An Empirical Study of Non-Uniform Sampling in Off-Policy Reinforcement Learning for Continuous Control

Nicholas Ioannidis
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4, Canada
nikolaos.ioannidis@alumni.ubc.ca

J. Wilder Lavington
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4, Canada
jola2372@cs.ubc.ca

Mark Schmidt
Department of Computer Science
University of British Columbia
Vancouver, BC V6T 1Z4, Canada
schmidtm@cs.ubc.ca

Abstract

Off-policy reinforcement learning (RL) algorithms can take advantage of samples generated from all previous interactions with the environment through "experience replay". Such methods outperform almost all on-policy and model based alternatives in complex tasks where a structured or well parameterized model of the world does not exist. This makes them desirable for practitioners who lack domain specific knowledge, but who still require high sample efficiency. However this high performance can come at a cost. Because of additional hyperparameters introduced to efficiently learn function approximators, off-policy RL can perform poorly on new problems. To address parameter sensitivity, we show how the correct choice of non-uniform sampling for experience replay can stabilize model performance under varying environmental conditions and hyper-parameters.

1 Introduction

Deep reinforcement learning (RL) represents a unique approach on how to produce programs that can take gathered experience, and turn it into policies able to complete complex tasks. What's more, its practical applications in autonomous vehicles, video-games and compiler optimization have proven its economic utility. However, alongside these potential applications lay a host of practical issues which stem from complicated algorithms, poor theoretical understanding, and the fundamental difficulty of the problem. To take full advantage of these methods as a research community, we need to better understand how these methods work, and why. To make this question more tractable, in this work we review a special class of RL algorithms known as "off-policy" algorithms, which have in recent years proven to be the most performant class of methods in deep RL (Haarnoja et al., 2019; Fujimoto et al., 2018; Lee et al., 2021a). These algorithms use experience gathered from all interactions with the environment to produce a better estimate of the expected cumulative reward; this contrasts with on-policy methods which only uses the most recent environment interaction or model based approaches which simulate examples using an internal model of the world. However, using this cache of experience effectively can be difficult in practice due to computational constraints and the bootstrapping error introduced by off-policy algorithms. These sources of error make changes to how interactions are added to replay buffer, and how previous interactions are sampled, important questions when presented with a new control problem; particularly in continuous control where both simulation and real world interactions can quickly become expensive.

We therefore consider how the sampling scheme affects performance of one of the most popular off-policy algorithms, Soft Actor-Critic (SAC) (Haarnoja et al., 2018). SAC is one of the most common baselines in deep reinforcement learning algorithms tailored to continuous control, and as we show below, can often be improved by properly chosen combinations of sampling methods and their hyper-parameters. We have chosen continuous control, because unlike discrete problems such as Atari, which have seen extensive experimentation across different sampling schemes, the effects of non-uniform sampling (NUS) on continuous control have remained largely ignored. In fact outside of broader sensitivity analyses which either exclude NUS (Islam et al., 2017; Henderson et al., 2018; Duan et al., 2016), or have focused specifically on on-policy methods (Engstrom et al., 2020; Andrychowicz et al., 2021), a modern practical survey on NUS for continuous control seems non-existent. Generally, the community has focused on the exploration method in the continuous control setting and assumes samples will be taken uniformly at random from the memory induced by the behavioral policy. Where to contrast with discrete control, PER has been shown to outperform other sampling schemes (Hessel et al., 2018; Schaul et al., 2016) across almost all standard benchmarks. To tackle this inconsistency, this paper covers the two broad classes of sampling algorithms for continuous control: generic priority based algorithms such as prioritized experience replay (PER) (Schaul et al., 2016), and recency based algorithms such emphasizing recent experience (ERE) (Wang & Ross, 2019). Picking the most popular variants of these two approaches, we answer the following questions:

1. Which sampling methods are most effective in the deterministic continuous control setting, and are any robust in the stochastic feedback setting?
2. How does the memory capacity, the maximum number of examples stored in the replay buffer, affect the performance of the algorithm. And can NUS improve performance in the low-capacity setting?
3. How do code-level design choices in NUS methods affect performance?
4. Which sampling methods are most robust to changes in hyper-parameters?
5. Can classic NUS algorithms like prioritized experience replay (Schaul et al., 2016) be improved through the use of different priority metrics?

In answering these questions, this paper first shows that while no individual sampler performs better then others across tasks in the deterministic setting, PER outperforms both uniform and ERE in most stochastic settings. Next we show that the memory capacity, independent of the NUS algorithm chosen, can destroy performance. And finally, we show that careful tuning of parameters like mini-batch size and number of policy updates can yield non-trivial improvement at a small increase in computational cost, while mixed sampling approaches for both the actor and critic networks beat the standard combined sampling approaches most commonly used in practice.

2 Background

2.1 Markov Decision Processes and Reinforcement Learning

We define a Markov decision process (MDP), $\mathcal{M}_\Phi(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}_0, \mathcal{T}, \Pi_\Phi)$, as a stochastic process which produces a sequence of tuples $\tau_t := \{a_t, s_t, s_{t+1}, r_t\}$, for a particular set of states $s_t \in \mathcal{S}$, actions $a_t \in \mathcal{A}$, initial state distribution $p(s_0) \in \mathcal{T}_0$, transition distribution $p(s_{t+1}|s_t, a_t) \in \mathcal{T}$, reward function $r_t : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, and policy $\pi_\phi \in \Pi_\phi : S \to A$ parameterized by $\phi \in \Phi$. The generative model for this finite horizon process is defined:

$$q_{\pi_\phi}(\tau) = p(s_0) \prod_{t=0}^{T} p(s_{t+1}|s_t, a_t)\pi_\phi(a_t|s_t). \tag{1}$$

We denote the marginal distribution with respect to a state s, under the trajectory distribution given in equation 1, as $q_{\pi_\phi}(s)$. In the finite horizon MDP setting, RL seeks to recover the policy π_ϕ, which maximizes expected cumulative reward over the trajectory induced, through seeking the solution of $\phi^* = \arg\max_{\phi\in\Phi} \mathbb{E}_{q_{\pi_\phi}(\tau)} \left[\sum_{t=0}^{T} r_t(s_t, a_t)\right]$. We can extend this problem class to an infinite horizon setting through the use of a discount factor which signifies the non-zero probability of termination after every time-step. In this setting it is common to arrive at a policy through the solution of a fixed point equation known as the bellman equation (Bertsekas, 2019; Sutton & Barto, 2018), by solving the following optimization problem:

$$\phi^* = \max_{\phi \in \Phi} \mathbb{E}_{s \sim d^{\pi_\phi}, a \sim \pi_\phi}[Q^{\pi_\phi}(a,s)], \quad \text{where} \quad d^{\pi_\phi}(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t q_{\pi_\phi}(s_t = s) \tag{2}$$

$$\text{and} \quad Q^{\pi_\phi}(a,s) = \mathbb{E}_{s' \sim p(s'|s,a)}\left[r(s,a) + \gamma \mathop{\mathbb{E}}_{\pi_\phi(a'|s')}[Q^{\pi_\phi}(a',s')] \right]. \tag{3}$$

Here the distribution over states, $d^{\pi_\phi}(s)$ in Equation 2, defines the *state occupancy*, while the function in Equation 3 defines the Q function. For practical purposes, the Q function is generally parameterized, and learned in an offline manner. In this offline setting the target of the Q function shifts from directly targeting the expected reward ahead of the current policy, and instead targets the expected reward ahead of a pessimistic approximation which is defined as the mixture over all previous behavior policies used to interact with the environment.

2.2 Soft Actor-Critic

One popular variant of this framework is known as soft actor-critic (SAC). This algorithm is similar to most off-policy actor critic algorithms, in that it learns to estimate the expected reward ahead given the state and action (the Q-function), using the bellman error:

$$F(\theta) = \mathop{\mathbb{E}}_{(a_t, s_t, s_{t+1}) \sim \mathcal{D}} \left[\left(Q_\theta^{\pi_\phi}(a_t, s_t) - (\hat{r}_t + \mathop{\mathbb{E}}_{a_{t+1} \sim \pi_\phi(a_{t+1}|s_{t+1})} \left[\gamma \bar{Q}_\theta^{\pi_\phi}(a_{t+1}, s_{t+1}) \right]) \right)^2 \right] \tag{4}$$

Here, D defines the set of examples gathered from the environment under a behavioral policy, or a sequence of behavioral policies. The primary distinction between SAC and other AC algorithms (Mnih et al., 2016; Fujimoto et al., 2018) however, is that r_t is replaced with a surrogate $\hat{r_t} = r_t - \mathcal{H}_t$, where $\mathcal{H}_t$ indicates the policy entropy at state s_t. Additionally in this setting, the bar over Q indicates a target network, generally defined by the use of parameter averaging (Mnih et al., 2013; Wang et al., 2016), a model ensemble (Chen et al., 2017), or both (Hessel et al., 2018). Using this Q function estimate, we can iteratively update our policy to match the so called "softmax-optimal" policy (Haarnoja et al., 2018, 2019; Levine, 2018). This approach defines its policy objective based on a divergence between the current policy and a normalized target Q function:

$$J_\theta(\phi) = \mathop{\mathbb{E}}_{s_t \sim \mathcal{D}} \left[\mathop{\mathbb{E}}_{a_t \sim \pi_\phi(a_t|s_t)} \left[Q_\theta^{\pi_\phi}(a_t, s_t) + \alpha \log\left(\frac{\pi_0(a_t|s_t)}{\pi_\phi(a_t|s_t)} \right) \right] \right], \tag{5}$$

Here the gradient is evaluated using the reparametrization trick (Le et al., 2017) in cases where the distribution over actions is reparameterizable (e.g. normal, gamma ect.), and with reinforce (Williams, 1992) otherwise. A description of SAC is included in Algorithm 1.

2.3 Prioritized Experience Replay

Prioritized Experience Replay (PER) (Schaul et al., 2016) is one of the most well known sampling methods in off-policy RL (Hessel et al., 2018), and was originally proposed for deep Q-learning (Hessel et al., 2018). PER assigns priorities to transition examples using absolute temporal difference(TD) error. The larger the absolute TD-error, the more likely the associated example is to be sampled. More specifically, the probability assigned to a transition is:

$$P(i) = \frac{p(i)^\alpha}{\sum_k p(k)^\alpha}, \quad \text{where} \quad p(i) = \mathop{\mathbb{E}}_{D}\left[\left| Q_\theta^{\pi_\phi}(a_t, s_t) - \left(r_t + \mathop{\mathbb{E}}_{\pi_\phi} \left[\gamma \bar{Q}_\theta^{\pi_\phi}(a_{t+1}, s_{t+1}) \right] \right) \right| \right] \tag{6}$$

In the original paper by Schaul et al. (2016), two prioritization schemes based on the above function are considered: a proportional prioritization, and a rank-based based prioritization. For the proportional prioritization $p(i) = |\delta_i| + \epsilon$ for the i$^{\text{th}}$ transition, where $|\delta_i|$ is the absolute TD-error and ϵ a small positive constant that ensures that $p(i) > 0 \; \forall \; i$. In rank-based prioritization $p(i) = \frac{1}{rank(i)}$, where the $rank(i)$ is sorted based on the absolute TD-error $|\delta_i|$. This work only considers proportional scoring scheme, as (Schaul et al., 2016) showed that, on average, it performed better than its ranked counterpart. Lastly, PER incorporates importance sampling to make the method more robust to noisy data, this is done by re-weighting the loss function by $w_i = (\frac{1}{N} \cdot \frac{1}{P(i)})^\beta$. Here N is the number of samples stored in the buffer and β is an annealing parameter which determines how close to uniform the sampler will be. The full PER algorithm is described in Algorithm 2.

2.4 Emphasizing Recent Experience

Emphasizing Recent Experience (ERE) (Wang & Ross, 2019) increases the importance of recently observed data by sampling more aggressively from recent experiences. This is done by uniformly sampling from the most recent c_k data points in the replay buffer, where for the implementation discussed in this paper, $c_k = \max(N \cdot \eta^{k\frac{1000}{K}}, M)$. Here, N defines the number of observations stored in the replay buffer total, while $\eta \in (0, 1]$ determines how aggressively to sample from recent data. K is included in cases where multiple (K) mini-batch updates are taken per environment interaction. The parameter M provides a fixed range of η values per update cycle, and helps to avoid over-fitting in small replay buffers. A more detailed description of the version of ERE used in this paper is included in Algorithm 3.

3 Related Work

Investigation into the effects of design decisions in experience replay has in recent years been sparse, with some exceptions. Notably for tabular and discrete settings, Zhang & Sutton (2017) explored the relationship of the size of the buffer capacity and model performance, while Liu & Zou (2018) explored the both effects of the buffer capacity and mini-batch size. Additionally, Fedus et al. (2020) studied the buffer capacity, and the effect of the model update frequency and demonstrated that larger buffers with higher update to sample ratio performed better on average in the Atari Arcade Learning Environment (Bellemare et al., 2013). To our knowledge, only a few extensions to continuous control have been made. Of these works, Wang & Ross (2019) studied the effects of replay buffer capacity on SAC (Haarnoja et al., 2018) where they found that decreased capacity again decreased in performance, while Islam et al. (2017) varied the batch size and found inconclusive results with respect to performance. Importantly, these studies were limited only to the uniform sampling, where we consider NUS methods.

Crucially for this paper, significant work has been done to find more suitable sampling schemes for different environmental settings. Hessel et al. (2018) reported PER (Schaul et al., 2016) as being one of the most significant modular improvements for DQN-like algorithms (Hessel et al., 2018; Mnih et al., 2013), with variations of it found in Lee et al. (2021b); Fujimoto et al. (2020). Other work like Ramicic & Bonarini (2017) defines new priority metrics for PER based on the perceived entropy of the state space, while Zha et al. (2019) and Oh et al. (2021) employed deep neural networks as a prioritization function to score stored transitions, learning both from local and global features. Other sampling schemes incorporate recency prioritization: Wang & Ross (2019) takes subsets of the buffer during mini-batch updates to increase the sampling frequency of more recent data, while Kong et al. (2021) samples over a half-normal distribution centered at the latest observation, and Zhang & Sutton (2017) includes the latest observation during model updates. More specialized sampling methods include Andrychowicz et al. (2017) and Fang et al. (2019), which are aimed to improve performance in challenging sparse environments using goal conditioning.

4 Experiments

Using Deepmind's Control Suite(Bellemare et al., 2013) we test Uniform, PER, and ERE sampling methods with SAC on 6 different tasks (WalkerWalk, CheetahRun, HumanoidStand, HumanoidRun, QuadrupedRun, HopperHop) using 5 random seeds for $1 \cdot 10^6$ environment interactions. The experiments are broken down into the following four subsections, which start with creating a baseline of algorithm performance for each sampling method (4.1). Next we consider sampling methods in the stochastic setting (4.2) where noise is added to the reward to measure relative performance from deterministic baselines. We then explore the sensitivity of standard replay buffer hyper-parameters(4.3) by varying: the buffer capacity, batch size, initial exploration steps, and number of policy updates. Lastly, we consider one possible degree of freedom with respect to implementation by varying priority metrics and splitting sampling procedures between actor and critic networks(4.4).

4.1 Baseline performance in Continuous Control

Starting with performance tests for Uniform, PER, and ERE with SAC on the Deepmind Control Suite Tassa et al. (2018) using standard hypeparameters found in Table 2. As shown in Figure 1, no single sampling method outperforms all the others across environments, though both ERE and PER seem to provide examples in which they significantly out-perform the other two samplers on a per-environment basis. In this setting, it seems clear that provided you search over NUS schemes, you can get non-trivial increases in performance over the uniform baseline.

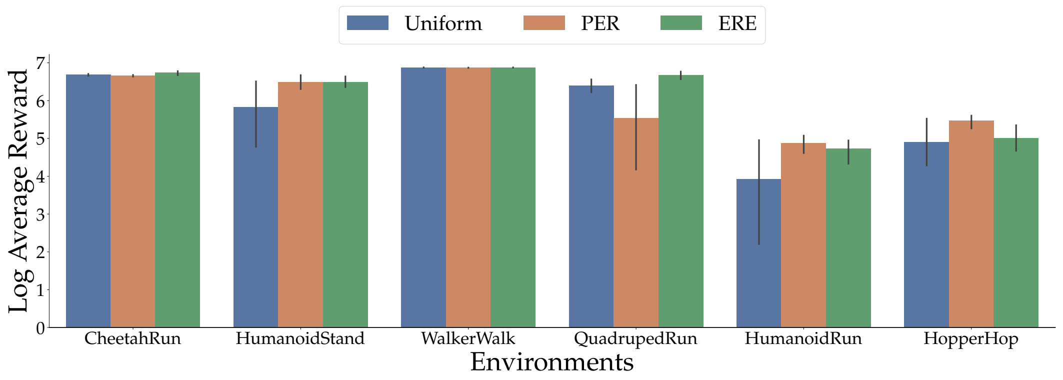

Figure 1: Baseline performance using Uniform, PER, and ERE sampling methods with SAC on 6 Deepmind Control tasks trained for 10^6 environment interactions. We report the log average reward and 95% confidence interval.

4.2 Sampling Methods in Stochastic Setting

Following the work of Romoff et al. (2018) we introduce noise in the reward function to each environment by sampling from three different noise distributions: Gaussian, Uniform and Sparse. For the Gaussian Noise example, we sample from a normal distribution $\Psi \sim N(0, \sigma^2)$ with increasing variance of $\sigma \in \{0.1, 0.2, 0.3, 0.4\}$ such that the new reward r_t^g at step t is calculated $r_t^g = r_t + \psi$. For the Uniform and Sparse noise the new rewards r_t^u and r_t^s are calculated respectively:

$$r_t^{\mathrm{u}} = \begin{cases} \xi, & \text{with probability } \epsilon_u \\ r_t, & \text{with probability } (1-\epsilon_u) \end{cases}, \quad r_t^{\mathrm{s}} = \begin{cases} 0, & \text{with probability } \epsilon_s \\ r_t, & \text{with probability } (1-\epsilon_s) \end{cases} \tag{7}$$

With $\xi \sim U(-1, 1)$ and probabilities $\epsilon_u \in \{0.1, 0.2, 0.3, 0.4\}$ and $\epsilon_s \in \{0.6, 0.7, 0.8, 0.9\}$. We compare the results by measuring the relative gain from the baseline which is calculated as

$$100 \times \frac{[\text{Stochastic Environment Performance}] - [\text{Deterministic Environment Performance}]}{|[\text{Deterministic Environment Performance}]|} \tag{8}$$

Each of the noise levels across 6 environments are included in Figure 2. This experiment, as expected, shows that the performance of each sampling method deteriorates as the noise becomes larger in magnitude. Interestingly, within the Gaussian Noise experiment QuadrupedRun environment $\sigma = 0.1$, and Sparsity experiment HopperHop environment with $\epsilon = 0.6$, PER and ERE respectively report a net positive gain, actually performing better than their respective noise-free baseline shown in Figure 1. This could indicate that some regularization might benefit learning in certain environments depending on the reward structure, or that additional seeds are required. Other than these anomalies, almost any noise appears to be catastrophic to the learning performance, though amongst sampling methods PER seems to perform the best in general. It should be noted however that PER does seem to be especially susceptible to the uniform noise regime, while both ERE and uniform are not. However the difference in performance in this one instance is somewhat negligible in comparison to the performance gains we see in other distributions of noise. This experiment therefore implies that without any prior knowledge of the distribution of noise which might apply to your environment, PER is the best choice of NUS algorithm.

4.3 Replay Buffer Hyperparameter Sensitivity Analysis

We now conduct a hyperparameter sensitivity analysis of replay buffers and their effect in Uniform, PER and ERE sampling methods. More specifically we will be considering the following settings:

- Decreasing Replay Buffer Capacity N_b to 10^5, 10^4, and 10^3

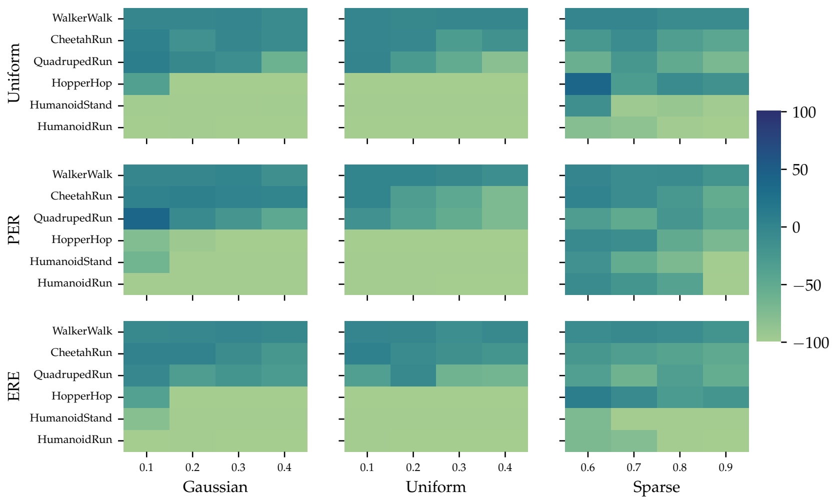

Figure 2: Relative gain of Uniform, PER, ERE sampling methods with added Gaussian, uniform and sparse noise on the reward function following Equation 7. Each example is trained for 10^6 environment interactions, and normalized using Equation 9. As noise intensity increases the learning performance of the algorithms drops substantially.

		Average Relative Gain		
	Parameter Value	Uniform	PER	ERE
Gaussian Reward Noise	$\sigma = 0.1$	-36.72	**-32.48**	-36.74
	$\sigma = 0.2$	-53.19	**-48.82**	-55.09
	$\sigma = 0.3$	**-52.12**	-52.89	-54.98
	$\sigma = 0.4$	-62.12	-59.51	**-59.07**
Uniform Reward Noise	$\epsilon = 0.1$	**-49.46**	-52.06	-54.40
	$\epsilon = 0.2$	-55.62	-60.89	**-52.04**
	$\epsilon = 0.3$	**-63.32**	-66.75	-64.30
	$\epsilon = 0.4$	-65.87	-75.51	**-64.87**
Sparse Reward Noise	$\epsilon = 0.6$	-22.48	**-9.66**	-33.40
	$\epsilon = 0.7$	-30.35	**-24.89**	-46.51
	$\epsilon = 0.8$	-47.26	**-35.60**	-51.35
	$\epsilon = 0.9$	**-55.65**	-63.42	-56.34

Table 1: Average relative gain of Uniform, PER, ERE sampling methods for different reward noise parameters over 6 Deepmind Control tasks shown in Figure 2

- Varying exploration steps E_s to 1000, 5000, 10000 ,and 25000
- Varying mini-batch size B_s to 128, 256, 512, and 1024
- Varying number of Policy updates P_u to 1, 3, 5, and 10

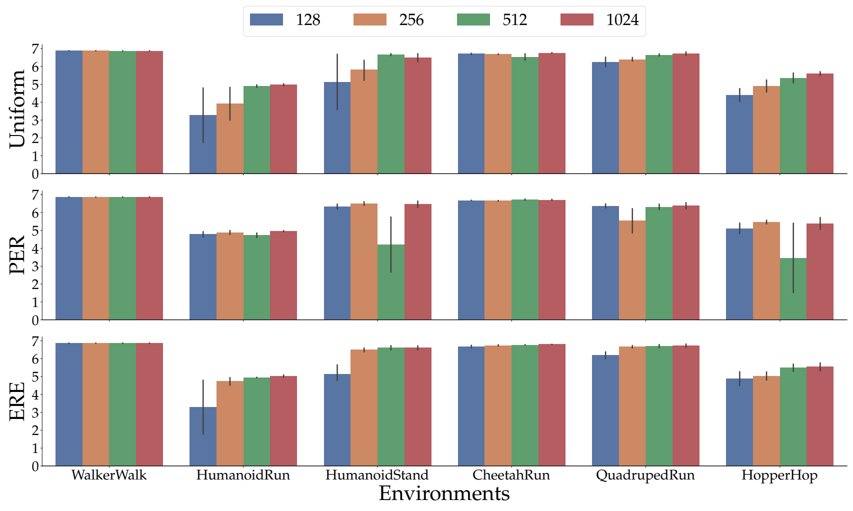

Figure 3: Varying mini-batch size $(B_s = (128, 256, 512, 1024))$ for Uniform, PER, ERE sampling methods on 6 Deepmind Control tasks trained over 10^6 environment interactions. Uniform and ERE benefit greatly from bigger mini-batch size, while PER handles smaller batches better.

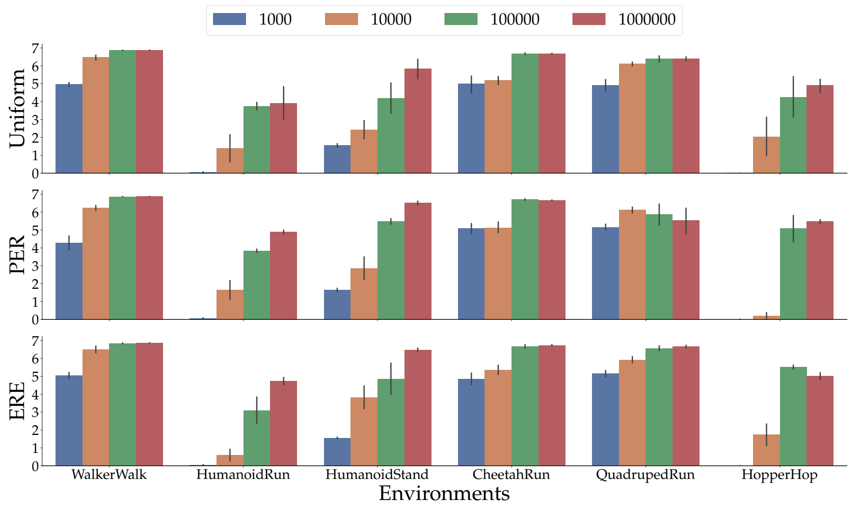

Figure 4: Varying replay buffer size $(N_b = (10^3, 10^4, 10^5, 10^6))$ for Uniform, PER, ERE sampling methods on 6 Deepmind Control tasks trained over 10^6 environment interactions. On average a larger replay buffer results to better learning performance.

These experiments confirm some of the existing hypothesis within the literature for continuous control in off-policy RL, while showing that these hypothesis do not quite hold for all NUS algorithms. Looking at the mini-batch size first, we see that while ERE and Uniform see monotonic improvement in the batch-size, there is a very negligible effect in the context of PER. This indicates that PER might be best suited to large models for which low batch sizes are required. Next, considering the size of the replay buffer, we can see that this is clearly one of the most important hyper-parameters of SAC independent of the sampling algorithm used. We also consider the effect of multiple policy steps per update where we find that there is approximately monotonic improvement as one increases the number of policy updates, though the improvement is generally negligible. This means that

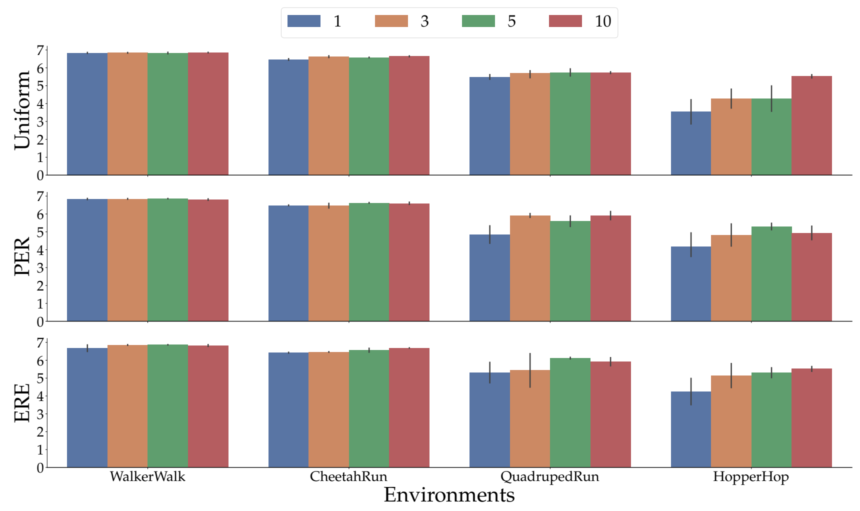

Figure 5: Varying the number of policy updates per environment interaction ($P_u = (1, 3, 5, 10)$ for Uniform, PER, ERE sampling methods on 4 Deepmind Control tasks trained over $0.25 \cdot 10^6$ environment interactions. Training steps were reduced relative to other experiments for computational efficiency. On average results show that more policy updates per environment interactions improve training performance.

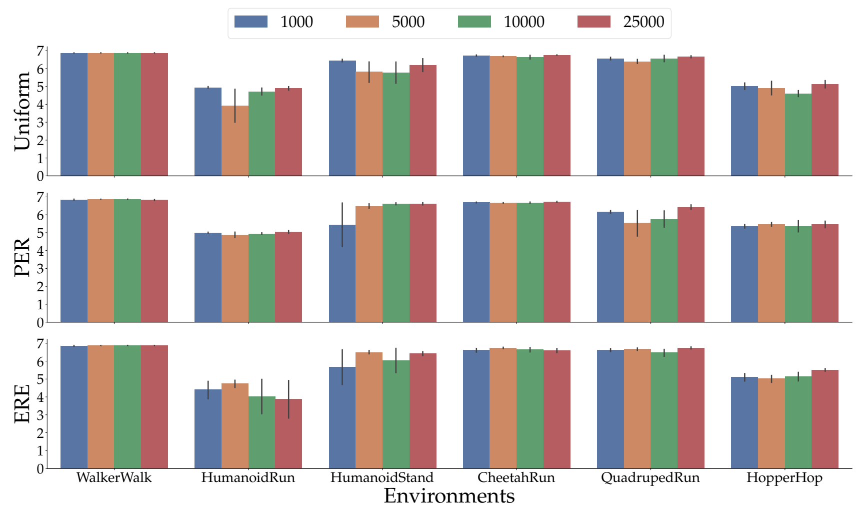

Figure 6: Varying initial exploration steps ($E_s = (10^3, 5 \cdot 10^3, 10^4, 2.5 \cdot 10^4)$ for Uniform, PER, ERE sampling methods on 6 Deepmind Control tasks trained over 10^6 environments. There are is no apparent trend throughout the sampling methods, showing the possible sensitivity of this hyperparameter.

throughout training, we are both targeting a useful objective, and that our policy remains close to the optimum (for any intermediate Q function) at any given step. The last plot, which varies exploration steps does not exhibit any major trends, indicating that tuning this parameter will be situation dependent.

4.4 Separate Sampling Methods and Varying Priority Metrics

In the final experiment we explore different variations of sampling method pairs. A central replay buffer is used for all variations, while each method per pair samples from this buffer independently from the other. We also explore different priority metrics for PER, like the mean-squared error from the critic's loss function, and the KL divergence from the actor's loss function. The results can be found in Figure 4 and are sorted by performance. Interestingly enough, a combination of ERE in the policy and uniform in the critic performs the best on average across standard benchmarks.

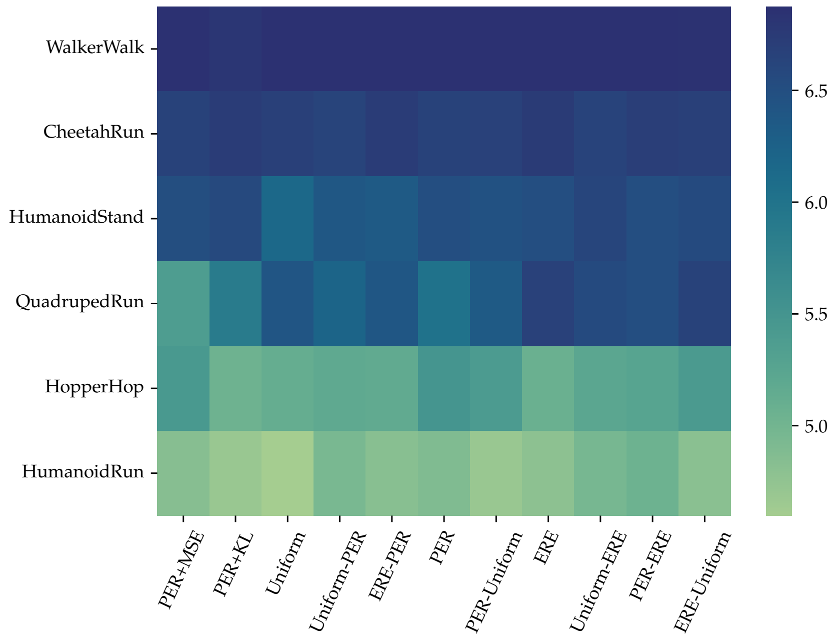

Figure 7: Log average reward of standard sampling methods (Uniform, PER, ERE), seperate sampling method per Actor-Critic network (Uniform-PER, Uniform-ERE, PER-Uniform, PER-ERE, ERE-Uniform, ERE-PER), and varying priority metrics for PER instead of TD-errror: PER using KL-Divergence (PER+KL), PER using Mean Squared Error (PER+MSE). Tested on 6 Deepmind Control tasks (WalkerWalk, CheetahRun, HumanoidStand, QuadrupedRun, HopperHop, HumanoidRun) over 10^6 environment interactions for 5 random seeds. Sampling methods are sorted by ascending order based on their average learning performance over the 6 tasks.

5 Conclusion

In this paper we investigated the behavior of NUS methods in off-policy RL. Our experiments showed the following results: first, in the deterministic setting no individual sampling method performed significantly better for all tasks. Second, in the stochastic setting PER performed significantly better then all other sampling methods considered. Third, changing hyper-parameters, particularly the memory capacity, can drastically change the relative performance of each algorithm. Lastly, by varying the types of samplers used for the actor and critic networks we gain non-trivial improvement in each of the environments over the single sampler baselines, indicating the optimal sampler used to update both the actor and critic networks might not be the same.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf`.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=nIAxjsniDzg`.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, May 2013. ISSN 1076-9757.

Dimitri Bertsekas. *Reinforcement and Optimal Control*. Athena Scientific, 2019.

Richard Y. Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles, 2017.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1329–1338, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/duan16.html`.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=r1etN1rtPB`.

Meng Fang, Cheng Zhou, Bei Shi, Boqing Gong, Weitao Xi, Tianzhou Wang, Jia Xu, and Tong Zhang. DHER: Hindsight experience replay for dynamic goals. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Byf5-30qFX`.

W. Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, H. Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *ICML*, 2020.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596, StockholmsmÃ¤ssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/fujimoto18a.html`.

Scott Fujimoto, David Meger, and Doina Precup. An equivalence between loss functions and non-uniform sampling in experience replay, 2020.

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*, pp. 1861–1870. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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.

Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Seung-Hyun Kong, I. Made Aswin Nahrendra, and Dong-Hee Paek. Enhanced off-policy reinforcement learning with focused experience replay. *IEEE Access*, 9:93152–93164, 2021. doi: 10.1109/ACCESS.2021.3085142.

Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential monte carlo. *arXiv preprint arXiv:1705.10306*, 2017.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 6131–6141. PMLR, 18–24 Jul 2021a. URL `https://proceedings.mlr.press/v139/lee21g.html`.

Sanghwa Lee, Jaeyoung Lee, and Ichiro Hasuo. Predictive per: balancing priority and diversity towards stable deep reinforcement learning. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10. IEEE, 2021b.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Ruishan Liu and James Zou. The effects of memory replay in reinforcement learning. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 478–485. IEEE, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. 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. PMLR, 2016.

Youngmin Oh, Kimin Lee, Jinwoo Shin, Eunho Yang, and Sung Ju Hwang. Learning to sample with local and global contexts in experience replay buffer. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=gJYlaqL8i8`.

Mirza Ramicic and Andrea Bonarini. Entropy-based prioritized sampling in deep q-learning. In *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, pp. 1068–1072, 2017. doi: 10.1109/ICIVC.2017.7984718.

Joshua Romoff, Peter Henderson, Alexandre Piche, Vincent Francois-Lavet, and Joelle Pineau. Reward estimation for variance reduction in deep reinforcement learning. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto (eds.), *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pp. 674–699. PMLR, 29–31 Oct 2018. URL `https://proceedings.mlr.press/v87/romoff18a.html`.

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

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

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.

Che Wang and Keith W. Ross. Boosting soft actor-critic: Emphasizing recent experience without forgetting the past. *CoRR*, abs/1906.04009, 2019. URL `http://arxiv.org/abs/1906.04009`.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pp. 1995–2003. PMLR, 2016.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and Xia Hu. Experience replay optimization. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 4243–4249. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/589. URL `https://doi.org/10.24963/ijcai.2019/589`.

Shangtong Zhang and Richard S Sutton. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.