Prioritizing Starting States for Reinforcement Learning

Arash Tavakoli [*1] Vitaly Levdik [*1] Riashat Islam [2] Petar Kormushev [1]

Abstract

Online, off-policy reinforcement learning algorithms are able to use an experience memory to remember and replay past experiences. In prior work, this approach was used to stabilize training by breaking the temporal correlations of the updates and avoiding the rapid forgetting of possibly rare experiences. In this work, we propose a conceptually simple framework that uses an experience memory to help exploration by prioritizing the starting states from which the agent starts acting in the environment, importantly, in a fashion that is also compatible with on-policy algorithms. Given the capacity to restart the agent in states corresponding to its past observations, we achieve this objective by (i) enabling the agent to restart in states belonging to significant past experiences (*e.g.*, nearby goals), and (ii) promoting faster coverage of the state space through starting from a more diverse set of states. While, using a good priority measure to identify significant past transitions, we expect case (i) to more considerably help exploration in certain domains (*e.g.*, sparse reward tasks), we hypothesize that case (ii) will generally be beneficial, even without any prioritization. We show empirically that our approach improves learning performance for both off-policy and on-policy deep reinforcement learning methods, with most notable gains in highly sparse reward tasks.

1. Introduction

Online reinforcement learning (RL) algorithms have demonstrated an impressive potential for tackling a wide range of complex tasks, with the majority of their success primarily being in simulated environments (Mnih et al., 2015; Lillicrap et al., 2016; Jaderberg et al., 2017; Silver et al., 2017). Scaling up RL algorithms to learn control policies for real practical systems (*e.g.*, robotic manipulation), nevertheless, is often more difficult due to the sample inefficiency of these algorithms. While richer, realistic environments facilitate the transfer of learned policies to reality (Tan et al., 2018), they are accompanied by increased cost of simulation. To be able to explore and learn faster in such simulated environments is, therefore, an important step towards bringing the application of RL to real systems.

Experience replay (Lin, 1992) has recently gained popularity in off-policy deep RL algorithms, such as Deep Q-Networks (DQN) (Mnih et al., 2015), as a means to improve sample efficiency over their on-policy counterparts. As such, on-policy algorithms are often sample inefficient as past transitions need to be thrown away soon, if not immediately, after they are experienced. Moreover, sample efficiency is also related to the ability to explore faster in complex domains. Yet, developing a generic exploration method that can easily be adapted to any RL algorithm remains an unsolved quest.

In this paper, we propose a conceptually simple and easily extendable framework that can, in principle, be applied to any existing on-policy or off-policy RL algorithm. Our approach is to prioritize over starting states from which an agent starts acting in the environment. By starting from significant regions that the agent has already encountered in its past experience, our approach can help improve exploration and, thus, sample efficiency in complex simulated domains.

Given the capacity to reset a simulator's state to those corresponding to agent's past observations, we draw inspiration from the idea of a *restart distribution* as in (Kakade & Langford, 2002) and propose a practical procedure for creating and adapting such distribution based on agent's past experiences. We maintain a restart distribution through a buffer from which an agent can draw starting states. By enabling the agent to restart from important regions of the environment rather than from a fixed reset state or a randomly-selected state from a designated set (as in most OpenAI Gym domains), we aim to explore faster and show that this particular approach is more effective in sparse reward tasks. We achieve this by prioritizing over the agent's past observations to identify important regions of the environment for restarting the agent. This approach tends to go hand in hand with diversifying the starting states that can also help exploration through faster coverage of the state space.

[*]Equal contribution [1]Imperial College London, United Kingdom [2]Montreal Institute for Learning Algorithms (Mila), McGill University, Canada. Correspondence to: Arash Tavakoli, Vitaly Levdik <a.tavakoli, v.levdik@imperial.ac.uk>.

Preprint. Work in Progress.

In this work, we present several variants for prioritizing starting states. In each variant, we consider maintaining a fixed proportion between sampling from our restart distribution—specifically, a *starting state buffer*—and from the environment's initial state distribution. The variants are as follows:

1. Our simplest variant is to store and sample states randomly from a uniform distribution in the same fashion as experience replay was used to handle transitions in (Mnih et al., 2015). That is, we do not prioritize over the buffered states, nor do we select what states to be stored in memory. In this case, we hypothesize that any performance improvement should be mainly due to the diversification of the starting states that in turn helps exploration via faster coverage of the state space.
2. An obvious extension is then to consider sampling starting states from regions with high expected learning progress as measured by the temporal-difference (TD) error, similar to the way transitions are prioritized in *prioritized experience replay* (Schaul et al., 2016).
3. Particularly suited for tackling domains with (significantly) sparse rewards, we consider a last prioritization scheme in which we use an *episodic starting state buffer* to store, only, those states that belong to experientially high-rewarding trajectories (*i.e.*, trajectories that achieve higher returns than previously experienced). This is similar in spirit to the criterion used by Oh et al. (2018) for selecting trajectories to imitate. We show that such an episodic buffer can be crucial in achieving robust learning from even an individual good experience, specifically beneficial for on-policy methods which cannot straightforwardly adopt experience replay to reuse past transitions.

We evaluate our proposed framework, as applied to both on-policy and off-policy algorithms, on numerous simulated physical environments, consisting of robotic manipulation tasks with sparse rewards and robotic locomotion tasks with dense rewards. Our results show improved performance on these domains, with the most significant improvements in sparse reward domains. In our study, we always evaluate the agent's performance based on the original starting state distribution, as we assume this to be the metric of interest.

2. Related Work

The idea of directly influencing the starting states distribution to learn good policies in RL has already drawn attention in the past (Kakade & Langford, 2002; Kormushev et al., 2011). Most notably, Kakade & Langford (2002) studied the notion of exploiting access to a *generative model* (Kearns et al., 2002) of the environment to allow training on a restart distribution (*i.e.*, a fixed, proposal initial state distribution) different from that of the environment. If properly chosen, this is proven to improve learning performance on the original starting state distribution. Nevertheless, no practical procedure is given to choose this new distribution, only suggesting to use a more uniform one over the state space. Also, enabling any such distribution assumes *a priori* knowledge of what constitutes a valid state. In our work, we provide a practical procedure for creating and adapting a starting state distribution during the training without such knowledge.

To improve learning of model-free RL algorithms, Popov et al. (2017) proposed to use expert demonstrations by modifying the starting state distribution to be uniform among the states visited by the provided trajectories. More recently, and concurrent to our work, Salimans & Chen (2018) reported achieving high levels of performance on the infamous Montezuma's Revenge ATARI game by restarting a standard deep RL agent from designated starting states, manually extracted from a single expert demonstration. These works resemble our episodic starting state buffer with a major difference being that, in our case, the agent progressively updates its buffered best trajectories and samples starts from them, thereby not relying on expert demonstrations or manually-designated starts.

Recent work in curriculum for RL presents a method for adaptive generation of curricula in the form of starting state distributions that start close to the goal state and gradually move away with agent's progress (Florensa et al., 2017). This method considers a specific class of goal-oriented tasks with clear goal states and assumes *a priori* access to such states. Contrary to this work, our framework is generally not limited to domains with clear goal states and does not require any prior knowledge of the task. Nevertheless, a similar behavior to curriculum generation in this manner could potentially emerge when using our approach with an appropriate priority measure, whereby any encounter of a goal state would bias the starting state distribution toward it.

Furthermore, in this work we provide an alternative perspective on how past experience can be harvested to assist learning, and remarkably, in a fashion that is compatible with both off-policy and on-policy learning algorithms. This is in contrast to the perspective of replaying past experience to improve the performance of RL agents (Lin, 1992; Mnih et al., 2015; Schaul et al., 2016), an approach that cannot straightforwardly be adopted by on-policy algorithms.

3. Background

3.1. Preliminaries

We consider the RL framework (Sutton & Barto, 1998; Szepesvári, 2010) in which a learning agent interacts with a stochastic environment over a sequence of discrete time steps in the standard fashion: at each time step, the agent chooses an action based on its current state, to which the environment responds with a reward and the next state. We model the environment as a *Markov decision process* (MDP) which comprises: a state space $\mathcal{S}$, an action space $\mathcal{A}$, a starting state distribution with density $p_1(s_1) = Pr\{S_1 = s_1\}$, a state transition kernel $p(s'|s,a) = Pr\{S_{t+1} = s'|S_t = s, A_t = a\}$, and an expected immediate reward function $r(s,a,s') = \mathbb{E}[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s']$, for all $s, s' \in \mathcal{S}, a \in \mathcal{A}, s_1 \in \mathcal{S}_1 \subset \mathcal{S}$.

In general, the agent's decision-making procedure is characterized by a stochastic policy $\pi(a|s) = Pr\{A_t = a|S_t = s\}$. In case of parameterized policies, such as those represented by artificial neural networks, we denote the policy by $\pi(a|s, \boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the vector of policy parameters, and where typically $d \ll |\mathcal{S}|$. The agent uses its policy to interact with the MDP to sample a trajectory $H_{1:T} = S_1, A_1, R_2, ..., S_T, A_T, R_{T+1}$ where T is the trajectory's horizon, which is in general a random variable. Throughout this paper we assume that T is finite, and that terminations could occur either due to terminal states in episodic tasks (*i.e.*, *concrete episodes*) or due to an arbitrary condition, such as timeouts, as could be the case for both continuing and episodic tasks (*i.e.*, *partial episodes*).

3.2. Assumptions

Several of our discussions in this paper are considered under the more generic assumption of learning from partial episodes and, therefore, are only relevant to *bootstrapping* methods. This includes, for instance, any algorithm that uses TD learning, such as Sarsa, Q-learning, and most actor-critic methods. Nevertheless, the main proposition of this paper applies also to *Monte Carlo* (non-bootstrapping) methods, in which case the episodes are strictly concrete.

We assume access to the capacity to restart the agent in states corresponding to its past observations—generally the case given a natural and common type of simulator of the environment. As in (Kearns et al., 2002), our assumption is considerably weaker than having knowledge of the environment's model. However, similar to (Kakade & Langford, 2002), it is a stronger assumption than having only irreversible experience, where the agent must follow a single trajectory without the ability to reset to obtain another trajectory from another state. Note that our assumption on reversibility is weaker than having knowledge of the MDP's starting state distribution.

Lastly, we assume throughout this paper that the agent's observations are *non-aliased* (Whitehead & Ballard, 1991). While such aliasing is generally problematic for RL agents, in our case it can further hurt performance of the agent under the original starting state distribution, as it can bias the agent's policy towards behaviors that may be less suitable for the more frequently-occurring underlying true state.

4. Prioritized Starting States

In this work, assuming access to the capacity to restart the agent in states corresponding to its past observations (as stated in Section 3.2), we propose using a starting state buffer from which the agent can prioritize and draw states to restart from in the next episode. We consider several variants of prioritization of starting states and present a generic and practical procedure for continual evolution of the starting state distribution for improved sample efficiency. Our main contribution is a flexible framework for gradually increasing the diversity of the starting states by storing the agent's previously encountered states in a buffer and enabling prioritized sampling of starting states.

4.1. Motivation

In control problems it is known that even finding an *optimal partial policy*, *i.e.*, a policy that is optimal for the relevant states but can specify arbitrary actions for the irrelevant ones,[1] using an on-policy trajectory-sampling control method in general requires exploring all state-action pairs an 'infinite' number of times (Sutton & Barto, 1998). Similarly, a known problem with policy gradient methods is that the original performance metric, which is what we ultimately seek to optimize, is insensitive to policy improvement at unlikely states despite the fact that policy improvement at these unlikely states might be necessary in order for the agent to achieve near-optimal performance (Kakade & Langford, 2002). For such cases, the diversification of starting states can indeed help better explore the state space beyond the originally reachable states.

In sparse reward environments, where the odds of stumbling upon an informative experience could be significantly low, it is critical for the agent to be able to maximally utilize its good experiences. Using the proposed approach, such trajectories can be recorded and used for sampling starting states, effectively increasing the chances of experiencing success. Moreover, a curriculum of starting states can be generated in this way from past experience by prioritizing the stored states. Employing the past experiences in this manner is especially important in on-policy learning methods which cannot straightforwardly use experience replay and which currently discard all recent experiences immediately after performing a single or multiple iterative updates.

4.2. Role of starting states in the performance objective

In this section, we concern ourselves with the following question: "*How does modifying the starting state distribution affect the original performance metric and, ultimately, the learned policy?*".

[1]The states that are unreachable from any of the MDP's designated starting states and under any optimal policy.

In order to answer this, we consider the case for tabular and approximate solution methods separately. In tabular methods, the learned values at each state are decoupled from one another such that an update at one state affects no other. Let us now consider the control problem in which the agent's goal is to maximize its value from the set of environment's starting states. As per the *principle of optimality* (Sutton & Barto, 1998), a policy achieves the optimal value from a state s, if and only if, for any state s' reachable from s, the policy achieves the optimal value. Therefore, by letting the agent to also start in states outside the designated set, we are, in principle, warranted to better optimize for the original set through better optimizing for the states that are reachable from the original set.

On the contrary, with approximation, an update at one state could affect many others as, by assumption, we have far more states than weights. Thus, making one state's estimate more accurate often means making others' less accurate (Sutton & Barto, 1998). Now let us consider, for instance, the prediction problem of approximating the action values of a given policy using a common loss function:

$$L(\mathbf{w}) \doteq \sum_{s\in\mathcal{S}} \rho(s) \sum_{a\in\mathcal{A}} \pi(a|s) \Big[q_\pi(s,a) - \hat{q}_\pi(s,a|\mathbf{w})\Big]^2.$$

As shown, the overall loss is weighted according to the (discounted) state distribution $\rho(s)$—which in episodic tasks depends on the policy as well as the starting state distribution. In effect, this results in the approximation of the value function to be more accurate at states that have higher visitation density. The same rationale holds for the approximate control methods, such as policy gradient algorithms (Sutton et al., 2000) and DQN. We established that, for approximate methods, changing the starting state distribution to a more diverse one, as in our case, could indeed bias the objective function. But does it result in different optimal policies? Unfortunately, the policy that maximizes our new objective within some restricted class of policies may potentially have poor performance according to the original objective. By using a parameterization that well affords the domain's underlying complexity, we may presume that an optimal policy that maximizes the modified objective—*i.e.*, one with a more diverse set of starts—will too maximize the original objective. While this assumption may seem impractical for large-scale problems, considering the relative simplicity of the current domains of interest with respect to the common high-capacity parametric representations (Li et al., 2018), it is often admissible (see, *e.g.*, (Florensa et al., 2017)).

4.3. Methods

For the set of buffered states $\mathcal{S}_b^i$ at training step i and the set of environment's starting states $\mathcal{S}_1$, where typically $|\mathcal{S}_1| \ll |\mathcal{S}|$, we enable sampling starting states from the increasingly more diverse set $\mathcal{S}_b^i \cup \mathcal{S}_1$ instead of the conventional approach of sampling from the fixed, original set $\mathcal{S}_1$. Assuming no prior knowledge of the environment's model, the original set of starting states $\mathcal{S}_1$ and the corresponding density p_1 are unknown. Nevertheless, as we assume access to a generative model of the environment, we can therefore sample on demand from the original starting state distribution. By maintaining a balance between sampling starting states from a distribution p_b^i over $\mathcal{S}_b^i$ and the original distribution p_1 over $\mathcal{S}_1$, we can ensure to further diversify the starting states by effectively sampling from a new distribution μ_1^i that encompasses both p_1 and p_b^i. We achieve this by sustaining a fixed ratio between the experienced states originating from the environment's starts and the buffered ones. Hence, this ratio is a hyperparameter of our approach.

We believe that when the problem domain is considered simple for the learning algorithm—*e.g.*, most continuous control benchmarks with *reward shaping* (Ng et al., 1999) for many deep RL methods (Duan et al., 2016; Tavakoli et al., 2018)—this ratio would then need to be selected to favor more experiences from the environment's starting states, in order to let the agent focus the majority of its estimator's resources on optimizing for the original performance metric. Therefore, in our dense reward experiments in Section 5.2 we only let 10% of the overall experiences to stem from the buffered starts (*i.e.*, ratio of 10%). However, in more challenging, sparse reward tasks (see Section 5.1), we find that increasing this ratio can significantly improve performance. We now describe the two memory types considered in this paper, namely flat and episodic starting state buffers.

4.3.1. Flat starting state buffer

Our flat starting state buffer simply follows the exact same mechanism as the experience replay memory in (Mnih et al., 2015), with the sole difference being that we store emulator states (*i.e.*, crucially not observations) as opposed to transitions. More specifically, same as the experience replay of (Mnih et al., 2015), our flat buffer has a fixed capacity and stores the most recently-encountered states, without any selection criteria.

In this case, the distribution p_b^i over the buffered states $\mathcal{S}_b^i$ at training step i could simply be uniform (similar to how Mnih et al. (2015) sample transitions to replay from their experience replay) or, alternatively, it could be biased towards important regions as identified by an appropriate priority measure, such as the state-value estimation's TD-error (van Seijen & Sutton, 2013; Schaul et al., 2016). We experiment with both these variants. When using TD-error prioritization, we calculate the probability of sampling states in the same way as proposed by Schaul et al. (2016) for calculating the probability of sampling transitions, and using their proportional prioritization scheme.

It is important to note that prioritization of starting states does not introduce learning bias in the same way as prioritization in the context of experience replay. For experience replay, biasing the sampling of buffered transitions directly alters the perceived state transition and reward distributions in a stochastic environment. This, however, is not the case for our approach, as the introduction of bias on the choice of starting state distribution does not change the perceived transition probabilities of the MDP, and as the experienced transitions are still sampled directly from the environment instead of a buffer.

We can further diversify the agent's experience by prematurely terminating the trajectories that originate from the buffered states after a short, fixed horizon. Doing so is possible for bootstrapping methods via *partial-episode bootstrapping* (PEB) (Pardo et al., 2018), where the agent continues to bootstrap from early terminations, thus allowing it to learn long-term policies from short, partial episodes. In our experiments in Section 5.2, we consider a short, fixed horizon of 10 steps for when interactions originate from buffered starts and compare its performance against having full-length interactions (*i.e.*, using the environment's default time limit of 1000 steps, as specified in OpenAI Gym).

4.3.2. Episodic starting state buffer

In our episodic starting state buffer, we apply a trajectory-centric selection criterion for storing states based on their corresponding episodic return. Specifically, we only allow storage of those states that belong to experientially high-rewarding trajectories (*i.e.*, those that achieve higher returns than previously experienced). In this way, the buffered states have an implicit prioritization associated with them (*i.e.*, that of their selection criterion) which we hypothesize to be useful for sampling starts in sparse reward domains.

Without any further considerations, storing trajectories in this manner could significantly hurt agent's learning performance by biasing its experiences towards very specific regions in the state space. To elucidate this, consider, *e.g.*, a multi-goal environment in which the agent's goal is different in each episode. Given the fact that the goal is part of the (emulator) state, sampling starts from buffered, single-goal episodic experiences could, potentially, cause the entirety of the episodic buffer to be filled with trajectories of a single goal. To address this, we consider a hierarchical approach for storage and sampling of starting states. In particular, for storage, we consider our episodic buffer to have a fixed capacity for the number of *main trajectories* (*i.e.*, those starting from an environment's initial state), as well as the number of per-main-trajectory's *sub trajectories* (*i.e.*, those starting from a state within a main one). We sample starts from our episodic buffer in the following fashion: first we sample a main trajectory class from the stored ones, then we sample a sub trajectory from the chosen main class, and ultimately we sample a state from the chosen sub trajectory. To make this clear, in our multi-goal domain example, using this hierarchical approach, all sub episodic trajectories from a main class will have the same goal, as they all have branched off from the same main episodic trajectory of a fixed goal throughout. This structure allows us to more effectively control the maximum number of buffered trajectories with a similar nature (*e.g.*, with the same goal).

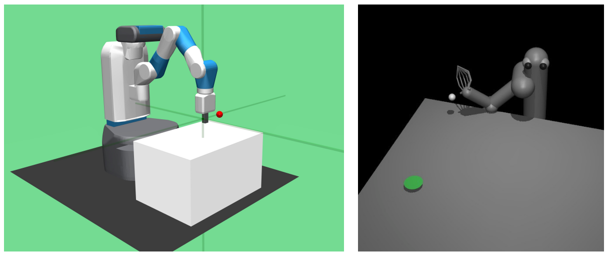
Figure 1. The two sparse reward manipulation environments in our experiments: FetchReach (left) and Thrower (right).

Moreover, to maintain episodic return as a comparable measure across main and sub trajectories for enforcing our storage criterion, we consider sub trajectories to consist of two segments: (i) sub episodic states and rewards for an interaction episode that initiated at a buffer-sampled start, and (ii) all states and rewards from the originating trajectory up to the state that was used to initiate the sub episode. Effectively, the latter serves as an augmentation that links a sub trajectory to an environment's starting state. Using this augmentation allows us to standardize sub episodic returns by assimilating full-length episodic returns. However, to achieve this, one last remaining consideration is then to account for the commonly-used, often fixed, episodic time limit T. In particular, we need to ensure that the time limit is accordingly shortened for sub episodic interactions. To do so, we keep track of the episodic time step t associated with each state from a buffered trajectory, where $t \in \{0, ..., T-1\}$. Having this information about a buffer-sampled initial state enables us to shorten the maximum length of the sub episodic interactions to ensure the augmented sub trajectories do not exceed the time limit T. In other words, for a buffer-sampled starting state s_t with the original episodic time step t, we enforce a horizon of $h = T - t$ for the resulting sub episode.

Lastly, we consider prioritizing over the buffered trajectories based on their corresponding episodic returns. To do so, we calculate the probability of sampling a trajectory τ as

$$P(\tau) = \frac{\bar{G}^{\tau}_{0:l_\tau} - \delta + \epsilon}{\sum_{\forall k \in \mathcal{D}} (\bar{G}^{k}_{0:l_k} - \delta + \epsilon)} \tag{1}$$

where $\delta = \min(0, \min_{\forall k \in \mathcal{D}} \bar{G}^{k}_{0:l_k})$. $\bar{G}^{k}_{0:l_k}$ denotes the undiscounted, episodic return of trajectory k with length

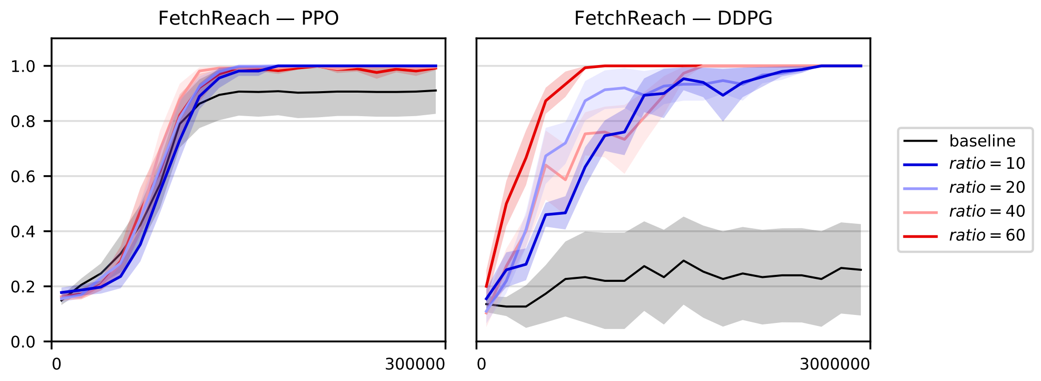

Figure 2. Mean test success rate (line) with standard error (shaded area) on our custom FetchReach environment, for PPO (left; 10 random seeds) and DDPG (right; 5 seeds) with and without our episodic buffer. We consider the performance of our method over a range of ratio hyperparameters. Test evaluations are performed every 200 and 2,000 steps for PPO and DDPG, respectively, and last for one episode.

$l_k \leq T$ from episodic buffer $\mathcal{D}$, ϵ is a small positive constant that prevents not sampling stored trajectories with minimum return, and δ is used as an offset variable to enable handling negative returns. We apply this procedure hierarchically, to first sample a main set and then to sample a sub trajectory from the chosen set. For sampling a main set, we use the highest possible return across the set's stored trajectories as representative of the set's priority. Finally, after selecting a trajectory, we consider uniform sampling of a state from it.

5. Experiments

In this section, we evaluate the performance of our proposed framework of prioritized starting states on a range of simulated physical environments using MuJoCo (v1.5) (Todorov et al., 2012), and empirically demonstrate the following:

1. Our framework can be easily adopted by any on-policy or off-policy RL algorithm through maintaining a starting state buffer. We empirically show this using one popular method from each category: on-policy Proximal Policy Optimization (PPO) (Schulman et al., 2017) and off-policy Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2016).
2. We illustrate that our proposed episodic starting state scheme (described in Section 4.3.2) can prove highly effective (even crucial in certain cases) in sparse reward domains. We demonstrate this on two custom robotic manipulation tasks with sparse rewards, adapted from the OpenAI Gym collections (Brockman et al., 2016).
3. Lastly, we evaluate our flat starting state buffer on a representative subset of the continuous control benchmarks with dense rewards from (Duan et al., 2016). We show that different variants of our approach can improve learning performance, with the performance improvement margin widening as the domains become more complex for the baseline algorithm.

We dedicate the entirety of Section 5.1 to Point 2, and illustrate Points 1 and 3 in Sections 5.1.1 and 5.2, respectively.

Throughout our experiments, we use the OpenAI Baselines (Hesse et al., 2017) implementation of PPO and DDPG, and generally with the same hyperparameter settings reported, respectively, in (Schulman et al., 2017) and (Lillicrap et al., 2016), unless stated otherwise. When using PPO, we always use partial-episode bootstrapping (Pardo et al., 2018), which improved performance in all cases. Throughout our experiments, we use the same discount factor of 0.99.

5.1. Sparse Reward Environments

In this section, we evaluate the performance of our episodic starting state buffer (described in Section 4.3.2). To do so, we consider two simulated, robotic manipulation tasks from OpenAI Gym: FetchReach and Thrower (see Figure 1).

Throughout the experiments in this section, $\epsilon = 10^{-8}$ is used for Equation 1. We begin sampling states from the buffer after 10,000 and 1,000 training steps, and use the main trajectory capacity of 100 and 50 episodes in FetchReach and Thrower, respectively. We use the same sub trajectory capacity of 10 in both domains.

For the DDPG agent, we use the same network size of three hidden layers and 64 units at each layer for both the actor and the critic, where we inject the action vector at the second layer of the critic network. We use a Polyak-averaging coefficient of 0.99 for updating the target (different from 0.999 used in (Lillicrap et al., 2016)). We use the same learning rate of 10^{-3} for both the actor and the critic, a batch size of 256, and an adaptive parameter noise (Plappert et al., 2018b) for exploration, with the same hyperparameters set in the OpenAI Baselines implementation of DDPG.

Due to the deterministic, off-policy nature of DDPG, all our test evaluations using this agent are performed using the greedy policy (*i.e.*, without any added noise for exploration).

For the PPO agent, we generally use a lower learning rate of 10^{-4} across all our experiments (including in Section 5.2), with the exception of the Thrower experiments for which the default learning rate of 10^{-3} from (Schulman et al., 2017) has worked well. In our preliminary experiments, we found lowering of the learning rate to generally help stabilize the performance of PPO on these domains.

5.1.1. Fetch reaching environment

In this section, we evaluate the performance of on-policy PPO and off-policy DDPG agents on a custom FetchReach environment from the OpenAI Gym's multi-goal collection (Plappert et al., 2018a), with and without our proposed episodic starting state buffer. FetchReach is a robotic manipulation domain based on a simulated model of the existing, Fetch robotic arm. In each episode, the agent's task is to control its 4 controllable joints in order to move the gripper to a target position, with the target being randomly sampled at the start of each new episode. Agent's observations include both the full state of the robot as well as the randomized, 3-dimensional goal position. We modified this task so as to eliminate the possibility of initialising the environment in an already solved state. This domain features a fixed time limit of 50 steps and has no terminal states. The agent receives a penalty of -1 for any step spent outside the goal region (*i.e.*, gripper not at target), and zero otherwise.

To study the impact of our ratio hyperparameter in domains with sparse rewards, we evaluate the performance of our agents over a range of ratios for sampling states from the proposed episodic starting state buffer. Specifically, we experimented with ratios 10%, 20%, 40%, and 60%.

Our results are shown in Figure 2. While the baseline PPO performs significantly better than the baseline DDPG on this task, in all cases, using our episodic buffer improves learning performance. For both agents, we observed that higher ratios (*i.e.*, sampling more often from our starts buffer) was beneficial to the learning performance, with the ratio of 60% offering best performance for DDPG, and the ratio of 40% to achieve fastest learning curve for PPO.

It is important to note that, for both agents in our experiments here, we use the same exact implementation of our episodic starting state buffer, with no specific considerations for either of the agents, regardless of their off-policy or on-policy nature. This supports our claim that our approach could be generally adopted by any RL algorithm.

5.1.2. Thrower environment

Next, we evaluate the performance of our proposed episodic starting state buffer on a custom, sparse reward Thrower domain from OpenAI Gym (see Figure 1 (right)). Our modified version of this task makes it particularly challenging as the agent only receives a reward of 1 for successfully throwing the ball in the goal region, and 0 otherwise. Moreover, the agent incurs a small torque penalty upon motion. The task terminates as soon as the ball hits the goal or the table, or upon reaching the time limit of 100 steps.

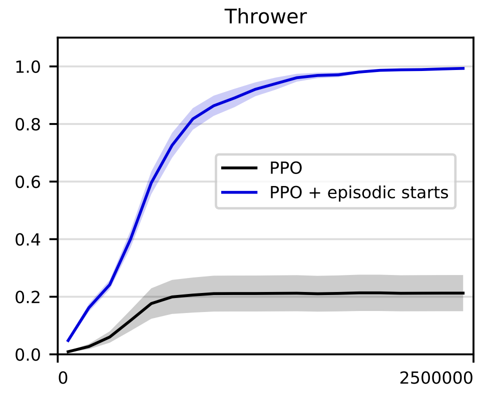

Figure 3. Mean test success rate (line) with standard error (shaded area) on our custom Thrower environment for PPO (42 random seeds) with and without our episodic buffer. While the majority of the baseline PPO instances (different random seeds) fail to learn from the significantly rare, good experience of throwing the ball in the goal region (*i.e.*, green area in Figure 1 (right)), our approach enables every instance to achieve a 100% success rate. Test evaluations are performed every 2,000 environment steps during training and last for 8 episodes each time.

We evaluate our proposed method as applied to the PPO algorithm and contrast it to a standard PPO baseline. We train the agents for 2.5M steps. Due to the complexity of this environment, the probability of the ball impacting the goal during initial exploration is very low, thus we only consider random seeds which have had an instance of the ball hitting the goal in their first 50,000 training steps and use an entropy coefficient of 0.02 to encourage exploration. We evaluate the agents for 8 episodes every 2,000 training steps. Due to the policy's stochasticity and the task's complexity, we mark an evaluation round a 'success' as long as at least one of the 8 trial episodes results in the ball impacting the goal.

Our results are shown in Figure 3. We find that using our episodic buffer, our approach can enable PPO to learn robustly across 42 unique seeds, whereas for 80% of the experiments, the original PPO completely fails to learn the task, with the failed seeds instead learning to balance or drop the ball so as to minimize the accumulated torque penalties.

5.2. Dense Reward Environments

Here, we evaluate the performance of our flat starting state buffer (described in Section 4.3.1). For this purpose, we consider a representative subset of the simulated, robotic locomotion domains with dense rewards from OpenAI Gym, namely HalfCheetah, Walker, and Humanoid.

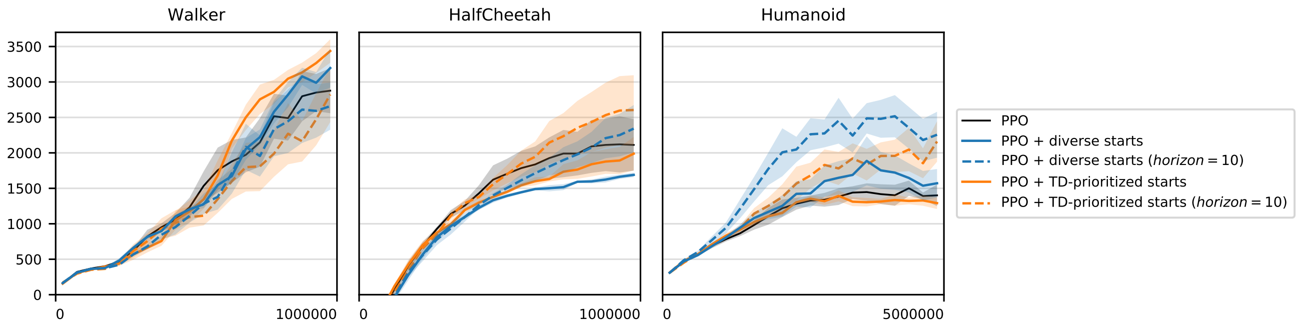

Figure 4. Mean test performances of our flat starting state buffer as applied to PPO (over 5 random seeds). We compare uniform sampling (blue graphs) against the TD-error prioritized variant of our approach (orange graphs) with $\alpha = 0.4$ (determines extent of prioritization). We also compare our approach using a short horizon for interactions originating from the buffered starts (dashed graphs) against using the default time limit of 1000 steps (solid graph). For all these experiments, we use a ratio of 10% and a buffer capacity of 20,000 states.

We only consider PPO in our experiments here, as it has been shown to generally achieve high levels of performance across the continuous control benchmarks. We run PPO with a lower learning rate than that reported in (Schulman et al., 2017) which appeared to perform more stably across the considered tasks and achieved much higher performance levels on the Humanoid domain for the baseline PPO agent. Once again, we use PPO with partial-episode bootstrapping as our baseline. This is due to the fact that the time limits in our benchmarks here are not environmental terminations and, thus, a bootstrapping agent such as PPO should continue to bootstrap from such terminations. Doing so improves the performance of PPO on the domains here, and enables experimenting with shorter horizons (or time limits) when starting from a buffer-sampled starting state. When not explicitly indicated the horizon, the agent is using the default time limit of 1,000 steps for interactions originating from the buffered starting states.

For all variants of our approach in this experiment, we use the ratio of 10% and a buffer capacity of 20,000 states. As described in Section 4.3.1, for prioritizing the buffered states in our method, we adopted the proportional prioritization scheme from (Schaul et al., 2016), indeed without the importance sampling correction, and with the prioritization hyperparameters $\alpha = 0.4$ and $\epsilon = 10^{-6}$ (where α determines how much prioritization is used and ϵ prevents the edge-case of states not being resampled once their corresponding TD-error is zero).

Our experimental results in Figure 4 illustrate that sampling starting states from the agent's past experiences can exhibit performance improvements, even in domains with dense, shaped rewards. Notably, on the Humanoid domain, which is the most complex domain among those considered in this section, we see that using our flat starting state buffer offers the most significant improvements. Particularly, the variants with the shortened horizon of 10 steps (dashed graphs) achieve the best performance on HalfCheetah and Humanoid. Note that, this variant of our approach effectively experiences highest diversification of starting states, as per our ratio hyperparameter which sustains a fixed proportion of states experienced from the environment's starting states and those originating from our buffer-sampled starts, thereby achieving higher sampling density of starts from our buffer. While using a shorter horizon seems to significantly improve performance in at least one domain (*i.e.*, Humanoid), we note that the difference between our TD-prioritized and uniform flat starting state buffers are not significant enough across the tasks here to be conclusive.

6. Conclusion and Future Work

In this work, we proposed a framework of prioritized starting states from which an agent can start acting in the environment. We achieve this by maintaining a buffer of agent's previously-encountered states from which we enable prioritized sampling of starting states. Our proposed framework can be easily adopted by any existing off-policy or on-policy RL algorithm. We applied our method on two popular RL algorithms, on-policy PPO and off-policy DDPG, and showed empirically that different variants of our approach can effectively improve upon the performance in almost all domains. Furthermore, we demonstrated how our approach can be used to robustly learn in problem domains that are characterized with sparse rewards, where a single informative trajectory can be of vital importance to the learning progress, especially of significance for on-policy algorithms which cannot straightforwardly adopt experience replay.

In future work, we aim to explore other prioritization signals for identifying significant states from the buffered ones. We believe it would be interesting to extend our framework to existing methods for intrinsic motivation and curiosity (Schmidhuber, 2010; Bellemare et al., 2016; Pathak et al., 2017) to further help in sparse reward tasks.

Acknowledgements

We thank Harm van Seijen for helpful suggestions on an earlier version of this manuscript, Antoine Cully for discussions about Quality-Diversity algorithms, and Fabio Pardo for providing us with his flexible experimental setup. The research presented in this paper has been supported by EPSRC, Samsung, and computation resources provided by Microsoft via the Microsoft Azure for Research Award program.

References

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1329–1338, 2016.

Florensa, C., Held, D., Wulfmeier, M., Zhang, M., and Abbeel, P. Reverse curriculum generation for reinforcement learning. In *Proceedings of the 1st Conference on Robot Learning*, volume 78, pp. 482–495, 2017.

Hesse, C., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. OpenAI Baselines. https://github.com/openai/baselines, 2017.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, pp. 267–274, 2002.

Kearns, M., Mansour, Y., and Ng, A. Y. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning*, 49(2-3):193–208, 2002.

Kormushev, P., Nomoto, K., Dong, F., and Hirota, K. Time hopping technique for faster reinforcement learning in simulations. *International Journal of Cybernetics and Information Technologies*, 11(3):42–59, 2011.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the intrinsic dimension of objective landscapes. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

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. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pp. 278–287, 1999.

Oh, J., Guo, Y., Singh, S., and Lee, H. Self-imitation learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3878–3887, 2018.

Pardo, F., Tavakoli, A., Levdik, V., and Kormushev, P. Time limits in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4045–4054, 2018.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2778–2787, 2017.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018a.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter space noise for exploration. In *Proceedings of the 6th International Conference on Learning Representations*, 2018b.

Popov, I., Heess, N., Lillicrap, T., Hafner, R., Barth-Maron, G., Vecerik, M., Lampe, T., Tassa, Y., Erez, T., and Riedmiller, M. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

Salimans, T. and Chen, R. Learning Montezuma's Revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

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

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

Szepesvári, C. *Algorithms for Reinforcement Learning*. Morgan and Claypool, 2010.

Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

Tavakoli, A., Pardo, F., and Kormushev, P. Action branching architectures for deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 4131–4138, 2018.

Todorov, E., Erez, T., and Tassa, Y. MuJoCo: A physics engine for model-based control. In *Proceedings of the 25th IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

van Seijen, H. and Sutton, R. Planning by prioritized sweeping with small backups. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 361–369, 2013.

Whitehead, S. D. and Ballard, D. H. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83, 1991.