Striving for Simplicity and Performance in Off-Policy DRL: Output Normalization and Non-Uniform Sampling

Che Wang [* 1 2] **Yanqiu Wu** [* 1 2] **Quan Vuong** [3] **Keith Ross** [1 2]

Abstract

We aim to develop off-policy DRL algorithms that not only exceed state-of-the-art performance but are also simple and minimalistic. For standard continuous control benchmarks, Soft Actor-Critic (SAC), which employs entropy maximization, currently provides state-of-the-art performance. We first demonstrate that the entropy term in SAC addresses action saturation due to the bounded nature of the action spaces, with this insight, we propose a streamlined algorithm with a simple normalization scheme or with inverted gradients. We show that both approaches can match SAC's sample efficiency performance without the need of entropy maximization, we then propose a simple non-uniform sampling method for selecting transitions from the replay buffer during training. Extensive experimental results demonstrate that our proposed sampling scheme leads to state of the art sample efficiency on challenging continuous control tasks. We combine all of our findings into one simple algorithm, which we call Streamlined Off Policy with Emphasizing Recent Experience, for which we provide robust public-domain code.

1. Introduction

Off-policy Deep Reinforcement Learning (RL) algorithms aim to improve sample efficiency by reusing past experience. Recently a number of new off-policy Deep RL algorithms have been proposed for control tasks with continuous state and action spaces, including Deep Deterministic Policy Gradient (DDPG) and Twin Delayed DDPG (TD3)

(Lillicrap et al., 2015; Fujimoto et al., 2018). TD3, which introduced clipped double-Q learning, delayed policy updates and target policy smoothing, has been shown to be significantly more sample efficient than popular on-policy methods for a wide range of MuJoCo benchmarks.

The field of Deep Reinforcement Learning (DRL) has also recently seen a surge in the popularity of maximum entropy RL algorithms. In particular, Soft Actor-Critic (SAC), which combines off-policy learning with maximum-entropy RL, not only has many attractive theoretical properties, but can also give superior performance on a wide-range of MuJoCo environments, including on the high-dimensional environment Humanoid for which both DDPG and TD3 perform poorly (Haarnoja et al., 2018a;b; Langlois et al., 2019). SAC and TD3 have similar off-policy structures with clipped double-Q learning, but SAC also employs maximum entropy reinforcement learning.

In this paper, we aim to develop off-policy DRL algorithms that not only provide state-of-the-art performance but are also simple and minimalistic. We first seek to understand the primary contribution of the entropy term to the performance of maximum entropy algorithms. For the MuJoCo benchmark, we demonstrate that when using the standard objective without entropy along with standard additive noise exploration, there is often insufficient exploration due to the bounded nature of the action spaces. Specifically, the outputs of the policy network are often way outside the bounds of the action space, so that they need to be squashed to fit within the action space. The squashing results in actions persistently taking on their maximal values, resulting in insufficient exploration. In contrast, the entropy term in the SAC objective forces the outputs to have sensible values, so that even with squashing, exploration is maintained. We conclude that, for the MuJoCo environments, the entropy term in the objective for Soft Actor-Critic principally addresses the bounded nature of the action spaces.

With this insight, we propose the Streamlined Off Policy (SOP) algorithm, which is a minimalistic off-policy algorithm that includes a simple but crucial output normalization. The normalization addresses the bounded nature of the action spaces, allowing satisfactory exploration throughout training. We also consider using inverting gra-

---

[*]Equal contribution [1]Department of Computer Science, New York University, New York, NY, USA [2]Department of Computer Science, NYU Shanghai, Shanghai, China [3]Department of Computer Science, University of California San Diego, San Diego, CA, USA. Correspondence to: Keith Ross <keith-wross@nyu.edu>.

dients (IG) (Hausknecht & Stone, 2015) with the streamlined scheme, which we refer to as SOP_IG. Both approaches use the standard objective without the entropy term. Our results show that SOP and SOP_IG match the sample efficiency and robust performance of SAC, including on the challenging Ant and Humanoid environments.

Having matched SAC performance without using entropy maximization, we then seek to attain state-of-the-art performance by employing a non-uniform sampling method for selecting transitions from the replay buffer during training. Priority Experience Replay (PER), a non-uniform sampling scheme, has been shown to significantly improve performance for the Atari games benchmark (Schaul et al., 2015), but requires sophisticated data structure for efficient sampling. Keeping with the theme of simplicity with the goal of meeting Occam's principle, we propose a novel and simple non-uniform sampling method for selecting transitions from the replay buffer during training. Our method, called Emphasizing Recent Experience (ERE), samples more aggressively recent experience while not neglecting past experience. Unlike PER, ERE is only a few lines of code and does not rely on any sophisticated data structures. We show that when SOP, SOP_IG, or SAC is combined with ERE, the resulting algorithm out-performs SAC and provides state of the art performance. For example, for Ant and Humanoid, SOP+ERE improves over SAC by 21% and 24%, respectively, with one million samples.

The contributions of this paper are thus threefold. First, we uncover the primary contribution of the entropy term of maximum entropy RL algorithms for the MuJoCo environments. Second, we propose a streamlined algorithm which does not employ entropy maximization but nevertheless matches the sampling efficiency and robust performance of SAC for the MuJoCo benchmarks. And third, we propose a simple non-uniform sampling scheme to achieve state-of-the art performance for the MuJoCo benchmarks. We provide public code for SOP+ERE for reproducibility [1].

2. Preliminaries

We represent an environment as a Markov Decision Process (MDP) which is defined by the tuple $(\mathcal{S}, \mathcal{A}, r, p, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are continuous multi-dimensional state and action spaces, $r(s, a)$ is a bounded reward function, $p(s'|s, a)$ is a transition function, and γ is the discount factor. Let $s(t)$ and $a(t)$ respectively denote the state of the environment and the action chosen at time t. Let $\pi = \pi(a|s)$, $s \in \mathcal{S}, a \in \mathcal{A}$ denote the policy. We further denote K for the dimension of the action space, and write a_k for the kth component of an action $a \in \mathcal{A}$, that is, $a = (a_1, \ldots, a_K)$.

The expected discounted return for policy π beginning in state s is given by:

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s(t), a(t))|s(0) = s] \qquad (1)$$

Standard MDP and RL problem formulations seek to maximize $V_\pi(s)$ over policies π. For finite state and action spaces, under suitable conditions for continuous state and action spaces, there exists an optimal policy that is deterministic (Puterman, 2014; Bertsekas & Tsitsiklis, 1996). In RL with unknown environment, exploration is required to learn a suitable policy.

In DRL with continuous action spaces, typically the policy is modeled by a parameterized policy network which takes as input a state s and outputs a value $\mu(s; \theta)$, where θ represents the current parameters of the policy network (Schulman et al., 2015; 2017; Vuong et al., 2018; Lillicrap et al., 2015; Fujimoto et al., 2018). During training, typically additive random noise is added for exploration, so that the actual action taken when in state s takes the form $a = \mu(s; \theta) + \epsilon$ where ϵ is a K-dimensional Gaussian random vector with each component having zero mean and variance σ. During testing, ϵ is set to zero.

2.1. Maximum Entropy Reinforcement Learning

Maximum entropy reinforcement learning takes a different approach than Equation (1) by optimizing policies to maximize both the expected return and the expected entropy of the policy (Ziebart et al., 2008; Ziebart, 2010; Todorov, 2008; Rawlik et al., 2013; Levine & Koltun, 2013; Levine et al., 2016; Nachum et al., 2017; Haarnoja et al., 2017; 2018a;b).

In particular, the maximum entropy RL objective is:

$$V_\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi[r(s(t), a(t))$$
$$+ \lambda H(\pi(\cdot|s(t)))|s(0) = s]$$

where $H(\pi(\cdot|s))$ is the entropy of the policy when in state s, and the temperature parameter λ determines the relative importance of the entropy term against the reward. For maximum entropy DRL, when given state s the policy network will typically output a K-dimensional vector $\sigma(s; \theta)$ in addition to the vector $\mu(s; \theta)$. The action selected when in state s is then modeled as $\mu(s; \theta) + \epsilon$ where $\epsilon \sim N(0, \sigma(s; \theta))$.

Maximum entropy RL has been touted to have a number of conceptual and practical advantages for DRL (Haarnoja et al., 2018a;b). For example, it has been argued that the policy is incentivized to explore more widely, while giving

---

[1] https://github.com/AutumnWu/Streamlined-Off-Policy-Learning

up on clearly unpromising avenues. It has also been argued that the policy can capture multiple modes of near-optimal behavior, that is, in problem settings where multiple actions seem equally attractive, the policy will commit equal probability mass to those actions. In this paper, we show for the MuJoCo benchmarks that the standard additive noise exploration suffices and can achieve the same performance as maximum entropy RL.

3. The Squashing Exploration Problem

3.1. Bounded Action Spaces

Continuous environments typically have bounded action spaces, that is, along each action dimension k there is a minimum possible action value $a_k^{\min}$ and a maximum possible action value $a_k^{\max}$. When selecting an action, the action needs to be selected within these bounds before the action can be taken. DRL algorithms often handle this by squashing the action so that it fits within the bounds. For example, if along any one dimension the value $\mu(s;\theta) + \epsilon$ exceeds $a^{\max}$, the action is set (clipped) to $a^{\max}$. Alternatively, a smooth form of squashing can be employed. For example, suppose $a_k^{\min} = -M$ and $a_k^{\max} = +M$ for some positive number M, then a smooth form of squashing could use $a = M \tanh(\mu(s;\theta) + \epsilon)$ in which $\tanh()$ is being applied to each component of the K-dimensional vector. DDPG (Hou et al., 2017) and TD3 (Fujimoto et al., 2018) use clipping, and SAC (Haarnoja et al., 2018a;b) uses smooth squashing with the $\tanh()$ function. For concreteness, henceforth we will assume that smooth squashing with the $\tanh()$ is employed.

We note that an environment may actually allow the agent to input actions that are outside the bounds. In this case, the environment will typically first clip the actions internally before passing them on to the "actual" environment (Fujita & Maeda, 2018).

We now make a simple but crucial observation: squashing actions to fit into a bounded action space can have a disastrous effect on additive-noise exploration strategies. To see this, let the output of the policy network be $\mu(s) = (\mu_1(s), \ldots, \mu_K(s))$. Consider an action taken along one dimension k, and suppose $\mu_k(s) >> 1$ and $|\epsilon_k|$ is relatively small compared to $\mu_k(s)$. Then the action $a_k = M \tanh(\mu_k(s) + \epsilon_k)$ will be very close (essentially equal) to M. If the condition $\mu_k(s) >> 1$ persists over many consecutive states, then a_k will remain close to 1 for all these states, and consequently there will be essentially no exploration along the kth dimension. We will refer to this problem as the *squashing exploration problem*. We will argue that algorithms using the standard objective (Equation 1) with additive noise exploration can be greatly impaired by squashing exploration.

3.2. How Does Entropy Maximization Help for the MuJoCo Environments?

SAC is a maximum-entropy off-policy DRL algorithm which provides good performance across all of the MuJoCo benchmark environments. To the best of our knowledge, it currently provides state of the art performance for the MuJoCo benchmark. In this section, we argue that the principal contribution of the entropy term in the SAC objective is to resolve the squashing exploration problem, thereby maintaining sufficient exploration when facing bounded action spaces. To argue this, we consider two DRL algorithms: SAC with adaptive temperature (Haarnoja et al., 2018b), and SAC with entropy removed altogether (temperature set to zero) but everything else the same. We refer to them as *SAC* and as *SAC without entropy*. For SAC without entropy, for exploration we use additive zero-mean Gaussian noise with σ fixed at 0.3. Both algorithms use $\tanh$ squashing. We compare these two algorithms on two MuJoCo environments: Humanoid-v2 and Walker-v2.

Figure 1 shows the performance of the two algorithms with 10 seeds. For Humanoid, SAC performs much better than SAC without entropy. However, for Walker, SAC without entropy performs nearly as well as SAC, implying maximum entropy RL is not as critical for this environment.

To understand why entropy maximization is important for one environment but less so for another, we examine the actions selected when training these two algorithms. Humanoid and Walker have action dimensions $K = 17$ and $K = 6$, respectively. Here we show representative results for one dimension for both environments. The top and bottom rows of Figure 2 shows results for Humanoid and Walker, respectively. The first column shows the μ_k values for an interval of 1,000 consecutive time steps, namely, for time steps 599,000 to 600,000. The second column shows the actual action values passed to the environment for these time steps. The third and fourth columns show a concatenation of 10 such intervals of 1000 time steps, with each interval coming from a larger interval of 100,000 time steps.

The top and bottom rows of Figure 2 are strikingly different. For Humanoid using SAC with entropy, the $|\mu_k|$ values are small, mostly in the range [-1.5,1.5], and fluctuate significantly. This allows the action values to also fluctuate significantly, providing exploration in the action space. On the other hand, for SAC without entropy the $|\mu_k|$ values are typically huge, most of which are well outside the interval [-10,10]. This causes the actions a_k to be persistently clustered at either M or $-M$, leading to essentially no exploration along that dimension. For Walker, we see that for both algorithms, the μ_k values are sensible, mostly in the range [-1,1] and therefore the actions chosen by both algorithms exhibit exploration.
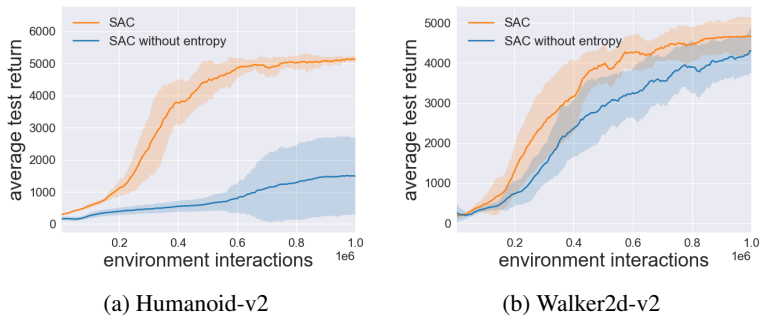
(a) Humanoid-v2

(b) Walker2d-v2

Figure 1: SAC performance with and without entropy maximization
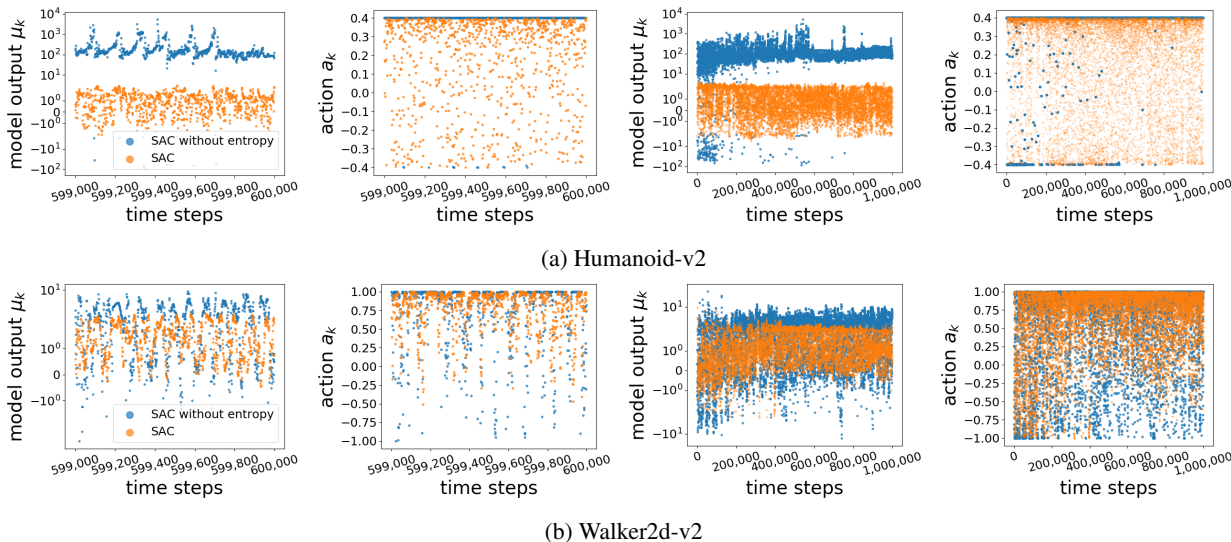

(a) Humanoid-v2

(b) Walker2d-v2

Figure 2: μ_k and a_k values from SAC and SAC without entropy maximization. See section 3.2 for a discussion.

In conclusion, the principal benefit of maximum entropy RL in SAC for the MuJoCo environments is that it resolves the squashing exploration problem. For some environments (such as Walker), the outputs of the policy network take on sensible values, so that sufficient exploration is maintained and overall good performance is achieved without the need for entropy maximization. For other environments (such as Humanoid), entropy maximization is needed to reduce the magnitudes of the outputs so that exploration is maintained and overall good performance is achieved.

4. Matching SOTA Performance without Entropy Maximization

In this paper we examine two approaches for matching SAC performance without using entropy maximization.

4.1. Output Normalization

As we observed in the previous section, in some environments the policy network output values $|\mu_k|$, $k = 1, \ldots, K$

can become persistently huge, which leads to insufficient exploration due to the squashing. We propose a simple solution of normalizing the outputs of the policy network when they collectively (across the action dimensions) become too large. To this end, let $\mu = (\mu_1, \ldots, \mu_K)$ be the output of the original policy network, and let $G = \sum_k |\mu_k|/K$. The G is simply the average of the magnitudes of the components of μ. The normalization procedure is as follows. If $G > 1$, then we reset $\mu_k \leftarrow \mu_k/G$ for all $k = 1, \ldots, K$; otherwise, we leave μ unchanged. With this simple normalization, we are assured that the average of the normalized magnitudes is never greater than one.

Our Streamlined Off Policy (SOP) algorithm is described in Algorithm 1. The algorithm is essentially TD3 minus the delayed policy updates and the target policy parameters but with the addition of the normalization described above. SOP also uses tanh squashing instead of clipping, since tanh gives somewhat better performance in our experiments. The SOP algorithm is "streamlined" as it has no entropy terms, temperature adaptation, target policy pa-
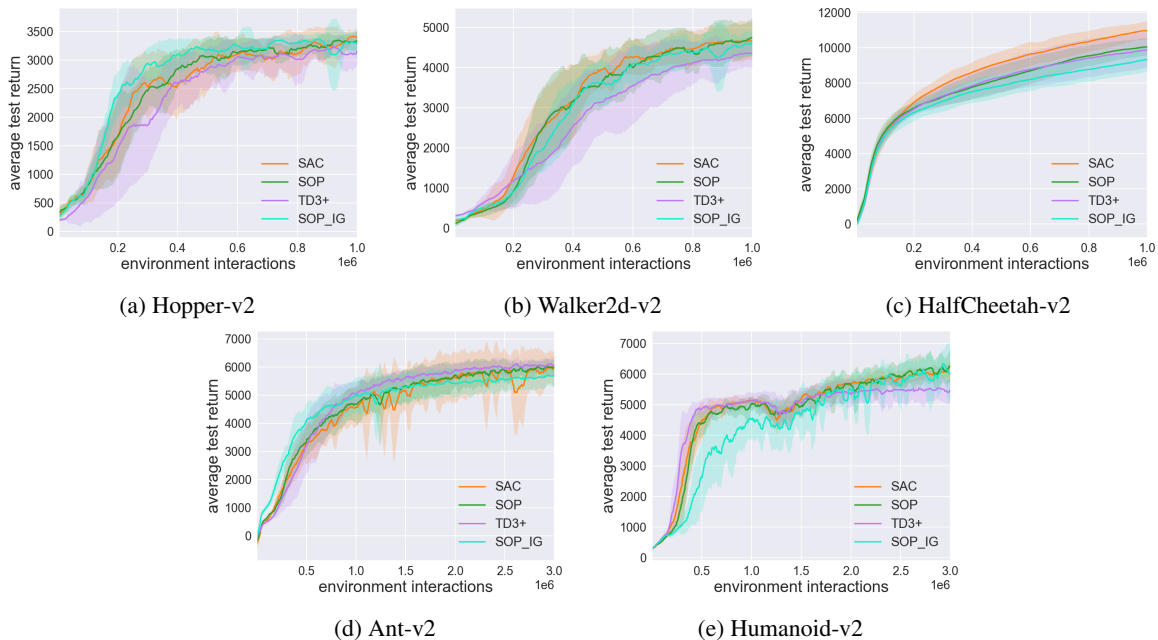
(a) Hopper-v2   (b) Walker2d-v2   (c) HalfCheetah-v2

(d) Ant-v2   (e) Humanoid-v2

Figure 3: Streamlined Off-Policy (SOP) versus SAC, SOP_IG and TD3

rameters or delayed policy updates.

4.2. Inverting Gradients

In our experiments, we also consider using SOP but replacing the output normalization with the IG scheme (Hausknecht & Stone, 2015). In this scheme, when gradients suggest increasing the action magnitudes, gradients are down scaled if actions are within the boundaries, and inverted if otherwise. More specifically, let p be the output of the last layer of the policy network, let $p_{\min}$ and $p_{\max}$ be the action boundaries. The IG approach can be summarized as follows (Hausknecht & Stone, 2015):

$$\nabla_p = \nabla_p \cdot \begin{cases} \frac{p_{\max}-p}{p_{\max}-p_{\min}} & \text{if } \nabla_p \text{ suggests} \\ & \text{increasing } p \\ \frac{p-p_{\min}}{p_{\max}-p_{\min}} & \text{otherwise} \end{cases} \quad (2)$$

Where ∇_p is the gradient of the policy loss w.r.t to p. Although IG is not complicated, it is not as simple and straightforward as simply normalizing the outputs. We refer to SOP with IG as SOP_IG. Implementation details can be found in the supplementary materials.

4.3. Experimental Results for SOP and SOP_IG

Figure 3 compares SAC (with temperature adaptation (Haarnoja et al., 2018a;b)) with SOP, SOP_IG, and TD3 plus the simple normalization (which we call TD3+) for five of the most challenging MuJoCo environments. Using the same baseline code, we train each of the algorithms with 10 seeds. Each algorithm performs five evaluation

rollouts every 5000 environment steps. The solid curves correspond to the mean, and the shaded region to the standard deviation of the returns over seeds. Results show that SOP, the simplest of all the schemes, performs as well or better than all other schemes. In particular, SAC and SOP have similar sample efficiency and robustness across all environments. TD3+ has slightly weaker asymptotic performance for Walker and Humanoid. SOP_IG initially learns slowly for Humanoid with high variance across random seeds, but gives similar asymptotic performance. These experiments confirm that the performance of SAC can be achieved without maximum entropy RL.

4.4. Ablation Study for SOP

In this ablation study, we separately examine the importance of (i) the normalization at the output of the policy network; (ii) the double Q networks; (iii) and randomization used in the line 9 of the SOP algorithm (that is, target policy smoothing (Fujimoto et al., 2018)).

Figure 4 shows the results for the five environments considered in this paper. In Figure 4, "no normalization" is SOP without the normalization of the outputs of the policy network; "single Q" is SOP with one Q-network instead of two; and "no smoothing" is SOP without the randomness in line 8 of the algorithm.

Figure 4 confirms that double Q-networks are critical for obtaining good performance (Van Hasselt et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018a). Figure 4 also shows that output normalization is critical. Without out-

put normalization, performance fluctuates wildly, and average performance can decrease dramatically, particularly for Humanoid and HalfCheetah. Target policy smoothing improves performance by a relatively small amount.

In addition, to better understand whether the simple normalization term in SOP achieves a similar effect compared to explicitly maximizing entropy, we plot the entropy values for SOP and SAC throughout training for all environments. We found that SOP and SAC have very similar entropy values across training, while removing the entropy term from SAC makes the entropy value much lower. This indicates that the effect of the action normalization is very similar to maximizing entropy. The results can be found in the supplementary materials.

5. Non-Uniform Sampling

In the previous section we showed that SOP, SOP_IG, and SAC all offer roughly equivalent sample-efficiency performance, with SOP being the simplest of the algorithms. We now show how a small change in the sampling scheme, which can be applied to any off-policy scheme (including SOP, SOP_IG and SAC), can achieve state of the art performance for the MuJoCo benchmark. We call this non-uniform sampling scheme Emphasizing Recent Experience (ERE). ERE has 3 core features: (i) It is a general method applicable to any off-policy algorithm; (ii) It requires no special data structure, is very simple to implement, and has near-zero computational overhead; (iii) It only introduces one additional important hyper-parameter.

The basic idea is that during the parameter update phase, the first mini-batch is sampled from the entire buffer, then for each subsequent mini-batch we gradually reduce our range of sampling to sample more from recent data. Specifically, assume that in the current update phase we are to make 1000 mini-batch updates. Let N be the max size of the buffer. Then for the kth update, we sample uniformly from the most recent c_k data points, where $c_k = N \cdot \eta^k$ and $\eta \in (0, 1]$ is a hyper-parameter that determines how much emphasis we put on recent data. $\eta = 1$ is uniform sampling. When $\eta < 1$, c_k decreases as we perform each update. η can be made to adapt to the learning speed of the agent so that we do not have to tune it for each environment. The algorithmic and implementation details of such an adaptive scheme is given in the supplementary material.

The effect of such a sampling formulation is twofold. The first is recent data have a higher chance of being sampled. The second is that sampling is done in an ordered way: we first sample from all the data in the buffer, and gradually shrink the range of sampling to only sample from the most recent data. This scheme reduces the chance of overwriting parameter changes made by new data with parameter changes made by old data (French, 1999; McClelland et al., 1995; McCloskey & Cohen, 1989; Ratcliff, 1990; Robins, 1995). This allows us to quickly obtain information from recent data, and better approximate the value functions near recently-visited states, while still maintaining an acceptable approximation near states visited in the more distant past.

What is the effect of replacing uniform sampling with ERE? First note if we uniformly sample several times from a fixed buffer (uniform fixed), where the buffer is filled, and no new data is coming in, then the expected number of times a data point has been sampled is the same for all data points.

Now consider a scenario where we have a buffer of size 1000 (FIFO queue), we collect one data at a time, and then perform one update with mini-batch size of one. If we start with an empty buffer and sample uniformly (uniform empty), as data fills the buffer, each data point gets less and less chance of being sampled. Specifically, start from timestep 0, over a period of 1000 updates, the expected number of times the tth data (the data point collected at tth timestep) has been sampled is: $\frac{1}{t} + \frac{1}{t+1} + \cdots + \frac{1}{1000}$. And if we start with a filled buffer and sample uniformly (uniform full), then the expected number of times the tth data has been sampled is $\sum_{t'=t}^{1000} \frac{1}{1000} = \frac{1000-t}{1000}$.

Figure 5f shows the expected number of times a data point has been sampled (at the end of 1000 updates) as a function of its position in the buffer. We see that when uniform sampling is used, older data are expected to get sampled much more than newer data, especially in the empty buffer case. This is undesirable because when the agent is improving and exploring new areas of the state space; new data points may contain more interesting information than the old ones, which have already been updated many times.

When we apply the ERE scheme, we effectively skew the curve towards assigning higher expected number of samples for the newer data, allowing the newer data to be frequently sampled soon after being collected, which can accelerate the learning process. In Figure 5f we can see that the curves for ERE (ERE empty and ERE full) are much closer to the horizontal line (Uniform fixed), compared to when uniform sampling is used. With ERE, at any point during training, we expect all data points currently in the buffer to have been sampled approximately the same number of times. Simply using a smaller buffer size will also allow recent data to be sampled more often, and can sometimes lead to a slightly faster learning speed in the early stage. However, it also tends to reduce the stability of learning, and damage long-term performance.

Another simple method is to sample data according to an exponential scheme, where more recent data points are as-

(a) Hopper-v2   (b) Walker2d-v2   (c) HalfCheetah-v2
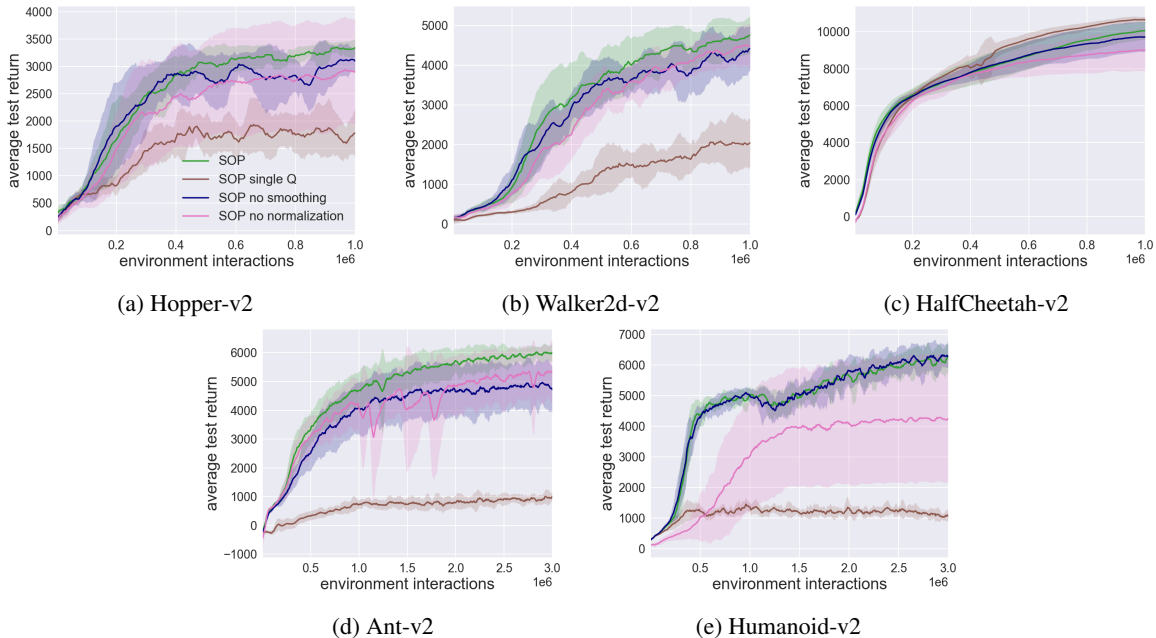
(d) Ant-v2   (e) Humanoid-v2

Figure 4: Ablation Study for SOP

signed exponentially higher probability of being sampled. In the supplementary materials, we provide further algorithmic detail and analysis on ERE, and compare ERE to the exponential sampling scheme, and show that ERE provides a stronger performance improvement. We also compare to another sampling scheme called Prioritized Experience Replay (PER) (Schaul et al., 2015). PER assigns higher probability to data points that give a high absolute TD error when used for the Q update, then it applies an importance sampling weight according to the probability of sampling. Performance comparison can also be found in the supplementary materials. Results show that in the MuJoCo environments, PER can sometimes give a performance gain, but it is not as strong as ERE and the exponential scheme.

5.1. Experimental Results for ERE

Figure 5 compares the performance of SAC (considered the baseline here), SAC+ERE, SOP+ERE, and SOP_IG+ERE. ERE gives a significant boost to all three algorithms, surpassing SAC and achieving a new SOTA. Among the three algorithms, SOP+ERE gives the best performance for Ant and Humanoid (the two most challenging environments) and performance roughly equivalent to SAC+ERE and SOP_IG+ERE for the other three environments.

In particular, for Ant and Humanoid, SOP+ERE improves performance by 21% and 24% over SAC at 1 million samples, respectively. For Humanoid, at 3 million samples, SOP+ERE improves performance by 15%. In conclusion,

SOP+ERE is not only a simple algorithm, but also exceeds state-of-the-art performance.

6. Related Work

In recent years, there has been significant progress in improving the sample efficiency of DRL for continuous robotic locomotion tasks with off-policy algorithms (Lillicrap et al., 2015; Fujimoto et al., 2018; Haarnoja et al., 2018a;b). There is also a significant body of research on maximum entropy RL methods (Ziebart et al., 2008; Ziebart, 2010; Todorov, 2008; Rawlik et al., 2013; Levine & Koltun, 2013; Levine et al., 2016; Nachum et al., 2017; Haarnoja et al., 2017; 2018a;b). Ahmed et al. (2019) very recently shed light on how entropy leads to a smoother optimization landscape. By taking clipping in the MuJoCo environments explicitly into account, Fujita & Maeda (2018) modified the policy gradient algorithm to reduce variance and provide superior performance among on-policy algorithms. Eisenach et al. (2018) extend the work of Fujita & Maeda (2018) for when an action may be direction. Hausknecht & Stone (2015) introduce Inverting Gradients, for which we provide experimintal results in this paper for the MuJoCo environments. Chou et al. (2017) also explores DRL in the context of bounded action spaces. Dalal et al. (2018) consider safe exploration in the context of constrained action spaces.

Experience replay (Lin, 1992) is a simple yet powerful method for enhancing the performance of an off-policy DRL algorithm. Experience replay stores past experience
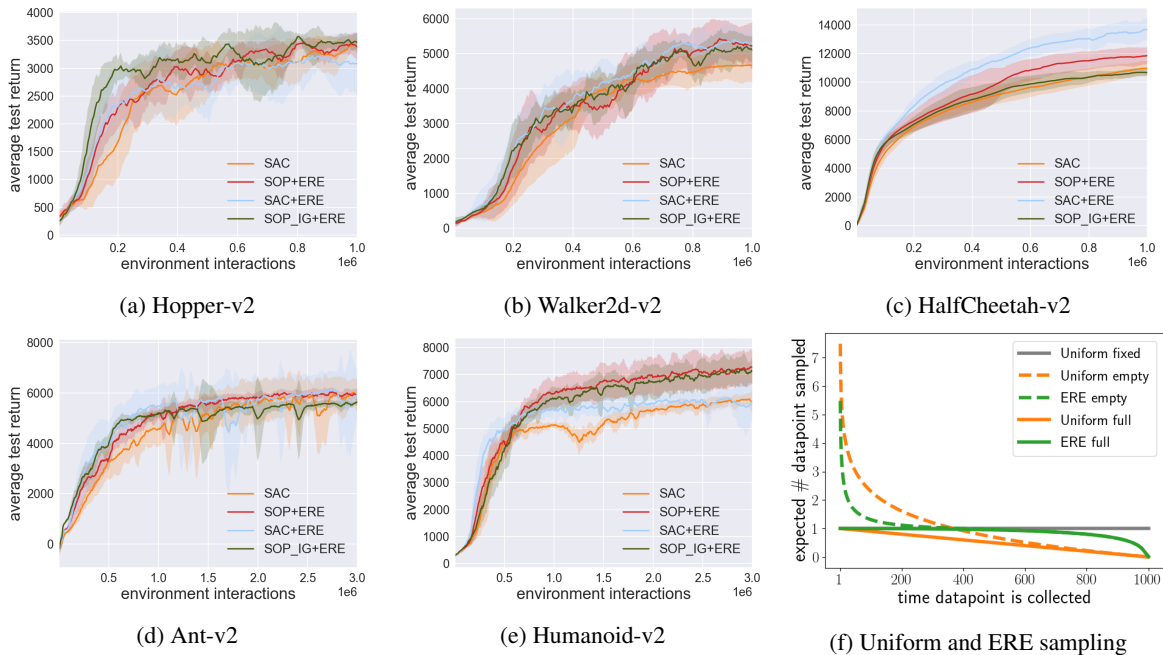
Figure 5: (a) to (e) show the performance of SAC baseline, SOP+ERE, SAC+ERE, and SOP_IG+ERE. (f) shows over a period of 1000 updates, the expected number of times the *t*th data point is sampled (with $\eta = 0.996$). ERE allows new data to be sampled many times soon after being collected.

in a replay buffer and reuses this past data when making updates. It achieved great successes in Deep Q-Networks (DQN) (Mnih et al., 2013; 2015).

Uniform sampling is the most common way to sample from a replay buffer. One of the most well-known alternatives is prioritized experience replay (PER) (Schaul et al., 2015). PER uses the absolute TD-error of a data point as the measure for priority, and data points with higher priority will have a higher chance of being sampled. This method has been tested on DQN (Mnih et al., 2015) and double DQN (DDQN) (Van Hasselt et al., 2016) with significant improvement and applied successfully in other algorithms (Wang et al., 2015; Schulze & Schulze, 2018; Hessel et al., 2018; Hou et al., 2017) and can be implemented in a distributed manner (Horgan et al., 2018).

When new data points lead to large TD errors in the Q update, PER will also assign high sampling probability to newer data points. However, PER has a different effect compared to ERE. PER tries to fit well on both old and new data points. While for ERE, old data points are always considered less important than newer data points even if these old data points start to give a high TD error. A performance comparison of PER and ERE are given in the supplementary materials.

There are other methods proposed to make better use of the replay buffer. The ACER algorithm has an on-policy part

and an off-policy part, with a hyper-parameter controlling the ratio of off-policy to on-policy updates (Wang et al., 2016). The RACER algorithm (Novati & Koumoutsakos, 2018) selectively removes data points from the buffer, based on the degree of "off-policyness," bringing improvement to DDPG (Lillicrap et al., 2015), NAF (Gu et al., 2016) and PPO (Schulman et al., 2017). In De Bruin et al. (2015), replay buffers of different sizes were tested, showing large buffer with data diversity can lead to better performance. Finally, with Hindsight Experience Replay (Andrychowicz et al., 2017), priority can be given to trajectories with lower density estimation (Zhao & Tresp, 2019) to tackle multi-goal, sparse reward environments.

7. Conclusion

In this paper we first showed that the primary role of maximum entropy RL for the MuJoCo benchmark is to maintain satisfactory exploration in the presence of bounded action spaces. We then developed a new streamlined algorithm which does not employ entropy maximization but nevertheless matches the sampling efficiency and robust performance of SAC for the MuJoCo benchmarks. Finally, we combined our streamlined algorithm with a simple non-uniform sampling scheme to create a simple algorithm that achieves state-of-the art performance for the MuJoCo benchmark.

Algorithm 1 Streamlined Off-Policy

1: Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
2: Throughout the output of the policy network $\mu_\theta(s)$ is normalized if $G > 1$. (See Section 4.1.)
3: Set target parameters equal to main parameters $\phi_{\text{targ}_i} \leftarrow \phi_i$ for i = 1, 2
4: **repeat**
5:     Generate an episode using actions $a = M\tanh(\mu_\theta(s) + \epsilon)$ where $\epsilon \sim \mathcal{N}(0, \sigma_1)$.
6:     **for** j in range(however many updates) **do**
7:         Randomly sample a batch of transitions, $B = \{(s, a, r, s)\}$ from $\mathcal{D}$
8:         Compute targets for Q functions:
$$y_q(r, s') = r + \gamma \min_{i=1,2} Q_{\phi_{\text{targ}_i}}(s', M\tanh(\mu_\theta(s') + \delta)) \quad \delta \sim \mathcal{N}(0, \sigma_2)$$
9:         Update Q-functions by one step of gradient descent using
$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s')\in B} \left(Q_{\phi_i}(s, a) - y_q(r, s')\right)^2 \text{ for } i = 1, 2$$
10:     Update policy by one step of gradient ascent using
$$\nabla_\theta \frac{1}{|B|} \sum_{s\in B} Q_{\phi_1}(s, M\tanh(\mu_\theta(s)))$$
11:     Update target networks with
$$\phi_{\text{targ}_i} \leftarrow \rho\phi_{\text{targ}_i} + (1 - \rho)\phi_i \text{ for } i = 1, 2$$
12:     **end for**
13: **until** Convergence

References

Ahmed, Z., Le Roux, N., Norouzi, M., and Schuurmans, D. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pp. 151–160, 2019.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

Chou, P.-W., Maturana, D., and Scherer, S. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 834–843. JMLR. org, 2017.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

De Bruin, T., Kober, J., Tuyls, K., and Babuška, R. The importance of experience replay database composition in deep reinforcement learning. In *Deep reinforcement learning workshop, NIPS*, 2015.

Eisenach, C., Yang, H., Liu, J., and Liu, H. Marginal policy gradients: A unified family of estimators for bounded action spaces with applications. *arXiv preprint arXiv:1806.05134*, 2018.

French, R. M. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

Fujita, Y. and Maeda, S.-i. Clipped action policy gradient. *arXiv preprint arXiv:1802.07564*, 2018.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pp. 2829–2838, 2016.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1352–1361. JMLR. org, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Hausknecht, M. and Stone, P. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., Van Hasselt, H., and Silver, D. Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*, 2018.

Hou, Y., Liu, L., Wei, Q., Xu, X., and Chen, C. A novel ddpg method with prioritized experience replay. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 316–321. IEEE, 2017.

Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Levine, S. and Koltun, V. Guided policy search. In *International Conference on Machine Learning*, pp. 1–9, 2013.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

McClelland, J. L., McNaughton, B. L., and O'reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529, 2015.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785, 2017.

Novati, G. and Koumoutsakos, P. Remember and forget for experience replay. *arXiv preprint arXiv:1807.05827*, 2018.

Puterman, M. L. *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

Ratcliff, R. Connectionist models of recognition memory: constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285, 1990.

Rawlik, K., Toussaint, M., and Vijayakumar, S. On stochastic optimal control and reinforcement learning by approximate inference. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

Robins, A. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Schulze, C. and Schulze, M. Vizdoom: Drqn with prioritized experience replay, double-q learning and snapshot ensembling. In *Proceedings of SAI Intelligent Systems Conference*, pp. 1–17. Springer, 2018.

Todorov, E. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pp. 4286–4292. IEEE, 2008.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, pp. 5. Phoenix, AZ, 2016.

Vuong, Q., Zhang, Y., and Ross, K. W. Supervised policy update for deep reinforcement learning. *arXiv preprint arXiv:1805.11706*, 2018.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

Zhao, R. and Tresp, V. Curiosity-driven experience prioritization via density estimation. *arXiv preprint arXiv:1902.08039*, 2019.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, figshare, 2010.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. 2008.