# WHAT GOVERNS LEARNING DYNAMICS IN MASSIVELY PARALLEL OFF-POLICY RL?

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

## ABSTRACT

GPU-accelerated simulators enable data collection from thousands of environments in parallel for off-policy RL. This parallelism increases data throughput, but also changes the dynamics of the replay buffer distribution during the training. As the replay buffer has finite capacity, increasing data throughput causes its samples to be replaced more rapidly. In this study, we analyze the learning dynamics of reinforcement learning with massively parallel environments. Specifically, we treat the replay buffer as a temporal mixture of data distributions under changing policies and distinguish the true distribution shape from the sample density. We also identify two sources of gradient variance, one induced by replay buffer construction from finite rollout samples and the other by mini-batch sampling from the replay buffer. Our empirical results highlight the importance of increasing the replay buffer size and batch size as the number of parallel environments grows, and reveal regimes in which additional parallelism yields little benefit. These findings provide practical guidance for choosing replay buffer and mini-batch sampling hyperparameters in massively parallel off-policy RL.

## 1 INTRODUCTION

GPU-accelerated simulators have made it practical to collect experience from thousands or even tens of thousands of environments in parallel (Makoviychuk et al., 2021; Freeman et al., 2021; Mittal et al., 2025; Zakka et al., 2025). This capability has enabled high-throughput off-policy reinforcement-learning systems such as PQL, FastTD3, and FlashSAC (Li et al., 2023; Seo et al., 2025; Kim et al., 2026), which are increasingly used to train and evaluate robotic control policies in simulation. Environment parallelism is often viewed primarily as a systems mechanism for accelerating data collection. In off-policy learning, however, changing the number of environments also changes how experience from successive policies is represented in the replay buffer. Scaling parallelism may therefore alter the learning dynamics rather than merely execute the same learning process faster.

This raises two related questions. First, which property of replay data governs the expected learning dynamics as environment parallelism is scaled? Second, even when that property is preserved, what determines whether learning remains consistent across finite scales? Answering these questions requires considering the number of environments, rollout length, replay buffer size, Update-to-Data (UTD) ratio, and mini-batch size jointly. Changing any one of them can affect both the distribution of replay data and the accuracy with which that distribution is represented.

Our key insight is that learning dynamics are governed by two properties of replay data. One is *temporal composition*, the range of past online policies represented in the buffer, determines the expected learning trajectory. The other is *finite-sample resolution*, how densely that policy history is represented by collected experience, determines scale-dependent deviations from that trajectory. We characterize the replay buffer distribution as a temporal mixture of distributions induced by successive policies and identify a time scale, determined by training hyperparameters, that controls its temporal extent. Scaling environment parallelism and replay buffer size together can preserve this extent while increasing its sampling density. Nevertheless, finite-scale learning may differ because gradients are estimated from a finite buffer and stochastic mini-batches. We separate these sampling errors and predict that learning trajectories should converge as finite-buffer gradient variance decreases, with the benefits of further parallelism eventually saturating.



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

We test these predictions using PQL (Li et al., 2023) on IsaacGym (Makoviychuk et al., 2021) CartPole and Ant, FastTD3 (Seo et al., 2025) and FlashSAC (Kim et al., 2026) on humanoid tasks in IsaacLab (Mittal et al., 2025) and MuJoCo Playground (Zakka et al., 2025). Across these settings, learning curves with the same temporal composition approach common dynamics as scale increases, while finite-buffer gradient variance decreases and the benefits of additional parallelism saturate.

**Main contributions** In light of the above, we highlight the main contributions of the paper.

- **Replay-buffer distribution.** We formalize the replay-buffer distribution as a temporal mixture of data distributions induced by successive online policies and characterize how the main hyperparameters determine the extent of policy history represented in the buffer.
- **Sample density and limits of parallel scaling.** We theoretically and empirically analyze how finite sample effects in the replay buffer and mini-batches shape gradient noise, and show how they govern cross-scale consistency and the limits of parallel scaling.
- **Hyperparameter guideline.** We identify which hyperparameters transfer across parallel scales and provide a practical tuning guideline for massively parallel off-policy RL, grounded in both our analysis and experiments.

## 2 Related Work

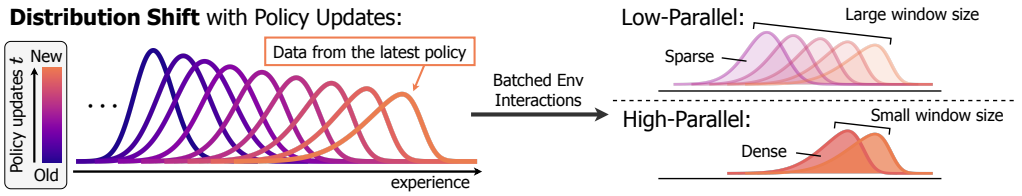
**RL with Highly Parallelized Environments.** Early distributed RL primarily relied on multi-CPU architectures, including the on-policy A3C (Mnih et al., 2016) and off-policy methods such as Ape-X (Horgan et al., 2018), D4PG (Barth-Maron et al., 2018), R2D2 (Kapturowski et al., 2019), and IMPALA (Espeholt et al., 2018). More recently, GPU-accelerated simulators such as IsaacGym (Makoviychuk et al., 2021), IsaacLab (Mittal et al., 2025), and MuJoCo Playground (Zakka et al., 2025) have made it practical to run tens of thousands of environments in parallel on a single GPU. This capability has been explored by PQN (Gallici et al., 2025), PQL (Li et al., 2023), FastTD3 (Seo et al., 2025), FlashSAC (Kim et al., 2026), and SEM (Obando Ceron et al., 2026) in off-policy learning. Complementary work has improved distributed learning through asymmetric architectures, as in SAPG (Singla et al., 2024) and CPO (Shitanda et al., 2026), and through multi-GPU systems such as GEAR (Wang et al., 2023) and TianJi (He et al., 2025).

**UTD Ratios in Off-policy RL.** Off-policy methods differ substantially in their Update-to-Data (UTD) ratio, defined as the number of gradient updates per environment interaction. While DQN (Mnih et al., 2013) uses a UTD ratio of 1 or lower, REDQ (Chen et al., 2021) and DroQ (Hiraoka et al., 2022) support ratios of around 20 through Q-function ensembles or dropout. Other approaches improve learning at high UTD ratios through periodic resets in SR-SAC (D'Oro et al., 2023), or larger networks in BBF (Schwarzer et al., 2023). At the other extreme, high-throughput methods such as PQL (Li et al., 2023) and FastTD3 (Seo et al., 2025) use ratios as low as $10^{-3}$, and FlashSAC (Kim et al., 2026) similarly adopts a low UTD ratio at scale. Low-UTD training is also used outside massively parallel settings to reduce gradient-computation costs in a variant of BRO (Nauman et al., 2024), improve efficiency through batch normalization in CrossQ (Bhatt et al., 2024), or alternate online collection with high-UTD offline learning in SPEQ (Romeo et al., 2025). Prior work has characterized the trade-off between data collection and model updates (D'Oro et al., 2023) and modeled them separately using UTD-dependent power laws (Rybkin et al., 2025; Fu et al., 2025), but has not systematically examined how the number of environments, replay buffer size, UTD ratio, and batch size jointly determine the composition of the replay buffer and the resulting learning dynamics.

## 3 Preliminaries and Notation

**Off-policy RL** We consider a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor.

A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps states to probability distributions over actions. The objective is to find the optimal policy $\pi^*$ that maximizes the expected cumulative discounted reward $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$.




**(a) True Online Distributions:** $P_{\text{online}}(e \mid \pi_t)$ **(b) Buffer Distributions:** $P_{\text{buffer}}(e \mid \pi_t)$

Figure 1: **Buffer Distribution with Different Environment Parallelisms.** With a fixed buffer size in low-parallel environments, the buffer distribution is formed by sparsely sampled data from largely changing policies. In contrast, high-parallel environments produce a distribution of densely sampled data from recent policies. This is represented by the UTC ratio, defined as the product of the replay buffer size and the UTD ratio, acting as the window size.

Off-policy reinforcement learning algorithms learn a target policy while following a potentially different behavior policy during data collection. These algorithms maintain a replay buffer $\mathcal{D}$ that stores experiences $(s, a, r, s')$ collected from environment interactions. The key advantage of off-policy methods is their ability to reuse past experiences, leading to improved sample efficiency.

**Parallel Data Collection** In distributed off-policy learning, multiple environments run in parallel, and data from each environment is stored in a common replay buffer. Let $E$ denote the number of parallel environments and $L$ denote the rollout length (i.e., the number of environment steps collected per model update). Thus, $EL$ samples are collected per model update.

There are several ways to express the balance between exploration and model updates. The Update-to-Data (UTD) ratio (Chen et al., 2021) is defined as $\sigma = 1/(EL)$ and represents the number of network updates per collected sample. The replay ratio is defined as $\sigma B = B/(EL)$ and represents the number of samples used for network updates per collected sample.

## 4 Buffer Distribution and Training Dynamics in Parallel Off-Policy RL

### 4.1 Replay Buffer Distribution

In distributed off-policy RL, multiple environments operate in parallel, and data collected is stored in a shared replay buffer. Let $E$ denote the number of parallel environments and $L$ denote the rollout length, the number of environment steps collected per model update. We define one cycle as the training steps required to completely replace the replay buffer of size $R$, so that exactly $R$ new samples are collected within a cycle. The UTD ratio $\sigma = 1/(EL)$ (Chen et al., 2021) gives the number of updates per collected sample, and the Update-to-Cycle (UTC) ratio $\tau = R\sigma$ gives the number of updates performed within one cycle:

$$\tau = R \cdot \sigma = \frac{R}{EL} \tag{1}$$

$R$ and $\tau$ thus describe the same cycle in two units, samples and updates, with $\sigma$ the rate converting between them. Since the replay buffer accumulates experiences collected from recent exploration policies, the distribution $P_{\text{buffer}}$ of data $e$ within the buffer is formed as a mixture of varying online distributions $P_{\text{online}}$. We show some additional properties regarding replay buffer distribution in Appendix A.The buffer distribution at training step $t$ corresponds to the accumulated data sample count $u = t/\sigma$ (where $t > \tau$ and $u > R$)

$$P_{\text{buffer}}(e \mid \pi_t) = \frac{1}{R} \sum_{u'=u-R+1}^{u} P_{\text{online}}(e \mid \pi_{\sigma u'}) = \frac{1}{\tau} \sum_{t'=t-\tau+1}^{t} P_{\text{online}}(e \mid \pi_{t'}) \tag{2}$$

Thus, $\tau$ determines the temporal extent of policy history represented in the replay buffer. Different combinations of $E$, $L$, and $R$ that yield the same $\tau$ induce the same population replay distribution in the infinite-sample limit. Since the population replay distribution determines the expected gradient

at a given model state, these configurations also induce the same expected learning dynamics in that limit when the remaining training conditions are fixed.

In conventional single-environment training with one environment step per update ($E = L = 1$), $\tau = R$. Increasing $E$ with $R$ and $L$ fixed decreases $\tau$ in inverse proportion to $E$. As illustrated in Figure 1, this changes the composition of the buffer. Non-parallel buffers contain data sparsely sampled from a long history of policies, while parallel buffers consist of data densely sampled from recent distributions. Ape-X (Horgan et al., 2018) and PQL (Li et al., 2023) implicitly adopt a small $\tau$ by maintaining standard buffer sizes under high parallelism, while FastTD3 (Seo et al., 2025) treats $\tau$ explicitly as a hyperparameter by scaling buffer size with parallelism. These approaches suggest that updating the model using data closer to the current policy is advantageous in data-rich regimes.

**Key Observations on *Replay Buffer Distribution***

- **Buffer as a $\tau$-window mixture**: The buffer distribution mixes online distributions over a window of width $\tau$. Different combinations of $E$, $L$, and $R$ that yield the same $\tau$ induce the same population replay buffer distribution.
- **Consistent learning dynamics**: The same population replay buffer distribution gives the same expected gradient and learning dynamics when the other training conditions are fixed.

### 4.2 Sample Density and Gradient Variance

Equation 2 describes the population replay distribution. Actual training uses a finite empirical buffer and a mini-batch sampled from that buffer. Let $\mathcal{D}_t$ denote the empirical replay buffer containing $R$ samples, and let $\widetilde{\mathcal{B}}_t$ denote a mini-batch of size $B$ sampled from $\mathcal{D}_t$. We define

$$\begin{aligned} g_t &:= \mathbb{E}_{e\sim P_{\text{buffer}}(\cdot|\pi_t)}[\nabla\ell(e)] = \frac{1}{\tau}\sum_{t'=t-\tau+1}^{t}\mathbb{E}_{e\sim P_{\text{online}}(\cdot|\pi_{t'})}[\nabla\ell(e)], \\ \widehat{g}_t &:= \frac{1}{R}\sum_{e\in\mathcal{D}_t}\nabla\ell(e), \qquad \widetilde{g}_t := \frac{1}{B}\sum_{e\in\widetilde{\mathcal{B}}_t}\nabla\ell(e) \end{aligned} \tag{3}$$

where $\nabla\ell(\cdot)$ denotes the per-sample gradient. To characterize the accumulated effect of the gradient estimation error, consider their average over $K$ consecutive updates

$$\Delta_{t,K} := \frac{1}{K}\sum_{k=0}^{K-1}(g_{t+k}-\widetilde{g}_{t+k}) = \underbrace{\frac{1}{K}\sum_{k=0}^{K-1}(g_{t+k}-\widehat{g}_{t+k})}_{\text{finite buffer error}} + \underbrace{\frac{1}{K}\sum_{k=0}^{K-1}(\widehat{g}_{t+k}-\widetilde{g}_{t+k})}_{\text{mini-batch error}}. \tag{4}$$

We use a local approximation in which the model and the population replay buffer distribution change negligibly over $K$ updates. Let $V_{\text{local}}$ denote the mean of the gradient variances within the online distributions, and let $V_{\text{buffer}}$ denote the total gradient variance under the population replay buffer distribution. The latter includes variation across online distributions. Their formal definitions and covariance decomposition are given in Appendix A. Let $\kappa R$ denote the effective buffer size with temporal correlation, where $\tau^{-1} \le \kappa \le 1$. Let $\gamma_K$ denote the correction due to FIFO replacement during the $K$ updates. Under standard unbiased sampling assumptions, the second moment of the average error is

$$\mathbb{E}\left[\|\Delta_{t,K}\|_2^2\right] \approx \frac{\gamma_K V_{\text{local}}}{\kappa R} + \frac{1}{KB}\left(V_{\text{buffer}} - \frac{V_{\text{local}}}{\kappa R}\right) \approx \frac{\gamma_K V_{\text{local}}}{\kappa R} + \frac{V_{\text{buffer}}}{KB} \tag{5}$$

assuming $\kappa R \gg 1$. See Appendix D for the full derivation. The first term depends on the local gradient variance and the effective replay buffer size, while the second depends on the total buffer variance and the mini-batch size.

The averaging horizon $K$ need not equal the buffer replacement horizon $\tau$. If samples retain comparable gradient influence throughout one complete buffer cycle, we may use $K = \tau$ as a full cycle approximation. Using $R/\tau = 1/\sigma$, Equation 5 then becomes

$$\mathbb{E}\left[\|\Delta_{t,\tau}\|_2^2\right] \approx \frac{1}{\tau}\left(\frac{\gamma_\tau V_{\text{local}}}{\kappa/\sigma} + \frac{V_{\text{buffer}}}{B}\right). \tag{6}$$

Thus, up to the temporal correlation factor $\kappa$ and the FIFO correction $\gamma_\tau$, the finite buffer and mini-batch terms scale with the sample resolutions $\sigma^{-1}$ and $B$. Changes in the model and bootstrapped targets may shorten the effective averaging horizon, in which case $K < \tau$.

Equation 6 reveals two distinct sources of gradient uncertainty. The first arises from constructing the replay buffer from finite samples and is governed by the local gradient variance $V_{\text{local}}$ and the effective replay buffer size $\kappa R$. The second arises from stochastic mini-batch sampling from the replay buffer and is governed by the total replay buffer variance $V_{\text{buffer}}$ and the mini-batch size $B$.

**Key Observations on *Sample Density and Gradient Variance***

- **Two controls on gradient variance.** The effective replay buffer size and the mini-batch size govern separate additive components of gradient estimation variance.
- **UTD and batch size correspondence.** Under the full cycle approximation $K = \tau$, the two components scale with the sample resolutions $\sigma^{-1}$ and $B$, with corrections for temporal correlation and FIFO replacement.

### 4.3 Scalability Limits of Environment Parallelism

Increasing environment parallelism $E$ while scaling $R$ to keep $\tau = R/(EL)$ fixed increases the sample density while leaving the population replay buffer distribution unchanged. Equation 6 shows that increasing $E$, and therefore decreasing $\sigma$, reduces the finite buffer contribution. The minibatch contribution is instead controlled by $B$ and is not directly reduced by increasing $E$.

Equation 6 indicates that the benefit of additional parallelism is expected to diminish once the finite-buffer contribution becomes smaller than the mini-batch contribution, since increasing E reduces only the former. Temporal correlation and FIFO replacement affect the scale at which this transition occurs through $\kappa$ and $\gamma_\tau$, but do not change the separation between the two contributions. Increasing $B$ separately reduces the minibatch contribution. This prediction motivates the separate measurements of buffer and minibatch variance in Section 5.3.

**Key Observations on *Scalability Limits***

- **Diminishing benefits of environment parallelism.** At a fixed UTC ratio, increasing parallelism reduces finite buffer error but does not directly reduce minibatch error. Its benefit is therefore expected to diminish once the latter becomes dominant.

## 5 Experiments

We study how the main hyperparameters of parallel off-policy RL shape scale consistency, gradient variance, and hyperparameter transfer across parallel scales.

### 5.1 Experimental Setup

We use three codebases across five tasks. PQL (Li et al., 2023), a TD3-based method, is evaluated on CartPole and Ant in IsaacGym (Makoviychuk et al., 2021), using state normalization, n-step returns, gradient clipping, and layer normalization for stability. FastTD3 (Seo et al., 2025), a TD3-based method using the SimbaV2 (Lee et al., 2025b) backbone, and FlashSAC (Kim et al., 2026), a SAC-based method with a backbone inspired by SimBa (Lee et al., 2025a), are evaluated on the humanoid robots G1 and H1 in IsaacLab (Mittal et al., 2025) (`Isaac-Velocity-Flat-G1-v0`, `Isaac-Velocity-Flat-H1-v0`) and T1 in MuJoCo Playground (Zakka et al., 2025) (`T1JoystickRoughTerrain` for FastTD3, `T1JoystickFlatTerrain` for FlashSAC). Together, these experiments span three codebases, two off-policy algorithm families, and three GPU-parallel simulators.

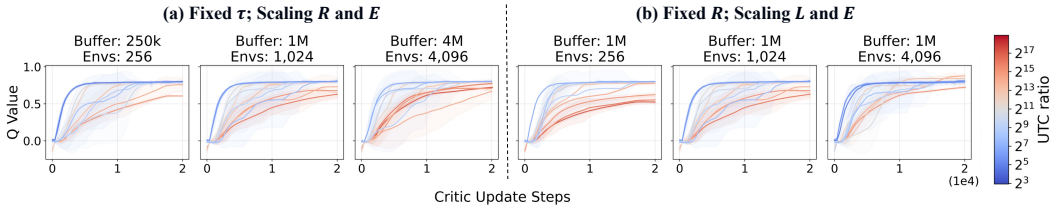



Figure 2: **CartPole consistency comparisons.** (a) The left panels show that training curves remain nearly identical when the buffer size and parallelism are increased linearly while preserving the UTC ratio. (b) The right panels show consistent training dynamics at the same UTD ratio and a constant replay buffer size when parallelism is increased and rollout length is decreased.

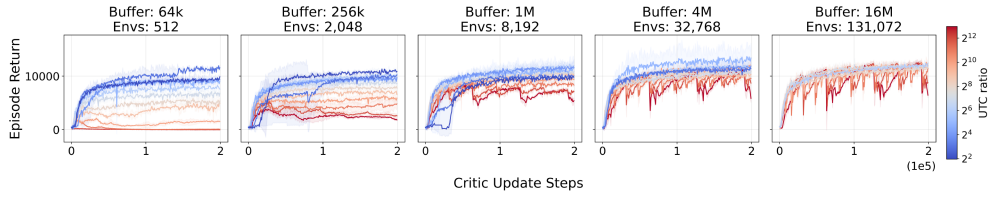



Figure 3: **Ant consistency comparisons.** Although the UTC ratio does not guarantee early-stage invariance in Ant, training dynamics stabilize at massive scales ($E \geq 32{,}768$). Some high-UTC configurations for the $E = 131{,}072$ setting are excluded due to resource limits.

### 5.2 Scale Consistency at a Fixed UTC Ratio

**Q1.** Do learning curves align as parallelism increases at a fixed UTC ratio, as suggested by our analysis of the replay buffer distribution in Section 4.1?

**Answer.** Yes, across most evaluated settings, learning curves with the same UTC ratio converge toward common dynamics once environment parallelism is sufficiently large.

We scale $E$ and $R$ proportionally while holding the UTC ratio $\tau$ and rollout length $L$ fixed. On CartPole, starting from $E = 256$ and $R = 250$k, we increase both by factors of 4 and 16. Figure 2 shows nearly identical learning curves across these scales at the same $\tau$, suggesting that the buffer distribution is already represented accurately at modest scale. On Ant, starting from $E = 512$ and $R = 64$k, we increase both by factors ranging from 4 to 256. Figure 3 indicates that Ant does not show the same small-scale consistency while the curves begin to stabilize toward a common trajectory at larger scales. We repeat this fixed-$\tau$ sweep with FastTD3 and FlashSAC, scaling $E$ from 16 to 65,536 and increasing $R$ proportionally. Figure 4 shows that curves with the same $\tau$ differ substantially at $E \leq 256$ but converge onto a common trajectory as $E$ grows, across all humanoid tasks.

As a further control, we vary $E$ and $L$ while keeping their product fixed and $R$ (hence $\tau$) constant, isolating the individual roles of $E$ and $L$. On CartPole, we evaluate $E = 256$ with $L$ ranging from $2^{-5}$ to $2^{5}$, increasing $E$ by factors of 4 and 16 while decreasing $L$ by $1/4$ and $1/16$ respectively. Figure 2 shows consistent training curves across these settings, confirming that the product $EL$, rather than $E$ and $L$ individually, is the relevant determinant of learning performance. In our setup, $E$ significantly exceeds $L$, consistent with GPU-based simulators such as IsaacGym, which are optimized for massive environment parallelism rather than large step ratios.

We quantify this alignment using the mean absolute error between each return curve and a reference curve at $E = 65{,}536$. Figure 5 reports comparisons across UTC ratios for FlashSAC on IsaacLab G1 and MuJoCo Playground T1. The upper and lower rows show G1 and T1, respectively. Each column uses the large scale curve at the indicated UTC ratio as its reference, while colors identify the UTC ratio of the compared curve. In both tasks, curves with a matching UTC ratio approach their large scale reference as $E$ increases, whereas curves with different UTC ratios generally remain farther away. This result shows that convergence across scales is strongest when the temporal composition of the replay buffer is preserved. Additional results for FlashSAC and FastTD3 are reported in Appendix B.2. The FastTD3 results are noisier and not always monotonic, but generally show smaller distances at larger values of $E$ for the higher UTC ratios.

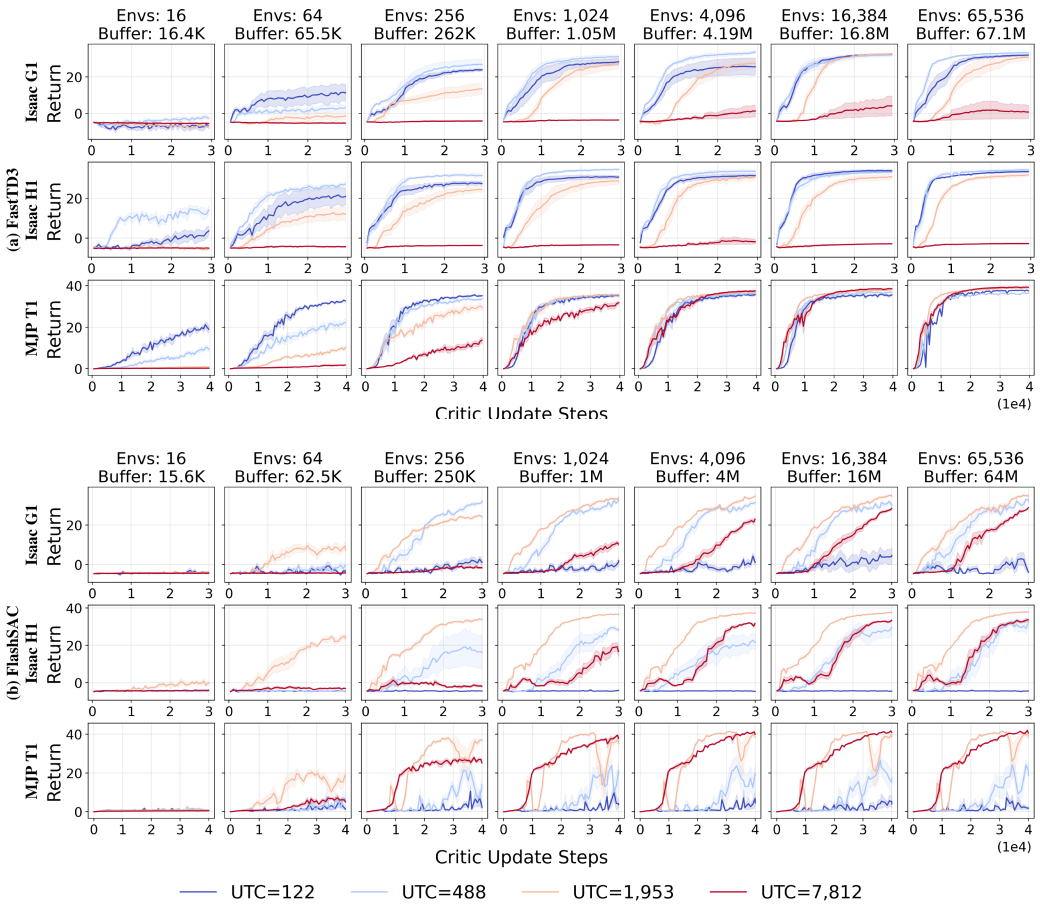



Figure 4: **Cross-scale convergence on humanoid locomotion.** (a) FastTD3 on IsaacLab G1, H1 and MuJoCo Playground T1 (Rough). (b) FlashSAC on IsaacLab G1, H1 and MuJoCo Playground T1 (Flat). Training curves are grouped by UTC ratio $\tau$ across seven scales of environment parallelism $E$ from 16 to 65,536. Curves with the same $\tau$ differ at small $E$ and converge toward a common trajectory as $E$ grows across both implementations and simulation settings.

### 5.3 Gradient Variance and Scalability Limits

**Q2.** Is gradient variance associated with scale consistency?

**Answer.** Yes, lower gradient variance is empirically associated with higher scale consistency.

We separately measure the replay buffer and mini-batch components of gradient variance across $E$. Figure 6 reports both gradient variance and the gradient noise scale proposed by McCandlish et al. (2018). Gradient variance measures the squared deviation of gradients from their mean, whereas gradient noise scale normalizes this variance by the squared norm of the mean gradient. The buffer variance corresponds to the term proportional to $1/R$ in Equation 5. Because computing gradients for every sample in the full replay buffer is expensive, we approximate this quantity using environment lanes. We compute the mean gradient within each lane, calculate the variance across the lane means, and divide it by the number of environments $E$. Samples within a lane are often temporally consecutive and correlated, so the variability captured by this estimate may differ from the variability of independently sampled transitions. The mini-batch variance corresponds to the term proportional to $1/B$ in Equation 6. We estimate the variance of a single mini-batch gradient using 50 independently sampled mini-batches of size $B = 8{,}192$ that are used only for evaluation. The buffer estimate is a snapshot at one update and does not account for the persistent effect of the same buffer error across $K$ updates. Since the true effective horizon $K$ is unknown, we interpret the buffer and mini-batch quantities as separate trends rather than directly comparing their absolute magnitudes.

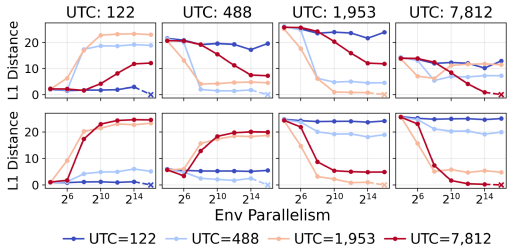



Figure 5: **Quantitative evaluation of UTC-invariance.** FlashSAC results on IsaacLab G1 (top) and MuJoCo Playground T1 (bottom).

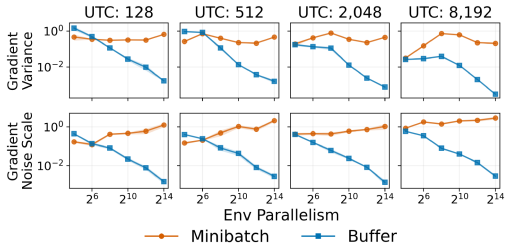



Figure 6: **Gradient variance and noise scale across parallel scales.** FastTD3 on IsaacLab G1 with $B = 8{,}192$.

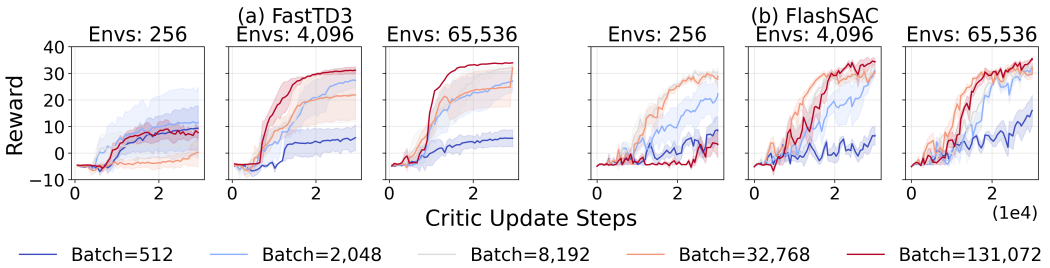



Figure 7: **Parallelism and batch size.** Training curves for FastTD3 and FlashSAC on IsaacLab G1 with varying batch sizes. Large batch sizes perform well at the two larger scales but do not consistently improve performance at $E = 256$.

Scale consistency improves while the buffer variance decreases. By contrast, the mini-batch variance does not decrease with $E$, so further increases in $E$ have less effect once the buffer variance becomes small. For FastTD3 G1 at $\tau = 2{,}048$, the learning curves change little beyond $E = 4{,}096$, which is consistent with the buffer variance having become too small to produce a visible change in the learning trajectory. This is an empirical association rather than a causal conclusion because the true effective horizon $K$ is unknown. The corresponding gradient variance and squared mean gradient norm are reported separately in Appendix B.1. A direct intervention on replay buffer sample density is reported in Appendix C.

### 5.4 Hyperparameter Transfer Across Scales

**Q3.** Do the UTC ratio and batch size selected at smaller scales remain effective at larger scales?

**Answer.** Yes for the UTC ratio, while batch size transfers from moderate to large scales but not consistently from small scales.

We examine whether the UTC ratio and batch size selected at smaller parallel scales remain effective as environment parallelism increases.

**UTC Ratio.** Figure 4 shows that UTC ratios that perform well at an intermediate scale generally remain competitive as environment parallelism increases. This pattern is particularly clear for FlashSAC, where the same range of UTC ratios performs well from intermediate scales up to $E = 65{,}536$ across G1, H1, and T1. The decreasing distance to the large scale reference in Figure 5 further shows that the learning dynamics associated with a fixed UTC ratio become increasingly stable as parallelism grows. FastTD3 shows the same overall transferability, although the differences among competitive UTC ratios become small at large scales for some tasks. The Ant results provide a more difficult case. The relative performance of different UTC ratios changes at small and intermediate scales, but becomes more stable once $E$ reaches $32{,}768$ or larger, as shown in Figure 3. These results indicate that a UTC ratio selected at a sufficiently large scale can be transferred to larger parallel scales within the same task and algorithm. The preferred UTC ratio still differs across tasks and algorithms.

**Batch Size.** Equation 6 predicts that increasing the batch size $B$ reduces mini-batch gradient variance. We vary $B$ for FastTD3 and FlashSAC on IsaacLab G1 at $E = 256$, $4{,}096$, and $65{,}536$. Figure 7 shows that large batch sizes perform well at $E = 4{,}096$ and remain effective at $E = 65{,}536$, but do not consistently improve performance at $E = 256$. One possible explanation is that finite buffer error remains substantial at small $E$, so reducing mini-batch variance alone does not sufficiently reduce the gradient noise affecting training. As $E$ and the replay buffer size increase, finite buffer error decreases and the reduction in mini-batch variance provided by a larger $B$ can have a greater effect. This interpretation is consistent with the variance trends in Figure 6.

Together, these results show different transfer behavior for the two hyperparameters. A UTC ratio selected at a sufficiently large scale generally remains competitive at larger parallel scales within the same task and algorithm. In contrast, the preferred batch size changes with scale, and a batch size selected at a smaller scale does not consistently remain effective as the parallelism increases.

## 6 Discussion and Conclusion

**Hyperparameter Search Guideline.** The UTC ratio can be tuned at a proxy scale once the preferred range of $\tau$ has stabilized across neighboring scales. The selected UTC ratio can then be transferred to larger scales, while the batch size should be reconsidered at the target scale.

**Guideline for Hyperparameter Search**

1. **Parameterize by $\tau$.** Scale $R$ in proportion to $EL$ so that $\tau = R/(EL)$ remains fixed.
2. **Find a proxy scale.** Use a coarse grid of $E$ and $\tau$ to find the smallest scale at which the preferred range of $\tau$ remains stable as $E$ increases.
3. **Tune $\tau$ at the proxy scale.** Sweep $\tau$ more finely by varying $R$ while keeping $E$, $L$, and the other training settings fixed.
4. **Scale up $E$ and $R$.** Increase $E$ and $R$ proportionally while keeping $L$ fixed until the learning curve changes little.
5. **Tune $B$ at the target scale.** Evaluate larger batch sizes as $E$ increases.

Step 1 is supported by the CartPole control, which produces consistent learning curves when $E$ and $L$ are varied while their product and $R$ remain fixed, as shown in Figure 2. Step 2 requires only a coarse grid of $E$ and $\tau$ to identify a proxy scale for the finer sweep in step 3. The required proxy scale depends on the task. CartPole already shows scale consistent curves from $E = 256$, while the humanoid tasks generally require larger values of $E$. For step 4, the finite-buffer gradient variance provides a candidate diagnostic. In FastTD3 G1 at $\tau = 2{,}048$, the learning curves change little beyond $E = 4{,}096$ as the finite buffer variance becomes small, while the mini-batch variance at $B = 8{,}192$ remains comparatively stable, as shown in Figure 6. Further increases in $E$ provide little visible benefit in this setting.

**Limitations.** Our variance analysis relies on local approximations. It assumes a model that remains approximately fixed over $K$ updates, and locally stable gradient variance and temporal correlation statistics. Temporal dependence within the replay buffer is summarized through an effective sample size, while FIFO replacement is represented through an approximate overlap correction. The effective horizon $K$ is unknown, so **Q2** shows that finite buffer variance and scale consistency are related, but does not explain why. Adaptive optimizers, parameter updates, and nonstationary bootstrapped targets may alter the horizon over which a finite buffer error influences learning.

**Conclusion.** We studied how massive environment parallelism changes off-policy learning through the replay buffer. By viewing the buffer as a temporal mixture of policy-induced distributions, we separated its temporal composition, set by the UTC ratio, from its finite-sample resolution, set by the buffer and mini-batch sizes. Across three codebases and five tasks, runs with the same UTC ratio converged toward common dynamics as parallelism grew, this convergence coincided with decreasing finite-buffer gradient variance, and the UTC ratio, unlike the batch size, generally transferred across parallel scales. These findings suggest selecting the UTC ratio at a stable scale and increasing parallelism and batch size until their benefits saturate.

## AI USE STATEMENT

In this work, we used generative AI tools to help develop theoretical models and conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose and refine hypotheses, implement methods, and assist with translation. We have not used generative AI tools to design or provide feedback on research methodology or experiments, or to interpret results; generating synthetic datasets, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures, suggest experimental parameters, create or edit software code, draft parts of the paper, discover research topics and identify gaps, brainstorm, search for information, identify relevant literature, improve readability, and format references. We have reviewed all AI-assisted work. All AI-assisted derivations and proofs were checked step by step by the authors, LLM-generated research ideas were checked for potential plagiarism through a manual literature survey, and LLM-generated code was reviewed and tested for correctness by the first author. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REPRODUCIBILITY STATEMENT

Our implementations build on publicly available implementations of PQL (Li et al., 2023), FastTD3 (Seo et al., 2025), and FlashSAC (Kim et al., 2026) and publicly available simulators including IsaacGym (Makoviychuk et al., 2021), IsaacLab (Mittal et al., 2025), and MuJoCo Playground (Zakka et al., 2025). Section 5 identifies the codebases, simulators, and task identifiers used in each experiment. Appendix E summarizes the training configurations, the resulting UTD and UTC ratios, the network architectures and optimization hyperparameters for each codebase, the number of seeds, the evaluation protocol, the procedures used to compute the curve distance metric and the buffer and mini-batch gradient variances, and the compute resources. The replay buffer properties used in Section 4 are derived in Appendix A. The gradient variance analysis used in Section 4.2 is derived in Appendix D.

## REFERENCES

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

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*, 2024.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double Q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2021.

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 *International Conference on Learning Representations*, 2023.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1407–1416, 2018.

Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - A differentiable physics engine for large scale rigid body simulation. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Preston Fu, Oleh Rybkin, Zhiyuan Paul Zhou, Michal Nauman, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Compute-optimal scaling for value-based deep RL. In *Advances in Neural Information Processing Systems*, pp. 177991–178030, 2025.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *International Conference on Learning Representations*, 2025.

Zhouyu He, Peng Qiao, Rongchun Li, Yong Dou, and Yusong Tan. Highly parallelized reinforcement learning training with relaxed assignment dependencies. In *AAAI Conference on Artificial Intelligence*, pp. 17159–17167, 2025.

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout Q-functions for doubly efficient reinforcement learning. In *International Conference on Learning Representations*, 2022.

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

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2019.

Donghu Kim, Youngdo Lee, Minho Park, Kinam Kim, Takuma Seno, I. Made Aswin Nahrendra, Sehee Min, Daniel Palenicek, Florian Vogt, Danica Kragic, Jan Peters, Jaegul Choo, and Hojoon Lee. FlashSAC: Fast and stable off-policy reinforcement learning for high-dimensional robot control. In *Robotics: Science and Systems*, 2026.

Hojoon Lee, Dongyoon Hwang, Donghu Kim, Hyunseung Kim, Jun Jet Tai, Kaushik Subramanian, Peter Wurman, Jaegul Choo, Peter Stone, and Takuma Seno. SimBa: Simplicity bias for scaling up parameters in deep reinforcement learning. In *International Conference on Learning Representations*, 2025a.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *International Conference on Machine Learning*, pp. 33352–33403, 2025b.

Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel $Q$-learning: Scaling off-policy reinforcement learning under massively parallel simulation. In *International Conference on Machine Learning*, pp. 19440–19459, 2023.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac Gym: High performance GPU based physics simulation for robot learning. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

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.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample efficient continuous control. In *Advances in Neural Information Processing Systems*, pp. 113038–113071, 2024.

Johan S Obando Ceron, Walter Mayor, Samuel Lavoie, Scott Fujimoto, Aaron Courville, and Pablo Samuel Castro. Simplicial embeddings improve sample efficiency in actor-critic agents. In *International Conference on Learning Representations*, 2026.

Carlo Romeo, Girolamo Macaluso, Alessandro Sestini, and Andrew D. Bagdanov. SPEQ: Offline stabilization phases for efficient Q-Learning in high update-to-data ratio reinforcement learning. *Reinforcement Learning Journal*, 6:1880–1893, 2025.

Oleh Rybkin, Michal Nauman, Preston Fu, Charlie Victor Snell, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Value-based deep RL scales predictably. In *International Conference on Machine Learning*, pp. 52422–52443, 2025.

Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level Atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380, 2023.

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.

Naoki Shitanda, Motoki Omura, Tatsuya Harada, and Takayuki Osa. Rethinking policy diversity in ensemble policy gradient in large-scale reinforcement learning. In *International Conference on Learning Representations*, 2026.

Jayesh Singla, Ananye Agarwal, and Deepak Pathak. SAPG: Split and aggregate policy gradients. In *International Conference on Machine Learning*, pp. 45759–45772, 2024.

Hanjing Wang, Man-Kit Sit, Congjie He, Ying Wen, Weinan Zhang, Jun Wang, Yaodong Yang, and Luo Mai. GEAR: A GPU-centric experience replay system for large reinforcement learning models. In *International Conference on Machine Learning*, pp. 36380–36390, 2023.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder Alexander Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. Demonstrating MuJoCo Playground. In *Robotics: Science and Systems*, 2025.

# A Properties of the Replay-Buffer Mixture

Assuming the distribution changes continuously, the buffer distribution at training step $t$ corresponds to the accumulated data sample count $s = t/\sigma$ (where $t > \tau$ and $s > R$):

$$P_{\text{buffer}}(e \mid \pi_t) = \frac{1}{R}\int_{s-R}^{s} P_{\text{online}}(e \mid \pi_{\sigma s'})\, ds' = \frac{1}{\tau}\int_{t-\tau}^{t} P_{\text{online}}(e \mid \pi_{t'})\, dt' \tag{7}$$

As the UTC ratio approaches zero, the replay-buffer distribution converges to the online distribution. If $P_{\text{online}}$ is continuous in training time, Equation 7 gives

$$\lim_{\tau\to 0} P_{\text{buffer}}(e \mid \pi_t) = \lim_{\tau\to 0}\frac{1}{\tau}\int_{t-\tau}^{t} P_{\text{online}}(e \mid \pi_{t'})\, dt' = P_{\text{online}}(e \mid \pi_t). \tag{8}$$

Using the law of total covariance, we obtain

$$\Sigma_{\text{buffer}} := \mathrm{Cov}_{e\sim P_{\text{buffer}}(\cdot\mid\pi_t)}[\nabla\ell(e)] \tag{9}$$

$$= \frac{1}{\tau}\int_{t-\tau}^{t} \mathrm{Cov}_{e\sim P_{\text{online}}(\cdot\mid\pi_{t'})}[\nabla\ell(e)]\, dt' + \mathrm{Cov}_{t'\sim\mathcal{U}([t-\tau,t])}\left[\mathbb{E}_{e\sim P_{\text{online}}(\cdot\mid\pi_{t'})}[\nabla\ell(e)]\right]. \tag{10}$$

The first term captures gradient variability within each online distribution, while the second captures variability across successive online policies.

We define $V_{\text{buffer}}$ as the trace of $\Sigma_{\text{buffer}}$ and $V_{\text{local}}$ as the trace of its first term.

$$V_{\text{buffer}} = \mathrm{tr}\left(\Sigma_{\text{buffer}}\right), \quad V_{\text{local}} = \mathrm{tr}\left(\frac{1}{\tau}\int_{t-\tau}^{t} \mathrm{Cov}_{e\sim P_{\text{online}}(\cdot\mid\pi_{t'})}[\nabla\ell(e)]\, dt'\right) \tag{11}$$

# B Additional Results on FastTD3 and FlashSAC

## B.1 Squared Norm of the Mean Gradient

Figure 6 in the main text reports both gradient variance and gradient noise scale. Figure 8 reports the squared norm of the mean gradient used to normalize the variance.

The mini-batch and environment-lane estimates closely track each other. The squared norm also tends to decrease as the UTC ratio increases. A larger UTC ratio keeps data in the replay buffer over more parameter updates, which allows the model to fit the stored data more closely and tends to reduce the remaining gradient magnitude. This normalization effect matters when noise scales are compared across UTC ratios. At a fixed UTC ratio, the decrease with $E$ in Figure 6 is driven by the variance term reported in the main text.

## B.2 Additional Quantitative Evaluation

We measure scale consistency using the mean absolute error between each return curve and the reference curve at $E = 65{,}536$ with the same UTC ratio. Figure 10 reports the complete FlashSAC results on G1, H1, and T1. Figure 9 reports the corresponding FastTD3 results on the same three tasks. At the higher UTC ratios, the distance to the reference generally decreases as $E$ increases. This pattern is clearer for FlashSAC. The FastTD3 results are noisier and are not always monotonic, although the distance is generally smaller at large values of $E$ than at small values of $E$.

# C Effect of Sample Density

The fixed-UTC-ratio experiments suggest that increasing environment parallelism improves performance mainly by increasing the finite-sample resolution of the replay buffer. We run a supplementary diagnostic experiment to test this sample density interpretation more directly.

We use FastTD3 with the SimbaV2 backbone on IsaacLab `Isaac-Velocity-Flat-G1-v0`, focusing on a single setting from the main FastTD3 sweep: $\tau = 2{,}048$ with per-environment buffer

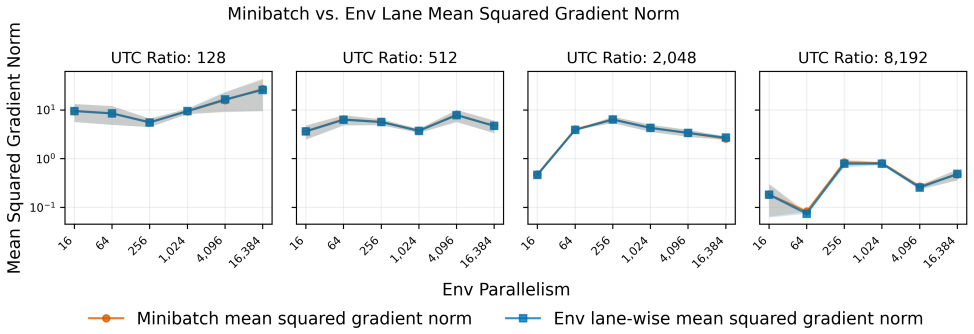



Figure 8: **Squared norm of the mean gradient.** Mini-batch and environment-lane estimates for FastTD3 on IsaacLab G1.

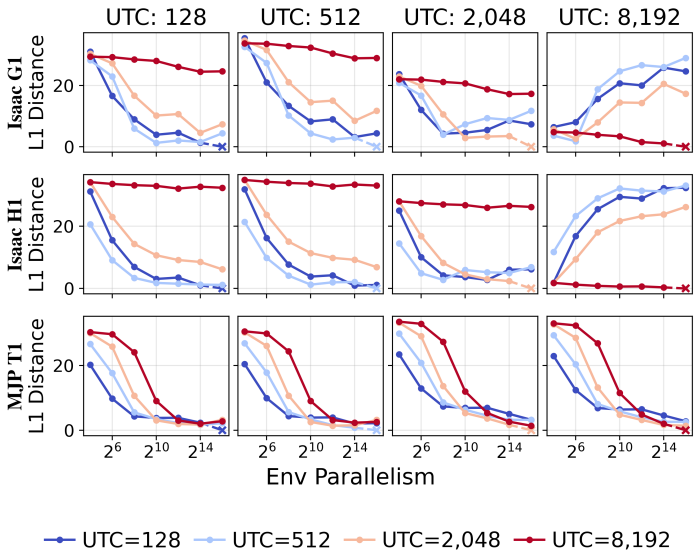



Figure 9: **Distance to the largest scale reference for FastTD3.** Results using the SimbaV2 backbone on IsaacLab G1 and H1 and MuJoCo Playground T1. For each task and UTC ratio, we report the mean absolute error between each return curve and the reference curve at $E = 65{,}536$.

size $R/E = 1{,}024$ and 2 updates per collection step. This experiment is intended as a targeted diagnostic rather than a full task sweep. We introduce a collector/follower design. A collector interacts with the environment and writes data into a replay buffer. Its number of active environment lanes is denoted by $N_c$. A follower does not interact with the environment, but trains off-policy from the collector's replay buffer. The number of replay buffer lanes available to the follower is denoted by $N_f$, and controls the sample density used for learning.

We compare three training protocols. The online collector is the standard online run with $N_c = x$. The aligned follower trains off-policy from the same collector stream using the same number of lanes, $N_f = N_c = x$. The full-density follower trains from the same collector stream but uses all available replay buffer lanes, $N_f = 16{,}384$. Thus, the comparison between the aligned follower and the full-density follower keeps the data stream fixed and changes only how densely that stream is sampled for learning. All comparisons keep the UTC ratio fixed.

Figure 11 shows that sample density is a bottleneck at intermediate parallel scales. At $N_c = 64$ and $N_c = 256$, the full-density follower substantially outperforms the aligned follower, with gains of roughly 10 return points or more. Since these two followers train from the same collector stream, this improvement cannot be attributed to a different online data source. It comes from using a denser

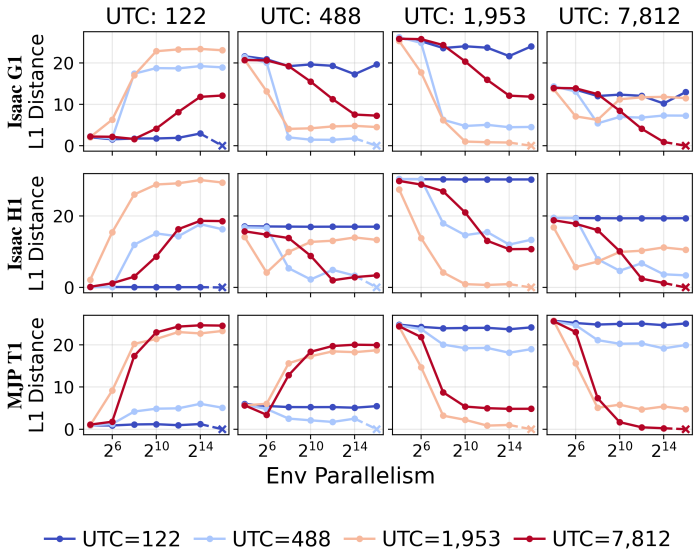



Figure 10: **Distance to the largest scale reference for FlashSAC.** Results on IsaacLab G1 and H1 and MuJoCo Playground T1. The metric and presentation follow Figure 9.

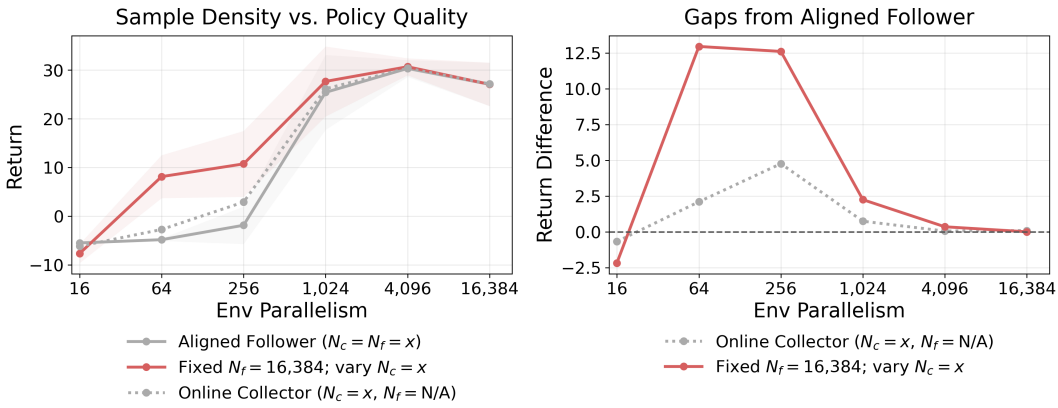



Figure 11: **Effect of sample density on FastTD3 G1.** Left: final return under three protocols: online training with $N_c = x$, an aligned off-policy follower with $N_c = N_f = x$, and a full-density off-policy follower that uses the same collector stream but samples from $N_f = 16{,}384$ lanes. Right: return difference from the aligned follower. The full-density follower gives a large gain at intermediate scales ($N_c = 64$ and 256), showing that the aligned follower is limited by sample density there.

subset of the replay buffer for off-policy updates. The gain becomes much smaller at $N_c = 1{,}024$ and nearly disappears by $N_c = 4{,}096$ and $16{,}384$, indicating that the sample-density bottleneck is gradually removed as the number of environment lanes increases.

# D Derivation of the Gradient Variance

## D.1 Notation

At model update step $t$, let the model parameters be $\theta_t$, and suppose that the online data obtained under the exploration policy $\pi_{\theta_t}$ follow $P_t(e \mid \pi_{\theta_t})$. The distribution of the entire buffer $P_t^{\text{buf}}$ is then

defined as the average of the online distributions over the past $\tau$ steps.

$$P_t^{\text{buf}} := \frac{1}{\tau} \sum_{t'=t-\tau+1}^{t} P_{t'} \tag{12}$$

Let $\mathcal{B}_t$ denote the set of online samples actually obtained through exploration, and let $\mathcal{D}_t = \bigcup_{t' \in [t-\tau+1, t]} \mathcal{B}_{t'}$ denote the set of all samples in the buffer. The number of samples collected at each step, $|\mathcal{B}_t| = R/\tau = 1/\sigma$, is constant and does not depend on the step index $t$. Here, $R = |\mathcal{D}_t|$ is the buffer size and $\sigma$ is the UTD ratio.

Let $P_{\mathcal{B}_t}$ and $P_{\mathcal{D}_t}$ denote the discrete empirical distributions followed by the samples in the online data $\mathcal{B}_t$ and the replay buffer $\mathcal{D}_t$, respectively. The empirical distribution of the replay buffer can then be expressed as the average of the online empirical distributions over the past $\tau$ steps.

$$P_{\mathcal{D}_t} := \frac{1}{\tau} \sum_{t'=t-\tau+1}^{t} P_{\mathcal{B}_{t'}} \tag{13}$$

For parameters $\theta$, let $\nabla \ell(\theta; e)$ denote the gradient computed from a data sample $e$. We write the expectation and variance of the gradient computed from samples following a distribution $P$ as follows.

$$\nabla \mathcal{L}(\theta; P) := \mathbb{E}_{e \sim P}[\nabla \ell(\theta; e)], \quad V(\theta; P) := \mathbb{E}_{e \sim P}\left[\|\nabla \ell(\theta; e) - \nabla \mathcal{L}(\theta; P)\|_2^2\right] \tag{14}$$

Let $\nu_t(\theta)$ denote the gradient error caused by randomness in exploration at each step, including randomness in the initial state and the policy. We define $\nu_t(\theta)$ as follows.

$$\nu_t(\theta) := \nabla \mathcal{L}(\theta; P_t) - \nabla \mathcal{L}(\theta; P_{\mathcal{B}_t}) \tag{15}$$

Let $\nu_t^{\text{buf}}(\theta)$ denote the gradient error over the entire buffer that is caused by randomness in exploration. By linearity of the gradient with respect to the data distribution, $\nu_t^{\text{buf}}(\theta)$ can be expressed as follows.

$$\nu_t^{\text{buf}}(\theta) := \nabla \mathcal{L}(\theta; P_t^{\text{buf}}) - \nabla \mathcal{L}(\theta; P_{\mathcal{D}_t}) \tag{16}$$

$$= \frac{1}{\tau} \sum_{t'=t-\tau+1}^{t} \nu_{t'}(\theta) \tag{17}$$

Unless explicitly stated otherwise, $\nu$ and $\nu^{\text{buf}}$ are evaluated at $\theta_t$.

Let $\widetilde{\mathcal{B}}_t$ denote the set of samples in a mini-batch, and let $P_{\widetilde{\mathcal{B}}_t}$ denote its empirical distribution. Let the batch size be $B = |\widetilde{\mathcal{B}}_t|$. Since the mini-batch is sampled from the buffer $\mathcal{D}_t$, we have $\widetilde{\mathcal{B}}_t \subseteq \mathcal{D}_t$. Let $\xi_t$ denote the gradient error caused by randomness in mini-batch sampling, which we define as follows.

$$\xi_t := \nabla \mathcal{L}(\theta_t; P_{\mathcal{D}_t}) - \nabla \mathcal{L}(\theta_t; P_{\widetilde{\mathcal{B}}_t}) \tag{18}$$

It follows that the difference between the mini-batch gradient and the true expected gradient can be expressed as the sum of the gradient errors caused by the finite buffer size and the finite mini-batch.

$$\nabla \mathcal{L}(\theta_t; P_t^{\text{buf}}) - \nabla \mathcal{L}(\theta_t; P_{\widetilde{\mathcal{B}}_t}) = \nu_t^{\text{buf}} + \xi_t = \frac{1}{\tau} \sum_{t'=t-\tau+1}^{t} \nu_{t'} + \xi_t \tag{19}$$

For clarity, this appendix uses a more explicit notation than Section 4.2. $P_t$ and $P_t^{\text{buf}}$ denote $P_{\text{online}}(\cdot \mid \pi_t)$ and $P_{\text{buffer}}(\cdot \mid \pi_t)$, and $g_t$, $\widehat{g}_t$, and $\widetilde{g}_t$ correspond to $\nabla \mathcal{L}(\theta_t; P)$ with $P = P_t^{\text{buf}}$, $P_{\mathcal{D}_t}$, and $P_{\widetilde{\mathcal{B}}_t}$, respectively.

### D.2 Conditional Expectation of Noise

At model update step $t$, exploration is performed under the policy $\pi_{\theta_t}$, the buffer is updated, and the model is then updated. Let $\mathcal{F}_t^-$ denote the history containing all information available at the

beginning of model update step $t$. Let $\mathcal{F}_t$ denote the history immediately after exploration and the subsequent buffer update.

We first consider the expectation of the gradient error. Since $\nu_t$ is the gradient error caused by randomness in exploration, we assume that its expectation is zero. When conditioning on $\mathcal{F}_t$, however, $\mathcal{B}_t$ has already been determined, so taking the expectation does not change its value.

$$\mathbb{E}[\nu_t] = 0, \quad \mathbb{E}[\nu_t \mid \mathcal{F}_t] = \nu_t \tag{20}$$

We also assume that the expectation of the mini-batch gradient error $\xi_t$ is zero with respect to sampling.

$$\mathbb{E}[\xi_t \mid \mathcal{F}_t^-] = \mathbb{E}[\xi_t \mid \mathcal{F}_t] = 0 \tag{21}$$

We define the average gradient error $\Delta_{t,K}$ over the next $K$ steps as follows.

$$\Delta_{t,K} := \frac{1}{K}\sum_{k=0}^{K-1}\left(\nabla\mathcal{L}(\theta_{t+k}; P_{t+k}^{\text{buf}}) - \nabla\mathcal{L}(\theta_{t+k}; P_{\widetilde{\mathcal{B}}_{t+k}})\right) \tag{22}$$

$$= \frac{1}{K}\sum_{k=0}^{K-1}\left(\nu_{t+k}^{\text{buf}} + \xi_{t+k}\right) \tag{23}$$

Since the expected mini-batch sampling error is zero at each step, we have

$$\mathbb{E}[\Delta_{t,K} \mid \mathcal{F}_t] = \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\nu_{t+k}^{\text{buf}} \mid \mathcal{F}_t] \tag{24}$$

### D.2.1 Fixed Model and Fixed Buffer Approximation

We approximate the model parameters as remaining unchanged over the next $K$ steps and set $\theta_{t+k} \approx \theta_t$. We also approximate the buffer as fixed and set $\mathcal{D}_{t+k} \approx \mathcal{D}_t$. Since $\nu_{t+k}^{\text{buf}} \approx \nu_t^{\text{buf}}$ under these approximations, the average gradient error over $K$ steps and its expectation can be expressed as follows.

$$\Delta_{t,K} \approx \nu_t^{\text{buf}} + \frac{1}{K}\sum_{k=0}^{K-1}\xi_{t+k}, \quad \mathbb{E}[\Delta_{t,K} \mid \mathcal{F}_t] \approx \nu_t^{\text{buf}} \tag{25}$$

Using the following second moment identity

$$\mathbb{E}[\|X\|_2^2 \mid \mathcal{F}] = \|\mathbb{E}[X \mid \mathcal{F}]\|_2^2 + \mathbb{E}[\|X - \mathbb{E}[X \mid \mathcal{F}]\|_2^2 \mid \mathcal{F}]. \tag{26}$$

the second moment of the average gradient error can be expressed as follows.

$$\mathbb{E}[\|\Delta_{t,K}\|_2^2 \mid \mathcal{F}_t] = \|\nu_t^{\text{buf}}\|_2^2 + \mathbb{E}\left[\left\|\frac{1}{K}\sum_{k=0}^{K-1}\xi_{t+k}\right\|_2^2 \middle| \mathcal{F}_t\right] \tag{27}$$

$$= \|\nu_t^{\text{buf}}\|_2^2 + \frac{1}{K^2}\sum_{k=0}^{K-1}\mathbb{E}[\|\xi_{t+k}\|_2^2 \mid \mathcal{F}_t] + \frac{1}{K^2}\sum_{k\neq l}\mathbb{E}[\langle\xi_{t+k}, \xi_{t+l}\rangle \mid \mathcal{F}_t] \tag{28}$$

$$= \|\nu_t^{\text{buf}}\|_2^2 + \frac{1}{K^2}\sum_{k=0}^{K-1}\mathbb{E}[\|\xi_{t+k}\|_2^2 \mid \mathcal{F}_t] \tag{29}$$

Here, we have used the fact that $\xi$ is sampled independently at each step, which makes the expected cross terms zero.

### D.2.2 Fixed Model Approximation with FIFO Buffer Replacement

As in Equation 23, we consider the average gradient error over the next $K$ steps. We use the fixed model approximation $\theta_{t+k} \approx \theta_t$, while allowing the buffer to be updated through first in, first out

replacement. Unlike the setting in Section D.2.1, the buffer is updated with samples collected under the fixed policy $\pi_{\theta_t}$ after step $t$.

By the tower property, the conditional expectation of the mini-batch sampling error $\xi_{t+k}$ given $\mathcal{F}_t$ is zero.

$$\mathbb{E}[\xi_{t+k} \mid \mathcal{F}_t] = \mathbb{E}[\mathbb{E}[\xi_{t+k} \mid \mathcal{F}_{t+k}] \mid \mathcal{F}_t] = 0 \tag{30}$$

Equation 20 also gives the following expression for the expectation of the buffer error $\nu^{\text{buf}}_{t+k}$. Note that $\nu_{t'}$ for $t' \leq t$ is determined when conditioning on $\mathcal{F}_t$, so taking the expectation does not change its value.

$$\mathbb{E}[\nu^{\text{buf}}_{t+k} \mid \mathcal{F}_t] = \frac{1}{\tau} \sum_{t'=t+k-\tau+1}^{t+k} \mathbb{E}[\nu_{t'} \mid \mathcal{F}_t] \tag{31}$$

$$= \frac{1}{\tau} \left( \sum_{t'=t+k-\tau+1}^{t} \nu_{t'} + \sum_{t'=t+1}^{t+k} \mathbb{E}[\nu_{t'} \mid \mathcal{F}_t] \right) \tag{32}$$

It follows that the expectation of the average gradient error $\Delta_{t,K}$ over $K$ steps, where $1 \leq K \leq \tau$, can be expressed as follows.

$$\mathbb{E}[\Delta_{t,K} \mid \mathcal{F}_t] = \frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}[\nu^{\text{buf}}_{t+k} \mid \mathcal{F}_t] \tag{33}$$

$$= \frac{1}{K\tau} \sum_{k=0}^{K-1} \left( \sum_{t'=t+k-\tau+1}^{t} \nu_{t'} + \sum_{t'=t+1}^{t+k} \mathbb{E}[\nu_{t'} \mid \mathcal{F}_t] \right) \tag{34}$$

Using the second moment identity in Equation 26, the second moment of the average gradient error can be expressed as follows.

$$\begin{aligned} \mathbb{E}[\|\Delta_{t,K}\|_2^2 \mid \mathcal{F}_t] = &\left\| \frac{1}{K} \sum_{k=0}^{K-1} \mathbb{E}[\nu^{\text{buf}}_{t+k} \mid \mathcal{F}_t] \right\|_2^2 \\ &+ \mathbb{E}\left[ \left\| \frac{1}{K\tau} \sum_{k=0}^{K-1} \sum_{t'=t+1}^{t+k} (\nu_{t'} - \mathbb{E}[\nu_{t'} \mid \mathcal{F}_t]) \right\|_2^2 \middle| \mathcal{F}_t \right] \\ &+ \frac{1}{K^2} \sum_{k=0}^{K-1} \mathbb{E}[\|\xi_{t+k}\|_2^2 \mid \mathcal{F}_t] \end{aligned} \tag{35}$$

$$= \mathbb{E}\left[ \left\| \frac{1}{K} \sum_{k=0}^{K-1} \nu^{\text{buf}}_{t+k} \right\|_2^2 \middle| \mathcal{F}_t \right] + \frac{1}{K^2} \sum_{k=0}^{K-1} \mathbb{E}[\|\xi_{t+k}\|_2^2 \mid \mathcal{F}_t] \tag{36}$$

Here, we have used the fact that $\xi$ is sampled independently at each step, which makes the expected cross terms among different $\xi$ terms and between $\xi$ and $\nu$ equal to zero.

### D.3 Expectation of Noise

We next evaluate the expectation of the second moment of the average gradient error over $K$ steps without conditioning on the history.

#### D.3.1 Effective Sample Size in Buffer

For the gradient variance defined in Equation 14, let $V_{\text{local}} = \mathbb{E}_{t'}[V(\theta_t; P_{t'})]$ denote the mean gradient variance within the online distributions represented in the buffer, and let $V_{\text{buffer}} = V(\theta_t; P_t^{\text{buf}})$ denote the total gradient variance under the buffer distribution.

For the buffer error, the environment lanes are mutually independent, so samples within the same step can be treated as independent. Since the number of samples collected in one step is $R/\tau$, the variance of the collection error for one step is

$$\mathbb{E}_{t'}[\mathbb{E}[\|\nu_{t'}\|_2^2]] = \frac{V_{\text{local}}}{R/\tau} = \frac{V_{\text{local}}\tau}{R} \tag{37}$$

Since samples within the same episode generally exhibit temporal correlation, the squared norm of the error over the entire buffer includes terms across rounds and can be expressed as follows.

$$\mathbb{E}[\|\nu_t^{\text{buf}}\|_2^2] = \frac{1}{\tau^2} \sum_{0 \le i,j \le \tau-1} \mathbb{E}[\langle \nu_{t-i}, \nu_{t-j} \rangle] \tag{38}$$

$$= \frac{V_{\text{local}}}{R} + \frac{1}{\tau^2} \sum_{\substack{0 \le i,j \le \tau-1 \\ i \ne j}} \mathbb{E}[\langle \nu_{t-i}, \nu_{t-j} \rangle] \tag{39}$$

$$=: \frac{V_{\text{local}}}{\kappa R} \tag{40}$$

Here, we introduce a constant $\kappa$, where $\tau^{-1} \le \kappa \le 1$, that corrects for temporal correlation among samples.

$$\kappa = \left( \frac{1}{\tau} \sum_{0 \le i,j \le \tau-1} \frac{\mathbb{E}[\langle \nu_{t-i}, \nu_{t-j} \rangle]}{\mathbb{E}[\langle \nu_t, \nu_t \rangle]} \right)^{-1} \tag{41}$$

$$= \left( 1 + \frac{R}{\tau^2 V_{\text{local}}} \sum_{\substack{0 \le i,j \le \tau-1 \\ i \ne j}} \mathbb{E}[\langle \nu_{t-i}, \nu_{t-j} \rangle] \right)^{-1} \tag{42}$$

Assuming that the covariance terms are always nonnegative, we have $\kappa = 1$ in the absence of temporal correlation. Strong temporal correlation gives $\kappa < 1$, which indicates a reduction in the effective number of samples $\kappa R$ in the buffer.

For the mini-batch sampling error, the mini-batch samples are drawn independently from the buffer, so the variance of the mini-batch sampling error can be expressed as follows.

$$\mathbb{E}[\|\xi_t\|_2^2] = \frac{\mathbb{E}[V(\theta_t; P_{\mathcal{D}_t})]}{B} \tag{43}$$

$$= \frac{V_{\text{buffer}} - \mathbb{E}[\|\nu_t^{\text{buf}}\|_2^2]}{B} \tag{44}$$

$$= \frac{1}{B} \left( V_{\text{buffer}} - \frac{V_{\text{local}}}{\kappa R} \right) \tag{45}$$

### D.3.2 Fixed Model and Fixed Buffer Approximation

Under the assumption used to derive Equation 29, which ignores buffer replacement, Equations 40 and 45 give the following expected second moment of the gradient error.

$$\mathbb{E}[\|\Delta_{t,K}\|_2^2] = \frac{V_{\text{local}}}{\kappa R} + \frac{1}{KB} \left( V_{\text{buffer}} - \frac{V_{\text{local}}}{\kappa R} \right) \tag{46}$$

$$\approx \frac{V_{\text{local}}}{\kappa R} + \frac{V_{\text{buffer}}}{KB} \tag{47}$$

where we have assumed $\kappa R \gg 1$.

### D.3.3 Fixed Model Approximation with FIFO Buffer Replacement

We define the coefficient $\gamma_K$, which measures how buffer replacement over $K$ steps changes the second moment of the buffer error, as follows.

$$\gamma_K := \frac{\mathbb{E}\left[\left\|\frac{1}{K}\sum_{k=0}^{K-1}\nu_{t+k}^{\text{buf}}\right\|_2^2\right]}{\mathbb{E}\left[\left\|\nu_t^{\text{buf}}\right\|_2^2\right]} \tag{48}$$

$$= \frac{1}{K^2}\sum_{0\le i,j\le K-1}\frac{\mathbb{E}[\langle\nu_{t+i}^{\text{buf}},\nu_{t+j}^{\text{buf}}\rangle]}{\mathbb{E}[\langle\nu_t^{\text{buf}},\nu_t^{\text{buf}}\rangle]} \tag{49}$$

While $\kappa$ represents a correction to the effective sample size at a given step, $\gamma_K$ represents the change in the second moment of the buffer error caused by buffer replacement over $K$ steps.

We assume that the covariance of the buffer error across steps depends only on the lag and not on absolute time. In other words, for $k \ge 0$, we assume that $\mathbb{E}[\langle\nu_t^{\text{buf}},\nu_{t+k}^{\text{buf}}\rangle] = \mathbb{E}[\langle\nu_{t-k}^{\text{buf}},\nu_t^{\text{buf}}\rangle]$. Let $S(h)$ denote the expected covariance at lag $h$. Equation 40 gives $S(0) = V_{\text{local}}/(\kappa R)$. Then $\gamma_K$ can be expressed as follows.

$$\gamma_K = \frac{1}{K^2}\sum_{0\le i,j\le K-1}\frac{S(|i-j|)}{S(0)} \tag{50}$$

$$= \frac{1}{K^2}\left(K + 2\sum_{h=1}^{K-1}(K-h)\frac{S(h)}{S(0)}\right) \tag{51}$$

Let $T$ denote the number of temporally correlated steps within an episode.

Among the terms in $S(h)$, the terms arising from within step variance remain in proportion to the length of the shared window. The covariance terms between different steps depend on the relative magnitudes of $T$ and $\tau$. We use $c$, where $\kappa \le c \le 1$, to capture this dependence. Then $S(h)$ can be expressed as follows.

$$S(h) = \left(1 - c\frac{h}{\tau}\right)S(0) \tag{52}$$

When $T \ll \tau$, we have $c \approx 1$, and almost none of the terms involving different steps remain. When $T \gg \tau$, we have $c \approx \kappa$, and the terms involving different steps remain unchanged.

Therefore,

$$\gamma_K = \frac{1}{K^2}\left(K + 2\sum_{h=1}^{K-1}(K-h)\left(1 - c\frac{h}{\tau}\right)\right) \tag{53}$$

$$= \frac{1}{K^2}\left(K + 2K(K-1) - \frac{2(Kc+\tau)}{\tau}\sum_{h=1}^{K-1}h + \frac{2c}{\tau}\sum_{h=1}^{K-1}h^2\right) \tag{54}$$

$$= \frac{1}{K^2}\left(2K^2 - K - \frac{2(Kc+\tau)}{\tau}\cdot\frac{(K-1)K}{2} + \frac{2c}{\tau}\cdot\frac{(K-1)K(2K-1)}{6}\right) \tag{55}$$

$$= 1 - c\frac{K^2-1}{3K\tau} \tag{56}$$

Equations 36, 40, 45, and 48 give the following expected second moment of the average gradient error over $K$ steps when FIFO buffer replacement is taken into account.

$$\mathbb{E}[\|\Delta_{t,K}\|_2^2] = \gamma_K\frac{V_{\text{local}}}{\kappa R} + \frac{1}{KB}\left(V_{\text{buffer}} - \frac{V_{\text{local}}}{\kappa R}\right) \tag{57}$$

$$\approx \frac{\gamma_K V_{\text{local}}}{\kappa R} + \frac{V_{\text{buffer}}}{KB} \tag{58}$$

Comparison with Equation 47 shows the additional correction factor $\gamma_K$ for FIFO buffer replacement. Equation 56 shows that $\gamma_K \to 1$ as $K/\tau \to 0$. When buffer replacement is negligible, we recover the same result as Equation 47, as expected.

When $K/\tau \to 1$, we instead have $\gamma_K \to 1-c/3$, and thus $2/3 \le \gamma_K \le 1-\kappa/3$. This shows that the second moment of the buffer error decreases when buffer replacement is not negligible. A shorter span of temporal correlation in the buffer results in more substantial replacement of correlated data and a larger reduction in $\gamma_K$.

# E Implementation Details

**Experimental systems.** Table 1 reports the training parameters varied in our experiments. These are the number of parallel environments $E$, the rollout length $L$, the replay buffer capacity $R$, and the mini-batch size $B$. CartPole and Ant use a TD3-style implementation based on the PQL codebase. FastTD3 uses the SimbaV2 backbone in all experiments reported in this paper. FastTD3 and FlashSAC use the same G1 and H1 tasks in IsaacLab, but use the RoughTerrain and FlatTerrain variants for T1 in MuJoCo Playground. The T1 results test whether the scaling pattern recurs across implementations and simulation settings rather than compare returns directly.

Table 1: **Training parameters.** The listed values are paired according to the controlled quantity in each experiment rather than evaluated as a full Cartesian product. For FastTD3 and FlashSAC, G1 and H1 share the same sweep.

| System | $E$ | $R$ | $L$ | $B$ |
|---|---|---|---|---|
| CartPole | $2^8, 2^{10}, 2^{12}$ | 250k, 1M, 4M | $2^{-7}, 2^{-6}, \ldots, 2^5$ | 8,192 |
| Ant | $2^9, 2^{11}, 2^{13}, 2^{15}, 2^{17}$ | $125E$ | $2^{-6}, 2^{-5}, \ldots, 2^1$ | 8,192 |
| FastTD3 | $2^4, 2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}$ | $1024E$ | $2^{-3}, 2^{-1}, 2^1, 2^3$ | 8,192 |
| FlashSAC | $2^4, 2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}$ | $1000E$ | $2^{-3}, 2^{-1}, 2^1, 2^3$ | 8,192 |

These parameters determine the UTD and UTC ratios according to $\sigma = 1/(EL)$ and $\tau = R/(EL)$. Table 2 reports the resulting values used in our experiments.

Table 2: **Resulting UTD and UTC ratios.** Derived from Table 1 via $\sigma = 1/(EL)$ and $\tau = R/(EL)$.

| System | UTD $\sigma$ | UTC $\tau$ |
|---|---|---|
| CartPole | $2^{-3}, 2^{-4}, \ldots, 2^{-15}$ | approx. $2^3, 2^5, \ldots, 2^{17}$ |
| Ant | $2^{-3}, 2^{-4}, \ldots, 2^{-18}$ | $62.5, 125, 250, 500, 1\text{k}, 2\text{k}, 4\text{k}, 8\text{k}$ |
| FastTD3 | $2^{-1}, 2^{-3}, \ldots, 2^{-19}$ | $2^7, 2^9, 2^{11}, 2^{13}$ |
| FlashSAC | $2^{-1}, 2^{-3}, \ldots, 2^{-19}$ | $122, 488, 1{,}953, 7{,}812$ |

The FlashSAC UTC ratios include a small offset arising from its implementation.

In the CartPole control experiment, we fix $R = 1\text{M}$ and vary $E$ and $L$ while preserving their product $EL$.

**PQL-based CartPole and Ant.** The actor is a two-hidden-layer MLP of width $512$ with LayerNorm and ReLU activations, followed by a tanh output. The twin critics use the same two-layer architecture of width $512$ with LayerNorm and ReLU. We use AdamW, discount $\gamma = 0.99$, 3-step returns, an actor update every 2 critic updates, reward scale $0.01$, and target-network coefficient $0.05$. The mini-batch size is 8,192. The Ant experiments use a learning rate of $10^{-3}$ and gradient clipping at $0.5$. The CartPole sweep primarily uses a learning rate of $10^{-4}$ and initializes from the same pretrained checkpoint across scaling conditions.

**FastTD3.** FastTD3 uses a deterministic actor and twin categorical critics with the SimbaV2 backbone. The actor consists of a `HyperEmbedder`, one `HyperLERP` block, and a `HyperTanhPolicy` head, with hidden dimension 256 and expansion factor $4$. Each critic uses a

`HyperEmbedder`, two `HyperLERP` blocks, hidden dimension $512$, and a categorical value head. We use $251$ atoms on IsaacLab and $101$ atoms on MuJoCo Playground, with value support $[-10, 10]$. Actor and critic are optimized with AdamW from an initial learning rate of $3 \times 10^{-4}$. The MuJoCo Playground configuration uses cosine decay to $3 \times 10^{-5}$. The target-network coefficient is $0.1$. We use $(\gamma, n) = (0.99, 8)$ for G1 and H1, and $(0.97, 1)$ for T1. Mixed-precision training with bfloat16 and compilation are enabled.

**FlashSAC.** FlashSAC uses a stochastic tanh-Gaussian actor, twin categorical critics, and a learned temperature. The actor has a FlashSACEmbedder followed by two FlashSAC blocks, RMSNorm, and a NormalTanhPolicy head with hidden dimension $128$. The critics use two ensemble FlashSAC blocks with hidden dimension $256$ and a $101$-bin categorical head. Rewards are normalized and the value support is $[-5, 5]$ after normalization. We use AdamW with learning-rate warmup to $3 \times 10^{-4}$ followed by cosine decay to $1.5 \times 10^{-4}$, an actor update period of 2, target-network coefficient $0.01$, initial temperature $0.01$, and target standard deviation $0.15$. We use $(\gamma, n) = (0.99, 3)$ for G1 and H1, and $(0.97, 1)$ for T1. Mixed-precision training and compilation are enabled. T1 additionally uses asymmetric observations.

**Seeds and evaluation.** CartPole results use $10$ seeds, while Ant, FastTD3, and FlashSAC use $5$ seeds per configuration. Each evaluation rollout has a horizon of $1{,}000$ environment steps. Curves show the mean and one standard deviation across seeds, and CartPole curves are smoothed with isotonic regression for visualization. For the curve-distance metric, we compute the MAE between the unsmoothed return curve of each seed and the reference curve at $E = 65{,}536$ with the same $\tau$, and then average the MAE across seeds. For gradient variance measurements, the gradient of each environment lane is computed as the mean gradient over all samples from that lane in the replay buffer. The buffer variance estimate is the variance of these lane mean gradients across lanes divided by the number of environments $E$. The mini-batch variance estimate is computed from $50$ independently sampled mini-batches of size $B = 8{,}192$ that are used only for evaluation.

**Compute resources.** All experiments used one GPU per run on an internal cluster. The primary GPU models were NVIDIA A100 GPUs with $40$ or $80$ GB of memory, NVIDIA Tesla V100 GPUs with $32$ GB of memory, and NVIDIA Quadro RTX 8000 GPUs with $48$ GB of memory. Host CPU configurations varied across workers and were not controlled because the workloads were limited primarily by GPU computation.

The main FastTD3 and FlashSAC sweeps comprised $670$ runs and required approximately $2{,}660$ allocated GPU hours. Most individual runs required between $1$ and $6$ allocated GPU hours, although runtimes varied substantially with the task, environment parallelism, and update setting. Including the associated preliminary studies, ablations, auxiliary evaluations, and reruns, we estimate that the experiments in this paper used approximately $7{,}000$ allocated GPU hours. This estimate includes approximately $1{,}000$ GPU hours for PQL CartPole and Ant. GPU hours are calculated from wall-clock runtime rather than measured GPU utilization. Persistent storage remained below $100$ GB.