behavior log-probabilities are captured during rollout or by scoring under the unchanged behavior policy before fresh optimization. The fresh batch undergoes the configured optimization phase before buffer insertion. Each outer rollout iteration then processes at most K replay batches, subject to warmup and sampling eligibility. Neither fresh nor replay optimization is restricted to a single optimizer step. Age is counted in outer rollout iterations.

Algorithm 1 FreshPER Training

Require: Policy π_θ, environment $\mathcal{E}$, buffer $\mathcal{B}_0$, decay timescale τ, maximum replay batches K

for rollout iteration $t = 1, 2, \ldots$ **do**
  Collect fresh batch $\mathcal{F}_t$ with behavior policy π_{μ_t}
  Record behavior log-probabilities before changing π_{μ_t}
  Optimize π_θ on $\mathcal{F}_t$ for the configured epochs and minibatches
  Form replay units from $\mathcal{F}_t$, preserving complete groups when required
  Set $\mathcal{B}_t \leftarrow \mathcal{B}_{t-1}$; insert admitted units with $c_i = t$ and base priorities
  Apply FIFO eviction if capacity is exceeded
  for $k = 1, \ldots, K$ **do**
    Complete refresh: $\bar{p}_i(t) = p_i^{\text{base}}(t) \exp[-(t - c_i)/\tau]$ for all $i \in \mathcal{B}_t$
    Form eligible set $\mathcal{I}_t$ under the configured age and signal filters
    if warmup is incomplete or too few eligible units are available **then**
      break
    end if
    Sample $\mathcal{S}_k$ from $\mathcal{I}_t$ using priorities $\bar{p}_i(t)^\alpha$
    Compute PER weights $u_i(t)$ from Eq. (2); use $\beta = 0$ if disabled
    Optimize on $\mathcal{S}_k$ with stored behavior log-probabilities and weights $u_i(t)$
    Write back base priorities if configured, without changing c_i
  end for
  Synchronize the behavior policy: $\pi_{\mu_{t+1}} \leftarrow \pi_\theta$
end for

A.2 Replay Buffer Architecture

The buffer stores *replay units*: complete trajectories or complete prompt groups, according to the algorithm configuration. Each trajectory retains its prompt, assistant turns, observations, reward, and behavior log-probabilities. Unit-level metadata includes collection iteration c_i, current base priority $p_i^{\text{base}}(t)$, and effective priority $\bar{p}_i(t)$. A group retains all member trajectories and their shared identity; its age is measured from their collection, not a later priority write-back. Capacities and batch sizes are reported with their units in Section F.

The experiments use FIFO eviction at capacity. The CPU-side freshness refresh covers all stored units before the next replay sample; an asynchronous refresh must finish before sampling proceeds. Warmup and minimum-age constraints determine whether replay can run. Probabilities and PER weights are normalized over the eligible population $\mathcal{I}_t$, with $N_t = |\mathcal{I}_t|$, rather than over ineligible buffer entries. Current-batch units have age zero and are excluded when the minimum sample age is one.

Group replay samples complete groups sequentially without replacement, renormalizing the remaining priority mass after each draw. The $P_t(i)$ used in Eq. (2) is the initial single-draw probability, not the group's batch-inclusion probability. Thus, these PER weights provide an approximate sampling correction; they do not yield an unbiased without-replacement estimator, even at $\beta = 1$. Values $\beta < 1$ apply partial correction. Each group shares its weight across its trajectories, with loss reduction determined by the underlying objective. Eligibility scans and whole-buffer freshness refresh mean that total sampling cost is not generally $O(\log N_t)$.

A.3 Framework Integration

We implement FreshPER on top of the ROLL framework (Wang et al., 2025b), which provides a single-controller architecture that decouples inference and training across GPU pools. Buffer storage and priority operations run on the CPU side. As in Algorithm 1, fresh optimization precedes buffer insertion, followed by at most K replay-batch optimization phases per outer iteration. The actual replay count depends on warmup, eligible population size, and the configured admission filters.

B Ablation and Applicability Details

B.1 Freshness Timescale

The age-decay comparison in Fig. 6 varies τ within reward-priority FreshPER on Sokoban Simple and text FrozenLake. These ablation runs use Qwen2.5-0.5B-Instruct and the ROLL REINFORCE++ variant, with 128 fresh trajectories per outer iteration and at most two replay batches. They are distinct from the eight-GPU Sokoban cross-algorithm study. On-policy is a reference, not a no-freshness replay ablation; a Priority comparison is equivalent to $\tau \to \infty$ only when the remaining settings are held fixed.

B.2 Importance-Sampling Correction

The IS comparison in Fig. 7(a) keeps $\tau = 500$ and reward-based priority fixed on text FrozenLake. The enabled configuration uses $\beta = 0.4$; the other disables the PER sampling correction. This toggle concerns non-uniform replay sampling, not removal of the behavior-policy ratio.

B.3 Boundary Cases

In Fig. 7(b,c), all three methods eventually reach score zero on CliffWalking, while replay introduces transient fluctuations. On GSM8K, the recorded evaluation starts near saturation, and the methods achieve similar accuracy. The high initial accuracy reflects the interaction and answer-attempt protocol used in these experiments, rather than standard single-attempt evaluation of the base model.

Table 1: Applicability boundaries retained alongside the positive comparisons. These observations are specific to the evaluated protocols.

Setting	Observation and scope
CliffWalking	All three methods reach score zero; replay does not improve the final score.
GSM8K	Curves occupy a narrow, high-accuracy range; FreshPER does not consistently improve on On-policy.
ScienceWorld, plain REINFORCE++	FreshPER does not outperform the two controls in the recorded configuration. This result is separate from the GRPO comparison.

C Additional Evaluation Details

C.1 Budget Accounting and Curve Conventions

The fresh-rollout axis is B_{fresh} defined in Section 4. Sokoban and ScienceWorld use cumulative trajectory counters. NQ and VLM curves use logged outer iteration multiplied by the associated fresh batch, with experiment-specific mappings documented below. Evaluation coordinates refer to the evaluated policy: when validation precedes the next rollout, that forthcoming batch is not counted. RLEP's preparatory training and pool collection count toward its acquisition budget even though they precede main training. Equal fresh-trajectory counts do not imply equal tokens, environment calls, optimizer updates, wall-clock time, or GPU-hours.

In the Sokoban cross-algorithm study, REINFORCE++ and PPO collect 128 fresh trajectories per iteration for 250 iterations (32,000 total); GRPO collects 144 for 222 iterations (31,968 total). VLM configurations collect 512 per iteration. Per-stage ScienceWorld counts and the distinction between

completed training and the last logged evaluation are provided in Section F.3. Training rollout success rate, shaped return, validation success rate, NQ exact match, and ScienceWorld Dev score are separate metrics, not interchangeable measures or inputs to a cross-task average.

Figure 5a counts actor optimizer updates completed before validation, using the preceding metrics row. Fresh and replay updates both count; critic updates and total GPU cost do not. The nine Sokoban runs are the same as in Fig. 3. Within the shared display ranges, replay ends at 238 updates for REINFORCE++/PPO and 213 for GRPO, while On-policy ends at 240 and 220, respectively. The three methods share an exact evaluation coordinate only at initialization. Curves use trailing means over up to three recorded evaluations, with partial initial windows and no inserted or extrapolated observations. These windows span different update intervals across methods.

For Fig. 5b, elapsed time is the recorded runtime minus each run's initial step-zero evaluation time. The main On-policy, Priority, and FreshPER comparison uses the REINFORCE++ configurations in Section F.2, with Qwen2.5-7B-Instruct, ROLL, and eight NVIDIA H200 GPUs. PPO and GRPO are contextual references with the model and framework settings in Section E. We compare scores within the runs' observed time overlap. Curves use trailing means over up to three evaluations without resampling. For visibility, the current plot offsets the On-policy and PPO curves by -1 and $+1$ percentage points when their absolute smoothed scores are at most 0.5%; GRPO is unshifted.

The result figures use blue squares for On-policy, amber upward triangles for Priority, and red circles for FreshPER; RAPER and RLEP use green diamonds and pink downward triangles. Algorithm-specific panels use solid lines. In Fig. 4(a), solid, dashed, and dotted lines identify REINFORCE++, GRPO, and PPO, respectively, without changing method colors or markers. The NQ wall-clock overlay instead uses solid lines throughout, with blue diamonds for GRPO and blue downward triangles for PPO. Timescale ablations separate configurations into columns and tasks into rows. The IS ablation explicitly distinguishes no IS (dark-gray hollow circles) from IS enabled (red filled circles). Captions identify smoothing where applicable. None of these displays is an independent-seed confidence interval.

C.2 Learning-Curve Details

Figure 9 provides the full training-step view of the raw Sokoban validation points together with NQ Search, visual FrozenLake, and MathVista. A training step denotes an outer rollout iteration, not an optimizer update; replay can perform additional updates within an iteration.

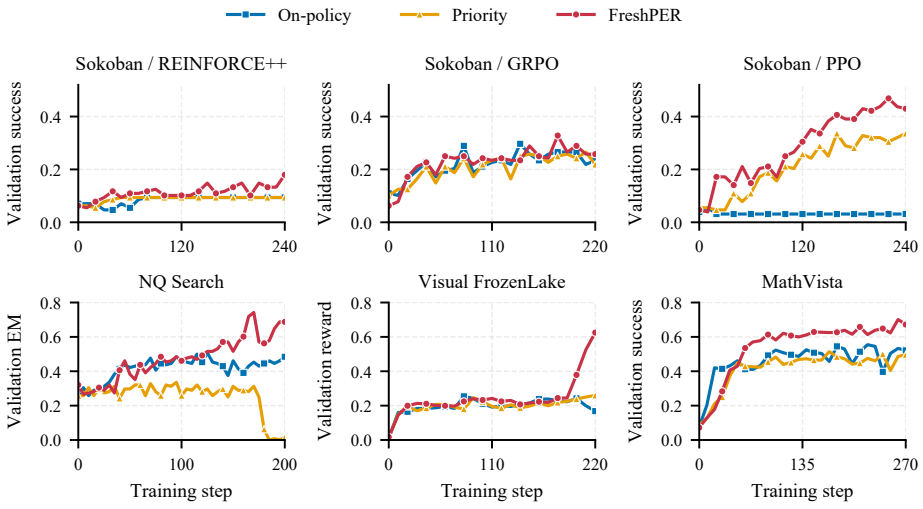


Figure 9: Raw validation versus training step (outer rollout iteration). Top: the nine Sokoban runs summarized in Fig. 3. Bottom: NQ Search (REINFORCE++), visual FrozenLake, and MathVista. Each panel uses a shared recorded interval; no smoothing, interpolation, or run averaging. Equal steps do not imply equal optimizer updates or compute (Section F).

Fixed-Budget Sokoban Validation. At outer iteration 240, corresponding to 30,720 cumulative fresh trajectories, PPO validation success rate is 43.0% for FreshPER, 33.6% for Priority, and 3.1% for On-policy. At the same fresh-trajectory budget, the ROLL REINFORCE++ variant reaches 18.0%, 9.4%, and 9.4%, respectively. These are the last shared validation points within each of these two algorithm-specific comparisons in Fig. 9, rounded to one decimal place, without smoothing, peak selection, or cross-run averaging. Matching fresh trajectories does not match optimizer updates.

For ScienceWorld (Fig. 4(b)), RLEP's last three evaluations occur at 2,800/2,960/3,120 fresh trajectories; the other methods' occur at 2,720/2,880/3,040. No additional evaluation at the final 3,200-trajectory checkpoint is implied. That panel does not connect RLEP's preparatory policy to its reset main-training policy or fill the pool-building interval.

Figure 4(a) shows the three NQ REINFORCE++ replay-study runs through step 200 together with additional on-policy PPO and GRPO references, displaying 0–25,600 fresh trajectories. Steps are multiplied by 128 for REINFORCE++ and PPO, and by 640 for GRPO; the visible GRPO interval ends at step 40. No measured values are smoothed or interpolated. The PPO and GRPO curves are contextual no-replay references rather than matched-model algorithm comparisons. VLM curves use 512 fresh trajectories per logged iteration.

D ESS Motivation and the Age Approximation

Let P and Q denote trajectory distributions under the current and collection policies, respectively. Assume $P \ll Q$ and that the importance ratio $\rho = dP/dQ$ has finite second moment under Q. Since $\mathbb{E}_Q[\rho] = 1$, the population ESS for n importance-weighted samples (Kong, 1992) satisfies

$$\frac{\mathrm{ESS}_{\mathrm{pop}}}{n} = \frac{1}{\mathbb{E}_Q[\rho^2]} = \exp[-D_2(P\|Q)] \le \exp[-D_{\mathrm{KL}}(P\|Q)]\,, \tag{3}$$

where $D_2(P\|Q) = \log \mathbb{E}_Q[\rho^2]$ is the order-two Rényi divergence. The inequality follows from the monotonicity of Rényi divergence in its order (van Erven & Harremoës, 2014). This relation concerns full trajectory importance ratios; it does not characterize the clipped surrogate used for optimization.

If policy divergence grows approximately linearly with collection age over a local interval, the right-hand side has an exponential dependence on age. We use this observation to motivate the age factor in Eq. (1), without estimating divergence for each historical policy. After priority exponentiation, its effective decay rate is α/τ. The relationship between age and policy drift depends on the learning rate, update schedule, and task, so we choose τ empirically.

E Environment and Model Details

We use Qwen2.5-0.5B-Instruct (Qwen et al., 2024) for Sokoban, the text-environment ablations, and boundary cases; Qwen2.5-7B-Instruct for the NQ Search (Jin et al., 2025) replay study; and Qwen3-8B (Yang et al., 2025) for ScienceWorld (Wang et al., 2022). Visual FrozenLake, GeoQA (Chen et al., 2021), and MathVista (Lu et al., 2023) use Qwen2.5-VL-3B-Instruct (Bai et al., 2025). CliffWalking and GSM8K (Cobbe et al., 2021) provide the boundary cases in Fig. 7(b,c). All replay studies use ROLL (Wang et al., 2025b); the additional NQ on-policy references use the model/framework identities described below.

Table 2: Task and model settings for the main figures and ablations. The training horizons in Table 3 distinguish configured limits from displayed evaluation ranges.

Task	Model	Sequence limit	Max turns
Sokoban cross-algorithm	Qwen2.5-0.5B-Instruct	2,048	10
NQ Search replay study	Qwen2.5-7B-Instruct	12,800	5
ScienceWorld	Qwen3-8B	8,192	30
Visual FrozenLake	Qwen2.5-VL-3B-Instruct	4,096	10
GeoQA	Qwen2.5-VL-3B-Instruct	4,096	3
MathVista	Qwen2.5-VL-3B-Instruct	4,096	3
Text FrozenLake / Sokoban ablations	Qwen2.5-0.5B-Instruct	2,048	10
CliffWalking control	Qwen2.5-0.5B-Instruct	2,048	200
GSM8K control	Qwen2.5-0.5B-Instruct	4,096	3

Sokoban and Navigation Tasks. The cross-algorithm study uses procedurally generated Simple-Sokoban boards (6×6, one box), a ten-action limit, and at most 128 generated tokens per action. Validation uses 128 fixed-seed boards every ten outer iterations, with stochastic decoding at temperature 0.99. Training shaped return and the fraction of solved boards are separate metrics. The Sokoban and text-FrozenLake ablations use the two-GPU configuration described below. Visual FrozenLake replaces the textual grid observation with an RGB rendering. CliffWalking uses its navigation-return metric and illustrates a setting where FreshPER offers no consistent advantage.

NQ Search. All NQ Search runs use eight NVIDIA H200 GPUs. The replay-study agent alternates reasoning, search queries, and a final answer. Search returns three passages per query from the E5/FAISS retrieval system, and validation measures normalized answer exact match. Figure 4(a) includes the Qwen2.5-7B-Instruct replay study together with additional no-replay PPO/Llama3.2-3B and GRPO/Qwen2.5-7B-Base references using verl. These references provide context only and are not used as matched-model comparisons of RL algorithms. Wall-clock accounting for Fig. 5b is documented in Section C.1.

ScienceWorld. Training covers all 30 task types with native train variations, no task simplifications, no valid-action list, and no thinking-mode output. An episode permits 30 commands, each with a 64-token generation limit. Training decoding uses temperature 0.7, top-$p = 0.8$, and top-$k = 20$. Dev uses 60 fixed task/variation pairs (two per task type) with a greedy generation override, evaluated every ten main-training iterations. Environment score increments are scaled by 0.01 for training; the figure instead uses native final Dev score. Full-task success is recorded separately. Shared task definitions and decoding do not guarantee identical observed Dev values across repeated executions.

Visual Reasoning and GSM8K. GeoQA and MathVista receive diagrams or other images alongside the question, with correctness feedback under the configured interaction limits. The plotted visual-FrozenLake metric is validation reward; GeoQA and MathVista report validation success rates. These task-specific metrics are not averaged across tasks. GSM8K retains the historical answer-attempt protocol and is not presented as a standard single-attempt benchmark result.

F Detailed Experimental Configurations

The configurations below specify the settings used in each experiment; replay hyperparameters are task-specific rather than universal.

Table 3: Training horizons and acquisition budgets. Fresh batch counts trajectories, not prompts. VLM horizons are associated configuration limits, not a claim that every repeat completed 400 iterations.

Study	GPUs	Fresh batch	Outer iterations	Displayed range
Sokoban REINFORCE++	8	128	250	Full recorded curves
Sokoban GRPO	8	144	222	Full recorded curves
Sokoban PPO	8	128	250	Full recorded curves
NQ replay study	8 H200	128	300 (configured)	Through step 200
ScienceWorld, non-RLEP	8	16	200	Dev steps 0–190
ScienceWorld, RLEP	8	16	50 + 75 + 75	Main Dev 0–70
Visual FrozenLake	8 H100	512	400 (configured)	Through step 220
GeoQA	8 H100	512	400 (configured)	Through step 400
MathVista	8 H100	512	400 (configured)	Through step 270
Text ablations	2 A100	128	400 (configured)	Recorded evaluations

F.1 Sokoban Cross-Algorithm Study

All plotted runs use actor learning rate 10^{-6}, cosine scheduling with ten warmup iterations, one fresh-update epoch, BF16, DeepSpeed ZeRO-2, and vLLM. Decoding uses temperature 0.99, top-$p = 0.99$, and top-$k = 100$. The ROLL REINFORCE++ variant first applies mean/std normalization to rewards within two-trajectory groups, computes reward-to-go, then whitens advantages across the full batch (`whiten_advantages=true`) before applying advantage clipping at 0.2. Group-level reward normalization and batch-wide advantage normalization are separate stages. This differs from plain REINFORCE++; it also retains local reward-standard-deviation scaling, unlike the published REINFORCE++ with-baseline variant (Hu et al., 2025). GRPO uses eight-trajectory groups, group-relative advantages, policy-ratio clipping at 0.2, and no additional advantage clip. PPO uses GAE with $\gamma = 1$ and $\lambda = 0.95$, policy/value clips of 0.2/0.5, advantage clip 2.0, and critic learning rate 10^{-5}. These configurations do not use a direct actor KL loss.

Table 4: Sokoban replay settings. Capacity and warmup are in complete groups of size G. Priority and FreshPER share the declared base priority within each algorithm; only FreshPER enables age decay.

Algorithm	G	Capacity	Warmup	Max replay batches	Base priority
REINFORCE++	2	50,000	64	2	Reward-based
GRPO	8	50,000	64	1	Group reward std
PPO	1	50,000	128	2	Reward-based

For all three algorithms, the replay configurations use FIFO eviction, $\alpha = 0.6$, and minimum sample age one. FreshPER uses $\tau = 500$; PER IS correction is disabled. GRPO accepts replay groups only when terminal-reward standard deviation exceeds 10^{-6}. Equal capacities in groups correspond to different numbers of trajectories across algorithms, so buffer capacity is not matched in trajectories. Each replay batch contains as many trajectories as a fresh batch. The plotted REINFORCE++/PPO replay runs process 498 replay batches (63,744 historical trajectories); GRPO replay runs process 215 (30,960 historical trajectories).

REINFORCE++ uses four actor/reference and four inference GPUs. GRPO uses six actor/reference and two inference GPUs, with the same effective fresh batch across training strategies. PPO uses four actor, two critic, and two inference GPUs. Replay PPO recomputes value/GAE using the current critic, so Priority and FreshPER share that PPO-replay variant. We do not compare wall-clock efficiency across these different GPU layouts.

F.2 NQ, VLM, AND ABLATION SETTINGS

Table 5: Replay configurations for NQ, VLM, and text-task ablations. Capacity is in trajectories; K is the maximum number of replay batches per outer rollout iteration. A dash indicates a disabled component.

Task	Method	Capacity	Base priority	K	τ
NQ Search	On-policy	–	–	0	–
	Priority	50,000	Advantage-based	2	–
	FreshPER	10,000	Reward	2	40
Visual FrozenLake	Priority / FreshPER	50,000	Reward	2	– / 1,000
GeoQA / MathVista	Priority / FreshPER	100,000	Reward	2	– / 1,000
Text ablations	FreshPER	50,000	Reward	2	500/1,000/1,500

The NQ settings in Table 5 describe the replay study in Fig. 4(a). They use four responses per prompt, $\alpha = 0.6$, and PER IS correction with $\beta = 0.4$ for replay. Priority uses a constant 10^{-6} learning rate; On-policy and FreshPER use a cosine schedule. This experiment evaluates the complete configurations listed here; the matched Sokoban experiments provide the component-level isolation of age decay.

The VLM configurations use the ROLL REINFORCE++ variant with group mean/std reward normalization followed by batch-wide advantage whitening, four responses per prompt, trajectory-level replay, actor learning rate 10^{-6}, cosine scheduling, and $\alpha = 0.6$. Replay uses reward-based priority, $\beta = 0.4$ IS correction, and the capacities in Table 5; FreshPER uses $\tau = 1000$. These configurations use eight H100 GPUs and up to 400 iterations. The shorter visual-FrozenLake and MathVista ranges reflect the common plotted interval, not different configured training horizons.

The text-task ablations use the same ROLL REINFORCE++ normalization variant with two responses per prompt, 128 fresh trajectories per iteration, $\alpha = 0.6$, and reward-based priority. The τ comparisons disable PER IS; the separate FrozenLake IS comparison fixes $\tau = 500$ and enables $\beta = 0.4$ in one configuration. These ablations use advantage clipping at 0.2 and disable direct KL and entropy losses.

F.3 SCIENCEWORLD FIVE-METHOD COMPARISON

All methods in Fig. 4(b) use the same ScienceWorld environment and Dev protocol, while each follows its method-specific optimization objective. Table 6 records these settings. All use actor learning rate 10^{-6} and the same 8,192-token sequence limit. The FreshPER configuration uses a constant schedule after warmup, fixed PER $\beta = 0.4$, and $\tau = 10$; Priority uses cosine scheduling and linearly increases β from 0.4 to 1.0.

Table 6: ScienceWorld optimization settings for the plotted runs. Warmup is the configured scheduler warmup count. Direct KL loss is distinct from reward-level KL.

Method	Fresh G	Loss aggregation	Schedule	Warmup	Actor KL
On-policy	8	Sequence/token mean	Cosine	20	0.04
Priority	8	Sequence/token mean	Cosine	20	0.04
FreshPER	8	Sequence/token mean	Constant	20	0.04
RAPER	8	Token mean	Constant	0	0
RLEP	16	Token mean	Constant	10	0

On-policy, Priority, and FreshPER use symmetric policy clipping at 0.2, an enabled reference model and direct actor KL loss, no token-level reward KL, and initial reward-KL coefficient zero. RAPER/RLEP use DAPO-style objectives with policy clips 0.2/0.28, dual clip 10, no reference model or KL loss, and weight decay 0.01/0.1, respectively; they are not labeled as complete DAPO implementations.

Priority and FreshPER store complete eight-trajectory groups, with capacity 64 groups (512 trajectories), warmup eight groups (64 trajectories), minimum sample age one, $\alpha = 0.6$, and at most one additional 16-trajectory replay batch. Their base priority is group terminal-reward standard deviation; replay groups with standard deviation at or below 10^{-6} are filtered. RAPER instead samples individual historical rollouts with birth-advantage priority $|A| + 10^{-6}$, $\alpha = 0.5$, a 30,000-trajectory capacity, 20-iteration warmup, and maximum age ten. Its replay-to-fresh ratio is 0.5 and optimization jointly uses fresh and sampled historical data, without an additional PER-IS loss weight.

RLEP uniformly samples up to two verified successes for the same task/variation as a 16-trajectory fresh group. It first trains a preparatory policy on 800 fresh trajectories, collects 1,200 additional trajectories to build the success pool, then resets the model and trains on another 1,200 fresh trajectories. When no successful trajectories matching the task and variation are available, RLEP uses only the fresh group. RLEP replays 12 trajectories across six of 75 scheduled main-training groups. Logical sample counts exclude zero-loss padding used for device divisibility.

The plotted On-policy run records 200 optimizer updates, compared with 394 for Priority and 391 for FreshPER. RAPER records 244, and RLEP records 125 including preparatory training. The historical-trajectory counts are 3,104 for Priority, 3,056 for FreshPER, 700 for RAPER, and 12 for RLEP. These counters describe method-dependent optimization cost, not compute matching.