When Does Non-Uniform Replay Matter in Reinforcement Learning?

Michal Korniak[† 1,2] **Mikołaj Czarnecki**[† 2] **Yarden As**[1]
Piotr Miłoś[2] **Pieter Abbeel**[3,4] **Michal Nauman**[2,3]

[1]ETH Zurich [2]University of Warsaw [3]UC Berkeley [4]Amazon FAR

[†]Equal contribution

Abstract

Modern off-policy reinforcement learning algorithms often rely on simple uniform replay sampling and it remains unclear *when and why* non-uniform replay improves over this strong baseline. Across diverse RL settings, we show that the effectiveness of non-uniform replay is governed by three factors: *replay volume*, the number of replayed transitions per environment step; *expected recency*, how recent sampled transitions are; and the *entropy* of the replay sampling distribution. Our main contribution is clarifying when non-uniform replay is beneficial and providing practical guidance for replay design in modern off-policy RL. Namely, we find that non-uniform replay is most beneficial when replay volume is low, and that high-entropy sampling is important even at comparable expected recency. Motivated by these findings, we adopt a simple Truncated Geometric replay that biases sampling toward recent experience while preserving high entropy and incurring negligible computational overhead. Across large-scale parallel simulation, single-task, and multi-task settings, including four modern and classic algorithms evaluated on six RL benchmark suites, this replay sampling strategy improves sample efficiency in low-volume regimes while remaining competitive when replay volume is high.

1 Introduction

Experience replay is one of the central components of off-policy RL [21, 32]. Many replay sampling strategies have been proposed, from uniform replay to distributions biased by error [28] or recency [36]. Yet recent continuous-control algorithms still often use uniform replay by default [5, 22, 18, 24]. Understanding when this choice is sufficient, and when it can be improved, is especially important in regimes such as parallel simulation, multi-task learning, and other settings where data collection outpaces optimization. In these regimes, individual transitions may be replayed only a few times, so the replay sampling distribution can have a larger effect on learning.

A natural way to analyze replay is to count how often each transition is reused over the course of training. This whole-training exposure view underlies prior analyses of recency-biased replay [36, 37]: changing the sampler changes how replay updates are distributed across transitions over their lifetime. However, whole-training exposure conflates two distinct factors. A learner can consume more recent data either by shifting the sampling distribution toward newer transitions, or by increasing the number of replayed transitions per environment step through a larger update-to-data ratio (UTD) or batch size. Moreover, whole-training exposure does not fully explain empirical behavior observed in off-policy RL. In runs where replay capacity is comparable to the training horizon, exposure can

Corresponding author: michael.korniak@gmail.com
Code available at https://github.com/M-Korniak/when-does-replay-matter
Project page at https://m-korniak.github.io/when-does-replay-matter

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

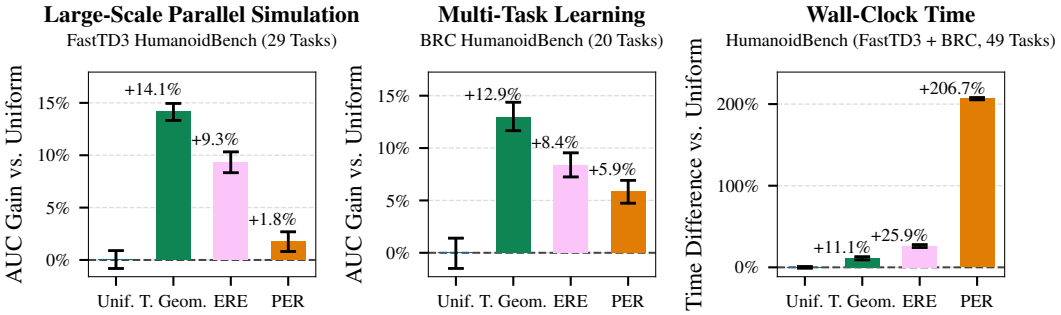

Figure 1: ***Performance and runtime trade-offs on HumanoidBench.*** Relative sample efficiency gains measured as area under (learning) curve (AUC) for large-scale parallel simulation FastTD3 (*left*) and multi-task BRC (*middle*), and wall-clock time difference aggregated across both settings (*right*) compared to uniform sampling. Error bars show 95% stratified bootstrap CI. We report results in large-scale parallel FastTD3 and multi-task BRC because they feature relatively low experience reuse per transition, where non-uniform replay is expected to have the greatest impact (Subsection 3.1). The recency-biased truncated geometric (described in detail in Section 4) sampling yields consistent gains with minimal overhead, while PER exhibits higher computational load. Experiment details in Section 5. We refer to Section E.1 for discussion on AUC as a suitable metric for *sample efficiency*.

be highly skewed, yet non-uniform replay often provides little benefit [22, 18]. Conversely, as we show in this paper, in high-throughput simulation, exposure can be nearly flat throughout training, yet recency-biased replay can substantially outperform uniform sampling. These observations suggest that whole-training exposure alone does not predict when non-uniform replay improves performance.

In this paper, we expand on this perspective by showing that replay performance also depends on factors that whole-training exposure does not account for. This view separates three factors: **expected recency** captures how recent sampled transitions are on average; **replay volume**, given by UTD times batch size, captures how many replayed transitions are consumed between consecutive environment steps; and **sampling entropy** captures how concentrated the replay sampling distribution is at each update. This decomposition separates changing the sampler from changing the amount of optimization, and distinguishes recency bias from the diversity of replayed samples.

Our empirical study reveals two main findings about replay in RL. First, the benefit of recency bias is strongly mediated by replay volume: when replay volume is small, biasing replay toward recent transitions yields substantial gains [37]. Conversely, when replay volume is large, the same bias offers limited benefit. Second, expected recency alone does not determine performance: smoothly biased, high-entropy samplers outperform strategies that obtain comparable expected recency by restricting the support of the replay sampling distribution. Together, these findings explain why non-uniform replay is especially effective in optimization-limited regimes such as multi-task learning and large-scale parallel simulation, while uniform replay remains competitive when data reuse is already high.

Motivated by these observations, we adopt a simple recency-biased replay sampler based on a truncated geometric distribution. The sampler directly targets the regime identified by our analysis: it increases expected recency smoothly while preserving high sampling entropy, requires no costly priority recomputation, and admits constant-time sampling. Across our experiments, it improves sample efficiency over uniform, priority-biased, and recency-biased replay in low-replay volume regimes (Figure 1) while remaining effectively neutral when replay volume is high. This makes it a practical replacement for uniform replay in off-policy RL, especially when optimization is limited relative to data collection. We list our contributions below:

- We provide a practical guide to replay sampling in modern off-policy RL through a large-scale empirical study across single-task, multi-task, and large-scale parallel training regimes.
- We propose a per-step view of replay that separates expected recency, replay volume, and sampling entropy. This decomposition shows that recency-bias matters most when replay volume is low, while high sampling entropy remains important even at comparable expected recency.
- We show that a simple truncated geometric performs competitively across diverse benchmarks, improving sample efficiency over uniform and priority-based replay in low-replay volume regimes with negligible computational overhead, while remaining neutral in high-replay volume regimes.

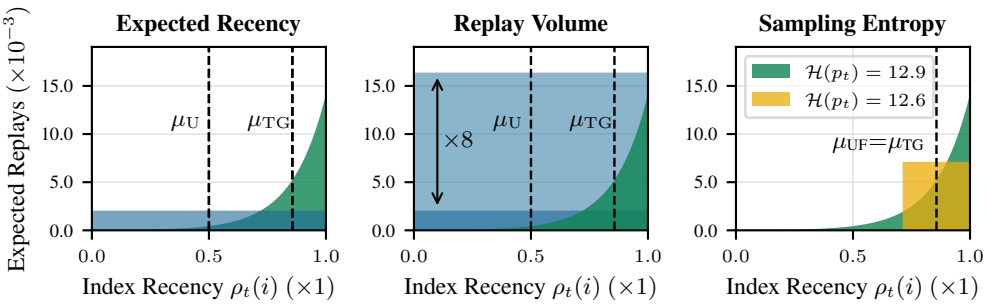

Figure 2: ***Effect of expected recency, replay volume, and sampling entropy.*** (*left*) By biasing samples toward recent data, the Truncated Geometric sampler (Section 4) achieves substantially higher expected recency than uniform replay. (*middle*) Increasing replay volume through UTD or batch size can make uniform replay match or exceed recency-biased sampling in the number of updates applied to recent transitions. (*right*) Replay distributions can have the same expected recency but different sampling entropy. The Truncated Geometric sampler achieves recency through a smooth full-support bias, increasing sampling entropy relative to a uniform buffer with the same expected recency; this is comparable to increasing the uniform sampling buffer size by roughly one third.

2 Background

Off-Policy RL. We consider infinite-horizon MDPs, where an agent interacts with an environment by selecting actions according to a policy and receives scalar rewards [26, 34]. We focus on off-policy deep RL methods, which improve sample efficiency by reusing data collected under previous policies [21, 10, 12]. By decoupling data collection from optimization, off-policy algorithms allow transitions to be leveraged across many gradient updates and have enabled applications including robotic control and real-world online learning during deployment [33, 29, 2]. The replay buffer enables data reuse and stabilizes learning by inducing a more slowly changing, approximately stationary training distribution [34, 21].

Replay Sampling Distributions. Many RL methods rely on first-in-first-out (FIFO) uniform replay, which is simple and effective across a wide range of tasks [19, 13, 30]. A substantial body of prior work has explored *non-uniform replay sampling*, in which transitions are sampled with unequal probability [15, 3, 11]. Representative approaches include TD-error-biased Prioritized Experience Replay [PER, 28] or Reliability-Adjusted Prioritized Experience Replay [ReaPER, 25], which bias sampling based on temporal-difference error, and recency-biased Emphasizing Recent Experience [ERE, 36], which biases sampling toward recent transitions. While non-uniform replay has shown improvements in some settings, it often introduces additional complexity, and it remains unclear in which regimes it consistently improves over uniform replay [22].

Recency-Biased Sampling. A key motivation for recency-biased replay is that newer transitions are more related to the current policy and as such may contain more informative signal, particularly when the agent adapts to new regions of the state space [37]. Prior work analyzes this through the lens of whole-training exposure: the expected number of times a transition is sampled while it remains eligible for replay. Crucially, when the buffer size is larger than the total number of steps the agent collects over its lifetime, older transitions are sampled disproportionately more often than newer ones. ERE addresses this by skewing back the sampling distribution towards recent transitions [36, 37]. While the exposure framing intuitively motivates recency-biased methods, we argue it can obscure factors that matter during optimization: two sampling strategies can share identical exposure yet differ greatly in performance, or exposure can be skewed while the choice of strategy barely matters. In contrast to prior work, we analyze replay per environment step, separating expected recency, replay volume, and sampling entropy.

3 What Matters in Non-Uniform Sampling?

Prior work has shown that increasing replay recency can improve learning in some settings [36, 37]. Our goal is therefore not to re-establish the importance of recency, but to identify the factors that determine *when* recency improves over other replay sampling distributions.

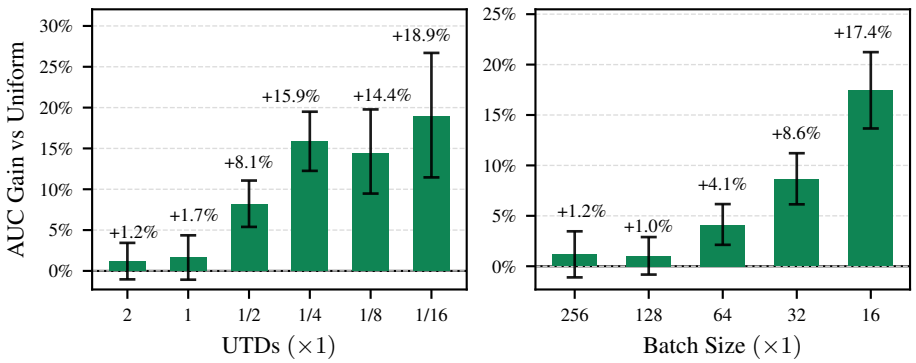

Figure 3: ***Replay volume matters.*** Improvement of recency-biased sampling over uniform replay as replay volume is varied through UTD (*left*) and batch size (*right*) (Section 3.1). In both panels, reducing replay volume increases the advantage of recency-biased replay: when replay volume is high, both methods perform similarly, whereas when replay volume is low, recency-biased sampling yields substantially larger gains. These results show that the benefit of recency bias depends on how much replay is performed per environment step. As such, the sampling distribution is especially important in optimization-limited regimes, such as multi-task learning or large-scale parallel simulation. PER follows a similar trend (Appendix F.1).

Let p_t denote the replay sampling distribution at environment step t over buffer indices $i = 0, \ldots, N$, where $i = 0$ denotes the oldest sample and N is the total size of the buffer. Let N_t be the number of transitions collected up to time t. We define the normalized recency of index i as $\rho_t(i) = i/(N_t - 1)$, where larger values correspond to more recent transitions. We define the **expected recency** of the replay sampling distribution p_t as $\mu_t = \mathbb{E}_{i \sim p_t}[\rho_t(i)]$. This quantity captures how recent the sampled transitions are on average. Higher expected recency means that the learner updates more often on the recent experience. Unlike whole-training exposure, which aggregates counts over a transition's lifetime, expected recency describes the "age" composition of the update at a single training step.

To determine the aggressiveness of optimization at a given timestep, we define **replay volume** as $\mathrm{UTD}_t \times B_t$, that is, the number of replayed transitions between two consecutive environment steps. Increasing replay volume increases the amount of optimization performed on the stored data without changing the sampling distribution itself. Thus, replay can change along two separable axes: expected recency determines *where* replay samples come from, while replay volume determines *how many times* a given transition is consumed. Importantly, when replay volume is high, even low-expected recency samplers can still update on many recent transitions; when replay volume is low, the sampling distribution plays a larger role in determining whether recent transitions are replayed at all.

Furthermore, expected recency does not fully characterize the shape of the replay sampling distribution. Two strategies can have the same expected recency while differing substantially in how concentrated their sampling probabilities are. We capture this concentration using the instantaneous **sampling entropy** $\mathcal{H}(p_t)$. As illustrated in the right panel of Figure 2, a sampler can obtain high expected recency by sampling only from a recent window, yielding low entropy, or by smoothly biasing probabilities over the full buffer, yielding higher entropy. These strategies may have similar expected recency, but differ in update diversity and optimization dynamics.

This per-step decomposition guides the rest of our analysis. In Section 3.1, we vary replay volume while keeping the replay sampling distributions fixed, showing that recency bias helps most when replay volume is low and has limited effect when replay volume is high. In Section 3.2, we compare replay strategies that bias sampling toward recent data but differ in sampling entropy, showing that recency alone is not sufficient: high-entropy, full-support sampling is also important. Together, these experiments demonstrate that separating expected recency, replay volume, and sampling entropy accounts for the observed behavior across different off-policy RL regimes.

3.1 Does Replay Volume Matter?

We first test whether replay volume modulates the effectiveness of recency-biased sampling.

Experiment. To isolate the role of replay volume, we vary UTD and batch size while keeping the replay buffer and replay sampling distributions fixed. Varying UTD changes the number of

Method	μ	$\mathcal{H}(p_t)$	AUC gain
Trunc. Geom.	0.85	**12.9**	**+12.9%**
Unif. FIFO	0.85	12.6	+4.6%
ERE	0.85	12.2	−41.4%

AUC gain relative to Uniform (1M).

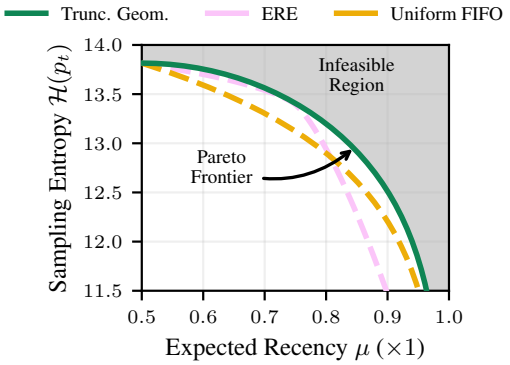

Figure 4: ***Sampling entropy matters.*** (*left*) ERE, Uniform FIFO (300k), and Truncated Geometric are matched to the same expected recency ($\mu \approx 0.85$), yet AUC gain relative to the Uniform (1M) baseline differs substantially with the sampling entropy $\mathcal{H}(p_t)$. At this μ, ERE has the lowest entropy and falls below the Uniform baseline (see Appendix E.4), while Truncated Geometric, which has the highest entropy at this μ, performs best. Interestingly, Uniform FIFO (300k) also outperforms the larger Uniform baseline (1M) despite using a smaller buffer. (*right*) The solid curve shows the maximum-entropy frontier for a 1M buffer: among all discrete replay sampling distributions with fixed expected recency, the Truncated Geometric distribution achieves the highest possible sampling entropy. The shaded region above the curve is infeasible. ERE and Uniform FIFO trace lower-entropy tradeoffs across hyperparameter settings, showing that they obtain recency less efficiently than Truncated Geometric; we discuss entropy interpretation in Appendix E.3.

gradient updates performed per environment step, while varying batch size changes the number of transitions consumed per update. Both therefore change replay volume without changing the expected recency or sampling entropy induced by a fixed sampling strategy. Both UTD and batch size are practically relevant hyperparameters and both reflect compute-data tradeoffs in modern off-policy RL systems [27, 9]. As such, contemporary algorithms operate across a wide range of UTD regimes and batch size settings.* We compare uniform replay to recency-biased Truncated Geometric sampling using SimbaV2 [18] on 13 HumanoidBench tasks [31]. We sweep UTD values $\{2, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$ and batch sizes $\{256, 128, 64, 32, 16\}$. As UTD or batch size decreases, replay volume decreases, entering regimes where each transition is replayed relatively few times. We detail the setup in Appendix C.

Results. As shown in Figure 3, uniform and recency-biased replay perform similarly at high UTD values and large batch sizes. In this regime, recency-biased replay has higher expected recency than uniform replay, but replay volume is also large, so even uniform replay updates on many recent transitions. Consequently, increasing expected recency provides limited additional benefit. As UTD or batch size decreases, however, the performance gap widens, with recency-biased replay increasingly outperforming uniform replay. Since the sampling strategies remain fixed across the sweep, their relative expected recency and sampling entropy stay unchanged, while replay volume decreases. The observed shift therefore shows that the benefit of recency bias depends critically on replay volume: when replay volume is limited, the replay sampling distribution becomes more consequential; when replay volume is large, changing expected recency has little effect. These results show that replay volume modulates when non-uniform replay improves over uniform replay.

Takeaway 1 – Replay Volume Matters

The benefit of increasing expected recency depends on replay volume. When replay volume is high, uniform replay already samples many recent transitions, so recency offers limited benefits. When replay volume is low, the sampling distribution becomes more consequential, and recency-biased replay can yield substantial gains. Thus, the replay distribution is especially important in optimization-limited regimes such as multi-task learning or large-scale parallel simulation.

*E.g., BRO [22] uses UTD $= 10$, whereas FastTD3 [30] uses UTD $= \frac{1}{64}$ due to large-scale parallel data collection.

3.2 Does Sampling Entropy Matter?

The preceding subsection shows that replay volume modulates the effectiveness of sampling. We now study the second factor: whether replay strategies with comparable expected recency can nevertheless differ in performance due to the entropy of their replay sampling distributions.

Experiment. To assess whether sampling entropy matters beyond recency, we compare four replay strategies on HumanoidBench using BRC [23]. Three strategies are matched for expected recency ($\mu \approx 0.85$) but differ in entropy: ERE with a 1M buffer, Uniform FIFO with a $\approx$ 300K buffer, and Truncated Geometric with a 1M buffer. Matching μ allows us to probe the effect of entropy while holding average recency fixed. As shown in Figure 4, entropy decreases substantially from Truncated Geometric to Uniform FIFO to ERE, with ERE being the most concentrated. We also include Uniform replay with a 1M buffer as a standard baseline. We detail the setup in Appendix C.

Results. As shown in Figure 4, performance tracks sampling entropy among the μ-matched recency-biased strategies: ERE, which has the lowest entropy, underperforms even the Uniform baseline; Uniform FIFO (300k) performs better; and Truncated Geometric, which has the highest entropy, achieves the best performance overall. Interestingly, Uniform FIFO (300k) slightly outperforms Uniform (1M) despite using a 3 times smaller buffer, which we attribute to its higher expected recency. These results show that *how* recency is achieved matters. Strategies that obtain recency by sharply restricting sampling support sacrifice entropy of the replay sampling distribution and can underperform even when their average recency is favorable. In contrast, smooth full-support bias, as in Truncated Geometric replay, preserves update diversity while shifting replay toward recent data.

Takeaway 2 – Sampling Entropy Matters

Expected recency is informative but not sufficient to determine performance. Strategies with comparable expected recency can differ substantially depending on whether they preserve high-entropy, full-support sampling. Effective non-uniform replay should therefore bias sampling smoothly while maintaining the diversity in the replay sampling distribution.

4 Efficient Truncated Geometric Sampling Procedure

The analysis in Section 3 suggests that an effective replay sampler should increase expected recency without sacrificing sampling entropy or adding computational overhead. We instantiate this principle with a truncated geometric distribution over replay-buffer indices. This sampler smoothly biases updates toward recent transitions, admits constant-time inverse-CDF sampling, and maximizes entropy among distributions with fixed expected recency. We summarize the procedure in Algorithm 1.

Truncated Geometric Sampler. To smoothly increase expected recency while retaining constant-time sampling, we use a truncated geometric distribution over replay-buffer indices. Let $N_{\max}$ denote the maximum replay-buffer capacity and N_{current} its size at optimization step t. Transitions are indexed from oldest ($i = 0$) to newest ($i = N_{\text{current}} - 1$). We sample an index $i \in \{0, \dots, N_{\text{current}} - 1\}$ according to

$$p(i) \propto 2^{\alpha \frac{i}{N_{\max}-1}},$$

where $\alpha > 0$ controls the strength of the recency bias. Larger values of α place more probability mass on recent transitions, while $\alpha \to 0$ recovers uniform replay. Unlike PER, we omit importance-sampling correction, which would counteract the recency bias. This sampler directly addresses the factors identified in Section 3. Firstly, it admits closed-form inverse-CDF sampling and therefore constant-time implementation. Secondly, in contrast to approaches like ERE, it increases expected

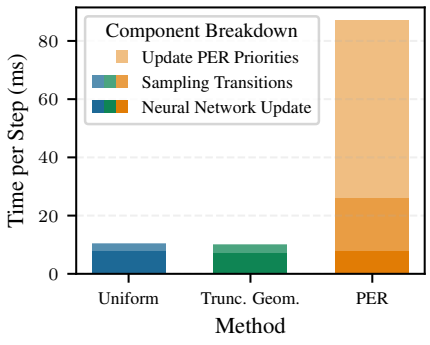

Figure 5: ***Decomposition of total latency.*** We compare the computational overhead of Uniform, Truncated Geometric, and PER. While the network update time is constant across methods, PER introduces significant latency due to priority tree management. In contrast, the truncated geometric sampling maintains a profile nearly identical to uniform due to efficient probability calculation.

Algorithm 1 Efficient Truncated Geometric Sampling

Require: Replay buffer $\mathcal{D}$, capacity $N_{\max}$, current size $N_{\text{current}} \leq N_{\max}$, recency parameter $\alpha > 0$.

Index transitions in $\mathcal{D}$ as $i = 0, \ldots, N_{\text{current}} - 1$, where $i = 0$ is the oldest and $i = N_{\text{current}} - 1$ is the most recent.
Set $k \leftarrow \alpha/(N_{\max} - 1)$.
Sample $U \sim \text{Unif}(0, 1)$.
Compute normalization constant $Z \leftarrow \frac{2^{kN_{\text{current}}}-1}{2^k-1}$.
Compute $r \leftarrow U \cdot Z$.
Compute index via inverse CDF: $i \leftarrow \lfloor \frac{\log_2(1+r(2^k-1))}{k} - 1 \rfloor$.
Clip i to $[0, N_{\text{current}} - 1]$.
return transition $\mathcal{D}[i]$.

recency smoothly, without restricting support to a recent window. Finally, among all distributions over buffer indices with fixed expected recency, the truncated geometric distribution maximizes sampling entropy [17, 7]. Thus, for a given expected recency, it preserves the highest update diversity. We visualize the tradeoff between expected recency and sampling entropy in the right panel of Figure 4.

Efficient Sampling. Sampling is performed using an efficient inverse transform. A single uniform random variable $U \sim \text{Unif}(0, 1)$ is mapped through the closed-form inverse cumulative distribution function of the truncated geometric distribution. The sampled index is given by:

$$i = \left\lfloor \frac{\log_2\big(1 + U \cdot Z \cdot (2^k - 1)\big)}{k} \right\rfloor - 1$$

clipped to $[0, N_{\text{current}} - 1]$, with $k = \alpha \,/\, (N_{\max} - 1)$, and $Z = (2^{kN_{\text{current}}} - 1) \,/\, (2^k - 1)$. While non-uniform replay is sometimes assumed to introduce additional wall-clock overhead, the proposed sampling procedure requires constant time, yielding $\mathcal{O}(1)$ complexity. This contrasts with the cost incurred by strategies whose sampling probabilities depend on updated priorities supported by segment trees [28], which yield $\mathcal{O}(\log N)$ complexity. As shown in Figure 5, the empirical wall-clock performance of the proposed truncated geometric sampling is comparable to uniform replay.

Interpretation of α. The proposed truncated geometric sampling is governed by a single hyperparameter α. As we now show, it provides intuitive control over **expected recency** in sampling. Approximately, for $\Delta \approx (N_{\max} - 1)/\alpha$,

$$\frac{p(i + \Delta)}{p(i)} \approx 2.$$

Thus, moving forward by roughly $(N_{\max} - 1)/\alpha$ positions doubles the sampling probability. Smaller α values approach uniform replay, while larger values increasingly emphasize recent transitions. We discuss how we choose α in Appendix E.5.

5 Experiments

The analysis in Section 3 predicts that recency-biased sampling should be most effective when replay volume is limited, provided the sampler preserves high entropy. We test this prediction across four representative off-policy protocols: large-scale parallel simulation, multi-task learning, single-task learning, and discrete-action tasks. Parallel simulation and multi-task learning are the primary low-replay volume regimes, since data collection is distributed across many environments or tasks.

For each regime, we start from the original algorithm and training setup of the corresponding source paper, modifying only the replay sampler. We compare the original uniform replay baseline against PER [28], ERE [36], and the proposed Truncated Geometric sampler. All base algorithm hyperparameters and training protocols are kept unchanged. PER and ERE hyperparameters are tuned following Appendix C.2, while Truncated Geometric uses a fixed $\alpha = 10$ across all experiments. Unless otherwise specified, all methods use a replay buffer of capacity 1M. We report 95% stratified bootstrap CIs computed with RLiable [1]. Performance is evaluated using normalized episodic returns and success rates where applicable, across multiple seeds. Full experimental details, including

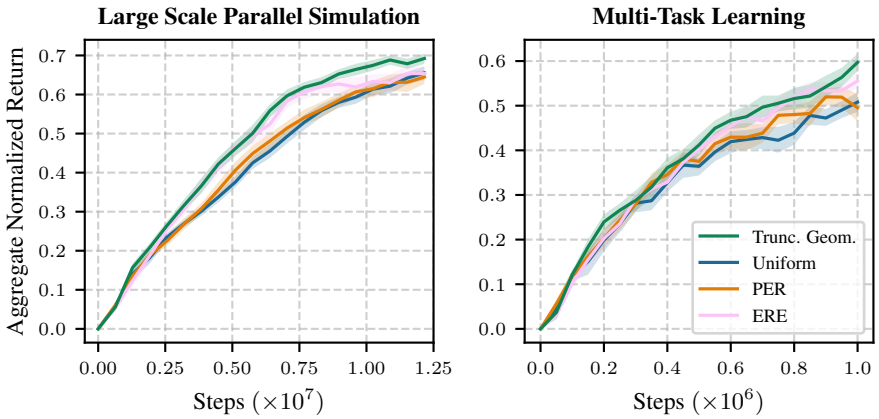

Figure 6: ***High-dimensional humanoid locomotion and manipulation tasks.*** We report aggregate mean return across 29 HumanoidBench tasks trained in a parallel simulation setup with FastTD3 (*left*) and 20 HumanoidBench tasks trained in a multi-task setup with BRC (*right*). Shaded regions show 95% CIs. Truncated Geometric sampling significantly improves over uniform replay on both benchmarks and outperforms PER and ERE, despite FastTD3 and BRC already being strong optimized baselines. These gains come with negligible computational and complexity overhead relative to uniform sampling. We detail the experimental setup in Section 5 and Appendix C.

hyperparameter settings, detailed figure descriptions, computational resources, environment lists, normalization procedures, and evaluation protocols are provided in Appendices C and D.

Large-Scale Parallel Simulation. We first evaluate replay sampling in a large-scale parallel simulation setting using FastTD3 [30]. We consider 29 high-dimensional continuous-control tasks from HumanoidBench [31] and 6 tasks from Isaac Lab [20]; full task lists are provided in Appendix D.2. In this setting, experience is collected rapidly from many environments in parallel, while relatively few gradient updates are performed per environment step, resulting in limited replay volume. On HumanoidBench, shown in the left panel of Figure 6, Truncated Geometric sampling improves sample efficiency over uniform replay, achieving up to $1.35\times$ faster learning in terms of environment interactions. PER performs comparably to uniform replay, while ERE provides intermediate gains between uniform replay and Truncated Geometric. Because Truncated Geometric sampling incurs small computational overhead relative to uniform, these sample-efficiency gains translate directly into faster wall-clock convergence, reaching comparable performance in approximately 80% of the training time required by standard FastTD3 with uniform replay. We observe similar trends on Isaac Lab, where recency-biased samplers also outperform uniform replay. In this benchmark, Truncated Geometric performs comparably to ERE in aggregate, with ERE benefiting from particularly strong performance on one task. We report the full Isaac Lab results in Appendix F.10.

Multi-Task Learning. We next evaluate replay sampling in the multi-task setting using BRC [23] on the HumanoidBench-Hard tasks [23] listed in Appendix F.3. As in large-scale parallel simulation, multi-task learning naturally limits replay volume, since experience is distributed across many tasks while optimization per task remains limited. As shown in the right panel of Figure 6, Truncated Geometric sampling achieves the best overall performance and improves substantially over uniform replay. ERE is competitive in this setting, though slightly below Truncated Geometric, in contrast to the low-entropy ERE configurations studied in Section 3.2, where aggressive concentration hurts performance. PER provides modest gains over uniform replay but incurs substantial computational overhead. These results further support the analysis in Section 3: when replay volume is limited relative to data collection, increasing expected recency through smooth, high-entropy recency bias improves performance while preserving computational efficiency. We also evaluate BRC with different replay strategies on DMC Dog [35] and Meta-World [38]; full results are provided in Appendix F.

Single-Task Learning. In Section 3.1, we evaluated SimbaV2 on 13 HumanoidBench Nohands tasks across UTD values from 2 to $1/16$ and batch sizes from 256 to 16, showing that Truncated Geometric sampling yields larger gains as replay volume becomes limited. Here, we return to the original SimbaV2 benchmark of 30 tasks under its standard high-UTD, standard-batch-size setting, where replay volume is substantial [18]. This experiment tests whether Truncated Geometric sampling remains safe in the high-replay volume regime, where our analysis predicts limited gains from increasing expected recency. As shown in Figure 15, Truncated Geometric sampling performs on

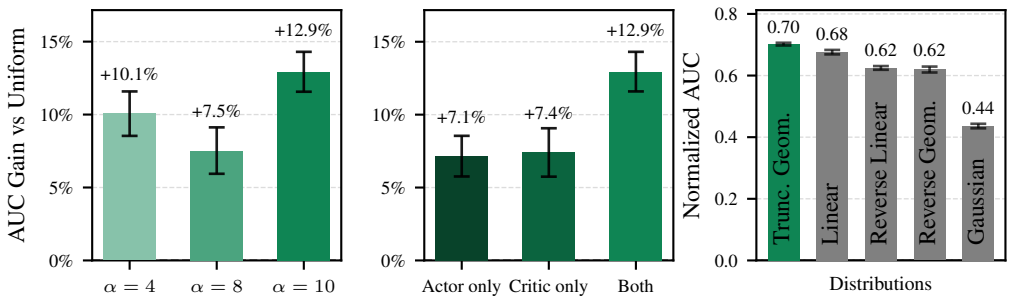

Figure 7: ***Ablations on Truncated Geometric replay.*** (*left*) Sample efficiency gains for Truncated Geometric replay with different recency parameter α, using 20 HumanoidBench tasks. Truncated Geometric consistently improves over uniform replay across all tested values of α, demonstrating robustness to hyperparameter selection. (*middle*) Sample efficiency gains when Truncated Geometric replay is applied separately to actor or critic, using 20 HumanoidBench tasks. These results show that Truncated Geometric sampling benefits actor and critic roughly equally. (*right*) Sampling distributions' sample efficiency investigated on the 4 DMC dog tasks, with Truncated Geometric as the top-performing method.

par with uniform replay, yielding a modest 1.2% improvement. These results support the prediction from Section 3: recency-biased replay improves performance when replay volume is limited, while remaining effectively neutral when replay volume is high.

Discrete Actions and Pixel Observations. To test whether our findings extend beyond continuous control, we evaluate Truncated Geometric sampling with DQN [21], using the CleanRL implementation [16], on 56 Atari games [4]. This setting combines pixel observations, discrete actions, and low replay volume (UTD $= 1/4$, batch size 32). Keeping all DQN hyperparameters unchanged and using the same $\alpha = 10$, Truncated Geometric improves the AUC of the IQM human-normalized score, the standard Atari metric [1, 21], by 18.9% over uniform replay, consistent with our prediction that recency bias helps most when replay volume is low. Full results are in Appendix F.11.

Ablation Studies. We ablate key design choices of Truncated Geometric sampling in Figure 7 using BRC on a variety of multi-task benchmarks. First, we evaluate sensitivity to the recency parameter α described in Section 4. Truncated Geometric sampling improves over uniform replay across a wide range of values, with gains between 7.5% and 12.9%, indicating that the Truncated Geometric sampling is robust to its only hyperparameter. Second, we ask whether the gains from recency-biased replay arise primarily through critic learning or through policy updates. This distinction is important because prior work has often found critic learning to be especially sensitive to algorithmic design choices in off-policy RL [8, 22]. We therefore apply Truncated Geometric sampling separately to the actor and critic updates, allowing for updates on independent batches of data. Shapley value analysis attributes 6.6% of the improvement to the critic and 6.3% to the actor, suggesting that both components benefit comparably from recency-biased replay. Finally, we compare against alternative non-uniform sampling distributions. Among the evaluated strategies that we describe in Appendix C.4, smooth recency-biased distributions achieve the strongest performance, supporting the role of high-entropy recency bias. We describe the experimental settings in detail in Appendix C.2.

Summary of the Experimental Results

Recency-biased replay provides the largest gains in low-replay volume regimes, such as large-scale parallel simulation and multi-task learning, while its benefits diminish when replay volume is high, as in high-UTD single-task learning. The results also support the role of sampling entropy: smooth, full-support recency bias outperforms strategies such as ERE that increase expected recency by sharply restricting sampling support. The same analysis explains Truncated Geometric sampling: it improves sample efficiency in low-replay volume regimes, incurs negligible overhead, and preserves performance when replay volume is high. Together, these findings support the analysis in Section 3 and clarify when recency-biased replay improves over uniform replay.

6 Conclusions

We revisited non-uniform replay sampling in off-policy reinforcement learning and showed that its effectiveness cannot be understood from expected recency alone. Instead, replay performance is mediated by two additional factors: replay volume and sampling entropy. Recency-biased replay is most beneficial in low-replay volume regimes, where transitions are replayed infrequently, and when the sampling distribution preserves high entropy rather than sharply restricting support. Practically, these findings suggest that replay design should be treated as a function of the learner–collector ratio: uniform replay is often sufficient when optimization is abundant, while smooth recency-biased replay is a simple and effective default when data collection outpaces learning.

Guided by this analysis, we introduced a simple Truncated Geometric replay sampler that smoothly increases expected recency while maintaining high entropy and constant-time sampling. Across large-scale parallel simulation, multi-task learning, and single-task settings, this approach improves sample efficiency in low-replay volume regimes, incurs negligible computational overhead with $\mathcal{O}(1)$ sampling complexity, and remains neutral when replay volume is high. Our results clarify when and why recency-biased replay improves learning, and provide a practical alternative to uniform and priority-based replay for modern off-policy RL. We discuss limitations of our analysis in Appendix A.

Acknowledgments

We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2025/018697. Mikołaj Czarnecki was supported by the National Science Centre under Grant No. 2023/50/E/ST6/00665. Yarden As received funding from a grant of the Hasler foundation (grant no. 21039) and the ETH AI Center. Pieter Abbeel holds concurrent appointments as a Professor at UC Berkeley and as an Amazon Scholar. This paper describes work performed at UC Berkeley and is not associated with Amazon.

References

[1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

[2] Yarden As, Dhruva Tirumala, René Zurbrügg, Chenhao Li, Stelian Coros, Andreas Krause, and Markus Wulfmeier. What matters for simulation to online reinforcement learning on real robots. *arXiv preprint arXiv:2602.20220*, 2026.

[3] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, TB Dhruva, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. In *International Conference on Learning Representations*, 2018.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[5] Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. CrossQ: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *International conference on learning representations (ICLR)*, 2024.

[6] Vittorio Caggiano, Huawei Wang, Guillaume Durandau, Massimo Sartori, and Vikash Kumar. Myosuite–a contact-rich simulation suite for musculoskeletal motor control. *arXiv preprint arXiv:2205.13600*, 2022.

[7] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[8] Pierluca D'Oro, Max Schwarzer, Evgenii Nikishin, Pierre-Luc Bacon, Marc G Bellemare, and Aaron Courville. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *The Eleventh International Conference on Learning Representations*, 2022.

[9] Preston Fu, Oleh Rybkin, Zhiyuan Zhou, Michal Nauman, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Compute-optimal scaling for value-based deep rl, 2025. URL https://arxiv.org/abs/2508.14881.

[10] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 2018.

[11] Scott Fujimoto, Pierluca D'Oro, Amy Zhang, Yuandong Tian, and Michael Rabbat. Towards general-purpose model-free reinforcement learning. *arXiv preprint arXiv:2501.16142*, 2025.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 2018.

[13] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2019.

[14] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[15] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

[17] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[18] Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv:2502.15280*, 2025.

[19] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[20] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, Lukasz Wawrzyniak, Milad Rakhsha, Alain Denzler, Eric Heiden, Ales Borovicka, Ossama Ahmed, Iretiayo Akinola, Abrar Anwar, Mark T. Carlson, Ji Yuan Feng, Animesh Garg, Renato Gasoto, Lionel Gulich, Yijie Guo, M. Gussert, Alex Hansen, Mihir Kulkarni, Chenran Li, Wei Liu, Viktor Makoviychuk, Grzegorz Malczyk, Hammad Mazhar, Masoud Moghani, Adithyavairavan Murali, Michael Noseworthy, Alexander Poddubny, Nathan Ratliff, Welf Rehberg, Clemens Schwarke, Ritvik Singh, James Latham Smith, Bingjie Tang, Ruchik Thaker, Matthew Trepte, Karl Van Wyk, Fangzhou Yu, Alex Millane, Vikram Ramasamy, Remo Steiner, Sangeeta Subramanian, Clemens Volk, CY Chen, Neel Jawale, Ashwin Varghese Kuruttukulam, Michael A. Lin, Ajay Mandlekar, Karsten Patzwaldt, John Welsh, Huihua Zhao, Fatima Anes, Jean-Francois Lafleche, Nicolas Moënne-Loccoz, Soowan Park, Rob Stepinski, Dirk Van Gelder, Chris Amevor, Jan Carius, Jumyung Chang, Anka He Chen, Pablo de Heras Ciechomski, Gilles Daviet, Mohammad Mohajerani, Julia von Muralt, Viktor Reutskyy, Michael Sauter, Simon Schirm, Eric L. Shi, Pierre Terdiman, Kenny Vilella, Tobias Widmer, Gordon Yeoman, Tiffany Chen, Sergey Grizan, Cathy Li, Lotus Li, Connor Smith, Rafael Wiltz, Kostas Alexis, Yan Chang, David Chu, Linxi "Jim" Fan, Farbod Farshidian, Ankur Handa, Spencer Huang, Marco Hutter, Yashraj Narang, Soha Pouya, Shiwei Sheng, Yuke Zhu, Miles Macklin, Adam Moravanszky, Philipp Reist, Yunrong Guo, David Hoeller, and Gavriel State. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

[21] 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.

[22] Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. *Advances in neural information processing systems*, 2024.

[23] Michal Nauman, Marek Cygan, Carmelo Sferrazza, Aviral Kumar, and Pieter Abbeel. Bigger, regularized, categorical: High-capacity value functions are efficient multi-task learners. *arXiv preprint arXiv:2505.23150*, 2025.

[24] Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. Xqc: Well-conditioned optimization accelerates deep reinforcement learning. *arXiv preprint arXiv:2509.25174*, 2025.

[25] Leonard S. Pleiss, Tobias Sutter, and Maximilian Schiffer. Reliability-adjusted prioritized experience replay, 2025. URL https://arxiv.org/abs/2506.18482.

[26] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[27] Oleh Rybkin, Michal Nauman, Preston Fu, Charlie Snell, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Value-based deep rl scales predictably. *arXiv preprint arXiv:2502.04327*, 2025.

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

[29] Younggyo Seo, Carmelo Sferrazza, Juyue Chen, Guanya Shi, Rocky Duan, and Pieter Abbeel. Learning sim-to-real humanoid locomotion in 15 minutes. *arXiv preprint arXiv:2512.01996*, 2025.

[30] Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

[31] Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. *arXiv preprint arXiv:2403.10506*, 2024.

[32] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[33] Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

[34] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[35] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[36] Che Wang and Keith Ross. Boosting soft actor-critic: Emphasizing recent experience without forgetting the past, 2019. URL https://arxiv.org/abs/1906.04009.

[37] Che Wang, Yanqiu Wu, Quan Vuong, and Keith Ross. Striving for simplicity and performance in off-policy drl: Output normalization and non-uniform sampling. In *International Conference on Machine Learning*, pp. 10070–10080. PMLR, 2020.

[38] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *CoRR*, abs/1910.10897, 2019. URL http://arxiv.org/abs/1910.10897.

A Limitations

Our experiments focus primarily on continuous-control off-policy RL, where replay buffers are central and low-replay volume regimes arise naturally in large-scale parallel and multi-task training. Although our ablations show that Truncated Geometric replay is robust to the recency parameter, adaptive schedules for recency bias may further improve performance. More broadly, our study is empirical: the decomposition into expected recency, replay volume, and sampling entropy provides an explanatory and practical framework, but not a complete theory of replay. Developing such a theory could clarify how replay sampling interacts with optimization, bootstrapping, distribution shift, and function approximation. Finally, while our analysis aligns well with aggregate trends across replay regimes, it does not fully account for task-level heterogeneity within a fixed regime. Identifying the environment or optimization properties that determine when recency-biased replay underperforms remains an important direction for future work.

B Truncated Geometric Sampling Procedure Details

In this section, we discuss the implementation details of the proposed Truncated Geometric Sampling Procedure.

B.1 Python Implementation

We provide a concise NumPy [14] implementation of the Truncated Geometric sampling. The implementation uses inverse-CDF sampling, making it $\mathcal{O}(1)$ and thus as computationally efficient as uniform sampling.

```python
import numpy as np

def geometric_sample(size, batch_size, alpha=10.0, capacity=1_000_000, offset=0):
    k = alpha / (capacity - 1)
    base = 2.0 ** k
    Z = (base ** size - 1.0) / (base - 1.0)
    r = np.random.uniform(0.0, Z, size=batch_size)
    i = np.log2(1.0 + r * (base - 1.0)) / k - 1.0
    i = np.clip(np.floor(i).astype(np.int64), 0, size - 1)
    if offset != 0:
        i = (i + offset) % capacity
    return i
```

B.2 Parallel Buffer Pseudocode

Algorithm 2 extends the single-task Truncated Geometric sampling procedure to the multi-task setting, where each task maintains an independent replay buffer. At each sampling step, a task is selected uniformly at random, and a transition is sampled from the corresponding buffer using the Truncated Geometric distribution, preserving recency bias across all tasks without introducing additional computational overhead.

C Experimental Details

C.1 Figures

Figure 1. We report the results in ***Low Replay Volume, High Expected Recency*** setting on HumanoidBench [31], with FastTD3 [30] evaluated across all 29 Hands tasks and BRC [23] across all 20 available tasks. Both algorithms operate in the low replay volume regime — FastTD3 [30] with a UTD ratio of $1/64$ and BRC [23] with $1/10$ — where recency bias is expected to yield the greatest benefit (Section 3.1). Wall-clock time is measured as the total running time of the algorithm and is aggregated across both settings, covering 49 tasks in total.

Figure 2. We show expected amount of replays we sample from each index in between each environment step. Index recency is normalized index in replay buffer with 0 corresponding to the oldest sample and 1 corresponding to the newest sample in the buffer. By default all figures correspond

Algorithm 2 Efficient Truncated Geometric Sampling Procedure

Require: Per-task replay buffer $\mathcal{D}$, max capacity $N_{\max}$, current size $N_{\text{current}} \leq N_{\max}$, number of tasks M, recency parameter $\alpha > 0$.

Sample task $j \sim \text{Unif}(\{0, \ldots, M-1\})$.
Index transitions in $\mathcal{D}[j]$ as $i = 0, \ldots, N_{\text{current}} - 1$, where $i = 0$ is the oldest and $i = N_{\text{current}} - 1$ is the most recent.
Set $k \leftarrow \alpha/(N_{\max} - 1)$.
Sample $U \sim \text{Unif}(0, 1)$.
Compute normalization constant $Z \leftarrow \frac{2^{k N_{\text{current}}} - 1}{2^k - 1}$.
Compute $r \leftarrow U \cdot Z$.
Compute index via inverse CDF: $i \leftarrow \lfloor \frac{\log_2(1 + r(2^k - 1))}{k} - 1 \rfloor$.
Clip i to $[0, N_{\text{current}} - 1]$.
return transition $\mathcal{D}[j][i]$.

to expected recency of training with UTD 2 and Batch Size 128. On left and middle plot mean of Uniform sampling distribution, μ_{U} has index recency $\rho_t(i) = 0.5$ and mean of Truncated Geometric distribution μ_{TG} has index recency $\rho_t(i) = 0.857$. On middle plot higher Uniform replay volume corresponds to training with UTD 16 and Batch Size 128. On right plot we provide numeric values of Entropy for Truncated Geometric distribution of size $1,000,000$ and Uniform FIFO distribution of size $288,000$. They both have the same mean $\mu_{\text{TG}} = \mu_{\text{UF}} \approx 0.85$.

Figure 3. We report the results in varying ***Replay Volume*** comparing ***High Expected Recency*** Truncated Geometric replay and ***Low Expected Recency*** Uniform replay on HumanoidBench Nohands [31] using SimbaV2 [18] with 95% stratified bootstrap CI computed via RLiable [1]. All experiments were run on $1,000,000$ steps with hyperparameters described in Table 1 in column 'SimbaV2'.

Figure 4. Results are reported in ***Low Replay Volume*** comparing ***High Expected Recency*** Truncated Geometric, ERE, and Uniform FIFO replays with ***Low Expected Recency*** Uniform replay on HumanoidBench [31] using BRC [23]. All results have 95% stratified bootstrap confidence intervals within $\pm 1.4\%$, computed with RLiable [1]. The entropy-recency frontier shown in the right panel is computed for a buffer of capacity 1M. Visualizations of the corresponding sampling distributions are provided in Figure 8. We discuss Uniform FIFO as a recency-biased strategy in Appendix E.2.

Figure 5. Measurements are conducted in ***Low Replay Volume*** comparing ***High Expected Recency*** Truncated Geometric replay with **Low *Expected Recency*** Uniform and PER replay on HumanoidBench using BRC across 20 tasks. PER maintains a separate segment tree per task, implemented in NumPy [14], to avoid Q-value scale mismatch across tasks. Latency is measured using `time.time` in Python; each operation is timed over 1000 consecutive steps and averaged to obtain a reliable estimate. All measurements are taken approximately at the midpoint of training.

Figure 6. Results on both plots are reported in ***Low Replay Volume*** setting on HumanoidBench [31] comparing ***High Expected Recency*** Truncated Geometric and ERE replays with ***Low Expected Recency*** Uniform and PER replays. Large Scale Parallel Simulation uses FastTD3 [30] algorithm. Multi-Task Learning uses BRC [23]. Shaded regions are 95% stratified bootstrap CI computed via RLiable [1]. On Large Scale Parallel Simulation we report total number of environment steps on x-axis. We tuned Episode Length and Decay Constant η in ERE buffer to achieve better results.

Figure 7. Results on α and actor/critic ablations are reported in ***Low Replay Volume*** setting on HumanoidBench [31], results on distribution ablations are reported on DMC Dogs [35]. All ablations use BRC [23] algorithm with 95% stratified bootstrap CI computed via RLiable [1]. Details of sampling distributions from right ablation are discussed in Section C.4.

C.2 Hyperparameters

Table 1 reports the full set of hyperparameters used across all experiments. For each algorithm, we distinguish between default hyperparameters (shown in **bold**), which are taken directly from the original implementations, and tuned hyperparameters (shown in *italic*), which were adjusted to accommodate the different replay sampling strategies.

Hyperparameter	FastTD3	BRC	SimbaV2	DQN
Environment and Execution				
Environment suite	HumanoidBench/Isaac Lab	HumanoidBench/DMC	**HumanoidBench Nohands**/DMC/MYOSuite	**Atari**
Parallel environments	**128 (HumanoidBench)/2048 (Isaac Lab)**	**20 (HB)**, **3 (DMC/Humanoids)**, **4 (DMC Dogs)**	–	**1**
Evaluation environments	**128 (HumanoidBench)/2048 (Isaac Lab)**	–	–	–
Total environment steps	**100,000**	**1,000,000**	**1,000,000**	**10,000,000**
Number of seeds	**5**	**3–6**	**5**	**5**
TD-n	**1 (HumanoidBench)/1–8 (Isaac Lab)**	**1**	**1**	**1**
Optimization				
Update-To-Data ratio	**1/64 (HumanoidBench)/1/128 (Isaac Lab)**	**1/10**	$\mathbf{2}, 1, 1/2, 1/4, 1/8, 1/16$	**1/4**
Actor learning rate	$\mathbf{3 \times 10^{-4}}$	$\mathbf{3 \times 10^{-4}}$	$\mathbf{1 \times 10^{-4}}$	–
Critic learning rate	$\mathbf{3 \times 10^{-4}}$	$\mathbf{3 \times 10^{-4}}$	$\mathbf{1 \times 10^{-4}}$	$\mathbf{1 \times 10^{-4}}$
Learning rate schedule	**Cosine annealing** ($\mathbf{3 \times 10^{-4} \to 3 \times 10^{-5}}$)	**Constant**	**Linear annealing** ($\mathbf{1 \times 10^{-4} \to 5 \times 10^{-5}}$)	**Constant**
Batch size	**32768**	**1024**	**256**	**32**
Discount factor γ	**0.99**	**0.99**	**0.95 (HumanoidBench/DMC)**, **0.995 (MYOSuite)**	**0.99**
Target update coefficient τ	**0.1**	**0.005**	**0.005**	**1.0 (every 1000 steps)**
Policy update frequency	**2**	**2**	**2**	–
Learning starts	**10 steps**	**5000 steps**	**5000**	**80,000 steps**
Weight decay	**0.1**	–	–	–
Replay Buffer				
Replay buffer capacity	**50,000 (HumanoidBench)** / $\mathbf{5 \times 1024 - 10 \times 1024}$ **(Isaac Lab)**	100,000, **1,000,000**	**1,000,000**	**1,000,000**
Sampling strategies	**Uniform** / PER / ERE / TG	**Uniform** / PER / ERE / TG	**Uniform** / PER / ERE / TG	**Uniform** / TG
Method-Specific Hyperparameters				
Truncated Geometric α	**10.0**	***10.0***	**10.0**	**10.0**
PER exponent α	**0.6**	*0.4*	**0.6**	–
PER importance-sampling β	$\mathbf{0.4 \to 1.0}$	$\mathbf{0.4 \to 1.0}$	$\mathbf{0.4 \to 1.0}$	–
ERE decay constant η	**0.996**	***0.996*** **(HB, DMC Dogs)**, ***0.994*** **(DMC Humanoids)**	0.994	–
ERE episode length	*100*	**1000**	**1000**	–
ERE mini-batch updates	*200*	**2000**	**2000**	–
ERE minimum buffer size $c_{\min}$	**10** $\times$ batch size	$10 \times$ batch size	5000	–
Architecture				
Actor hidden dimension	**512**	**256**	**128**	–
Critic hidden dimension	**1024**	**4096**	**512**	**512 (Nature CNN)**
Critic blocks	**2**	**2**	**2**	–
Actor blocks	**1**	**1**	**1**	–
Distributional atoms	**101**	**101**	**101**	–
Value support $(v_{\min}, v_{\max})$	$\mathbf{(-250, 250)}$ **(HumanoidBench)** / $\mathbf{(-50, 50)}$, $\mathbf{(-10, 10)}$	$\mathbf{(-10, 10)}$	$\mathbf{(-5, 5)}$	–
Task embedding dimension	–	**32**	–	–
Stability and Systems				
Observation normalization	**Enabled**	**Disabled**	**Disabled**	**Disabled**
Reward normalization	**Disabled**	**Enabled**	**Enabled**	**Disabled**
Clipped Double Q	**Enabled**	**Disabled**	**Enabled**	**Disabled**
AMP	**Enabled (bf16)**	–	–	–
`torch.compile`	**Enabled** (Excluding ERE)	–	–	–
Gradient clipping	**Disabled**	**Disabled**	**Disabled**	**Disabled**

Table 1: Algorithm-specific training and replay hyperparameters. FastTD3 values are taken directly from the default configuration (`BaseArgs`). BRC values are taken from BRC flag defaults. SimbaV2 is default configuration (`configs`). DQN values are the CleanRL defaults (`dqn_atari_jax.py`) with no changes; for DQN, critic rows refer to the Q-network, and exploration uses ϵ-greedy with ϵ annealed linearly from 1.0 to 0.01 over the first 10% of training. **Bold** denotes default hyperparameters, *italic* denotes tuned hyperparameters.

C.3 Sampling Distributions

We visualize example sampling distributions for all used replay strategies in Figure 8. The Truncated Geometric, ERE, and Uniform FIFO distributions are matched to the same expected recency μ as used in Section 3.2.

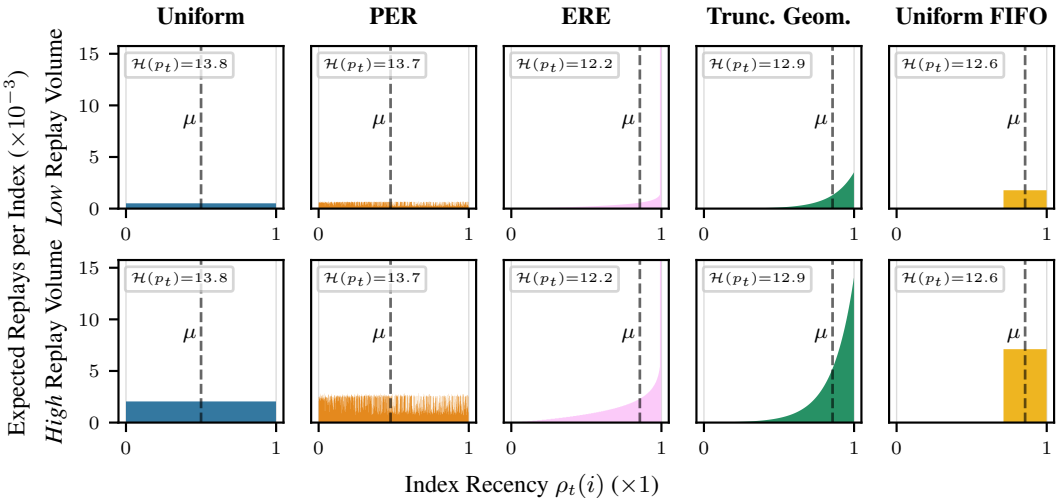

Figure 8: ***Sampling distributions and expected replay counts across methods.*** Each column shows the sampling distribution of a replay strategy visualized as expected number of replays per buffer index, in the *low* (top) and *high* (bottom) replay volume regimes. The dashed vertical line denotes the expected recency μ for each strategy, and $\mathcal{H}(p_t)$ reports the sampling entropy. ERE, Truncated Geometric, and Uniform FIFO are matched to the same expected index μ, yet differ noticeably in their sampling entropy. Truncated Geometric achieves the highest entropy at this μ. We discuss the interpretation of entropy differences in Appendix E.3.

C.4 Details of Distributions From Ablations

In distribution ablation from Figure 7 we train BRC [23] with different sampling distributions:

- ***High Expected Recency*** Linear.
- ***Low Expected Recency*** Reverse Linear.
- ***Low Expected Recency*** Gaussian.
- ***Low Expected Recency*** Reverse Truncated Geometric.

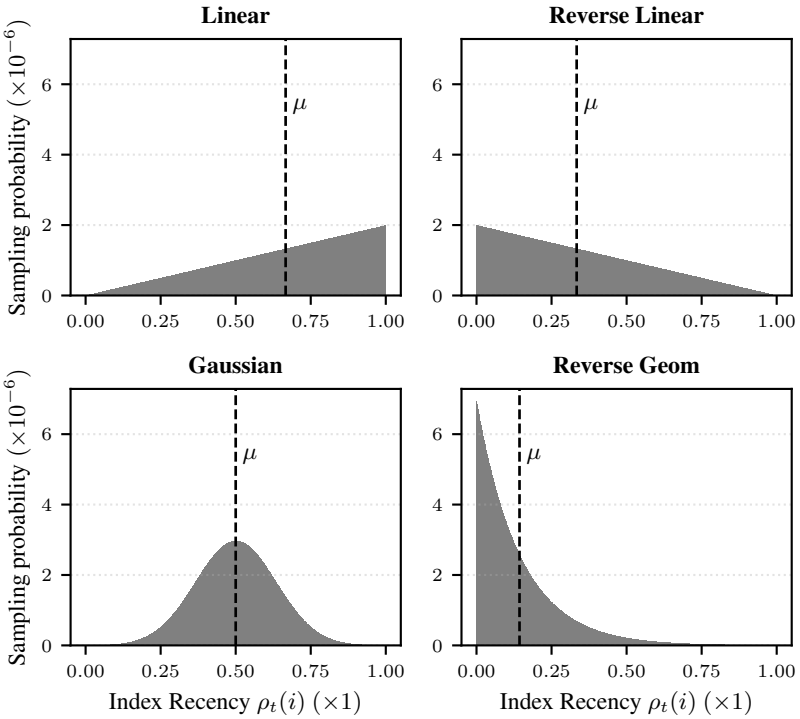

Figure 9: **Sampling distributions from Figure 7.** All four distributions are normalized so that the highest-probability transition is 2^{10} times more likely to be sampled than the lowest-probability transition.

C.5 Computational Resources

All experiments were conducted on NVIDIA A100 GPUs. Single-task and large-scale parallel runs required approximately 3 hours to complete, while multi-task runs required approximately 40 hours.

C.6 Reproducibility

All environment lists, hyperparameter configurations, training scripts are provided. Code is available at `https://github.com/M-Korniak/when-does-replay-matter`.

D Evaluation Protocols

This section provides full details of the experimental setup used throughout the paper, including replay buffer configurations, algorithm-specific hyperparameters, normalization procedures, and evaluation protocols. All experiments are conducted using the original implementations of the corresponding algorithms, with minimal modifications required to integrate alternative replay sampling strategies.

D.1 Policy Evaluation

All policies are evaluated periodically during training without exploration noise. Each evaluation consists of multiple episodes per task, and results are averaged. All reported curves show the mean performance over random seeds, with 95% confidence intervals computed using nonparametric bootstrapping via RLiable [1]. Unless otherwise specified, all experiments are run with three to six random seeds (see Table 1).

D.2 Benchmarks

In this section we describe benchmarks considered in our studies:

D.2.1 Multi-task benchmarks

All tasks are available in BRC [23] repository `https://github.com/naumix/BiggerRegularizedCategorical`.

- **BRC HumanoidBench Hands** - we use all 20 tasks of the HumanoidBench-Hard benchmark from [23], built on [31].

  Tasks: *walk, stand, run, stair, crawl, pole, slide, hurdle, maze, sit_simple, sit_hard, balance_simple, balance_hard, reach, spoon, window, insert_small, insert_normal, bookshelf_simple, bookshelf_hard*

- **BRC Meta-World** - we use benchmark of all 50 tasks from [38].

  Tasks: *assembly-v2-goal-observable, basketball-v2-goal-observable, bin-picking-v2-goal-observable, box-close-v2-goal-observable, button-press-topdown-v2-goal-observable, button-press-topdown-wall-v2-goal-observable, button-press-v2-goal-observable, button-press-wall-v2-goal-observable, coffee-button-v2-goal-observable, coffee-pull-v2-goal-observable, coffee-push-v2-goal-observable, dial-turn-v2-goal-observable, disassemble-v2-goal-observable, door-close-v2-goal-observable, door-lock-v2-goal-observable, door-open-v2-goal-observable, door-unlock-v2-goal-observable, hand-insert-v2-goal-observable, drawer-close-v2-goal-observable, drawer-open-v2-goal-observable, faucet-open-v2-goal-observable, faucet-close-v2-goal-observable, hammer-v2-goal-observable, handle-press-side-v2-goal-observable, handle-press-v2-goal-observable, handle-pull-side-v2-goal-observable, handle-pull-v2-goal-observable, lever-pull-v2-goal-observable, pick-place-wall-v2-goal-observable, pick-out-of-hole-v2-goal-observable, pick-place-v2-goal-observable, plate-slide-v2-goal-observable, plate-slide-side-v2-goal-observable, plate-slide-back-v2-goal-observable, plate-slide-back-side-v2-goal-observable, peg-insert-side-v2-goal-observable, peg-unplug-side-v2-goal-observable, soccer-v2-goal-observable, stick-push-v2-goal-observable, stick-pull-v2-goal-observable, push-v2-goal-observable, push-wall-v2-goal-observable, push-back-v2-goal-observable, reach-v2-goal-observable, reach-wall-v2-goal-observable, shelf-place-v2-goal-observable, sweep-into-v2-goal-observable, sweep-v2-goal-observable, window-open-v2-goal-observable, window-close-v2-goal-observable*

- **BRC DMC Dogs** - we use benchmark of all 4 DMC Dogs tasks [35].

  Tasks: *dog-stand, dog-walk, dog-trot, dog-run*

- **BRC DMC Humanoids** - we use benchmark of all 3 DMC Humanoid tasks [35].

  Tasks: *humanoid-stand, humanoid-walk, humanoid-run*

D.2.2 Large Scale Parallel Learning benchmarks

All tasks are available in FastTD3 [30] repository `https://github.com/younggyoseo/FastTD3`

- **FastTD3 HumanoidBench Hands** - we use benchmark of 29 HumanoidBench Hands tasks from [31]

Tasks: *balance_hard, balance_simple, basketball, bookshelf_ hard, bookshelf_simple, cabinet, crawl, cube, door, hurdle, insert_normal, insert_small, maze, package, pole, powerlift, push, reach, room, run, sit_hard, sit_simple, slide, spoon, stair, stand, truck, walk, window*

- **FastTD3 Isaac Lab** - we use benchmark of 6 Isaac Lab [20] tasks

  Tasks: *Velocity-Flat-G1, Velocity-Flat-H1, Velocity-Rough-G1, Velocity-Rough-H1, Repose-Cube-Allegro-Direct, Repose-Cube-Shadow-Direct*

D.2.3 Single-task benchmarks

All tasks are available for use in SimbaV2 [18] repository `https://github.com/DAVIAN-Robotics/SimbaV2`

- **SimbaV2 HumanoidBench No Hands** - we use benchmark of 13 HumanoidBench No Hands tasks from [31].

  Tasks: *walk, stand, run, hurdle, crawl, maze, sit_simple, sit_hard, balance_simple, balance_hard, stair, slide, pole*

- **SimbaV2 MyoSuite** - we use benchmark of 10 MyoSuite tasks from [6].

  Tasks: *reach, reach-hard, pose, pose-hard, obj-hold, obj-hold-hard, key-turn, key-turn-hard, pen-twirl, pen-twirl-hard*

- **SimbaV2 DMC** - we use 7 tasks from combined DMC Dogs and DMC Humanoids from [35].

  Tasks: *humanoid-stand, humanoid-walk, humanoid-run, dog-stand, dog-walk, dog-trot, dog-run*

D.2.4 Discrete-action benchmarks

All games are available through the Arcade Learning Environment [4]; we use the DQN implementation from the CleanRL [16] repository `https://github.com/vwxyzjn/cleanrl`

- **CleanRL DQN Atari** - we use 56 Atari games (the standard Atari-57 set without *Surround*, which is unavailable in the `NoFrameskip-v4` environments).

  Tasks: *Alien, Amidar, Assault, Asterix, Asteroids, Atlantis, BankHeist, BattleZone, BeamRider, Berzerk, Bowling, Boxing, Breakout, Centipede, ChopperCommand, CrazyClimber, Defender, DemonAttack, DoubleDunk, Enduro, FishingDerby, Freeway, Frostbite, Gopher, Gravitar, Hero, IceHockey, Jamesbond, Kangaroo, Krull, KungFuMaster, MontezumaRevenge, MsPacman, NameThisGame, Phoenix, Pitfall, Pong, PrivateEye, Qbert, Riverraid, RoadRunner, Robotank, Seaquest, Skiing, Solaris, SpaceInvaders, StarGunner, Tennis, TimePilot, Tutankham, UpNDown, Venture, VideoPinball, WizardOfWor, YarsRevenge, Zaxxon*

D.3 Normalization and Metrics

Performance is evaluated using standard task-specific metrics, including normalized episodic returns and success rates where applicable. Normalized returns are computed as

$$\text{Score} = \frac{R - R_{\text{random}}}{R_{\text{success}} - R_{\text{random}}},$$

where R_{random} and R_{success} are task-specific constants provided by the benchmark.

Table 2 reports normalization constants and success thresholds for all tasks sourced from Sferrazza et al. [31].

Task	Random Score R_{random}	Success Threshold R_{success}
h1-balance_hard-v0	9.044	800.0
h1-balance_simple-v0	9.391	800.0
h1-crawl-v0	272.658	700.0
h1-hurdle-v0	2.214	700.0
h1-maze-v0	106.441	1200.0
h1-pole-v0	20.090	700.0
h1-reach-v0	260.302	12000.0
h1-run-v0	2.020	700.0
h1-sit_hard-v0	2.448	750.0
h1-sit_simple-v0	9.393	750.0
h1-slide-v0	3.191	700.0
h1-stair-v0	3.112	700.0
h1-stand-v0	10.545	800.0
h1-walk-v0	2.377	700.0
h1hand-balance_hard-v0	10.032	800.0
h1hand-balance_simple-v0	10.170	800.0
h1hand-basketball-v0	8.979	1200.0
h1hand-bookshelf_hard-v0	14.848	2000.0
h1hand-bookshelf_simple-v0	16.777	2000.0
h1hand-cabinet-v0	37.733	2500.0
h1hand-crawl-v0	278.868	800.0
h1hand-cube-v0	4.787	370.0
h1hand-door-v0	2.771	600.0
h1hand-hurdle-v0	2.371	700.0
h1hand-insert_normal-v0	1.673	350.0
h1hand-insert_small-v0	1.653	350.0
h1hand-kitchen-v0	0.000	4.0
h1hand-maze-v0	106.233	1200.0
h1hand-package-v0	-10040.932	1500.0
h1hand-pole-v0	19.721	700.0
h1hand-powerlift-v0	17.638	800.0
h1hand-push-v0	-526.800	700.0
h1hand-reach-v0	-50.024	12000.0
h1hand-room-v0	3.018	400.0
h1hand-run-v0	1.927	700.0
h1hand-sit_hard-v0	2.477	750.0
h1hand-sit_simple-v0	10.768	750.0
h1hand-slide-v0	3.142	700.0
h1hand-spoon-v0	4.661	650.0
h1hand-stair-v0	3.161	700.0
h1hand-stand-v0	11.973	800.0
h1hand-truck-v0	562.419	3000.0
h1hand-walk-v0	2.505	700.0
h1hand-window-v0	2.713	650.0

Table 2: Normalization constants (R_{random}) and success thresholds (R_{success}) for HumanoidBench tasks.

E Discussion

E.1 Normalized AUC for Sample Efficiency

To evaluate sample efficiency, we report the *Normalized Area Under the Curve* (Normalized AUC). Rather than measuring performance only at the final training step, this metric captures learning progress throughout training and relates to the cumulative regret that the agent incurs during learning.

Formally, let R_t denote the normalized return at evaluation step t. Given equally spaced evaluation intervals $\{t_1, \ldots, t_K\}$, we compute

$$\text{Normalized AUC} = \frac{1}{K} \sum_{k=1}^{K} R_{t_k}.$$

For each run and task, we first compute the temporal mean of normalized returns across evaluation checkpoints. We then average these per-task Normalized AUC values across tasks and finally report the mean across runs. Confidence intervals are estimated using stratified bootstrap resampling over runs, following the RLiable protocol [1].

Normalized AUC provides a more faithful measure of sample efficiency than reporting only the final performance. Final-step metrics can obscure differences in convergence speed and learning stability, whereas Normalized AUC captures both how quickly and how consistently an agent improves during training.

This evaluation protocol follows prior work that emphasizes area-under-curve metrics for sample-efficient reinforcement learning [1].

E.2 Uniform FIFO as a Recency-Biased Method

A Uniform FIFO replay buffer can be interpreted as a recency-biased method. At each step, the oldest transition is discarded, which is equivalent to assigning it a sampling probability of zero. In this sense, FIFO implements recency bias through the discarding of oldest transitions rather than through non-uniform sampling probabilities.

Interestingly, in our preliminary experiments, alternative discarding strategies — such as discarding transitions based on Q-values, estimated advantages, or novelty estimated via random network distillation — did not improve over standard FIFO. This suggests that FIFO may be a surprisingly strong discarding baseline, consistent with the broader literature [36, 37] suggesting that recency — implicitly induced by FIFO through discarding — is an important factor in replay buffer design. We leave a systematic investigation of discarding strategies as an interesting direction for future work.

E.3 Entropy Scale and Buffer Size Equivalence

Since sampling entropy $\mathcal{H}(p_t)$ is a logarithmic quantity, small numerical differences correspond to substantial changes in sampling diversity. For intuition, consider a uniform distribution over a buffer of size N, which has entropy $\ln N$. A change of 1 nat in entropy is therefore equivalent to scaling the effective buffer size by a factor of $e \approx 2.71$. Consequently, differences in entropy of 0.3 nats — which may appear modest — correspond to a factor of $e^{0.3} \approx 1.35$ change in effective buffer size, i.e., a 35% increase or decrease in sampling diversity.

An alternative metric is perplexity, defined as $e^{\mathcal{H}(p_t)}$, which trades the logarithmic scale of entropy for a linear one, potentially making scalar values easier to interpret directly. However, perplexity and entropy are bijectively equivalent, and since entropy is the more standard metric in the literature, we report entropy throughout this work. We encourage readers to keep the logarithmic scale in mind when interpreting entropy differences reported in our figures.

E.4 ERE Buffer Shortcomings

On the left panel of Figure 4, the ERE buffer [36] underperforms severely. We believe this is caused by the low entropy of the distribution from which ERE samples. To make its expected recency match that of the Truncated Geometric with $\alpha = 10$, we tune the Decay Constant η by decreasing it. In the ERE buffer, replays are sampled in episodes k increasing cyclically from 1 to the episode length K; in each episode, we sample uniformly from $c_k = \max(N \cdot \eta^{1000k/K}, c_{\text{min}})$ most recent samples, where N is the length of the buffer and c_{min} is the minimal sampling size, usually set to 5000 for a buffer size of $1{,}000{,}000$. By decreasing the base of the exponent, i.e., the decay constant η, we increase the number of episodes for which we sample from the minimal window $c_k = c_{\text{min}}$; this drastically reduces sampling entropy, as visible for high expected recency μ in the right subplot of Figure 4. For the experiments in Figure 6, we tune the ERE parameters for every algorithm and benchmark

individually to achieve competitive performance comparable to other methods. In comparison, Truncated Geometric replay used a fixed hyperparameter of $\alpha = 10$ across all experiments.

E.5 Choosing α in Truncated Geometric Sampler

We choose α so that the expected recency lies near the elbow of the recency–entropy frontier (Figure 4, right). This point is recent enough to emphasize recent data, but comes before entropy starts to fall sharply. In all experiments, we use $\alpha = 10$ with no per-benchmark or per-task tuning. For a full buffer, this gives an expected recency of $\mu \approx 0.85$, which lies near the elbow. This choice of α performs consistently well across low-replay-volume settings (Section 5), and the gains also hold for $\alpha \in \{4, 8\}$ (Figure 7, left). We view $\alpha = 10$ as a practical, benchmark-independent heuristic rather than a proven optimum; identifying the optimal α is left to future work.

F Extended Results and Learning Curves

We provide extended results and learning curves for the experiments presented in Sections 3 and 5. In Figure 11 we present the performance of all sampling baselines when using FastTD3. In Figures 12 to 14 we illustrate the learning curves for our experiments using BRC on HumanoidBench and DeepMind Control Suite. In figures Figure 16 to Figure 21 we illustrate the learning curves for experiment from Figure 3. In sections F.9, F.10 and F.11 we report extra experiments mentioned in Section 5. In Section F.12 we report gains from SimbaV2 with Truncated Geometric replay compared to uniform replay. In Section F.13 we investigate behavior of SimbaV2 in Low Replay Volumewhen actions are perturbed randomly.

F.1 Prioritized Experience Replay UTD and Batch Size ablation

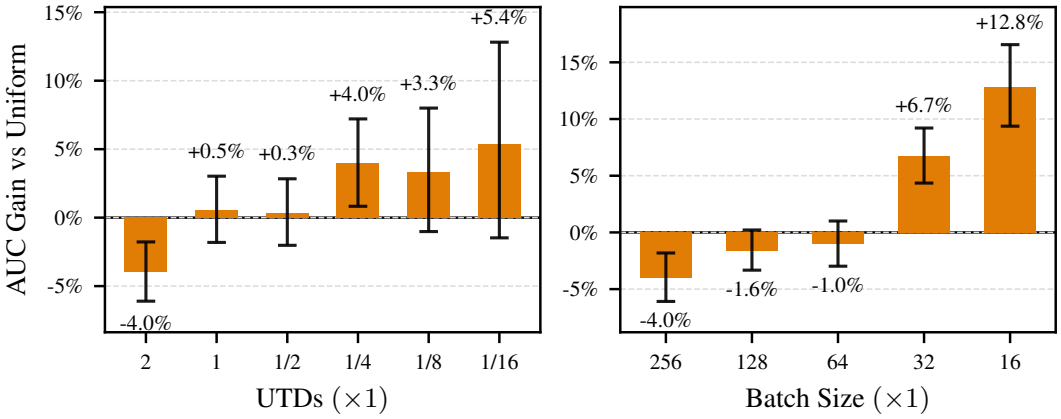

Figure 10: ***SimbaV2 on HumanoidBench (13 tasks).*** Improvement of Prioritized Experience Replay over uniform replay as **replay volume** is varied through UTD *(left)* and batch size *(right)*. We report a gain as a ratio of normalized AUCs from normalized returns over 5 seeds for each method. Error bars show 95% stratified bootstrap CIs computed with `rliable`.

F.2 FastTD3

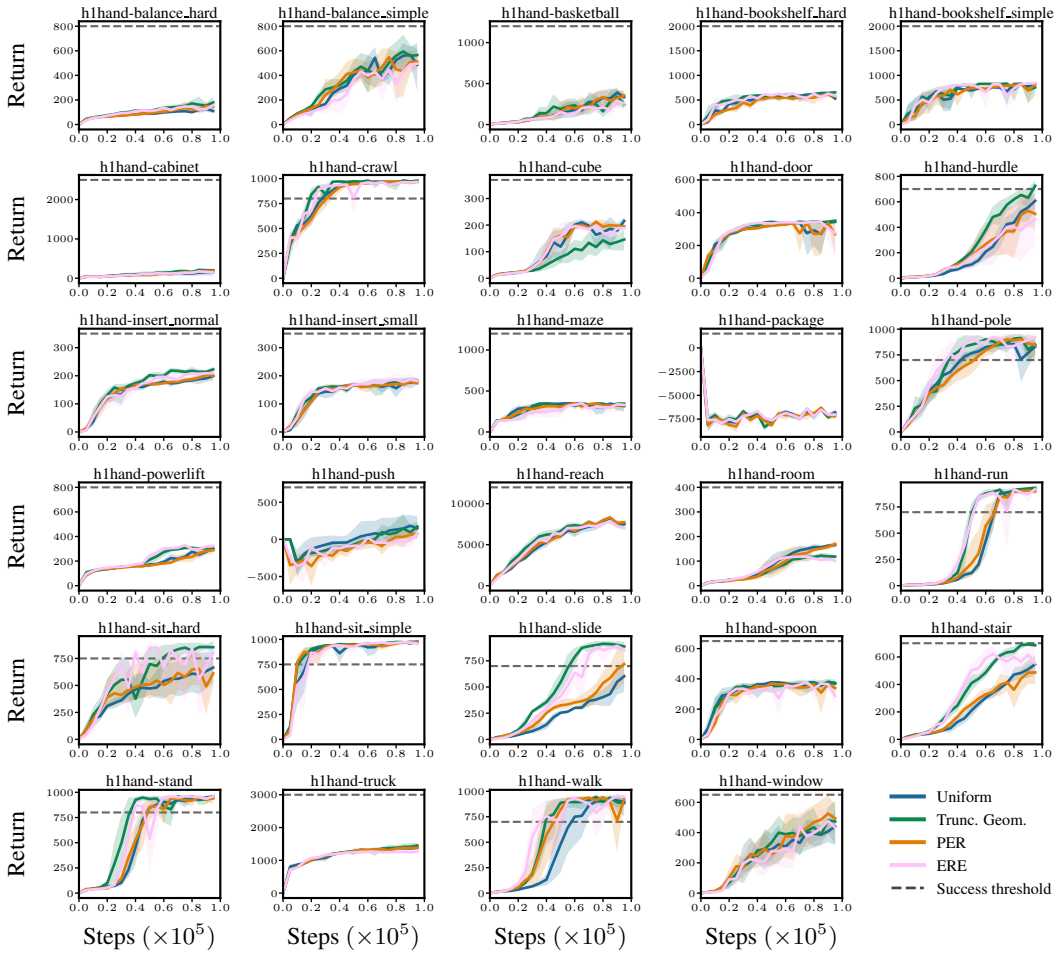

Figure 11: ***(Low Replay Volume) Performance across all 29 tasks.*** We compare FastTD3 [30] with different replay strategies: ***Low expected recency*** - Uniform (blue), PER (orange) and ***High expected recency*** - Truncated Geometric (green), and ERE (pink). Solid lines represent the mean return over 5 seeds, and shaded regions denote the 95% bootstrap confidence intervals computed via `rliable` on HumanoidBench. The dashed grey line in each subplot indicates the task-specific success threshold. All x-axes represent environment steps ($\times 10^7$). On average, Truncated Geometric achieves a 14.1% gain over Uniform in AUC.

F.3 BRC HumanoidBench

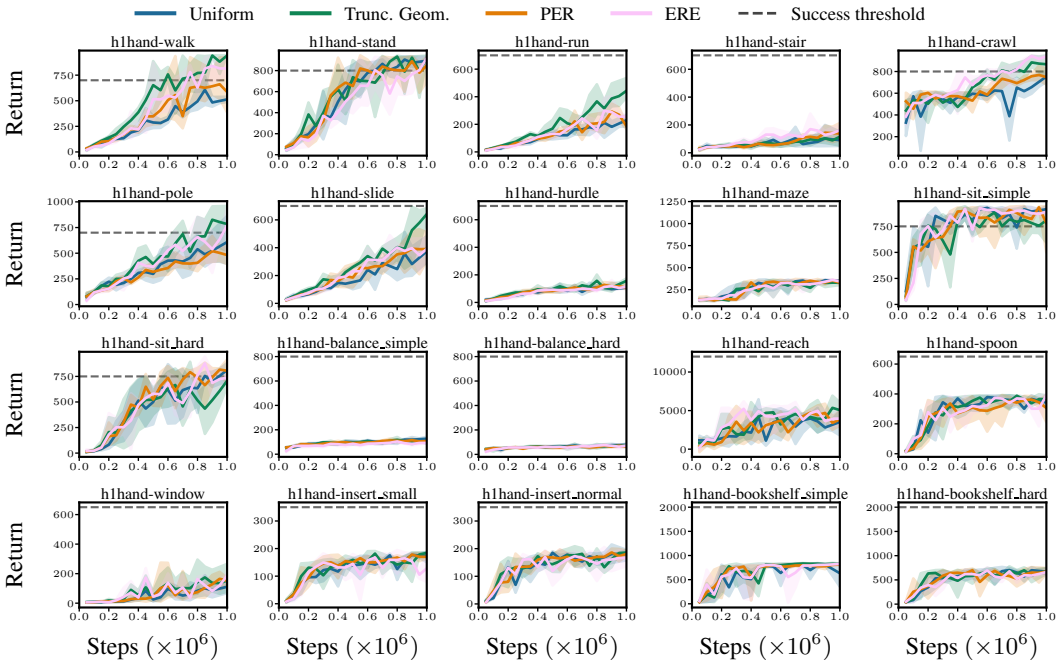

Figure 12: ***(Low Replay Volume)*** *Performance across all 20 BRC tasks.* We compare different replay strategies: ***Low expected recency*** - Uniform (blue), PER (orange) and ***High expected recency*** - Truncated Geometric (green), and ERE (pink). Solid lines represent the mean return over 5 seeds, and shaded regions denote the 95% bootstrap confidence intervals computed via `rliable` on HumanoidBench. The dashed grey line in each subplot indicates the task-specific success threshold. All x-axes represent environment steps ($\times 10^6$). On average, Truncated Geometric achieves a 12.9% gain over Uniform in AUC and approximately 20% improvement in final performance.

F.4 BRC DeepMind Control Suite Humanoids

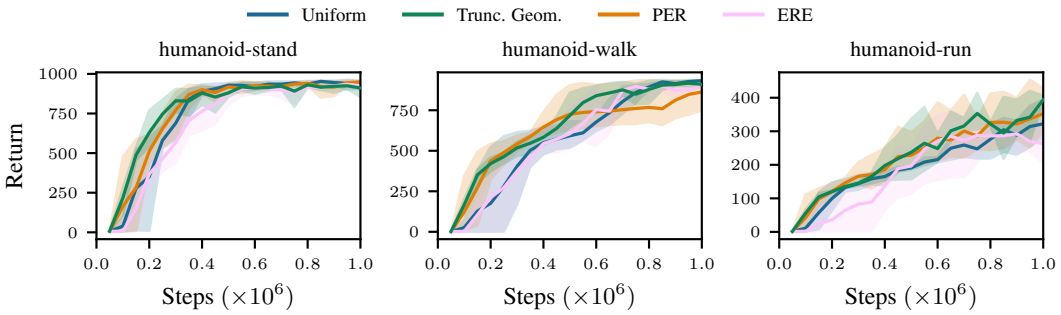

Figure 13: ***(Moderate Replay Volume) Performance on DMC Humanoids tasks [35].*** We compare different replay strategies: ***Low expected recency*** - Uniform (blue), PER (orange) and ***High expected recency*** - Truncated Geometric (green), and ERE (pink) across three humanoid tasks: `stand`, `walk`, and `run`. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns. On average, Truncated Geometric achieves a 10% gain over Uniform in AUC.

F.5 BRC DeepMind Control Suite Dogs

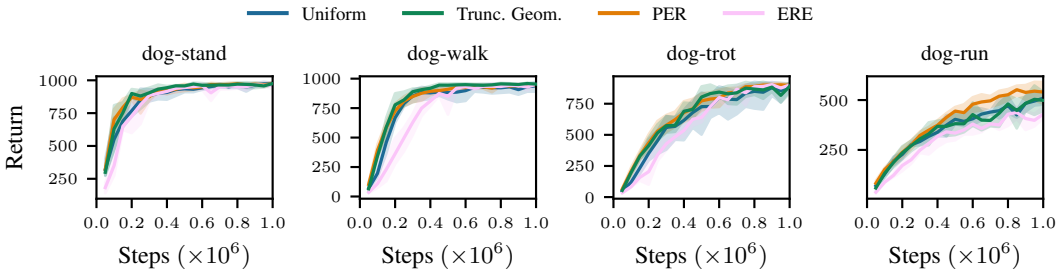

Figure 14: ***(Moderate Replay Volume) Performance on DMC Dog tasks.*** We compare different replay strategies on BRC [23]: ***Low expected recency*** - Uniform (blue), PER (orange) and ***High expected recency*** - Truncated Geometric (green), and ERE (pink) across four dog locomotion tasks: `stand`, `walk`, `trot`, and `run`. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns. On average, Truncated Geometric achieves a 4.2% gain over Uniform in AUC.

F.6 SimbaV2 at default UTD learning curve

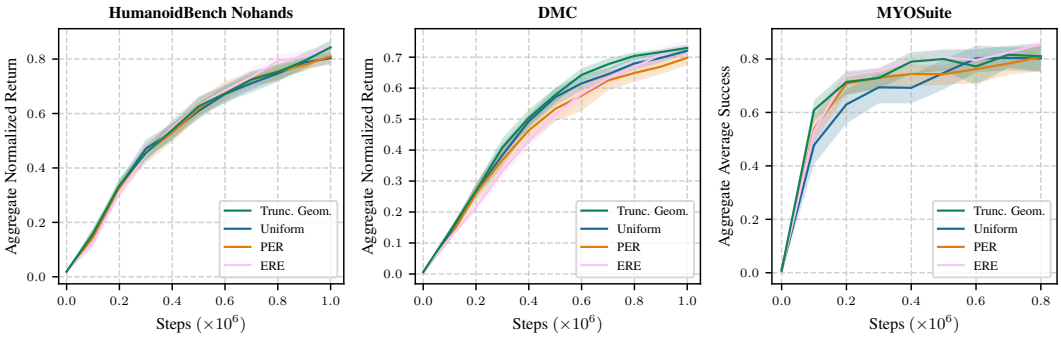

Figure 15: ***(High Replay Volume) Performance on HumanoidBench Nohands tasks, DMC, and MYOSuite.*** We compare different replay strategies on SimbaV2 [18]: ***Low expected recency*** - Uniform (blue), PER (orange) and ***High expected recency*** - Truncated Geometric (green), and ERE (pink). We use UTD=2, standard UTD value taken from original paper on benchmarks Humanoid-Bench Nohands, DMC. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the task returns. Truncated Geometric remains effectively neutral when replay volume is high.

F.7 SimbaV2 UTD

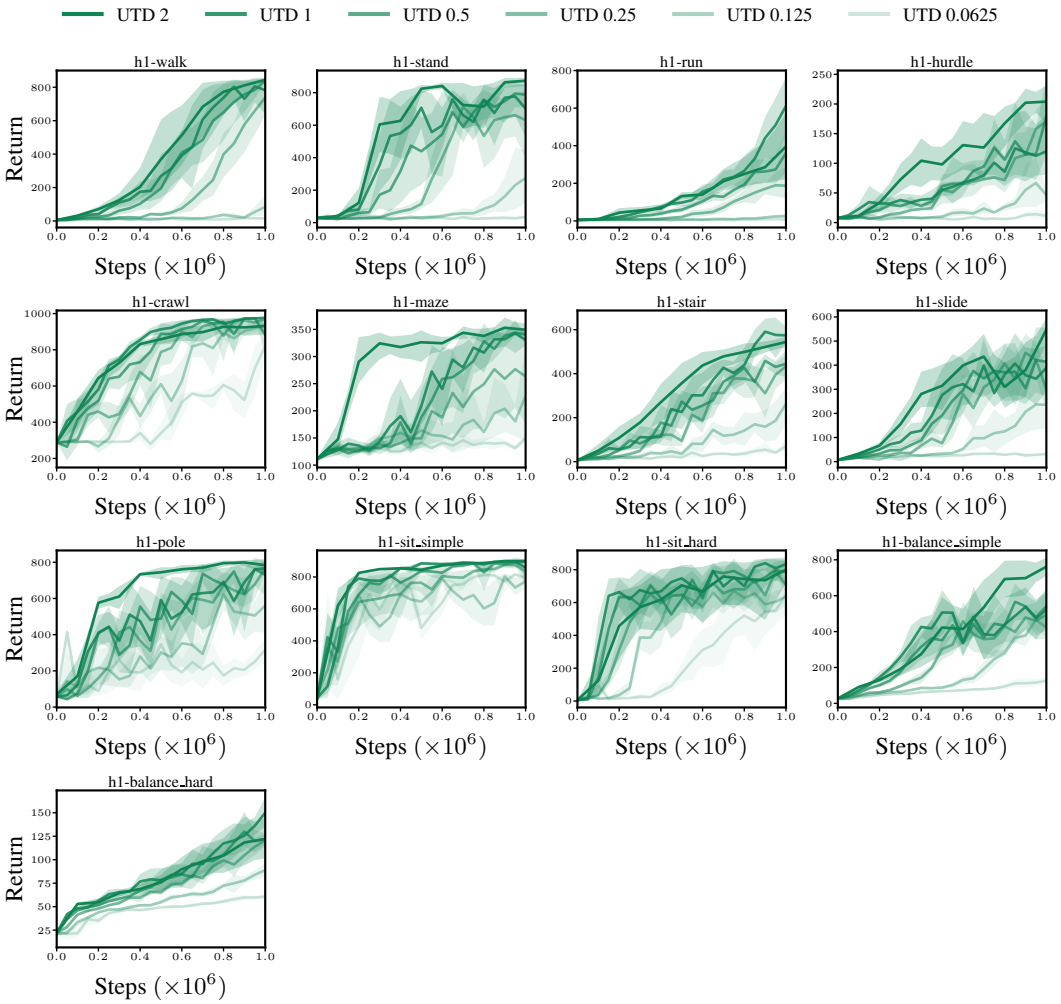

Figure 16: ***(Changing Replay Volume, High Expected Recency) Performance on HumanoidBench Nohands tasks.*** We compare performance of SimbaV2 with Truncated Geometric replay for all UTDs reported in Figure 3. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns.

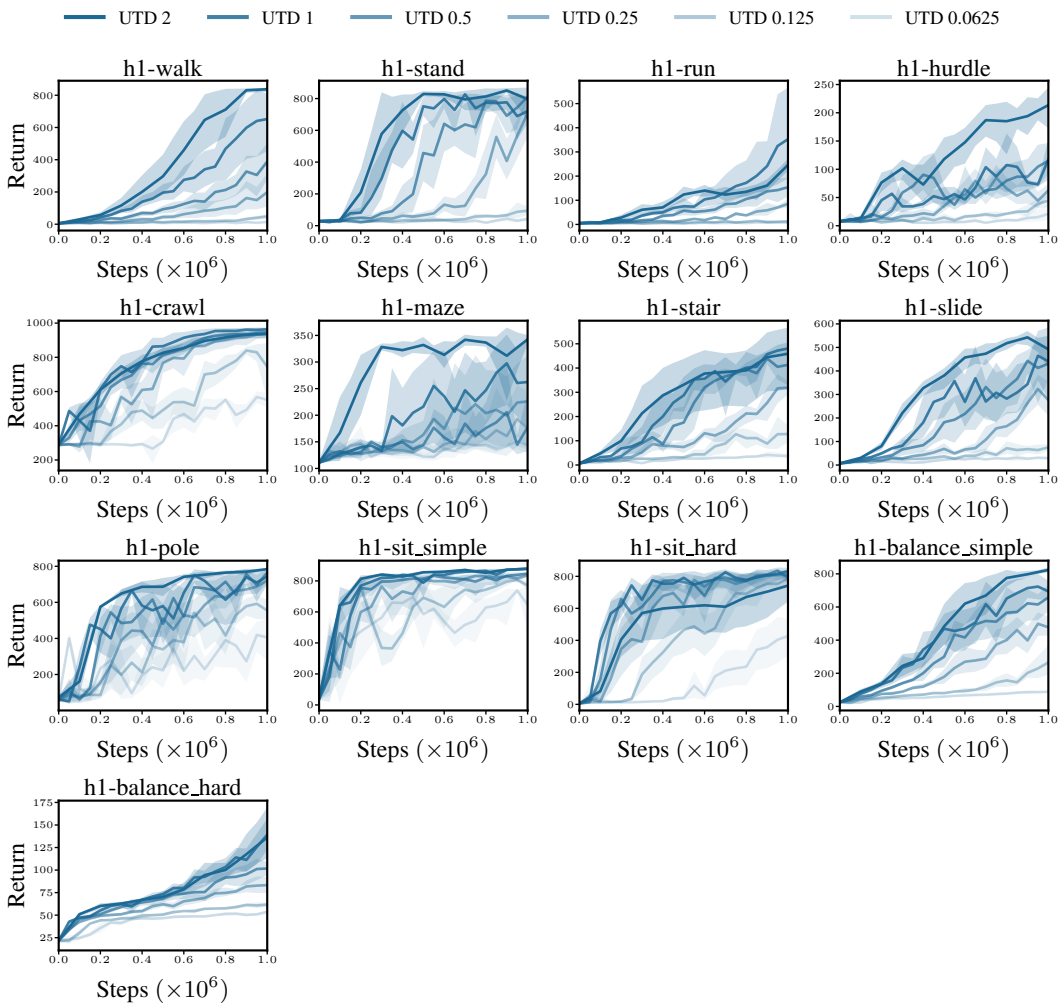

Figure 17: ***(Changing Replay Volume, Low Expected Recency) Performance of SimbaV2 with Uniform replay on HumanoidBench Nohands tasks.*** We report performance for all UTDs from Figure 3. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns.

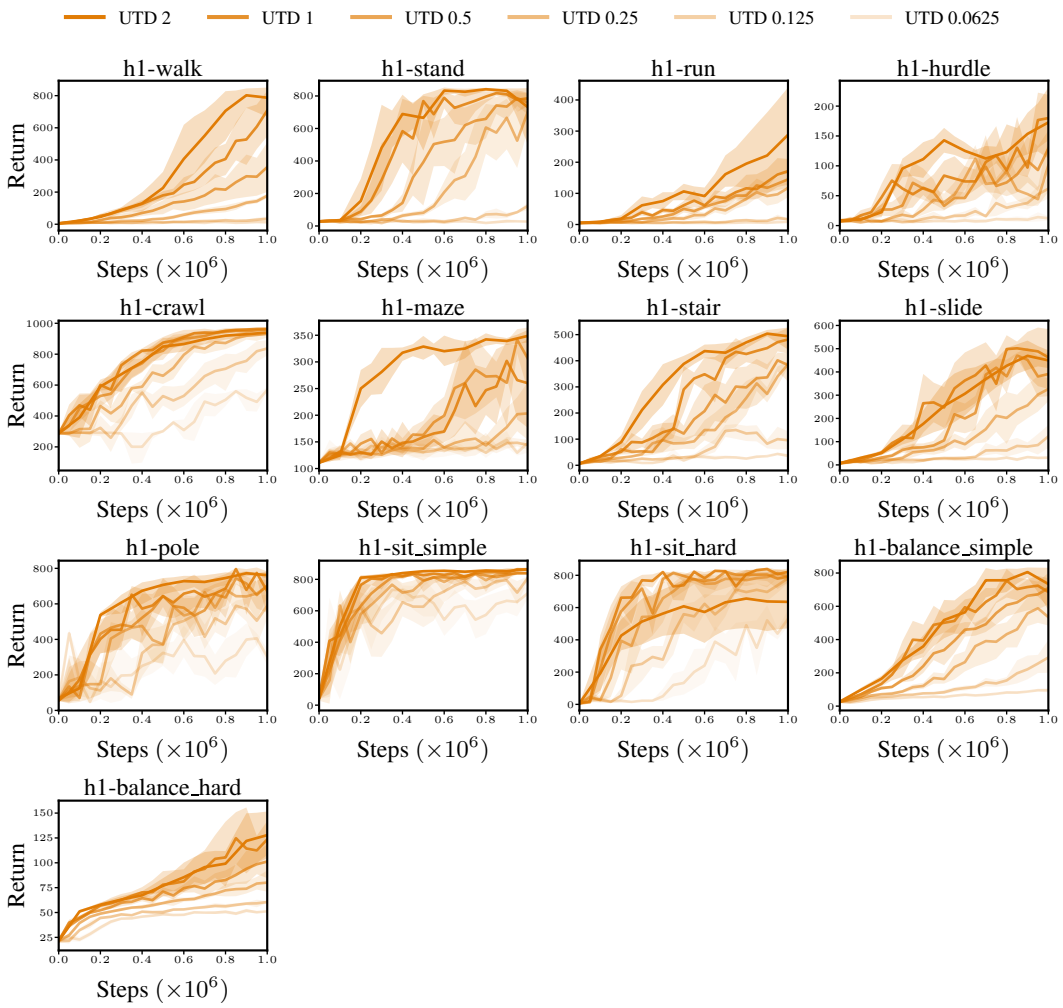

Figure 18: ***(Changing Replay Volume, Low Expected Recency) Performance of SimbaV2 with Prioritized replay on HumanoidBench Nohands tasks.*** We report performance for all UTDs from Figure 10. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns.

F.8 SimbaV2 batch size

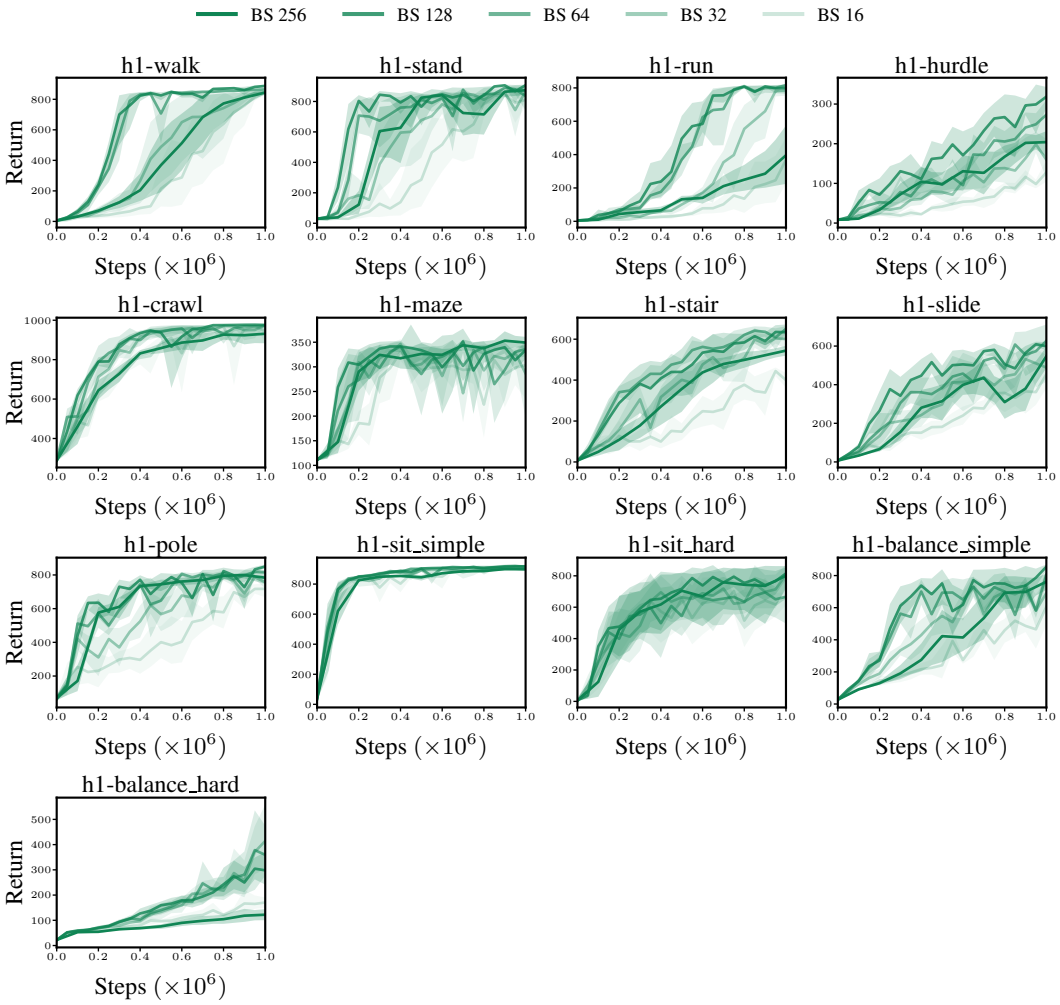

Figure 19: ***(Changing Replay Volume, High Expected Recency) Performance on HumanoidBench Nohands tasks.*** We compare performance of SimbaV2 with Truncated Geometric replay for all batch sizes reported in Figure 3. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns.

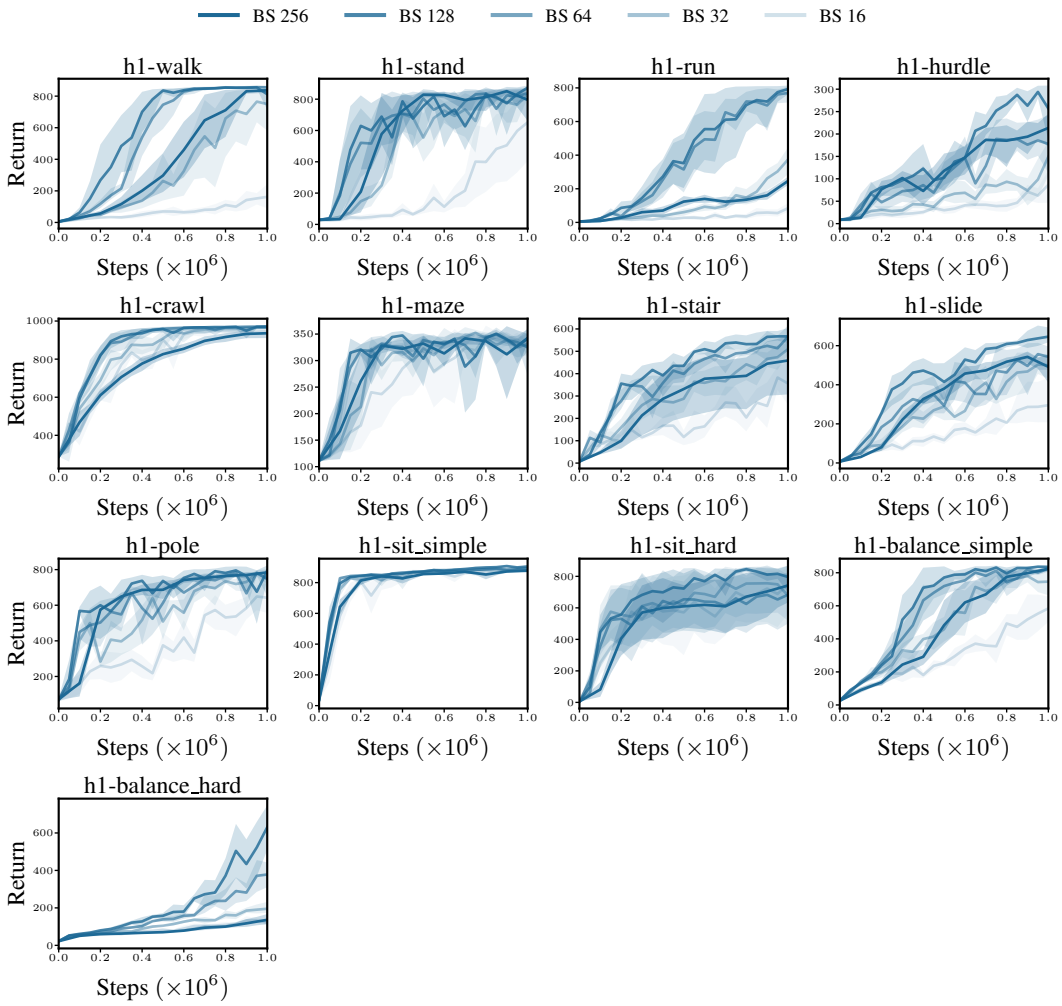

Figure 20: ***(Changing Replay Volume, Low Expected Recency) Performance of SimbaV2 with Uniform replay on HumanoidBench Nohands tasks.*** We report performance for all batch sizes from Figure 3. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns.

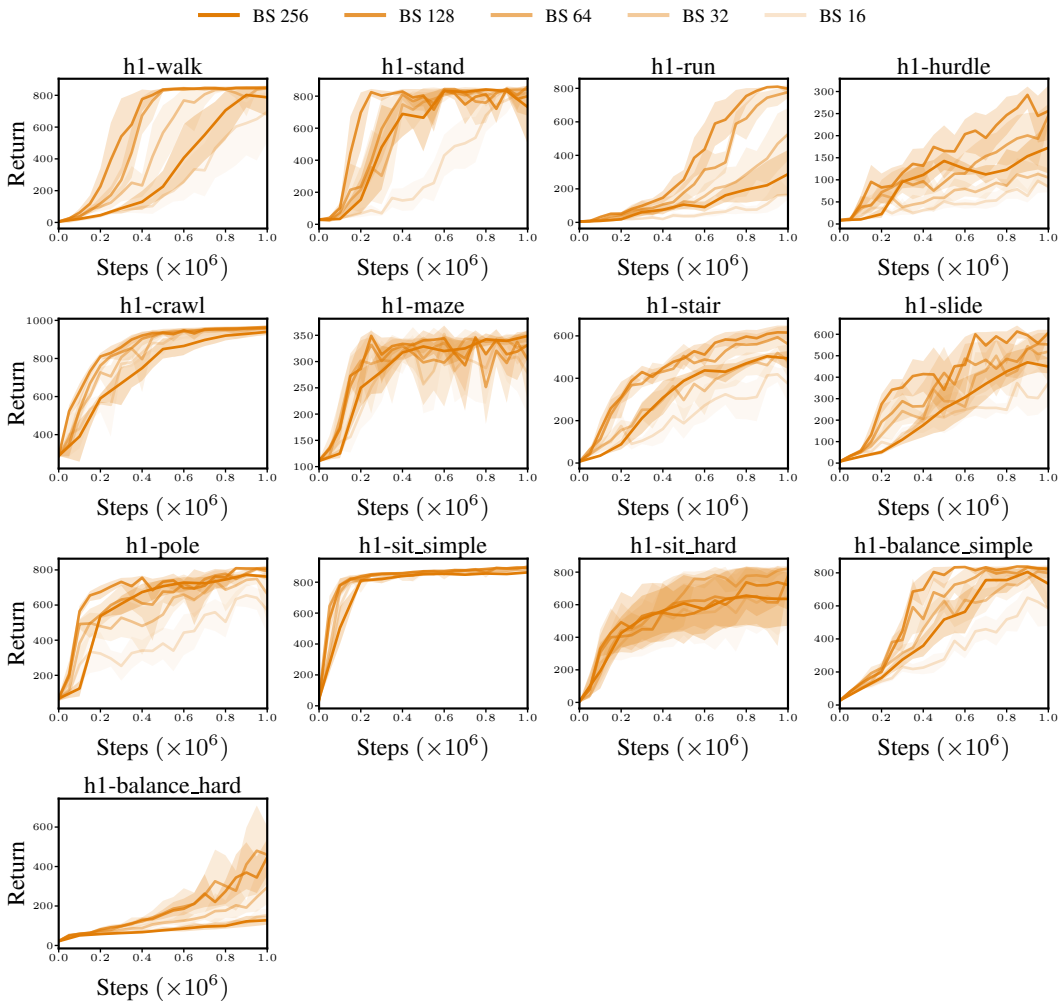

Figure 21: ***(Changing Replay Volume, Low Expected Recency) Performance of SimbaV2 with Prioritized replay on HumanoidBench Nohands tasks.*** We report performance for all batch sizes from Figure 10. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns.

F.9 BRC Meta-World

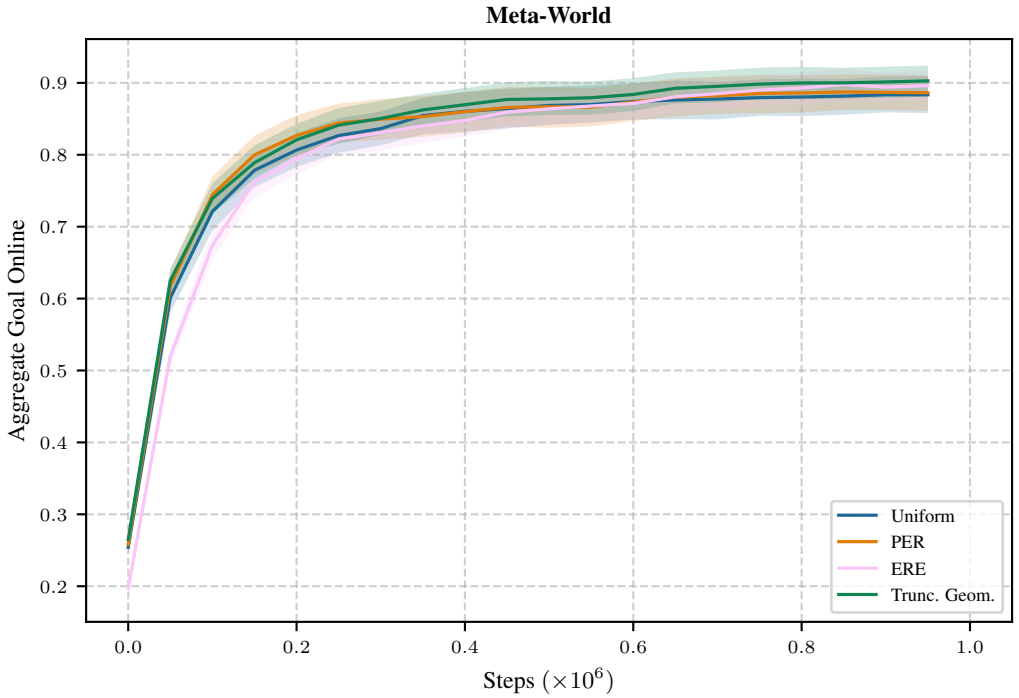

Figure 22: ***(Low Replay Volume) Mean performance of BRC [23] on Meta-World*** with ***Low expected recency*** - Uniform (blue), PER (orange) and ***High expected recency*** - Truncated Geometric (green), and ERE (pink) replay strategies. Solid lines show the mean goal online over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. Truncated Geometric achieves a 2 percentage point improvement over Uniform (90% vs. 88% aggregate success rate), which is noticeable given that Meta-World is largely saturated at this performance level, leaving limited room for improvement.

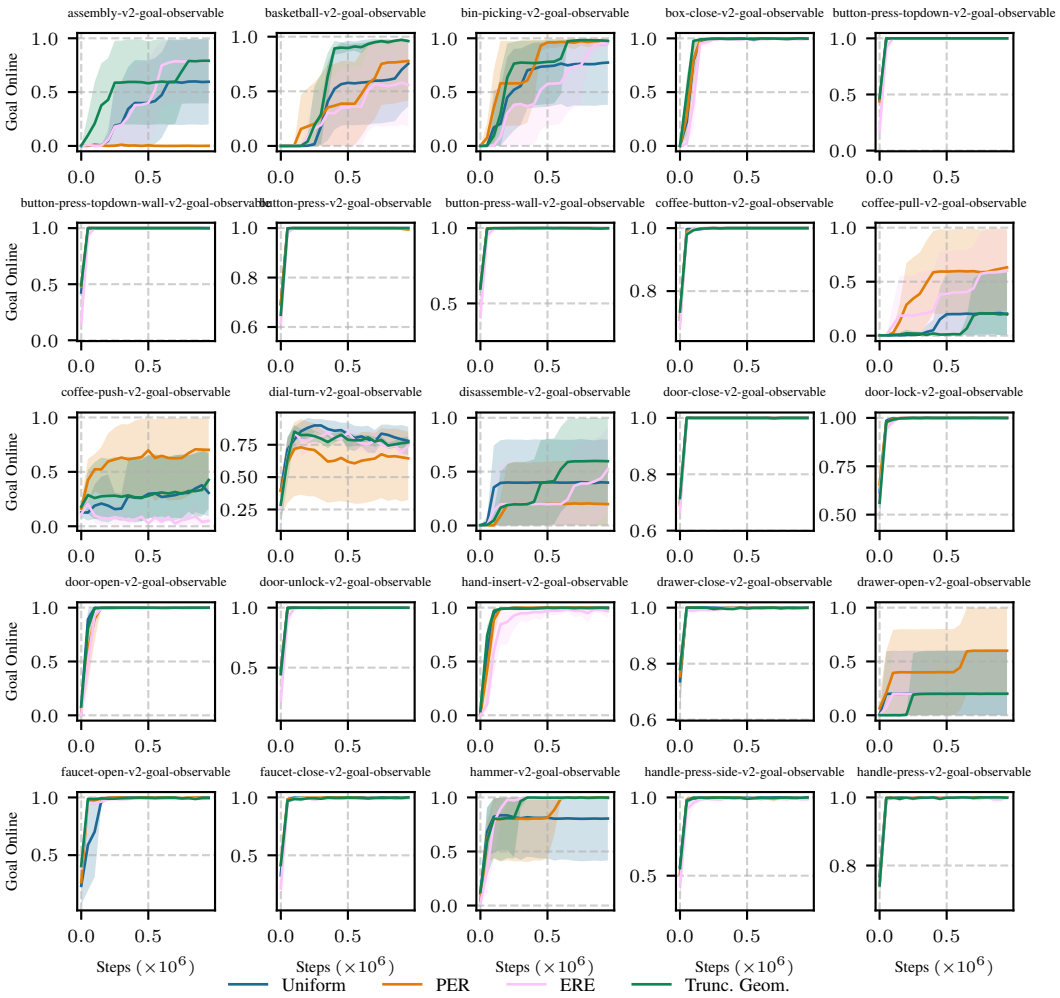

Figure 23: ***(Low Replay Volume) Mean performance of BRC [23] with various replay strategies on Meta-World.*** We use ***Low expected recency*** - Uniform (blue), PER (orange) and ***Low expected recency*** - Truncated Geometric (green), and ERE (pink). Solid lines show the mean goal online over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`.

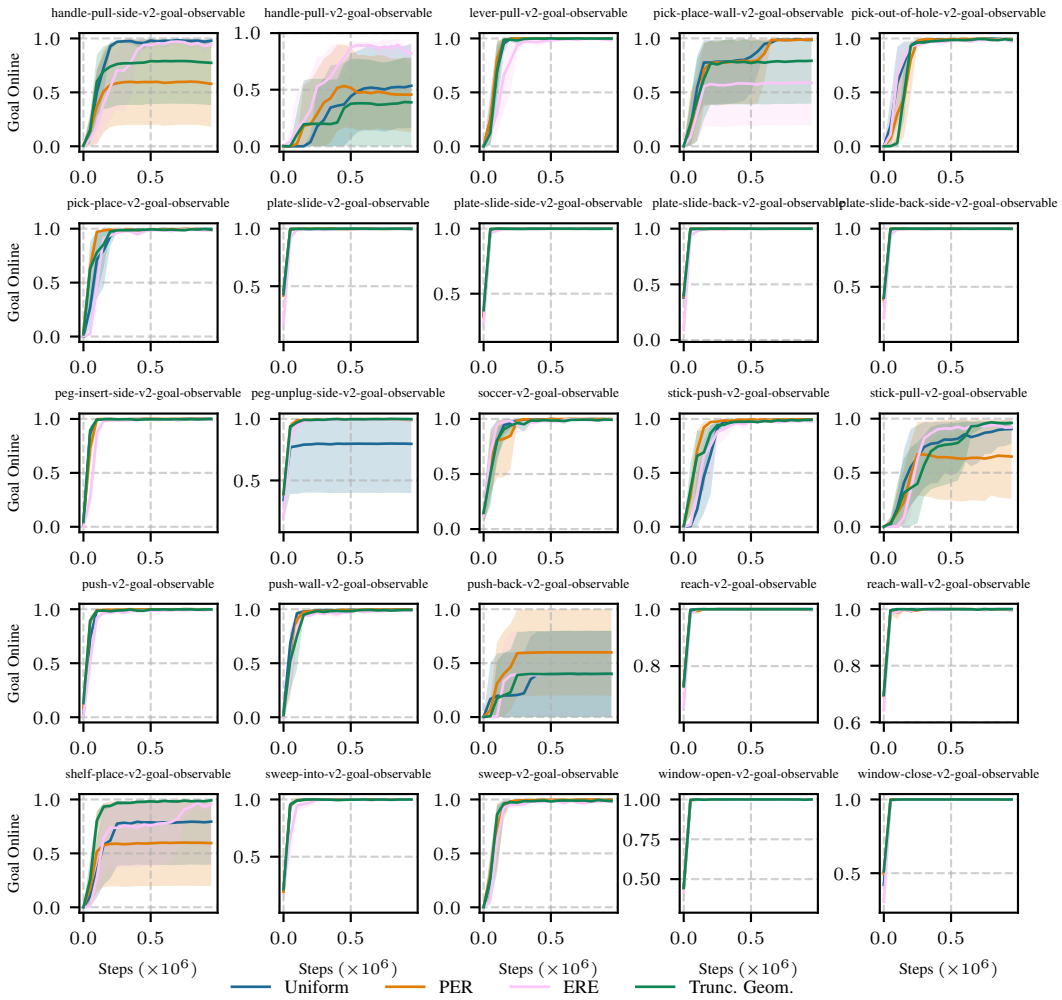

Figure 24: ***(Low Replay Volume) Mean performance of BRC [23] with various replay strategies on Meta-World.*** We use ***Low expected recency*** - Uniform (blue), PER (orange) and ***Low expected recency*** - Truncated Geometric (green), and ERE (pink). Solid lines show the mean goal online over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`.

F.10 FastTD3 Isaac Lab

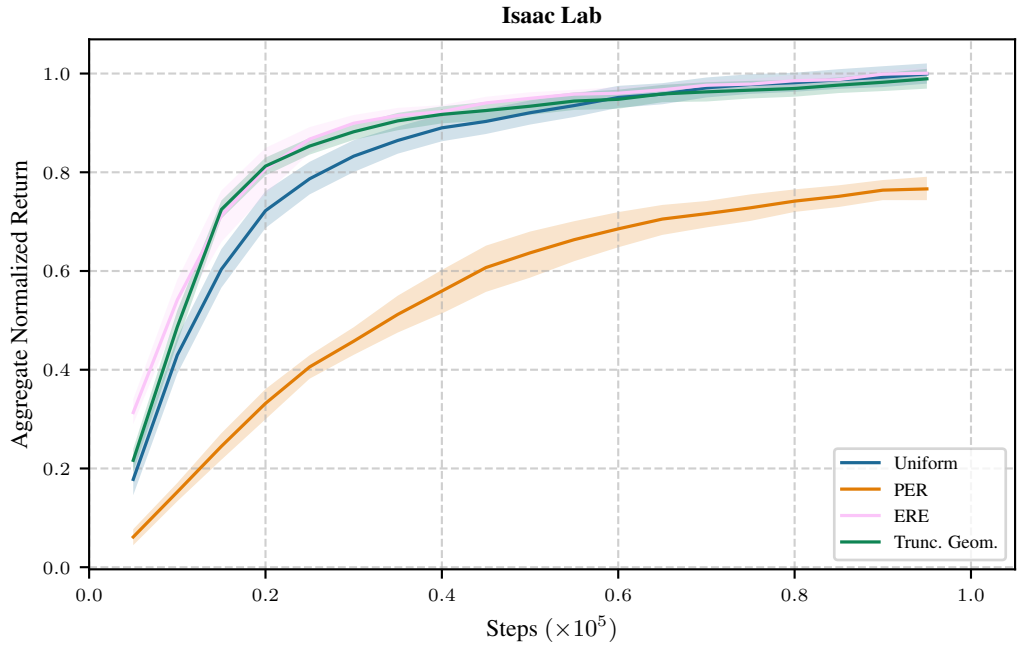

Figure 25: ***(Low Replay Volume, High Expected Recency) Mean performance of FastTD3 [30] with various replay strategies on Isaac Lab.*** In this setting due to low buffer size all replay schemes produce high expected recency replay. Solid lines show mean returns over 5 seeds normalized for each task by dividing returns by mean return from final timestep from uniform sampling strategy. Shaded regions indicate 95% bootstrap confidence intervals computed with `rliable` [1]. Truncated Geometric performs comparably to ERE in aggregate, with ERE benefiting from particularly strong performance on one task.

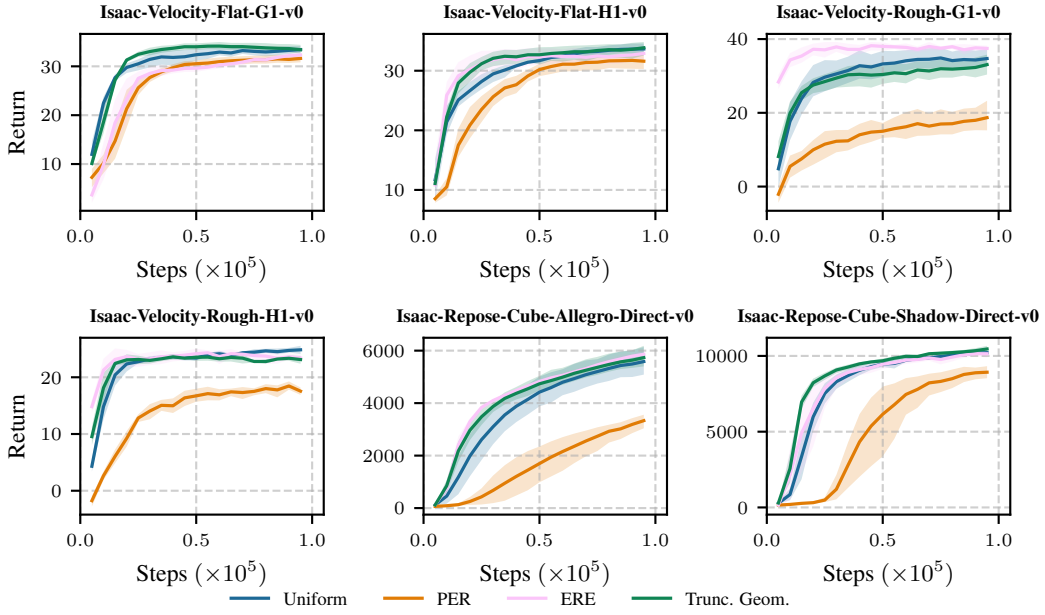

Figure 26: ***(Low Replay Volume, High Expected Recency) Mean performance of FastTD3 on various replay strategies for each Isaac Lab task.*** Solid lines show mean returns over 5 seeds. Shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`.

F.11 DQN Atari

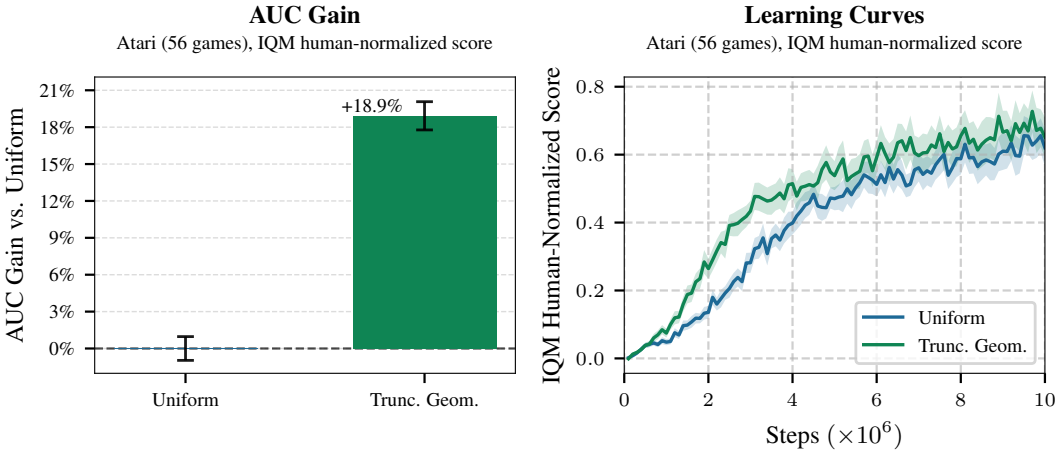

Figure 27: ***(Low Replay Volume) DQN on Atari (56 games): Truncated Geometric vs. Uniform replay.*** *Left:* AUC gain over Uniform, where AUC is the time-average of the IQM learning curve; Truncated Geometric improves AUC by 18.9%. *Right:* IQM human-normalized score over 56 games $\times$ 5 seeds. Following standard practice [1], we report the interquartile mean (IQM), which is robust to outlier games whose normalized scores dominate the mean. Error bars and shaded regions show 95% stratified bootstrap CIs computed with `rliable`.

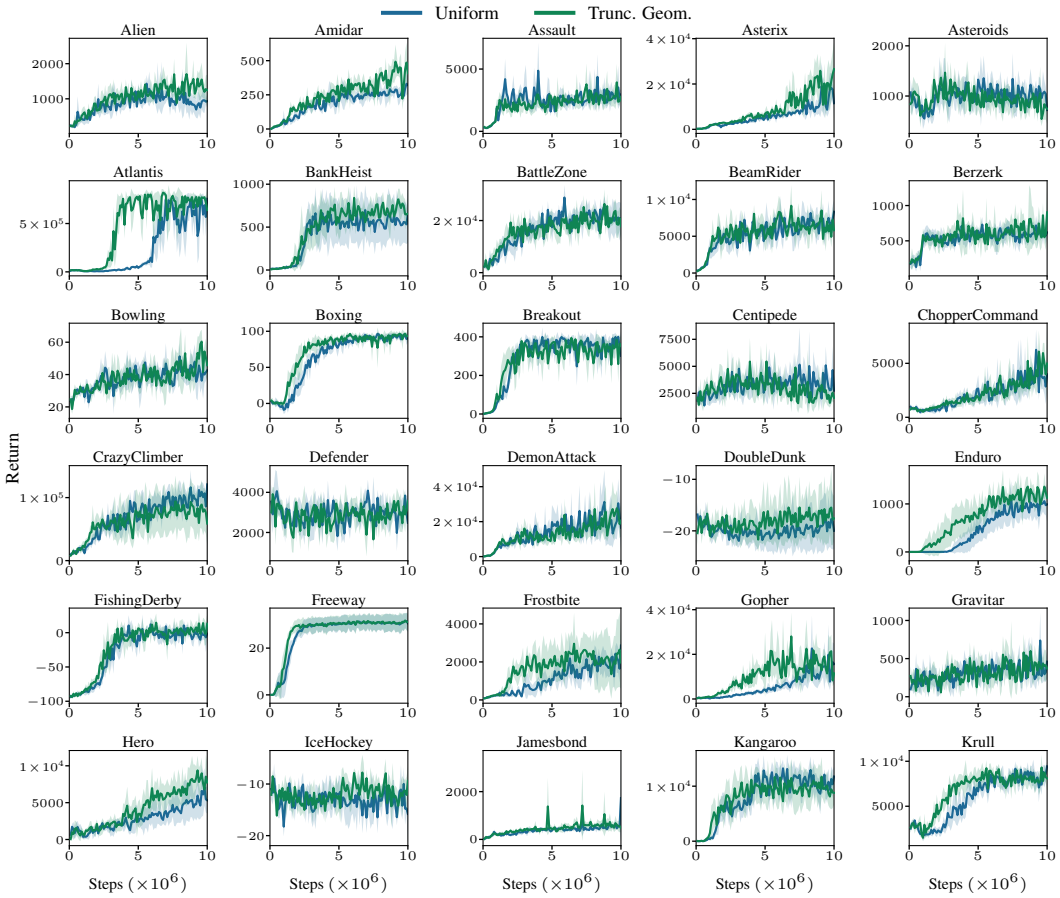

Figure 28: ***(Low Replay Volume) Per-game performance of DQN [21] with two replay strategies on Atari (Part 1).*** We use ***Low expected recency*** - Uniform (blue) and ***High expected recency*** - Truncated Geometric (green). Solid lines show the mean unclipped episodic return over 5 seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`.

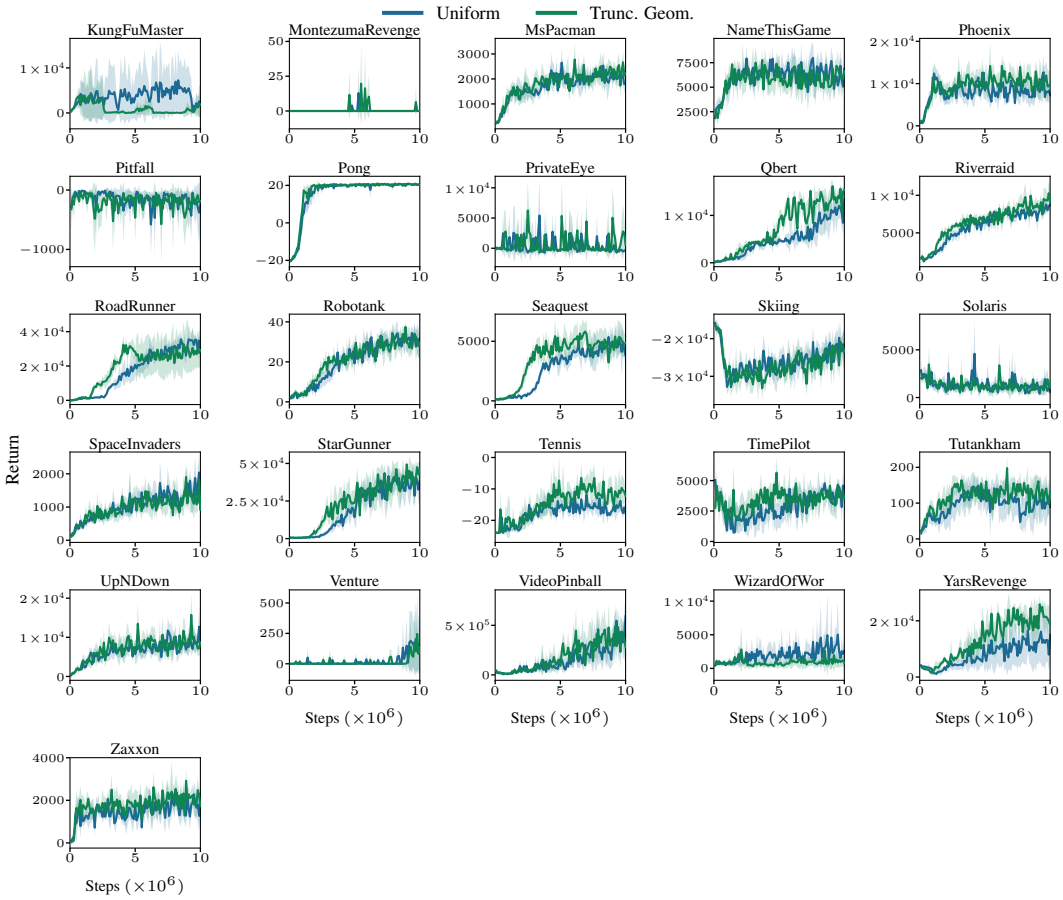

Figure 29: ***(Low Replay Volume) Per-game performance of DQN [21] with two replay strategies on Atari (Part 2).*** We use ***Low expected recency*** - Uniform (blue) and ***High expected recency*** - Truncated Geometric (green). Solid lines show the mean unclipped episodic return over 5 seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`.

F.12 Constant replay volume on SimbaV2

UTD	Batch size	Uniform AUC	TG AUC	Gap	AUC gain
2	32	0.528 [0.522, 0.534]	0.574 [0.567, 0.580]	+0.045	+8.6% [+6.1, +11.3]
1	64	0.338 [0.333, 0.344]	0.381 [0.374, 0.387]	+0.042	+12.4% [+8.8, +16.2]
$\frac{1}{2}$	128	0.367 [0.362, 0.372]	0.402 [0.395, 0.408]	+0.035	+9.5% [+6.2, +12.9]
$\frac{1}{4}$	256	0.326 [0.321, 0.331]	0.377 [0.372, 0.383]	+0.052	**+15.9%** [+12.3, +19.5]

Table 3: **Matched replay volume.** AUC of SimbaV2 with Uniform and Truncated Geometric (TG, $\alpha = 10$) sampling on HumanoidBench (13 tasks, 5 seeds each), at four UTD/batch-size settings with the same number of replayed samples per environment step (UTD $\times$ BS $= 64$). AUC is the normalized return averaged over evaluations up to 1M agent steps. Gap = TG $-$ Uniform; AUC gain is relative to Uniform. Brackets show 95% stratified bootstrap CIs. Action repeat equals the UTD, so the UTD 2 setting repeats each action twice.

F.13 Stochastic Action SimbaV2

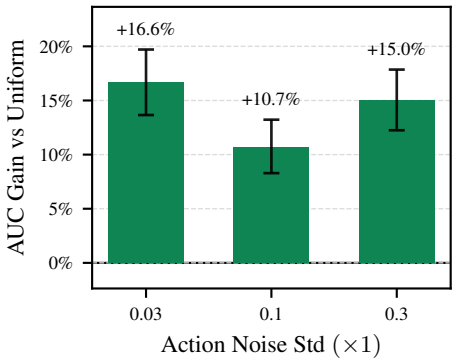

Figure 30: ***(Low Replay Volume) Noisy SimbaV2 on HumanoidBench (13 tasks).*** Improvement of Truncated Geometric replay over uniform replay at UTD $= 1/4$, BS $= 256$ in stochastic env SimbaV2. Stochasticity is introduced by adding zero-mean Gaussian noise with a given standard deviation to every action before it is executed in the environment (actions are normalized to $[-1, 1]$ and clipped back to that range after the noise is added). We report a gain as a ratio of normalized AUCs from normalized returns over 3 seeds for each method. Error bars show 95% stratified bootstrap CIs computed with `rliable`.

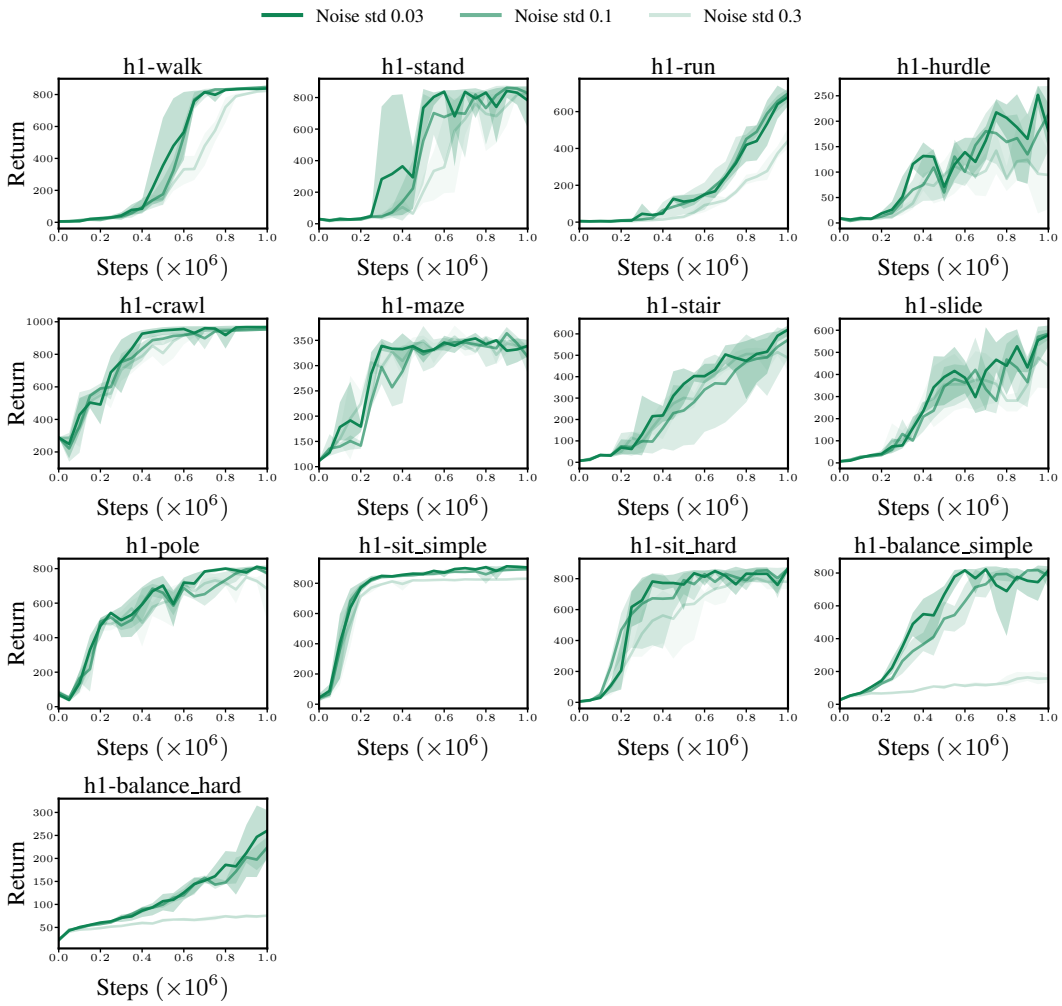

Figure 31: ***(Low Replay Volume, High Expected Recency) Performance of noisy SimbaV2 with Truncated Geometric replay on HumanoidBench Nohands tasks.*** We report performance for all noise stds from Figure 30. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns.

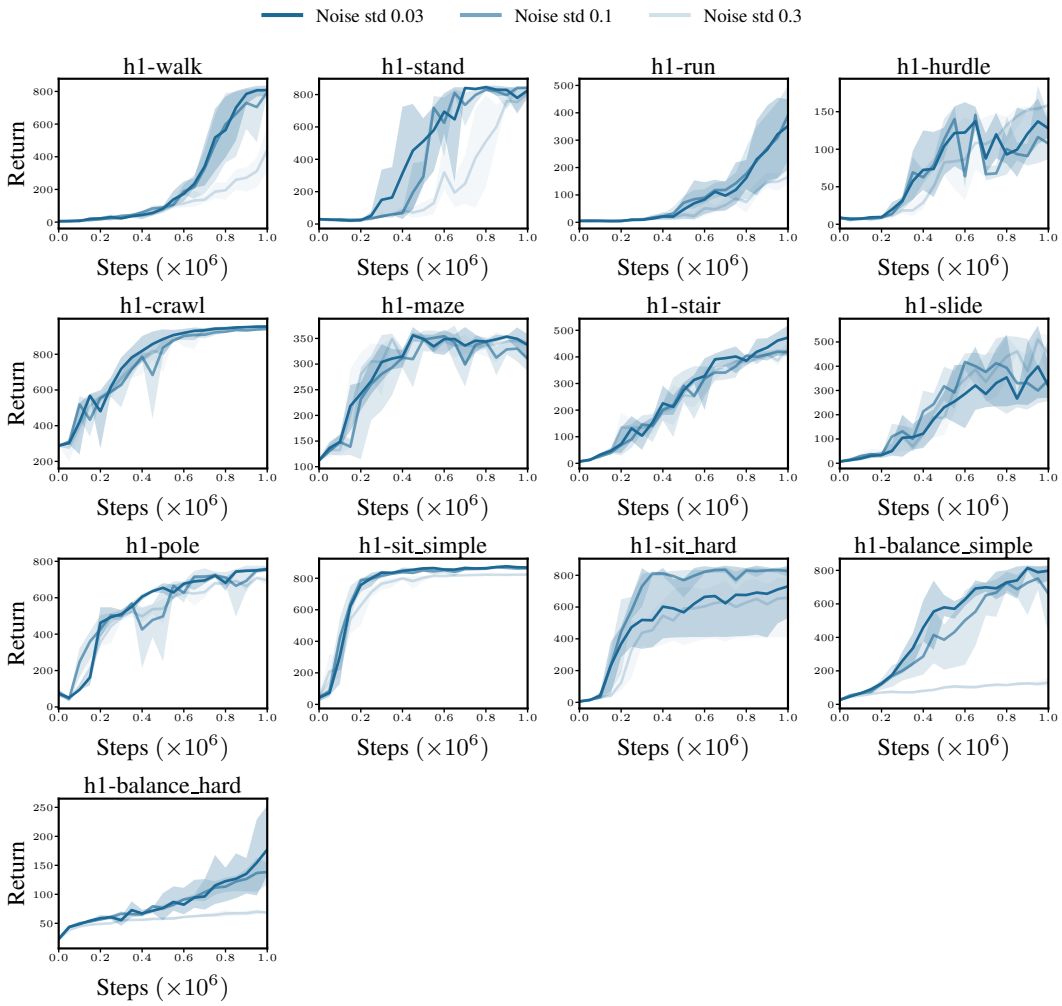

Figure 32: ***(Low Replay Volume, Low Expected Recency) Performance of noisy SimbaV2 with uniform replay on HumanoidBench Nohands tasks.*** We report performance for all noise stds from Figure 30. Solid lines show the mean return over all available seeds for each method, and shaded regions indicate 95% bootstrap confidence intervals computed with `rliable`. All x-axes represent environment steps ($\times 10^6$), and y-axes show the denormalized task returns.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's contributions and scope, as detailed in Section 1 and further supported by the experimental results in Section 3 and Section 5.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Limitations are discussed in the appendix A, acknowledging that experiments focus on continuous control settings and identifying a formal theoretical framework as a promising direction for future work.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper is empirical and does not include novel theoretical results; the Truncated Geometric Pareto frontier referenced in Section 4 is a known result with citations provided.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: All information needed to reproduce the experiments is provided, including the algorithms used, benchmark environments (Section 5), pseudocode of the proposed method (Section 4), and hyperparameters detailed in the appendix C.2.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: Code is provided in the supplementary material and experiments are conducted on publicly available open-source reinforcement learning environments.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Full training details, hyperparameters, evaluation protocols, and metrics are provided in the appendix D.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: All results report 95% confidence intervals computed via stratified bootstrap resampling over runs following the RLiable [1] protocol, with the aggregation procedure detailed in the appendix E.1.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Computational resources are detailed in Appendix C.5, including GPU type and estimated runtimes for single-task, large scale parallel and multi-task experiments.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research fully conforms with the NeurIPS Code of Ethics. It involves no human subjects, sensitive data, or foreseeable harmful applications.

   Guidelines:

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

10. **Broader impacts**

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

Answer: [N/A]

Justification: The paper presents foundational research on replay sampling strategies for reinforcement learning, with no direct path to negative societal impacts or specific real-world deployment concerns.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper releases only reinforcement learning code and poses no risk of misuse; no safeguards are required.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: All benchmarks used are open-source and properly cited in Section 5; all models are trained from scratch with no additional assets requiring licensing.

Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The code released alongside the paper is documented and provided as an anonymized zip file in the supplementary material.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

Justification: The paper does not involve human subjects and therefore requires no IRB approval.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: LLMs are not used as part of the core methodology; any use was limited to writing and editing purposes only.

Guidelines:

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