Remember and Forget for Experience Replay

Guido Novati[1] **Petros Koumoutsakos**[1]

Abstract

Experience replay (ER) is a fundamental component of off-policy deep reinforcement learning (RL). ER recalls experiences from past iterations to compute gradient estimates for the current policy, increasing data-efficiency. However, the accuracy of such updates may deteriorate when the policy diverges from past behaviors and can undermine the performance of ER. Many algorithms mitigate this issue by tuning hyper-parameters to slow down policy changes. An alternative is to actively enforce the similarity between policy and the experiences in the replay memory. We introduce *Remember and Forget Experience Replay* (`ReF-ER`), a novel method that can enhance RL algorithms with parameterized policies. `ReF-ER` (1) skips gradients computed from experiences that are too unlikely with the current policy and (2) regulates policy changes within a trust region of the replayed behaviors. We couple `ReF-ER` with Q-learning, deterministic policy gradient and off-policy gradient methods. We find that `ReF-ER` consistently improves the performance of continuous-action, off-policy RL on fully observable benchmarks and partially observable flow control problems.

1. Introduction

Deep reinforcement learning (RL) has an ever increasing number of success stories ranging from realistic simulated environments (Schulman et al., 2015; Mnih et al., 2016), robotics (Levine et al., 2016) and games (Mnih et al., 2015; Silver et al., 2016). Experience Replay (ER) (Lin, 1992) enhances RL algorithms by using information collected in past policy (μ) iterations to compute updates for the current policy (π). ER has become one of the mainstay techniques to improve the sample-efficiency of off-policy deep RL.

[1]Computational Science & Engineering Laboratory, ETH Zurich, Zurich, Switzerland. Correspondence to: Guido Novati <novatig@ethz.ch>, Petros Koumoutsakos <petros@ethz.ch>.

Sampling from a replay memory (RM) stabilizes stochastic gradient descent (SGD) by disrupting temporal correlations and extracts information from useful experiences over multiple updates (Schaul et al., 2015b). However, when π is parameterized by a neural network (NN), SGD updates may result in significant changes to the policy, thereby shifting the distribution of states observed from the environment. In this case sampling the RM for further updates may lead to incorrect gradient estimates, therefore deep RL methods must account for and limit the dissimilarity between π and behaviors in the RM. Previous works employed trust region methods to bound policy updates (Schulman et al., 2015; Wang et al., 2017). Despite several successes, deep RL algorithms are known to suffer from instabilities and exhibit high-variance of outcomes (Islam et al., 2017; Henderson et al., 2017), especially continuous-action methods employing the stochastic (Sutton et al., 2000) or deterministic (Silver et al., 2014) policy gradients (PG or DPG).

In this work we redesign ER in order to control the similarity between the replay behaviors μ used to compute updates and the policy π. More specifically, we classify experiences either as "near-policy" or "far-policy", depending on the ratio ρ of probabilities of selecting the associated action with π and that with μ. The weight ρ appears in many estimators that are used with ER such as the off-policy policy gradients (off-PG) (Degris et al., 2012) and the off-policy return-based evaluation algorithm Retrace (Munos et al., 2016). Here we propose and analyze Remember and Forget Experience Replay (`ReF-ER`), an ER method that can be applied to any off-policy RL algorithm with parameterized policies. `ReF-ER` limits the fraction of far-policy samples in the RM, and computes gradient estimates only from near-policy experiences. Furthermore, these hyper-parameters can be gradually annealed during training to obtain increasingly accurate updates from nearly on-policy experiences. We show that `ReF-ER` allows better stability and performance than conventional ER in all three main classes of continuous-actions off-policy deep RL algorithms: methods based on the DPG (ie. `DDPG` (Lillicrap et al., 2016)), methods based on Q-learning (ie. `NAF` (Gu et al., 2016)), and with off-PG (Degris et al., 2012; Wang et al., 2017).

In recent years, there is a growing interest in coupling RL with high-fidelity physics simulations (Reddy et al., 2016; Novati et al., 2017; Colabrese et al., 2017; Verma et al.,

2018). The computational cost of these simulations calls for reliable and data-efficient RL methods that do not require problem-specific tweaks to the hyper-parameters (HP). Moreover, while on-policy training of simple architectures has been shown to be sufficient in some benchmarks (Rajeswaran et al., 2017), agents aiming to solve complex problems with partially observable dynamics might require deep or recurrent models that can be trained more efficiently with off-policy methods. We analyze `ReF-ER` on the OpenAI Gym (Brockman et al., 2016) as well as fluid-dynamics simulations to show that it reliably obtains competitive results without requiring extensive HP optimization.

2. Methods

Consider the sequential decision process of an agent aiming to optimize its interaction with the environment. At each step t, the agent observes its state $s_t \in \mathbb{R}^{d_S}$, performs an action by sampling a policy $a_t \sim \mu_t(a|s_t) \in \mathbb{R}^{d_A}$, and transitions to a new state $s_{t+1} \sim \mathcal{D}(s|a_t, s_t)$ with reward $r_{t+1} \in \mathbb{R}$. The experiences $\{s_t, a_t, r_t, \mu_t\}$ are stored in a RM, which constitutes the data used by off-policy RL to train the parametric policy $\pi^{\mathtt{w}}(a|s)$. The importance weight $\rho_t = \pi^{\mathtt{w}}(a_t|s_t)/\mu_t(a_t|s_t)$ is the ratio between the probability of selecting a_t with the current $\pi^{\mathtt{w}}$ and with the behavior μ_t, which gradually becomes dissimilar from $\pi^{\mathtt{w}}$ as the latter is trained. The on-policy state-action value $Q^{\pi^{\mathtt{w}}}(s, a)$ measures the expected returns from (s, a) following $\pi^{\mathtt{w}}$:

$$Q^{\pi^{\mathtt{w}}}(s, a) = \mathop{\mathbb{E}}_{\substack{s_t \sim \mathcal{D} \\ a_t \sim \pi^{\mathtt{w}}}} \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \,\middle|\, s_0 = s, \ a_0 = a \right] \quad (1)$$

Here γ is a discount factor. The value of state s is the on-policy expectation $V^{\pi^{\mathtt{w}}}(s) = \mathbb{E}_{a \sim \pi^{\mathtt{w}}} \left[Q^{\pi^{\mathtt{w}}}(s, a) \right]$. In this work we focus on three deep-RL algorithms, each representing one class of off-policy continuous action RL methods.

DDPG (Lillicrap et al., 2016) is a method based on deterministic PG which trains two networks by ER. The value-network (a.k.a. critic) outputs $Q^{\mathtt{w}'}(s, a)$ and is trained to minimize the L2 distance from the temporal difference (TD) target $\hat{Q}_t = r_{t+1} + \gamma \mathbb{E}_{a' \sim \pi} Q^{\mathtt{w}'}(s_{t+1}, a')$:

$$\mathcal{L}^{\mathrm{Q}}(\mathtt{w}') = \frac{1}{2} \mathop{\mathbb{E}}_{\substack{s_k \sim B(s) \\ a_k \sim \mu_k}} \left[Q^{\mathtt{w}'}(s_k, a_k) - \hat{Q}_k \right]^2 \quad (2)$$

Here $B(s) \propto \sum_{k=t-N}^{t} P(s_k = s \mid s_0, \ a_k \sim \mu_k)$ is the probability of sampling state s from a RM containing the last N experiences of the agent acting with policies μ_k. The policy-network is trained to output actions $\mathbf{m}^{\mathtt{w}}$ that maximize the returns predicted by the critic (Silver et al., 2014):

$$\mathcal{L}^{\mathrm{DPG}}(\mathtt{w}) = - \mathbb{E}_{s_k \sim B(s)} \left[Q^{\mathtt{w}'}(s_k, \mathbf{m}^{\mathtt{w}}(s_k)) \right] \quad (3)$$

NAF (Gu et al., 2016) is the state-of-the-art of Q-learning

based algorithms for continuous-action problems. It employs a quadratic-form approximation of the $Q^{\mathtt{w}}$ value:

$$Q_{\mathrm{NAF}}^{\mathtt{w}}(s, a) = V^{\mathtt{w}}(s) - [a - \mathbf{m}^{\mathtt{w}}(s)]^{\mathsf{T}} \, \mathbf{L}^{\mathtt{w}} \, [\mathbf{L}^{\mathtt{w}}]^{\mathsf{T}} \, [a - \mathbf{m}^{\mathtt{w}}(s)] \quad (4)$$

Given a state s, a single network estimates its value $V^{\mathtt{w}}$, the optimal action $\mathbf{m}^{\mathtt{w}}(s)$, and the lower-triangular matrix $\mathbf{L}^{\mathtt{w}}(s)$ that parameterizes the advantage. Like `DDPG`, `NAF` is trained by ER with the Q-learning target (Eq. 2). For both `DDPG` and `NAF`, we include exploratory Gaussian noise in the policy $\pi^{\mathtt{w}} = \mathbf{m}^{\mathtt{w}} + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ with $\sigma = 0.2$ (to compute ρ_t or the Kullback-Leibler divergence D_{KL} between policies).

V-RACER is the method we propose to analyze off-policy PG (off-PG) and ER. Given s, a single NN outputs the value $V^{\mathtt{w}}$, the mean $\mathbf{m}^{\mathtt{w}}$ and diagonal covariance $\mathbf{\Sigma}^{\mathtt{w}}$ of the Gaussian policy $\pi^{\mathtt{w}}(a|s)$. The policy is updated with the off-policy objective (Degris et al., 2012):

$$\mathcal{L}^{\text{off-PG}}(\mathtt{w}) = - \mathop{\mathbb{E}}_{\substack{s_k \sim B(s) \\ a_k \sim \mu_k}} \left[\rho_k \left(\hat{Q}_k^{\mathrm{ret}} - V^{\mathtt{w}}(s_k) \right) \right] \quad (5)$$

On-policy returns are estimated with Retrace (Munos et al., 2016), which takes into account rewards obtained by μ_t:

$$\hat{Q}_{t-1}^{\mathrm{ret}} = r_t + \gamma V^{\mathtt{w}}(s_t) + \gamma \bar{\rho}_t \left[\hat{Q}_t^{\mathrm{ret}} - Q^{\mathtt{w}}(s_t, a_t) \right] \quad (6)$$

Here we defined $\bar{\rho}_t = \min\{1, \rho_t\}$. V-RACER avoids training a NN for the action value by approximating $Q^{\mathtt{w}} := V^{\mathtt{w}}$ (i.e. it assumes that any individual action has a small effect on returns (Tucker et al., 2018)). The on-policy state value is estimated with the "variance truncation and bias correction trick" (TBC) (Wang et al., 2017):

$$\hat{V}_t^{\mathrm{tbc}} = V^{\mathtt{w}}(s_t) + \bar{\rho}_t [\hat{Q}_t^{\mathrm{ret}} - Q^{\mathtt{w}}(s_t, a_t)] \quad (7)$$

From Eq. 6 and 7 we obtain $\hat{Q}_t^{\mathrm{ret}} = r_{t+1} + \gamma \hat{V}_{t+1}^{\mathrm{tbc}}$. From this, Eq. 7 and $Q^{\mathtt{w}} := V^{\mathtt{w}}$, we obtain a recursive estimator for the on-policy state value that depends on $V^{\mathtt{w}}$ alone:

$$\hat{V}_t^{\mathrm{tbc}} = V^{\mathtt{w}}(s_t) + \bar{\rho}_t \left[r_{t+1} + \gamma \hat{V}_{t+1}^{\mathrm{tbc}} - V^{\mathtt{w}}(s_t) \right] \quad (8)$$

This target is equivalent to the recently proposed V-trace estimator (Espeholt et al., 2018) when all importance weights are clipped at 1, which was empirically found by the authors to be the best-performing solution. Finally, the value estimate is trained to minimize the loss:

$$\mathcal{L}^{\mathrm{tbc}}(\mathtt{w}) = \frac{1}{2} \mathop{\mathbb{E}}_{\substack{s_k \sim B(s) \\ a_k \sim \mu_k}} \left[V^{\mathtt{w}}(s_t) - \hat{V}_t^{\mathrm{tbc}} \right]^2 \quad (9)$$

In order to estimate $\hat{V}_t^{\mathrm{tbc}}$ for a sampled time step t, Eq. 8 requires $V^{\mathtt{w}}$ and ρ_t for all following steps in sample t's episode. These are naturally computed when training from batches of episodes (as in `ACER` (Wang et al., 2017)) rather than time steps (as in `DDPG` and `NAF`). However, the information contained in consecutive steps is correlated, worsening

the quality of the gradient estimate, and episodes may be composed of thousands of time steps, increasing the computational cost. To efficiently train from uncorrelated time steps, `V-RACER` stores for each sample the most recently computed estimates of $V^{\mathtt{w}}(s_k)$, ρ_k and $\hat{V}_k^{\text{tbc}}$. When a time step is sampled, the stored $\hat{V}_k^{\text{tbc}}$ is used to compute the gradients. At the same time, the current NN outputs are used to update $V^{\mathtt{w}}(s_k)$, ρ_k and to correct $\hat{V}^{\text{tbc}}$ for all prior timesteps in the episode with Eq. 8. Each algorithm and the remaining implementation details are described in App. A.

3. Remember and Forget Experience Replay

In off-policy RL it is common to maximize on-policy returns estimated over the distribution of states contained in a RM. In fact, each method introduced in Sec. 2 relies on computing estimates over the distribution $B(s)$ of states observed by the agent following behaviors μ_k over prior steps k. However, as $\pi^{\mathtt{w}}$ gradually shifts away from previous behaviors, $B(s)$ is increasingly dissimilar from the on-policy distribution, and trying to increase an off-policy performance metric may not improve on-policy outcomes. This issue can be compounded with algorithm-specific concerns. For example, the dissimilarity between μ_k and $\pi^{\mathtt{w}}$ may cause vanishing or diverging importance weights ρ_k, thereby increasing the variance of the off-PG and deteriorating the convergence speed of Retrace (and V-trace) by inducing "trace-cutting" (Munos et al., 2016). Multiple remedies have been proposed to address these issues. For example, `ACER` tunes the learning rate and uses a target-network (Mnih et al., 2015), updated as a delayed copy of the policy-network, to constrain policy updates. Target-networks are also employed in `DDPG` to slow down the feedback loop between value-network and policy-network optimizations. This feedback loop causes overestimated action values that can only be corrected by acquiring new on-policy samples. Recent works (Henderson et al., 2017) have shown the opaque variability of outcomes of continuous-action deep RL algorithms depending on hyper-parameters. Target-networks may be one of the sources of this unpredictability. In fact, when using deep approximators, there is no guarantee that the small weight changes imposed by target-networks correspond to small changes in the network's output.

This work explores the benefits of actively managing the "off-policyness" of the experiences used by ER. We propose a set of simple techniques, collectively referred to as Remember and Forget ER (`ReF-ER`), that can be applied to any off-policy RL method with parameterized policies.

- The cost functions are minimized by estimating the gradients $\hat{\mathbf{g}}$ with mini-batches of experiences drawn from a RM. We compute the importance weight ρ_t of each experience and classify it as "near-policy" if $1/c_{\max}<\rho_t<c_{\max}$

with $c_{\max}>1$. Samples with vanishing ($\rho_t<1/c_{\max}$) or exploding ($\rho_t>c_{\max}$) importance weights are classified as "far-policy". When computing off-policy estimators with finite batch-sizes, such as $\hat{Q}^{\text{ret}}$ or the off-PG, "far-policy" samples may either be irrelevant or increase the variance. For this reason, **(Rule 1:) the gradients computed from far-policy samples are clipped to zero.** In order to efficiently approximate the number of far-policy samples in the RM, we store for each step its most recent ρ_t.

- **(Rule 2:) Policy updates are penalized in order to attract the current policy $\pi^{\mathtt{w}}$ towards past behaviors**:

$$\hat{\mathbf{g}}^{\text{ReF-ER}}(\mathtt{w}) = \begin{cases} \beta\hat{\mathbf{g}}(\mathtt{w}) - (1-\beta)\hat{\mathbf{g}}^D(\mathtt{w}) & \text{if } \frac{1}{c_{\max}}<\rho_t<c_{\max} \\ -(1-\beta)\hat{\mathbf{g}}^D(\mathtt{w}) & \text{otherwise} \end{cases} \tag{10}$$

Here we penalize the "off-policyness" of the RM with:

$$\hat{\mathbf{g}}^D(\mathtt{w}) = \mathbb{E}_{s_k\sim B(s)}\left[\nabla D_{\text{KL}}\left(\mu_k\|\pi^{\mathtt{w}}(\cdot|s_k)\right)\right] \tag{11}$$

The coefficient $\beta \in [0,1]$ is updated at each step such that a set fraction $D \in (0,1)$ of samples are far-policy:

$$\beta \leftarrow \begin{cases} (1-\eta)\beta & \text{if } n_{\text{far}}/N > D \\ (1-\eta)\beta + \eta, & \text{otherwise} \end{cases} \tag{12}$$

Here η is the NN's learning rate, N is the number of experiences in the RM, of which n_{far} are far-policy. Note that iteratively updating β with Eq. 12 has fixed points in $\beta=0$ for $n_{\text{far}}/N>D$ and in $\beta=1$ otherwise.

We remark that alternative metrics of the relevance of training samples were considered, such as D_{KL} or its discounted cumulative sum, before settling on the present formulation. `ReF-ER` aims to reduce the sensitivity on the NN architecture and HP by controlling the rate at which the policy can deviate from the replayed behaviors. For $c_{\max}\to 1$ and $D\to 0$, `ReF-ER` becomes asymptotically equivalent to computing updates from an on-policy dataset. Therefore, we anneal `ReF-ER`'s $c_{\max}$ and the NN's learning rate according to:

$$c_{\max}(t) = 1+C/(1 + A \cdot t), \quad \eta(t) = \eta/(1 + A \cdot t) \tag{13}$$

Here t is the time step index, A regulates annealing, and η is the initial learning rate. $c_{\max}$ determines how much $\pi^{\mathtt{w}}$ is allowed to differ from the replayed behaviors. By annealing $c_{\max}$ we allow fast improvements at the beginning of training, when inaccurate policy gradients might be sufficient to estimate a good direction for the update. Conversely, during the later stages of training, precise updates can be computed from almost on-policy samples. We use $A=5\cdot 10^{-7}$, $C=4$, $D=0.1$, and $N=2^{18}$ for all results with `ReF-ER` in the main text. The effect of these hyper parameters is further discussed in a detailed sensitivity analysis reported in the Supplementary Material.

4. Related work

The rules that determine which samples are kept in the RM and how they are used for training can be designed to address specific objectives. For example, it may be necessary to properly plan ER to prevent lifelong learning agents from forgetting previously mastered tasks (Isele & Cosgun, 2018). ER can be used to train transition models in planning-based RL (Pan et al., 2018), or to help shape NN features by training off-policy learners on auxiliary tasks (Schaul et al., 2015a; Jaderberg et al., 2017). When rewards are sparse, RL agents can be trained to repeat previous outcomes (Andrychowicz et al., 2017) or to reproduce successful states or episodes (Oh et al., 2018; Goyal et al., 2018).

In the next section we compare ReF-ER to conventional ER and *Prioritized Experience Replay* (Schaul et al., 2015b) (PER). PER improves the performance of DQN (Mnih et al., 2015) by biasing sampling in favor of experiences that cause large temporal-difference (TD) errors. TD errors may signal rare events that would convey useful information to the learner. de Bruin et al. (2015) proposes a modification to ER that increases the diversity of behaviors contained in the RM, which is the opposite of what ReF-ER achieves. Because the ideas proposed by de Bruin et al. (2015) cannot readily be applied to complex tasks (the authors state that their method is not suitable when the policy is advanced for many iterations), we compare ReF-ER only to PER and conventional ER. We assume that if increasing the diversity of experiences in the RM were beneficial to off-policy RL then either PER or ER would outperform ReF-ER.

ReF-ER is inspired by the techniques developed for on-policy RL to bound policy changes in PPO (Schulman et al., 2017). Rule 1 of ReF-ER is similar to the clipped objective function of PPO (gradients are zero if ρ_t is outside of some range). However, Rule 1 is not affected by the sign of the advantage estimate and clips both policy and value gradients. Another variant of PPO penalizes $D_{KL}(\mu_t||\pi^{\mathtt{w}})$ in a similar manner to Rule 2 (also Schulman et al. (2015) and Wang et al. (2017) employ trust-region schemes in the on- and off-policy setting respectively). PPO picks one of the two techniques, and the authors find that gradient-clipping performs better than penalization. Conversely, in ReF-ER Rules 1 and 2 complement each other and can be applied to most off-policy RL methods with parametric policies.

V-RACER shares many similarities with ACER (Wang et al., 2017) and IMPALA (Espeholt et al., 2018) and is a secondary contribution of this work. The improvements introduced by V-RACER have the purpose of aiding our analysis of ReF-ER: (1) V-RACER employs a single NN; not requiring expensive architectures eases reproducibility and exploration of the HP (e.g. continuous-ACER uses 9 NN evaluations per gradient). (2) V-RACER samples time steps rather than episodes (like DDPG and NAF and unlike ACER

and IMPALA), further reducing its cost (episodes may consist of thousands of steps). (3) V-RACER does not introduce techniques that would interfere with ReF-ER and affect its analysis. Specifically, ACER uses the TBC (Sec. 2) to clip policy gradients, employs a target-network to bound policy updates with a trust-region scheme, and modifies Retrace to use $\sqrt[d]{\rho}$ instead of ρ. Lacking these techniques, we expect V-RACER to require ReF-ER to deal with unbounded importance weights. Because of points (1) and (2), the computational complexity of V-RACER is approximately two orders of magnitude lower than that of ACER.

5. Results

In this section we couple ReF-ER, conventional ER and PER with one method from each of the three main classes of deep continuous-action RL algorithms: DDPG, NAF, and V-RACER. In order to separate the effects of its two components, we distinguish between ReF-ER-1, which uses only Rule 1, ReF-ER-2, using only Rule 2, and the full ReF-ER. The performance of each combination of algorithms is measured on the MuJoCo (Todorov et al., 2012) tasks of OpenAI Gym (Brockman et al., 2016) by plotting the mean cumulative reward $R = \sum_t r_t$. Each plot tracks the average R among all episodes entering the RM within intervals of $2 \cdot 10^5$ time steps averaged over five differently seeded training trials. For clarity, we highlight the contours of the 20th to 80th percentiles of R only of the best performing alternatives to the proposed methods. The code to reproduce all present results is available on GitHub.[1]

5.1. Results for DDPG

The performance of DDPG is sensitive to hyper-parameter (HP) tuning (Henderson et al., 2017). We find the critic's weight decay and temporally-correlated exploration noise to be necessary to stabilize DDPG with ER and PER. Without this tuning, the returns for DDPG can fall to large negative values, especially in tasks that include the actuation cost in the reward (e.g. Ant). This is explained by the critic not having learned local maxima with respect to the action (Silver et al., 2014). Fig. 1 shows that replacing ER with ReF-ER stabilizes DDPG and greatly improves its performance, especially for tasks with complex dynamics (e.g. Humanoid and Ant). We note that with ReF-ER we do not use temporally-correlated noise and that annealing η worsened the instability of DDPG with regular ER and PER.

In Fig. 2, we report the average $D_{KL}(\mu_t||\pi^{\mathtt{w}})$ as a measure of the RM's "off-policyness". With ReF-ER, despite its reliance on approximating of the total number of far-policy samples n_{far} in Eq. 12 from outdated importance weights, the D_{KL} smoothly decreases during training due to the

---
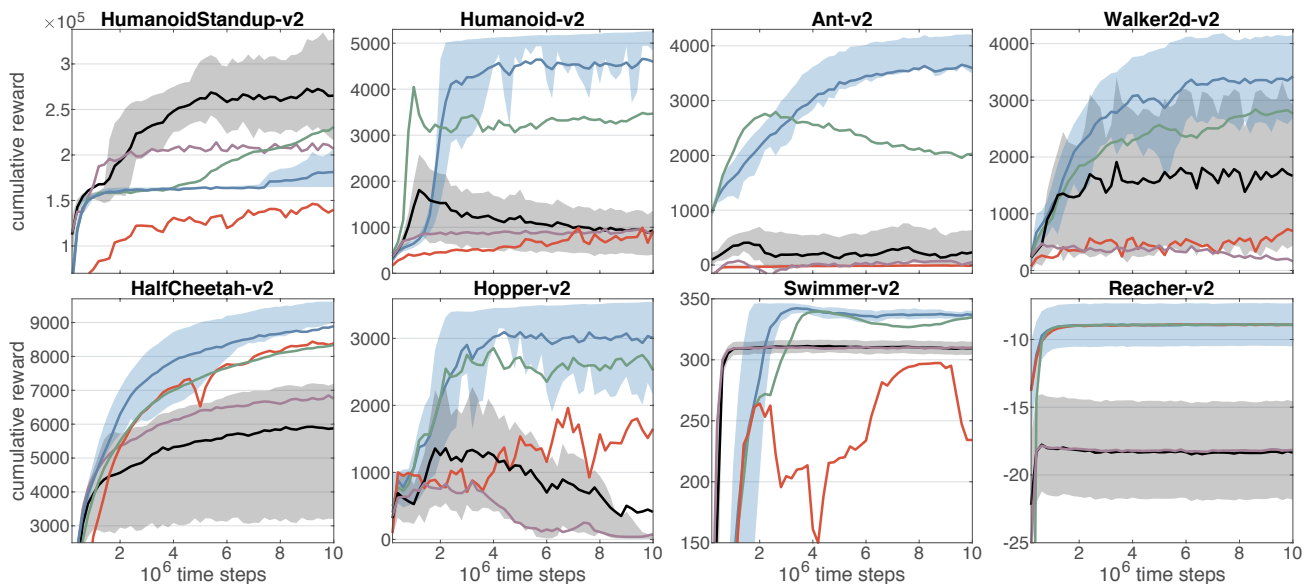
[1] https://github.com/cselab/smarties

Figure 1. Cumulative rewards on OpenAI MuJoCo tasks for `DDPG` (black line), `DDPG` with rank-based `PER` (purple line), `DDPG` with `ReF-ER` (blue), with `ReF-ER-1` (red), and with `ReF-ER-2` (green). Implementation details in App. A.
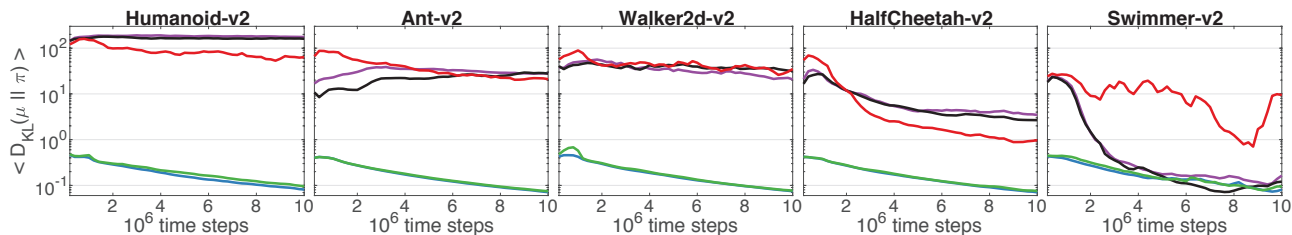

Figure 2. Kullback-Leibler divergence D_{KL} between $\pi^{\mathrm{w}} = m^{\mathrm{w}} + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ trained by `DDPG` and the replayed behaviors. Same colors as above. Note: the average D_{KL} for each algorithm is 0 at the beginning of training and is updated after every 1e5 time steps.

annealing process. This validates that Rule 2 of `ReF-ER` achieves its intended goal with minimal computational overhead. With regular ER, even after lowering η by one order of magnitude from the original paper (we use $\eta = 10^{-4}$ for the critic and $\eta = 10^{-5}$ for the policy), D_{KL} may span the entire action space. In fact, in many tasks the average D_{KL} with ER is of similar order of magnitude as its maximum $2d_A/\sigma^2$ (`DDPG` by construction bounds $\mathbf{m}^{\mathrm{w}}$ to the hyperbox $(-1, 1)^{d_A}$). For example, for $\sigma = 0.2$, the maximum D_{KL} is 850 for Humanoid and 300 for Walker and it oscillates during training around 100 and 50 respectively. This indicates that $\mathbf{m}^{\mathrm{w}}$ swings between the extrema of the action space likely due to the critic not learning local maxima for Q^{w}. Without policy constrains, `DDPG` often finds only "bang-bang" control schemes, which explains why bounding the action space is necessary to ensure the stability of `DDPG`.

When comparing the components of `ReF-ER`, we note that relying on gradient clipping alone (`ReF-ER-1`) does not produce good results. `ReF-ER-1` may cause many zero-

valued gradients, especially in high-dimensional tasks where even small changes to $\mathbf{m}^{\mathrm{w}}$ may push ρ_t outside of the near-policy region. However, it's on these tasks that combining the two rules brings a measurable improvement in performance over `ReF-ER-2`. Training from only near-policy samples, provides the critic with multiple examples of trajectories that are possible with the current policy. This focuses the representation capacity of the critic, enabling it to extrapolate the effect of a marginal change of action on the expected returns, and therefore increasing the accuracy of the DPG. Any misstep of the DPG is weighted with a penalization term that attracts the policy towards past behaviors. This allows time for the learner to gather experiences with the new policy, improve the value-network, and correct the misstep. This reasoning is almost diametrically opposed to that behind `PER`, which generally obtains worse outcomes than regular ER. In `PER` observations associated with larger TD errors are sampled more frequently. In the continuous-action setting, however, TD errors may be caused by actions that are farther from $\mathbf{m}^{\mathrm{w}}$. Therefore, precisely estimating
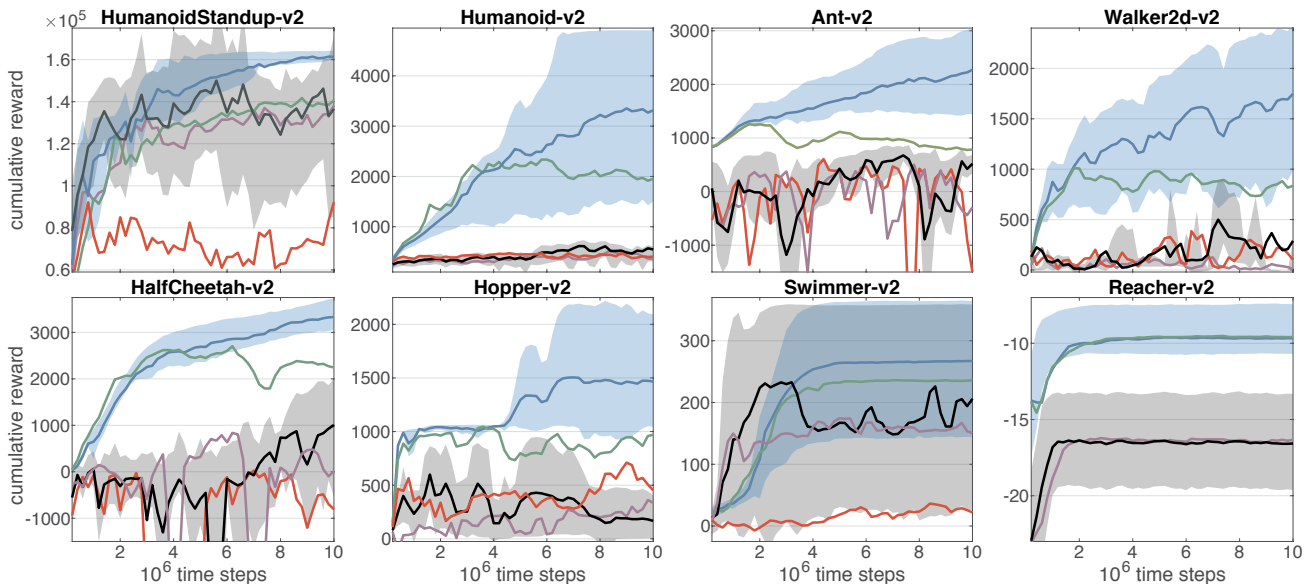
Figure 3. Cumulative rewards on OpenAI MuJoCo tasks for `NAF` (black line), `NAF` with rank-based `PER` (purple line), `NAF` with `ReF-ER` (blue), with `ReF-ER-1` (red), and with `ReF-ER-2` (green). Implementation details in App. A.

their value might not help the critic in yielding an accurate estimate of the DPG. The Swimmer and HumanoidStandup tasks highlight that ER is faster than `ReF-ER` in finding bang–bang policies. The bounds imposed by `DDPG` on $\mathbf{m}^{\mathtt{w}}$ allow learning these behaviors without numerical instability and without finding local maxima of $Q^{\mathtt{w}}$. The methods we consider next learn unbounded policies. These methods do not require prior knowledge of optimal action bounds, but may not enjoy the same stability guarantees.

5.2. Results for `NAF`

Figure 3 shows how `NAF` is affected by the choice of ER algorithm. While Q-learning based methods are thought to be less sensitive than PG-based methods to the dissimilarity between policy and stored behaviors owing to the bootstrapped Q-learning target, `NAF` benefits from both rules of `REF-ER`. Like for `DDPG`, Rule 2 provides NAF with more near-policy samples to compute the off-policy estimators. Moreover, the performance of `NAF` is more distinctly improved by combining Rule 1 and 2 of `REF-ER` over using `REF-ER-2`. This is because Q^{π} is likely to be approximated well by the quadratic $Q^{\mathtt{w}}_{\mathrm{NAF}}$ in a small neighborhood near its local maxima. When $Q^{\mathtt{w}}_{\mathrm{NAF}}$ learns a poor fit of Q^{π} (e.g. when the return landscape is multi-modal), `NAF` may fail to choose good actions. Rule 1 clips the gradients from actions outside of this neighborhood and prevents large TD errors from disrupting the locally-accurate approximation $Q^{\mathtt{w}}_{\mathrm{NAF}}$. This intuition is supported by observing that rank-based `PER` (the better performing variant of `PER` also in this case), often worsens the performance of `NAF`. `PER` aims at biasing sampling in favor of larger TD errors, which are more likely to

be farther from $\mathbf{m}^{\mathtt{w}}(s)$, and their accurate prediction might not help the learner in fine-tuning the policy by improving a local approximation of the advantage. Lastly, $Q^{\mathtt{w}}_{\mathrm{NAF}}$ is unbounded, therefore training from actions that are farther from $\mathbf{m}^{\mathtt{w}}$ increases the variance of the gradient estimates.

5.3. Results for `V-RACER`

Here we compare `V-RACER` to `ACER` and to `PPO`, an algorithm that owing to its simplicity and good performance on MuJoCo tasks is often used as baseline. For clarity, we omit from Fig. 4 results from coupling `V-RACER` with ER or `PER`, which generally yield similar or worse results than `ReF-ER-1`. Without Rule 1 of `ReF-ER`, `V-RACER` has no means to deal with unbounded importance weights, which cause off-PG estimates to diverge and disrupt prior learning progress. In fact, also `ReF-ER-2` is affected by unbounded ρ_t because even small policy differences can cause ρ_t to overflow if computed for actions at the tails of the policy. For this reason, the results of `ReF-ER-2` are obtained by clipping all importance weights $\rho_t \leftarrow \min(\rho_t, 10^3)$.

Similarly to `ReF-ER`, `ACER`'s techniques (summarized in Sec. 4) guard against the numerical instability of the off-PG. `ACER` partly relies on constraining policy updates around a target-network with tuned learning and target-update rates. However, when using deep NN, small parameter updates do not guarantee small differences in the NN's outputs. Therefore, tuning the learning rates does not ensure similarity between $\pi^{\mathtt{w}}$ and RM behaviors. This can be observed in Fig. 5 from `ACER`'s superlinear relation between policy changes and the NN's learning rate η. By lowering η from
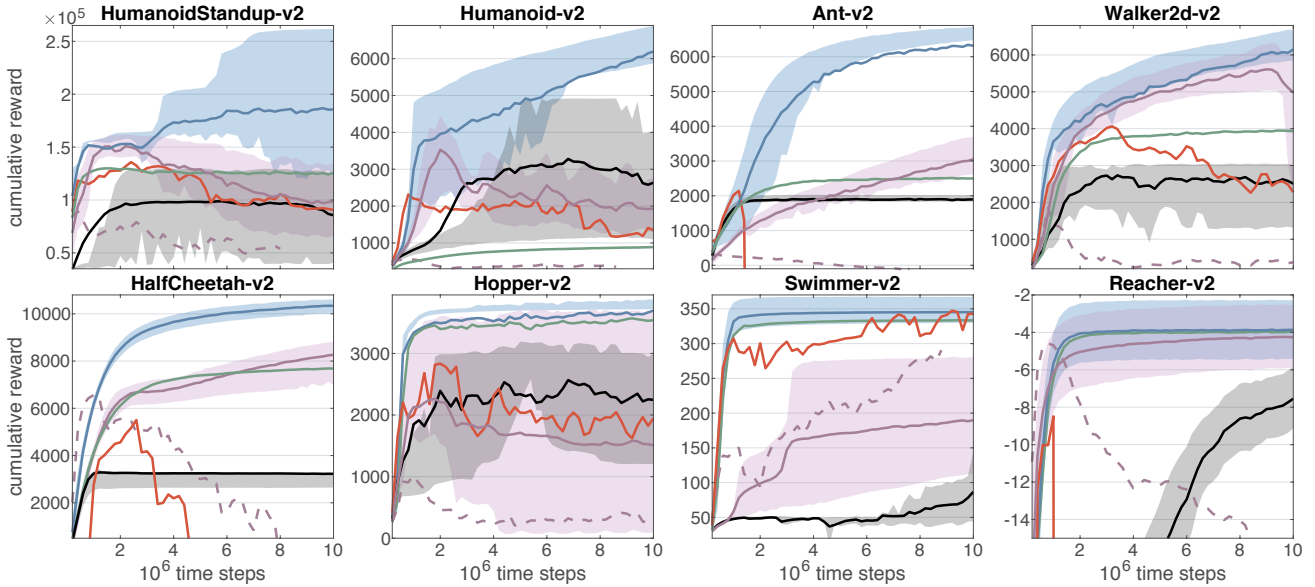
Figure 4. Average cumulative rewards on MuJoCo OpenAI Gym tasks obtained by `PPO` (black line), `ACER` (purple dashed line for $\eta = 10^{-4}$ and full line for $\eta = 10^{-5}$) and `V-RACER` with `ReF-ER` (blue), `ReF-ER-1` (red), `ReF-ER-2` (green).
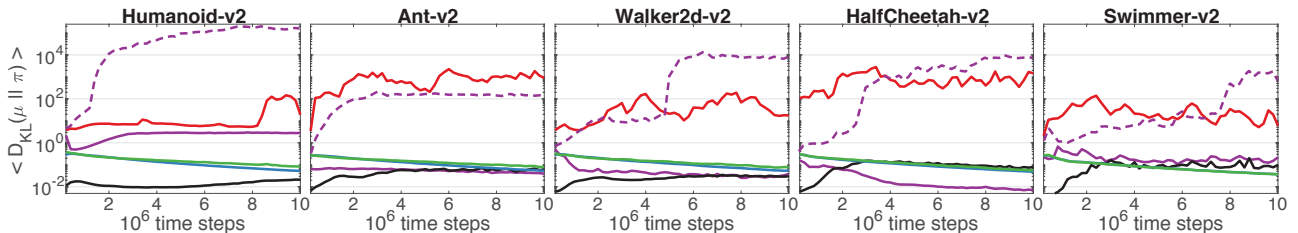

Figure 5. Kullback-Leibler divergence between π^{w} and the replayed behaviors obtained by the PG-based methods. Same colors as above.

10^{-4} to 10^{-5}, $D_{KL}(\mu_t \parallel \pi^{\text{w}})$ is reduced by multiple orders of magnitude (depending on the task). This corresponds to a large disparity in performance between the two choices of HP. For $\eta = 10^{-4}$, as $D_{KL}(\mu_t \parallel \pi^{\text{w}})$ grows orders of magnitude more than with other algorithms, off-PG estimates become inaccurate, causing `ACER` to be often outperformed by `PPO`. These experiments, together with the analysis of `DDPG`, illustrate the difficulty of controlling off-policyness in deep RL by enforcing slow parameter changes.

`ReF-ER` aids off-policy PG methods in two ways. As discussed for `DDPG` and `NAF`, Rule 2 ensures a RM of valuable experiences for estimating on-policy quantities with a finite batch size. In fact, we observe from Fig. 4 that `ReF-ER-2` alone often matches or surpasses the performance of `ACER`. Rule 1 prevents unbounded importance weights from increasing the variance of the PG and from increasing the amount of "trace-cutting" incurred by Retrace (Munos et al., 2016). Trace-cutting reduces the speed at which Q^{ret} converges to the on-policy $Q^{\pi^{\text{w}}}$ after each change to π^{w}, and consequently affects the accuracy of the loss functions. On

the other hand, skipping far-policy samples without penalties or without extremely large batch sizes (OpenAI, 2018) causes `ReF-ER-1` to have many zero-valued gradients (reducing the effective batch size) and unreliable outcomes.

Annealing $c_{\max}$ eventually provides `V-RACER` with a RM of experiences that are almost as on-policy as those used by `PPO`. In fact, while considered on-policy, `PPO` alternates gathering a small RM (usually 2^{11} experiences) and performing few optimization steps on the samples. Fig. 5 shows the average $D_{KL}(\mu_t \parallel \pi^{\text{w}})$ converging to similar values for both methods. While a small RM may not contain enough diversity of samples for the learner to accurately estimate the gradients. The much larger RM of `ReF-ER` (here $N = 2^{18}$ samples), and possibly the continually-updated value targets, allow `V-RACER` to obtain much higher returns. The Supplementary Material contains extended analysis of `V-RACER`'s most relevant HP. For many tasks presented here, `V-RACER` combined with `ReF-ER` outperforms the best result from `DDPG` (Sec. 5.1), `NAF` (Sec. 5.2), `PPO` and `ACER` and is competitive with the best published results, which to our
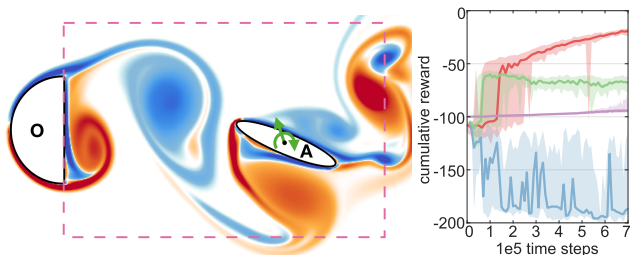
Figure 6. (left) Contours of the vorticity field (red and blue for anti- and clockwise rotation respectively) of the 2D flow control problem: the D-section cylinder is moving leftward, the agent is marked by A and by the highlighted control force and torque. (right) Returns obtained by `V-RACER` (red), `ACER` (purple), `DDPG` with ER (blue), `DDPG` with `ReF-ER` (green).

knowledge were achieved by the on-policy algorithms Trust Region Policy Optimization (Schulman et al., 2015), Policy Search with Natural Gradient (Rajeswaran et al., 2017), and Soft Actor-Critic (Haarnoja et al., 2018).

5.4. Results for a partially-observable flow control task

The problems considered so far have been modeled by ordinary differential equations (ODE), with the agent having access to the entire state of the system. We now apply the considered methods to systems described by non-linear Partial Differential Equations (PDE), here the Navier Stokes Equations (NSE) that govern continuum fluid flows. Such PDEs are used to describe many problems of scientific (e.g. turbulence, fish swimming) and industrial interests (e.g. wind farms, combustion engines). These problems pose two challenges: First, accurate simulations of PDEs may entail significant computational costs and large scale computing resources which exceed by several orders of magnitude what is required by ODEs. Second, the NSE are usually solved on spatial grids with millions or even trillions of degrees of freedom. It would be excessive to provide all that information to the agent, and therefore the state is generally measured by a finite number of sensors. Consequently, the assumption of Markovian dynamics at the core of most RL methods is voided. This may be remedied by using recurrent NN (RNN) for function approximation. In turn, RNNs add to the challenges of RL the increased complexity of properly training them. Here we consider the small 2D flow control problem of agent A, an elliptical body of major-axis D and aspect ratio 0.2, interacting with an unsteady wake. The wake is created by a D-section cylinder of diameter D (O in Fig. 6) moving at constant speed (one length D per time-unit $\mathcal{T}$) at Reynolds number $D^2/(\nu\mathcal{T})=400$. Agent A performs one action per unit $\mathcal{T}$ by imposing a force and a torque on the flow $a_t:=\{f_X, f_Y, \tau\}$. The state $s_t\in\mathbb{R}^{14}$ contains A's position, orientation and velocity relative to O and has 4 flow-speed sensors located

at A's 4 vertices. The reward is $r_{t+1}=-\|a_t\|^2$. If A exits the area denoted by a dashed line in Fig. 6, the terminal reward is -100 and the simulation restarts with random initial conditions. Otherwise, the maximum duration of the simulation is 400 actions. We attempt this problem with three differently-seeded runs of each method considered so far. Instead of maximizing the performances by HP tuning, we only substitute the MLPs used for function approximation with LSTM networks (2 layers of 32 cells with back-propagation window of 16 steps).

If correctly navigated, drafting in the momentum released into the flow by the motion of O allows A to maintain its position with minimal actuation cost. Fig. 6 shows that the optimal HP found for `ACER` (small η) in the ODE tasks, together with the lack of feature-sharing between policy and value-networks and with the variance of the off-PG, cause the method to make little progress during training. `DDPG` with ER incurs large actuation costs, while `DDPG` with `ReF-ER` is the fastest at learning to avoid the distance limits sketched in Fig. 6. In fact the critic quickly learns that A needs to accelerate leftward to avoid being left behind, and the policy adopts the behavior rapidly due to the lower variance of the DPG (Silver et al., 2014). Eventually, the best performance is reached by `V-RACER` with `ReF-ER` (an animation of a trained policy is provided in the Supplementary Material). `V-RACER` has the added benefit of having an unbounded action space and of feature-sharing: a single NN receives the combined feedback of V^{w} and π^{w} on how to shape its internal representation of the dynamics.

6. Conclusion

Many RL algorithms update a policy π^{w} from experiences collected with off-policy behaviors μ. We present evidence that off-policy continuous-action deep RL methods benefit from actively maintaining similarity between policy and replay behaviors. We propose a novel ER algorithm (`ReF-ER`) which consists of: (1) Characterizing past behaviors either as "near-policy" or "far-policy" by the deviation from one of the importance weight $\rho=\pi^{\text{w}}(a|s)/\mu(a|s)$ and computing gradients only from near-policy experiences. (2) Regulating the pace at which π^{w} is allowed to deviate from μ through penalty terms that reduce $D_{KL}(\mu||\pi^{\text{w}})$. This allows time for the learner to gather experiences with the new policy, improve the value estimators, and increase the accuracy of the next steps. We analyze the two components of `ReF-ER` and show their effects on continuous-action RL algorithms employing off-policy PG, deterministic PG (`DDPG`) and Q-learning (`NAF`). Moreover, we introduce `V-RACER`, a novel algorithm based on the off-policy PG which emphasizes simplicity and computational efficiency. The combination of `ReF-ER` and `V-RACER` reliably yields performance that is competitive with the state-of-the-art.

References

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

P. Angot, C. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81(4):497–520, 1999.

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

A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of computational physics*, 2(1): 12–26, 1967.

S. Colabrese, K. Gustavsson, A. Celani, and L. Biferale. Flow navigation by smart microswimmers via reinforcement learning. *Physical Review Letters*, 118(15):158004, 2017.

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

T. Degris, M. White, and R. S. Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

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

L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

A. Goyal, P. Brakel, W. Fedus, T. Lillicrap, S. Levine, H. Larochelle, and Y. Bengio. Recall traces: Backtracking models for efficient reinforcement learning. *arXiv preprint arXiv:1804.00379*, 2018.

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

S. Gu, T. Lillicrap, Z. Ghahramani, R. E Turner, and S. Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *International Conference on Learning Representations (ICLR)*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

D. Isele and A. Cosgun. Selective experience replay for lifelong learning. *arXiv preprint arXiv:1802.10269*, 2018.

R. Islam, P. Henderson, M. Gomrokchi, and D. Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

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

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

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

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

L. H. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3/4): 69–97, 1992.

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

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

R. Munos, T. Stepleton, A. Harutyunyan, and M. Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

G. Novati, S. Verma, D. Alexeev, D. Rossinelli, W. M. van Rees, and P. Koumoutsakos. Synchronisation through learning for two self-propelled swimmers. *Bioinspiration & Biomimetics*, 12(3):036001, 2017.

J. Oh, Y. Guo, S. Singh, and H. Lee. Self-imitation learning. *arXiv preprint arXiv:1806.05635*, 2018.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Y. Pan, M. Zaheer, A. White, A. Patterson, and M. White. Organizing experience: A deeper look at replay mechanisms for sample-based planning in continuous state domains. *arXiv preprint arXiv:1806.04624*, 2018.

K. B. Petersen, M. S. Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pp. 6553–6564, 2017.

G. Reddy, A. Celani, T. J. Sejnowski, and M. Vergassola. Learning to soar in turbulent environments. *Proceedings of the National Academy of Sciences*, 113(33):E4877–E4884, 2016.

T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015a.

T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2015b.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1889–1897, 2015.

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

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

D. Silver, A. Huang, C. J Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

G. Tucker, S. Bhupatiraju, S. Gu, R. E Turner, Z. Ghahramani, and S. Levine. The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031*, 2018.

S. Verma, G. Novati, and P. Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 2018.

Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Koray, and N. de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations (ICLR)*, 2017.