Deep Coherent Exploration for Continuous Control

Yijie Zhang [1] **Herke van Hoof** [2]

Abstract

In policy search methods for reinforcement learning (RL), exploration is often performed by injecting noise either in action space at each step independently or in parameter space over each full trajectory. In prior work, it has been shown that with linear policies, a more balanced trade-off between these two exploration strategies is beneficial. However, that method did not scale to policies using deep neural networks. In this paper, we introduce deep coherent exploration, a general and scalable exploration framework for deep RL algorithms for continuous control, that generalizes step-based and trajectory-based exploration. This framework models the last layer parameters of the policy network as latent variables and uses a recursive inference step within the policy update to handle these latent variables in a scalable manner. We find that deep coherent exploration improves the speed and stability of learning of A2C, PPO, and SAC on several continuous control tasks.

1. Introduction

The balance of exploration and exploitation (Kearns & Singh, 2002; Jaksch et al., 2010) is a longstanding challenge in reinforcement learning (RL). With insufficient exploration, states and actions with high rewards can be missed, resulting in policies prematurely converging to bad local optima. In contrast, with too much exploration, agents could waste their resources trying suboptimal states and actions, without leveraging their experiences efficiently. To learn successful strategies, this trade-off between exploration and exploitation must be balanced well, and this is known as the *exploration vs. exploitation dilemma*.

At a high level, exploration can be divided into directed strategies and undirected strategies (Thrun, 1992; Plappert et al., 2018). While directed strategies aim to extract useful information from existing experiences for better exploration, undirected strategies rely on injecting randomness into the agent's decision-making. Over the years, many sophisticated directed exploration strategies have been proposed (Tang et al., 2017; Ostrovski et al., 2017; Houthooft et al., 2016; Pathak et al., 2017). However, since these strategies still require lower-level exploration to collect the experiences, or are either complicated or computationally intensive, they are usually only used for 'hard exploration' problems where rewards are sparse, delayed, or deceptive. Except for those 'hard exploration' tasks, undirected exploration strategies are commonly used in practice and in RL literature, where some well-known examples are ϵ-greedy (Sutton, 1995) for discrete action space and additive Gaussian noise for continuous action space (Williams, 1992). Such strategies explore by randomly perturbing agents' actions at different steps independently and hence are referred to as performing *step-based* exploration in *action space* (Deisenroth et al., 2013).

As an alternative to those exploration strategies in action space, exploration by perturbing the weights of linear policies has been proposed (Rückstieß et al., 2010; Sehnke et al., 2010; Kober & Peters, 2008). Since these strategies in *parameter space* naturally explore conditioned on the states and are usually *trajectory-based*, they have the advantages of being more consistent, structured, and global (Deisenroth et al., 2013). Later, van Hoof et al. (2017) proposed a generalized exploration (GE) scheme, bridging the gap between step-based and trajectory-based exploration in parameter space. With the advance of deep RL, NoisyNet (Fortunato et al., 2018) and parameter space noise for exploration (PSNE) (Plappert et al., 2018) were introduced, extending parameter-space exploration strategies for policies using deep neural networks.

Although GE (van Hoof et al., 2017), NoisyNet (Fortunato et al., 2018), and PSNE (Plappert et al., 2018) improved the vanilla exploration strategies in parameter space and were shown leading to more global and consistent exploration, they still suffer from several limitations. Our contribution consists of a new exploration scheme to overcome these limitations. To the best of our knowledge, ours is the first scheme that combines the following characteristics:

[1]University of Copenhagen, Copenhagen, Denmark (work done while YZ was a master student at the University of Amsterdam) [2]University of Amsterdam, Amsterdam, the Netherlands. Correspondence to: Yijie Zhang <yizh@di.ku.dk>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. **Generalizing Step-based and Trajectory-based Exploration** Since both NoisyNet and PSNE are trajectory-based exploration strategies, they are considered relatively inefficient and bring insufficient stochasticity (Deisenroth et al., 2013). Following van Hoof et al. (2017), our method improves by interpolating between step-based and trajectory-based exploration in parameter space, where a more balanced trade-off between stability and stochasticity can be achieved.

2. **Recursive Analytical Integration of Latent Exploring Policies** NoisyNet and PSNE address the uncertainty from sampling exploring policies using Monte Carlo integration, while GE uses analytical integration on full trajectories, which scales poorly in the number of time steps. In contrast, we apply analytical and recurrent integration after each step, which leads to low-variance and scalable updates.

3. **Perturbing Last Layers of Policy Networks** Both NoisyNet and PSNE perturb all layers of the policy network. However, in general, only the uncertainty in parameters of the last (linear) layer can be integrated analytically. Furthermore, it is not clear that deep neural networks can be perturbed in meaningful ways for exploration (Plappert et al., 2018). We thus propose and evaluate an architecture where perturbation is only applied on the parameters of the last layer.

We will refer to our method that combines these characteristics as *deep coherent exploration*. We evaluate the coherent versions of A2C (Mnih et al., 2016), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018), where the experiments on OpenAI MuJoCo (Todorov et al., 2012; Brockman et al., 2016) tasks show that deep coherent exploration outperforms other exploration strategies in terms of both learning speed and stability.

2. Related Work

As discussed, exploration can broadly be classified into directed and undirected strategies (Thrun, 1992; Plappert et al., 2018), with undirected strategies being commonly used in practice because of their simplicity. Well known methods such as ϵ-greedy (Sutton, 1995) or additive Gaussian noise (Williams, 1992) randomly perturb the action at each time step independently. These high-frequency perturbations, however, can result in poor coverage of the state-action space due to random-walk behavior (Rückstieß et al., 2010; Deisenroth et al., 2013), washing-out of exploration by the environment dynamics (Kober & Peters, 2008; Rückstieß et al., 2010; Deisenroth et al., 2013), and potential damages to mechanical systems (Koryakovskiy et al., 2017).

One alternative is to instead perturb the agent's behavior at the beginning of the trajectory, with the perturbation held fixed during the trajectory. Possible mechanisms include posterior sampling techniques, which behave according to the solution of a randomly sampled MDP for the duration of a trajectory (Strens, 2000); drawing value functions from a posterior distribution (Osband et al., 2019); or sampling the policy parameters from a search distribution (Rückstieß et al., 2008; Sehnke et al., 2010). In particular, compared to their action-space counterparts (Sutton, 1995; Williams, 1992), Rückstieß et al. (2010) and Sehnke et al. (2010) showed that such parameter-space methods could bring improved exploration behaviors because of reduced variance and faster convergence, when combined with REINFORCE (Williams, 1992) or natural actor-critic (Peters et al., 2005).

Another alternative to independent action-space perturbations, is to correlate the noises applied at subsequent actions (Morimoto & Doya, 2000; Wawrzynski, 2015; Lillicrap et al., 2016), for example by generating perturbations from an Ornstein-Uhlenbeck (OU) process (Uhlenbeck & Ornstein, 1930). Later, van Hoof et al. (2017) used the same stochastic process but in the parameter space of the policy. This approach uses a temporally coherent exploring policy, which unifies step-based and trajectory-based exploration. Moreover, the authors showed that, with linear policies, a more delicate balance between these two extreme strategies could have better performance. However, this approach was derived in a batch mode setting and requires storing the full trajectory history and the inversion of a matrix growing with the number of time steps. Thus, it does not scale well to long trajectories or complex models.

Although these methods pioneered the research of exploration in parameter space, their applicability is limited. More precisely, these methods were only evaluated with extremely shallow (often linear) policies and relatively simple tasks with low-dimensional state spaces and action spaces. Given this, NoisyNet (Fortunato et al., 2018), PSNE (Plappert et al., 2018), and Stochastic A3C (Shang et al., 2019) were proposed, introducing more general and scalable methods for deep RL algorithms.

All three of these methods can be seen as learning a distribution over policies for trajectory-based exploration in parameter space. These exploring policies are sampled by perturbing the weights across all layers of a deep neural network, with the uncertainty from sampling being addressed by Monte Carlo integration. Whereas NoisyNet learns the magnitudes of the noises for each parameter, PSNE heuristically adapts a single magnitude for all parameters.

While showing good performance in practice (Fortunato et al., 2018; Plappert et al., 2018), these methods suffer from two potential limitations. Firstly, trajectory-based strategies can be inefficient as only one strategy can be evaluated for a potentially long trajectory (Deisenroth et al., 2013), which

could fail to escape local optima. Secondly, Monte Carlo integration results in high-variance gradient estimates that could lead to oscillating updates.

3. Background

3.1. Reinforcement Learning

Reinforcement learning is a sub-field of machine learning that studies how an agent learns strategies with high returns through trial-and-error by interacting with an environment. This interaction between an agent and an environment is described using Markov Decision Processes (MDPs). A MDP is a tuple $(\mathcal{S}, \mathcal{A}, r, P, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function with $r_t = r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the transition probability function, and γ is a discount factor indicating the preference of short-term rewards.

In RL with continuous action space, an agent aims to learn a parametrized (e.g. Gaussian) policy $\pi_{\boldsymbol{\theta}}(\mathbf{a}|\mathbf{s}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$, with parameters $\boldsymbol{\theta}$, that maximizes the expected return over trajectories:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim p(\tau|\pi_{\boldsymbol{\theta}})}[R(\tau)], \tag{1}$$

where $\tau = (\mathbf{s}_0, \mathbf{a}_0, ..., \mathbf{a}_{T-1}, \mathbf{s}_T)$ is a trajectory and $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$ is the discounted return.

3.2. Deep Reinforcement Leaning Algorithms

Deep reinforcement learning combines deep learning and reinforcement learning, where policies and value functions are represented by deep neural networks for more sophisticated and powerful function approximation. In our experiments, we consider the following three deep RL algorithms.

Advantage Actor-Critic (A2C) Closely related to REINFORCE (Williams, 1992), A2C is an on-policy algorithm proposed as the synchronous version of the original asynchronous advantage actor-critic (A3C) (Mnih et al., 2016). The gradient of A2C can be written as:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim p(\tau|\pi_{\boldsymbol{\theta}})} \left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) A^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) \right], \tag{2}$$

where $A^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) = Q^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t)$ is the advantage function that measures how much on average a specific action $\mathbf{a}_t$ is better than other actions in state $\mathbf{s}_t$ when following policy $\pi_{\boldsymbol{\theta}}$. Here $Q^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\tau \sim p(\tau|\pi_{\boldsymbol{\theta}})}[R_t(\tau) \mid \mathbf{S}_t = \mathbf{s}, \mathbf{A}_t = \mathbf{a}]$ is the action-value function (also known as the Q-function) and $V^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t) = \mathbb{E}_{\tau \sim p(\tau|\pi_{\boldsymbol{\theta}})}[R_t(\tau) \mid \mathbf{S}_t = \mathbf{s}]$ is the state-value function (also known as the V-function), where $R_t(\tau) = \sum_{t'=t}^{T} \gamma^{(t'-t)} r_{t'}$ is the discounted rewards-to-go, defined as the sum of discounted rewards starting from step t.

Proximal Policy Optimization (PPO) PPO (Schulman et al., 2017) is an on-policy algorithm developed to determine the largest step for update while still keeping the updated policy close to the old policy in terms of Kullback–Leibler (KL) divergence. Instead of using a second-order method as in trust region policy optimization (TRPO) (Schulman et al., 2015), PPO applies a first-order method and combines several tricks to relieve the complexity. We consider the primary variant PPO-Clip with the following surrogate objective:

$$L^{CLIP}_{\boldsymbol{\theta}_k}(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim p(\tau|\boldsymbol{\theta}_k)} \left[\sum_{t=0}^{T-1} \left[\min\left(r_t(\boldsymbol{\theta}), \text{clip}\left(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon\right)\right) A_t^{\pi_{\boldsymbol{\theta}_k}} \right] \right], \tag{3}$$

where $r_t(\boldsymbol{\theta}) = \dfrac{\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{\boldsymbol{\theta}_k}(\mathbf{a}_t|\mathbf{s}_t)}$ and ϵ is a small threshold that approximately restricts the distance between the new policy and the old policy. In practice, to prevent the new policy from changing too fast, the KL divergence between the new policy and the old policy approximated on a sampled batch is often used as a further constraint.

Soft Actor-Critic (SAC) SAC (Haarnoja et al., 2018) is an entropy-regularized (Ziebart et al., 2008) off-policy actor-critic method (Lillicrap et al., 2016; Fujimoto et al., 2018) with a stochastic policy. Using 'soft' Bellman back-ups with off-policy data, SAC learns the optimal entropy-regularized Q-function defined as:

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}', \tilde{\mathbf{a}}'}\left[r + \gamma \left(Q^{\pi}(\mathbf{s}', \tilde{\mathbf{a}}') + \alpha H\left(\pi(\tilde{\mathbf{a}}'|\mathbf{s}')\right)\right)\right], \tag{4}$$

where $\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a}), \tilde{\mathbf{a}}' \sim \pi(\tilde{\mathbf{a}}'|\mathbf{s}')$, H is the entropy, and α is the temperature parameter. The policy is then learned by maximizing the expected entropy-regularized Q-function via the reparameterization trick (Kingma et al., 2015).

3.3. Undirected Exploration Strategies

While directed exploration exploits global information to decide which action to try systematically, undirected exploration is realised by local and random perturbations (Thrun, 1992). Roughly speaking, undirected strategies can be classified into two dimensions (Deisenroth et al., 2013).

Action Space vs. Parameter Space Exploration In continuous control tasks, exploration in action space is usually performed by adding spherical Gaussian noise to the sampled action. In contrast, exploration in parameter space is often implemented by imposing Gaussian noise on the policy parameters. In practice, exploration in action space is sometimes preferred (Baxter & Bartlett, 2000; Sutton et al., 1999; Williams, 1992) because it is straightforward and easy to understand. In contrast, exploration in parameter space

has the advantages of being more consistent, structured, and global as it naturally explores conditioned on the states (Sehnke et al., 2010; Rückstieß et al., 2008; Deisenroth et al., 2013; Plappert et al., 2018).

Trajectory-based vs. Step-based Exploration Step-based exploration strategies rely on injecting exploration noise at each step independently and trajectory-based exploration strategies often add exploration noise at the beginning of a trajectory. Step-based exploration strategies are more random, leading to unreproducible action sequences. The effects of the perturbations can sometimes be hard to estimate as they can be washed out by the system dynamics (Kober & Peters, 2008; Rückstieß et al., 2008; Deisenroth et al., 2013). However, this randomness could sometimes be helpful as it could make the policy less prone to getting trapped in a local optimum. In contrast, trajectory-based exploration produces reproducible action sequences and are often more stable (Deisenroth et al., 2013). This increased stability is also helpful for more consistent policy evaluation and leads to more reliable policy updates (Stulp & Sigaud, 2012; Sehnke et al., 2010; Deisenroth et al., 2013).

4. Deep Coherent Exploration for Continuous Control

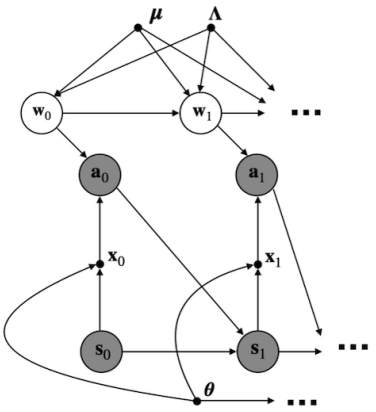


Figure 1. Graphical model of deep coherent exploration.

To achieve the desiderata in Section 1, we propose deep coherent exploration, a method that models the policy as a generative model with latent variables. This policy is represented as $\pi_{\mathbf{w}_t,\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t) = \mathcal{N}(\mathbf{W}_t f_{\boldsymbol{\theta}}(\mathbf{s}_t) + \mathbf{b}_t, \boldsymbol{\Lambda}_a^{-1})$. Here $\mathbf{w}_t$ denotes all last layer parameters of the policy network at step t by combining $\mathbf{W}_t$ and $\mathbf{b}_t$, $\boldsymbol{\theta}$ denotes the parameters of the policy network except for the last layer, and $\boldsymbol{\Lambda}_a$ is a fixed and diagonal precision matrix. Our method treats the last layer parameters $\mathbf{w}_t$ as latent variables with marginal distribution $\mathbf{w}_t \sim \mathcal{N}\left(\boldsymbol{\mu}_t, \boldsymbol{\Lambda}_t^{-1}\right)$, where $\boldsymbol{\mu}_t$ and $\boldsymbol{\Lambda}_t$ are functions of learnable parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$ respectively. In this model, all learnable parameters can be denoted as $\zeta = (\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\theta})$. We provide the graphical model of deep coherent exploration in Figure 1.

As in van Hoof et al. (2017), deep coherent exploration generalizes step-based and trajectory-based exploration by constructing a Markov chain of $\mathbf{w}_t$. This Markov chain specifies joint probabilities through an initial distribution $p_0(\mathbf{w}_0) = \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right)$ and the conditional distribution $p(\mathbf{w}_t|\mathbf{w}_{t-1})$. This latter term explicitly expresses temporal coherence between subsequent parameter vectors. In this setting, step-based exploration corresponds to the extreme case when $p(\mathbf{w}_t|\mathbf{w}_{t-1}) = p_0(\mathbf{w}_t)$, and trajectory-based exploration corresponds to another extreme case when $p(\mathbf{w}_t|\mathbf{w}_{t-1}) = \delta(\mathbf{w}_t - \mathbf{w}_{t-1})$, where δ is the Dirac delta function. To ensure the marginal distribution of $\mathbf{w}_t$ will be equal to the initial distribution p_0 at any step t, we directly follow van Hoof et al. (2017) with the following transition distribution for $\mathbf{w}_t$:

$$p(\mathbf{w}_t|\mathbf{w}_{t-1}) = \mathcal{N}\left((1-\beta)\,\mathbf{w}_{t-1} + \beta\boldsymbol{\mu}, (2\beta - \beta^2)\boldsymbol{\Lambda}^{-1}\right), \tag{5}$$

where β is a hyperparameter that controls the temporal coherency of $\mathbf{w}_t$ and $\mathbf{w}_{t-1}$. Then, the two extreme cases correspond to $\beta = 0$ for trajectory-based exploration and $\beta = 1$ for step-based exploration, while the intermediate exploration corresponds to $\beta \in (0, 1)$. For intermediate values of β, we obtain smoothly changing policies that sufficiently explore, while reducing high-frequency perturbations.

4.1. On-Policy Deep Coherent Exploration

Our method can be combined with all on-policy policy gradient methods and here we present this adaptation with REINFORCE (Williams, 1992). Starting from the RL objective in Equation 1:

$$\nabla_\zeta J(\zeta) = \mathbb{E}_{\tau\sim p(\tau|\zeta)}\left[\nabla_\zeta \log p(\tau|\zeta) R(\tau)\right], \tag{6}$$

the gradients w.r.t. the sampled trajectory can be obtained using standard chain rule:

$$\nabla_\zeta \log p(\tau|\zeta) = \sum_{t=0}^{T-1} \left(\nabla_\zeta \log p(\mathbf{a}_t|\mathbf{s}_{[0:t]}, \mathbf{a}_{[0:t-1]}, \zeta)\right). \tag{7}$$

Here, since information can still flow through the unobserved latent variable $\mathbf{w}_t$, our policy is *not Markov* anymore. To simplify this dependency, we introduce $\mathbf{w}_t$ into $p(\mathbf{a}_t|\mathbf{s}_{[0:t]}, \mathbf{a}_{[0:t-1]}, \zeta)$:

$$\begin{aligned} &p(\mathbf{a}_t|\mathbf{s}_{[0:t]}, \mathbf{a}_{[0:t-1]}, \zeta) \\ &= \int p(\mathbf{a}_t, \mathbf{w}_t|\mathbf{s}_{[0:t]}, \mathbf{a}_{[0:t-1]}, \zeta) d\mathbf{w}_t \qquad (8) \\ &= \int \underbrace{p(\mathbf{a}_t|\mathbf{s}_t; \mathbf{w}_t, \boldsymbol{\theta})}_{\text{Gaussian policy}} \underbrace{p(\mathbf{w}_t|\mathbf{s}_{[0:t-1]}, \mathbf{a}_{[0:t-1]}, \boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\theta})}_{\text{forward message } \alpha(\mathbf{w}_t)} d\mathbf{w}_t, \qquad (9) \end{aligned}$$

where the first factor is the Gaussian action probability given by the policy and the second factor can be interpreted as the forward message $\alpha(\mathbf{w}_t)$ along the chain. We decompose $\alpha(\mathbf{w}_t)$ by introducing $\mathbf{w}_{t-1}$:

$$\alpha(\mathbf{w}_t) = \int p(\mathbf{w}_t, \mathbf{w}_{t-1}|\mathbf{s}_{[0:t-1]}, \mathbf{a}_{[0:t-1]}, \zeta) d\mathbf{w}_{t-1} \quad (10)$$

$$= \int \underbrace{p(\mathbf{w}_t|\mathbf{w}_{t-1}; \boldsymbol{\mu}, \boldsymbol{\Lambda})}_{\text{transition probability of } \mathbf{w}_t} \frac{p(\mathbf{a}_{t-1}|\mathbf{s}_{t-1}; \mathbf{w}_{t-1}, \boldsymbol{\theta})\alpha(\mathbf{w}_{t-1})}{\mathbf{Z}_{t-1}} d\mathbf{w}_{t-1}, \quad (11)$$

where the first factor is the transition probability of $\mathbf{w}_t$ (Equation 5) and $\mathbf{Z}_{t-1}$ is a normalizing constant. Since Gaussians are closed under marginalization and condition, the second factor can be obtained analytically without the need of computing the normalizing constant $\mathbf{Z}_{t-1}$. Moreover, $\alpha(\mathbf{w}_{t-1})$ is a Gaussian by mathematical induction from the initial step. As a result, we arrive at an efficient recursive expression for exact inference of $\mathbf{w}_t$ (Equation 11). Again, with the property of Gaussians, all integrals appearing above can be solved analytically, where the marginal action probability given the history at each step t can be obtained and used for policy updates.

Summarizing, non-Markov policies require substituting the regular $p(\mathbf{a}_t|\mathbf{s}_t; \boldsymbol{\theta})$ term in the update equation with $p(\mathbf{a}_t|\mathbf{s}_{[0:t]}, \mathbf{a}_{[0:t-1]}, \zeta)$ (Equation 7). Equations 8-11 show how this expression can be efficiently calculated recursively. Except for this substitution, learning algorithms like A2C and PPO can proceed as normal. Because of the parameter sampling and recursive exact inference, our method has a added complexity of $\mathcal{O}\left(nd_{\text{last}}^3\right)$ [1], where n is the number of time steps and d_{last} is the dimension of the parameter vector in the last layer. For detailed mathematical derivation, please refer to the supplementary material.

4.2. Off-Policy Deep Coherent Exploration

Combining our method with off-policy methods (Lillicrap et al., 2016; Fujimoto et al., 2018; Haarnoja et al., 2018) requires defining both the behavior policy and the update equation. The behavior policy is the same as in on-policy methods (Equation 5). The policy update procedure may require adjustments for specific algorithms. Here, we show how to adapt our method for SAC (Haarnoja et al., 2018). In the SAC policy update, the target policy is adapted towards the exponential of the new Q-function.

However, since SAC is off-policy, its exploration and policy update are separated, making the recursive update described before inapplicable. During exploration, coherent exploration takes place as described before. In the SAC update, states are sampled *randomly* from the replay buffer, which violates our graphical model (Figure 1), and thus sampling $\mathbf{w}_t$ per step sequentially is no longer a sensible choice for off-policy policy update. Instead, for low-variance gradient estimates we choose the marginal policy $p\left(\mathbf{a}_t|\mathbf{s}_t, \zeta\right)$ rather than the policy conditioned on the sampled last layer parameters $\mathbf{w}$ to be the target policy. The marginal policy $p\left(\mathbf{a}_t|\mathbf{s}_t, \zeta\right)$ can be obtained analytically for Gaussian distributions:

$$p\left(\mathbf{a}_t|\mathbf{s}_t, \zeta\right) = \int \underbrace{p\left(\mathbf{a}_t|\mathbf{s}_t; \mathbf{w}, \boldsymbol{\theta}\right)}_{\text{Gaussian policy}} \underbrace{p_0(\mathbf{w}|\boldsymbol{\mu}, \boldsymbol{\Lambda})}_{\text{initial probability}} d\mathbf{w}. \quad (12)$$

This consideration leads to the following objective for policy update:

$$J(\zeta) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\text{KL}\left(p\left(\mathbf{a}_t|\mathbf{s}_t, \zeta\right) \middle\| \frac{\exp\left(Q_{\boldsymbol{\phi}}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right)}{Z_{\boldsymbol{\phi}}\left(\mathbf{s}_t\right)}\right)\right] \quad (13)$$

$$= \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \mathbf{a}_t \sim p(\mathbf{a}_t|\mathbf{s}_t, \zeta)}\left[\log p(\mathbf{a}_t|\mathbf{s}_t, \zeta) - Q_{\boldsymbol{\phi}}\left(\mathbf{s}_t, \mathbf{a}_t\right)\right], \quad (14)$$

where $\mathcal{D}$ denotes the replay buffer and $\boldsymbol{\phi}$ denotes the parameters of the Q-function. The expected value over actions is approximated using a sample from the marginal policy. Then, all parameters can be learned from the sampled action $\mathbf{a}_t$ via the reparameterization trick (Kingma et al., 2015). When combining with SAC, our method has an extra complexity of $\mathcal{O}(nd_{\text{last}} + d_{\text{last}}^3)$, as the marginalization is performed only once for each policy update.

5. Experiments

For the experiments, we compare our method with NoisyNet (Fortunato et al., 2018), PSNE (Plappert et al., 2018) and standard action noise (Williams, 1992). This comparison is evaluated in combination of A2C (Mnih et al., 2016), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018) on OpenAI Gym MuJoCo (Todorov et al., 2012; Brockman et al., 2016) continuous control tasks.

For exploration in parameter space, we use a fixed action noise with a standard deviation of 0.1. For A2C and PPO, their standard deviations of parameter noise are all initialized at 0.017, as suggested in Fortunato et al. (2018). For SAC, we initialize the standard deviation of parameter noise at 0.034 for both our method and PSNE as it gave better results in practice. Besides, deep coherent exploration learns the logarithm of parameter noise, while NoisyNet learns the parameter noise directly, and PSNE adapts the parameter noise. We consider five values of β (0.0, 0.01, 0.1, 0.5, and 1.0) for deep coherent exploration, where we use $\beta = 0.01$ for comparative evaluation with other exploration strategies.

[1] The cubic complexity arises from the naive algorithms for matrix multiplication (Schoolbook matrix multiplication) and inversion (Gauss–Jordan elimination). In practice, faster algorithms could be used.

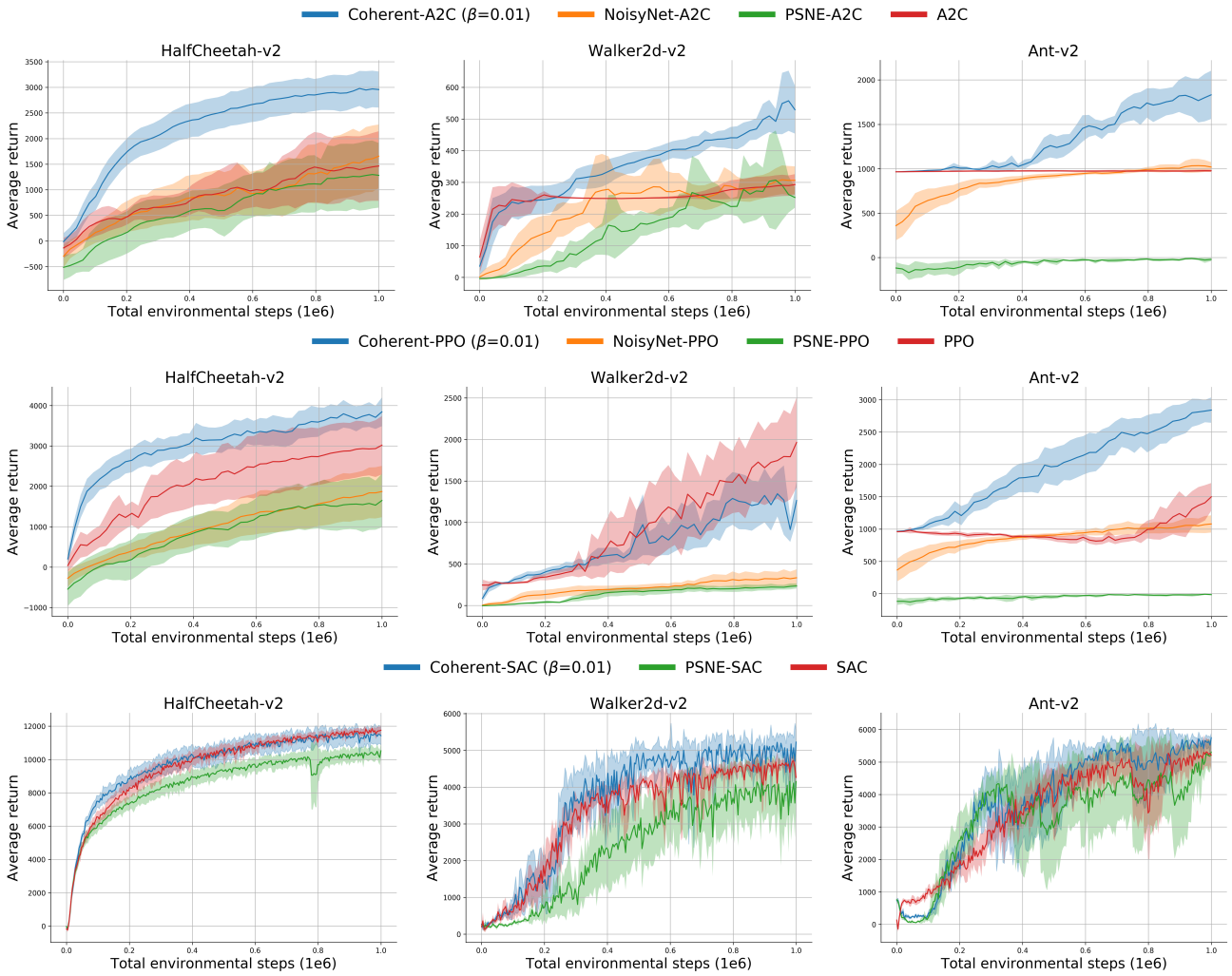


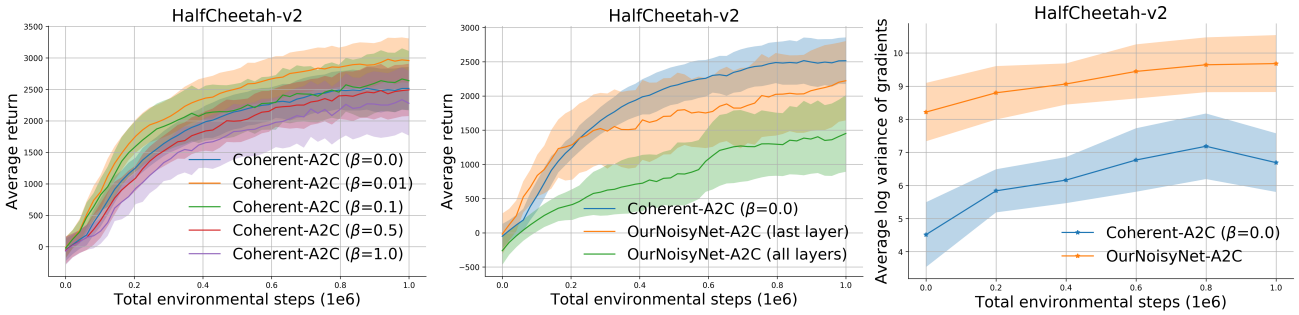
Figure 2. Learning curves for deep RL algorithms with different exploration strategies on OpenAI MuJoCo continuous control tasks, where the top, middle and bottom row corresponds to results of A2C, PPO, and SAC respectively. The solid curves correspond to the mean, and the shaped region represents two times the standard error of the average return over 10 random seeds.

Our implementation of NoisyNet is based on the code from Kaixhin[2]. For PSNE, we refer to the authors' implementation in OpenAI Baselines[3] (Dhariwal et al., 2017) and the original paper, where we set the KL threshold for A2C and PPO to 0.01 and the MSE threshold for SAC to 0.1. On the other hand, exploration with action noise uses the default setting proposed by Achiam (2018)[4], where the standard deviation of action noise is initialized at around 0.6 for A2C and PPO. For SAC, in the baseline setting the standard deviation of action noise is decided by the policy network and the state.

In all experiments, agents are trained with a total of 10^6 environmental steps, where they are updated after each epoch. A2C and PPO use four parallel workers, where each worker collects a trajectory of 1000 steps for each epoch, resulting in epochs with 4000 steps in total. After each epoch, both A2C and PPO update their value functions for 80 gradient steps. At the same time, A2C updates its policy for one gradient step, while PPO updates its policy for up to 80 gradient steps until the KL constraint is satisfied. SAC uses a single worker, with a step size of 4000 for each epoch. After every 50 environmental steps, both the policy and the value function are updated for 50 gradient steps. To make the parameter noise stable, we learn or adapt the standard deviation of parameter noise after each epoch.

Our implementation of A2C, PPO, and SAC with different exploration strategies are adapted based on OpenAI Spin-

[2]https://github.com/Kaixhin/NoisyNet-A3C
[3]https://github.com/openai/baselines/tree/master/baselines/ddpg
[4]https://spinningup.openai.com/en/latest/



(a) (b) (c)

Figure 3. Results of Coherent-A2C with different settings for HalfCheetah-v2, where Figure 3a and Figure 3b show the learning curves and Figure 3c shows the average log variance of gradients during six stages in learning. The solid curves correspond to the mean, and the shaped region represents two times the standard error of the average return over 10 random seeds.

ning Up (Achiam, 2018) with default settings. All three algorithms use two-layer feedforward neural networks with the same network architectures for both policy and value function. Precisely, A2C and PPO use a network architecture with 64 and 64 hidden nodes activated by tanh units. In comparison, SAC uses a network architecture of 256 and 256 hidden nodes activated by rectified linear units (ReLU). Parameters of policies and value functions in all three algorithms are updated using Adam (Kingma & Ba, 2015). A2C and PPO use a learning rate of $3 \cdot 10^{-4}$ for the policies and a learning rate of 10^{-3} for the value functions. SAC uses a single learning rate of 10^{-3} for both policy and value function.

For each task, we evaluate the performance of agents after every 5 epochs, with no exploration noise. Additionally, each evaluation reports the average reward of 10 episodes for each worker. To mitigate the randomness within environments and policies, we report our results as the average over 10 random seeds. All settings not explicitly explained in this section, are set to the code base defaults. For more details, please refer to documents and source code from OpenAI Spinning Up (Achiam, 2018) and our implementation[5].

5.1. Comparative Evaluation

In this section, we present the results for A2C (Mnih et al., 2016), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018) on HalfCheetah-v2, Walker2d-v2, and Ant-v2, with the additional results on Hopper-v2, Reacher-v2, and InvertedDoublePendulum-v2 shown in the supplementary material. Figure 2 shows that, overall, Coherent-A2C outperforms all other A2C-based methods in terms of learning speed, final performance, and algorithm stability. In particular, given that Ant-v2 is considered a challenging task, deep coherent exploration considerably accelerates learning speed. For PPO-based methods, deep coherent exploration still outperforms NoisyNet and PSNE significantly in all tasks. However, our method's advantage compared to standard action noise is smaller. Particularly, Coherent-PPO underperforms PPO in Walker2d-v2. Two reasons might explain this. Firstly, some environments might be more unstable and require a larger degree of exploration, which favors PPO as it initializes its action noise with a much greater value. Secondly, because of having extra parameters, policies of Coherent-PPO, NoisyNet-PPO, and PSNE-PPO tend to satisfy the KL constraint in fewer update steps, which leads to slower learning. The latter together with the high variance of gradient estimates also explain the poor performance of NoisyNet-PPO and PSNE-PPO, as the more oscillating updates of the parameters could result in significantly KL-different policies and hence even fewer policy updates. This is observed in our experiments (Figure 5) and also supported by the similar performance of NoisyNet and PSNE when combined with PPO and A2C. For SAC, the advantages of deep coherent exploration are smaller compared to A2C and PPO. More specifically, Coherent-SAC learns slightly faster than SAC in HalfCheetah-v2 and achieves the highest average returns in Walker2d-v2 and Ant-v2. Furthermore, between different random seeds, Coherent-SAC shows standard errors lower than PSNE-SAC but higher than the baseline SAC.

5.2. Ablation Studies

In this section, we present three separate ablation studies to clarify the effects of each characteristic discussed in Section 1. These ablation studies are performed with A2C, to ensure that all characteristics are applicable and also the fixed number of gradient steps for policy updates puts different variants on equal footing. The three ablation studies are performed on HalfCheetah-v2.

Generalizing Step-based and Trajectory-based Exploration As shown in Figure 3a, we evaluate five different

[5]https://github.com/pyijiezhang/deep-coherent-exploration-for-continuous-control

values 0.0, 0.01, 0.1, 0.5, and 1.0 of β for Coherent-A2C. Except for the large value of $\beta = 0.5$, the two intermediate strategies ($\beta = 0.01$ and 0.1) both outperform step-based strategy ($\beta = 1.0$) and trajectory-based strategy ($\beta = 0.0$) quite significantly. Coherent-A2C with $\beta = 0.01$ seems to achieve the best balance between randomness and stability, with a considerably higher return than the other four.

Analytical Integration of Latent Exploring Policies We introduce OurNoisyNet for comparison. OurNoisyNet equips a noisy linear layer for only its last layer, and this layer learns the logarithm of standard deviation, as in deep coherent exploration. We compare Coherent-A2C using $\beta = 0.0$ and OurNoisyNet-A2C, with the only difference thus being whether we integrate analytically or use the reparameterization trick (Kingma et al., 2015).

We first measure the variance of gradient estimates in both variants. This variance is measured by computing the trace of the covariance matrix using 10 gradient samples. We report this measure in six stages during training, as shown in Figure 3c. We can observe that analytical integration leads to lower-variance gradient estimates across all training stages for HalfCheetah-v2. We further present the learning curves of both variants in Figure 3b, where Coherent-A2C with $\beta = 0.0$ shows higher return than OurNoisyNet-A2C. Interestingly, Coherent-A2C displays a much lower standard error across different random seeds. Furthermore, the lower-variance gradient estimates of Coherent-A2C could enable a larger learning rate for faster training without making policy updates unstable.

Perturbing Last Layers of Policy Networks In this part, we compare OurNoisyNet-A2C perturbed over all layers and OurNoisyNet-A2C perturbed over only the last layer. The result is shown in Figure 3b. We can see that the latter performs considerably better with a significantly higher return. This is somewhat to our surprise since we chose to perturb only the last layer such that exact inference and tractable computation are possible. There are several possible reasons. Firstly, since it is unknown how the parameter noise (especially in lower layers) is realized in action noise, perturbing all layers of the policy network may lead to uncontrollable perturbations. Such excess exploratory noise could inhibit exploitation. Secondly, perturbing all layers might disturb the representation learning of states, which is undesirable for learning a good policy. Thirdly, perturbing only the last layer could also lead to fewer parameters for NoisyNet.

5.3. Environments with Sparse Rewards

Having confirmed that deep coherent exploration could bring improved performance in environments with dense rewards, we now turn to much more challenging continuous control tasks where the reward signals are sparse. We consider the task of SparseContinuousCartpole-v0, where the reward is only available when the paddle is lifted over a given angle. In addition, we consider PPO for the experiments because Coherent-PPO combines all the characteristics and performs well in most environments. The results are shown in Figure 4. We can observe that Coherent-PPO performs the best, with faster learning and lower standard error across different random seeds than all the other exploration strategies. However, considering SparseContinuousCartpole-v0 is a relatively simple task, we also try all the PPO-based exploration strategies on more difficult continuous control tasks such as SparseHalfCheetah-v2 and SparseWalker2d-v2, where the reward is only available when the agent moves forward over a specific number of units. In those experiments, all of the methods fail, clearly showing that undirect strategies are not competent in, or even capable of solving 'hard exploration' problems. Thus, we conclude that undirected exploration strategies should only be used in environments with dense rewards.

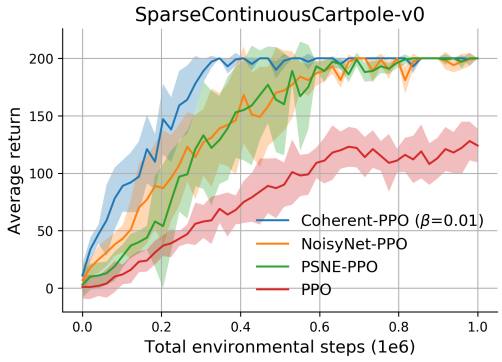


Figure 4. Learning curves for PPO with different exploration strategies on SparseContinuousCartpole-v0. The solid curves correspond to the mean, and the shaped region represents two times the standard error of the average return over 10 random seeds.

5.4. Stochasticity in Last Layers of Policy Networks

Policy search methods with undirected exploration strategies tend to reduce the magnitude of exploration noise during policy updates, which can result in premature convergence to suboptimal solutions (Peters et al., 2010). In this section, we analyze the average stochasticity in last layers of policy networks to see if this could also happen for deep coherent exploration. We consider PPO-based methods for analysis as they provide a greater number of policy updates such that the patterns become clearer to observe. For deep coherent exploration and NoisyNet, we measure the average standard deviation over the dimension of last layer parameters of the policy networks (NoisyNet uses absolute values before averaging). For PSNE, we use the single standard deviation for all parameters. Figure 5 shows the patterns of last layer

stochasticity during policy learning for each of the 10 different random seeds, where each curve has a different number of total policy updates. We should also note that such last layer stochasticity is all randomness in parameter space for deep coherent exploration, while part of the randomness in parameter space for NoisyNet and PSNE. In general, the last layer stochasticity of both Coherent-PPO and NoisyNet-PPO has a stable and consistent decrease, indicating steady trends towards exploitation. More specially, the last layer stochasticity of Coherent-PPO seems to have a more persistent change. Unlike Coherent-PPO and NoisyNet-PPO, PSNE-PPO has a rapidly decreasing last layer stochasticity and the three outliers show that the differences between random seeds could be huge. This is understandable as PSNE heuristically set a value for the distance measure to adapt its stochasticty, while a good value can be hard to determine and could be highly different in various settings. Moreover, in high-dimensional control tasks, different action dimensions could require different levels of randomness for exploration that are correlated. Thus, adapting a single stochasticity metric for all parameters could ignore such correlations and provide suboptimal solutions. Critically, the last layer stochasticity is not the only factor in exploration, so these results have to be interpreted with care. However, it does seem that deep coherent exploration well preserves its exploration behavior over the course of learning.

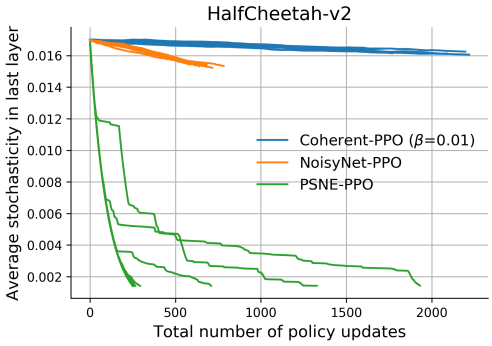


Figure 5. Average stochasticity over dimension of last layer parameters of policy networks for PPO with different exploration strategies on HalfCheetah-v2. The solid curves show the last layer stochasticity for each of the 10 random seeds during training.

6. Conclusion

In this paper, we have presented a general and scalable exploration framework that extends the GE scheme (van Hoof et al., 2017) for continuous deep RL algorithms. In particular, recursive calculation of marginal action probabilities allows handling long trajectories and high-dimensional parameter vectors. Compared with NoisyNet (Fortunato et al., 2018) and PSNE (Plappert et al., 2018), our method has three improvements. Firstly, deep coherent exploration generalizes step-based and trajectory-based exploration in parameter space, which allows a more balanced trade-off between stochasticity and coherence. Secondly, deep coherent exploration analytically marginalizes the latent policy parameters, yielding lower-variance gradient estimates that stabilize and accelerate learning. Thirdly, by perturbing only the last layer of the policy network, deep coherent exploration provides better control of the injected noise.

When combining with A2C (Mnih et al., 2016), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018), we empirically show that deep coherent exploration outperforms other exploration strategies on most of the MuJoCo continuous control tasks tested. Furthermore, the ablation studies show that, while each of the improvements is beneficial, combining them leads to even faster and more stable learning. For future work, since deep coherent exploration uses a fixed and small action noise, we believe one interesting direction is to study whether the learnable perturbations in action space can be combined with our method in a meaningful way for even more effective exploration.

Acknowledgements

We thank SURFsara[6] for providing the computational resources needed for this paper.

References

Achiam, J. Spinning Up in Deep Reinforcement Learning. 2018.

Baxter, J. and Bartlett, P. L. Direct gradient-based reinforcement learning. In *ISCAS*, pp. 271–274. IEEE, 2000.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Deisenroth, M. P., Neumann, G., and Peters, J. A survey on policy search for robotics. *Found. Trends Robotics*, 2 (1-2):1–142, 2013.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines, 2017.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. Noisy networks for exploration. In *ICLR (Poster)*. OpenReview.net, 2018.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018.

[6] https://www.surf.nl/

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018.

Houthooft, R., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. VIME: variational information maximizing exploration. In *NIPS*, pp. 1109–1117, 2016.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *J. Mach. Learn. Res.*, 11:1563–1600, 2010.

Kearns, M. J. and Singh, S. P. Near-optimal reinforcement learning in polynomial time. *Mach. Learn.*, 49(2-3):209–232, 2002.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. *CoRR*, abs/1506.02557, 2015.

Kober, J. and Peters, J. Policy search for motor primitives in robotics. In *NIPS*, pp. 849–856. Curran Associates, Inc., 2008.

Koryakovskiy, I., Vallery, H., Babuška, R., and Caarls, W. Evaluation of physical damage associated with action selection strategies in reinforcement learning. *IFAC-PapersOnLine*, 50(1):6928–6933, 2017. ISSN 1474-6670. doi: 10.1016/j.ifacol.2017.08.1218. 20th World Congress of the International Federation of Automatic Control (IFAC), 2017, IFAC 2017 ; Conference date: 09-07-2017 Through 14-07-2017.

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 *ICLR (Poster)*, 2016.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1928–1937. JMLR.org, 2016.

Morimoto, J. and Doya, K. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. In *ICML*, pp. 623–630. Morgan Kaufmann, 2000.

Osband, I., Roy, B. V., Russo, D. J., and Wen, Z. Deep exploration via randomized value functions. *J. Mach. Learn. Res.*, 20:124:1–124:62, 2019.

Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. Count-based exploration with neural density models. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2721–2730. PMLR, 2017.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2778–2787. PMLR, 2017.

Peters, J., Vijayakumar, S., and Schaal, S. Natural actor-critic. In *ECML*, volume 3720 of *Lecture Notes in Computer Science*, pp. 280–291. Springer, 2005.

Peters, J., Mülling, K., and Altun, Y. Relative entropy policy search. In *AAAI*. AAAI Press, 2010.

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 *ICLR (Poster)*. OpenReview.net, 2018.

Rückstieß, T., Felder, M., and Schmidhuber, J. State-dependent exploration for policy gradient methods. In *ECML/PKDD (2)*, volume 5212 of *Lecture Notes in Computer Science*, pp. 234–249. Springer, 2008.

Rückstieß, T., Sehnke, F., Schaul, T., Wierstra, D., Sun, Y., and Schmidhuber, J. Exploring parameter space in reinforcement learning. *Paladyn J. Behav. Robotics*, 1(1): 14–24, 2010.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 1889–1897. JMLR.org, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

Shang, W., van der Wal, D., van Hoof, H., and Welling, M. Stochastic activation actor critic methods. In *ECML/PKDD (3)*, volume 11908 of *Lecture Notes in Computer Science*, pp. 103–117. Springer, 2019.

Strens, M. J. A. A bayesian framework for reinforcement learning. In *ICML*, pp. 943–950. Morgan Kaufmann, 2000.

Stulp, F. and Sigaud, O. Path integral policy improvement with covariance matrix adaptation. In *ICML*. icml.cc / Omnipress, 2012.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *NIPS*, pp. 1038–1044. MIT Press, 1995.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pp. 1057–1063. The MIT Press, 1999.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. #exploration: A study of count-based exploration for deep reinforcement learning. In *NIPS*, pp. 2753–2762, 2017.

Thrun, S. The role of exploration in learning control. In *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, Kentucky, January 1992.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS*, pp. 5026–5033. IEEE, 2012.

Uhlenbeck, G. E. and Ornstein, L. S. On the Theory of the Brownian Motion. *Physical Review*, 36(5):823–841, September 1930.

van Hoof, H., Tanneberg, D., and Peters, J. Generalized exploration in policy search. *Mach. Learn.*, 106(9-10): 1705–1724, 2017.

Wawrzynski, P. Control policy with autocorrelated noise in reinforcement learning for robotics. *International Journal of Machine Learning and Computing*, 5:91–95, 04 2015. doi: 10.7763/IJMLC.2015.V5.489.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8:229–256, 1992.

Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, pp. 1433–1438. AAAI Press, 2008.