Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)

Autoregressive Policies for Continuous Control Deep Reinforcement Learning

Dmytro Korenkevych , **A. Rupam Mahmood** , **Gautham Vasan** and **James Bergstra**
Kindred AI
{dmytro.korenkevych, rupam, gautham.vasan, james}@kindred.ai

Abstract

Reinforcement learning algorithms rely on exploration to discover new behaviors, which is typically achieved by following a stochastic policy. In continuous control tasks, policies with a Gaussian distribution have been widely adopted. Gaussian exploration however does not result in smooth trajectories that generally correspond to safe and rewarding behaviors in practical tasks. In addition, Gaussian policies do not result in an effective exploration of an environment and become increasingly inefficient as the action rate increases. This contributes to a low sample efficiency often observed in learning continuous control tasks. We introduce a family of stationary autoregressive (AR) stochastic processes to facilitate exploration in continuous control domains. We show that proposed processes possess two desirable features: subsequent process observations are temporally coherent with continuously adjustable degree of coherence, and the process stationary distribution is standard normal. We derive an autoregressive policy (ARP) that implements such processes maintaining the standard agent-environment interface. We show how ARPs can be easily used with the existing off-the-shelf learning algorithms. Empirically we demonstrate that using ARPs results in improved exploration and sample efficiency in both simulated and real world domains, and, furthermore, provides smooth exploration trajectories that enable safe operation of robotic hardware.

1 Introduction

Reinforcement Learning (RL) is a promising approach to solving complex real world tasks with physical robots, supported by recent successes [Andrychowicz *et al.*, 2018; Kalashnikov *et al.*, 2018; Haarnoja *et al.*, 2018]. Exploration is an integral part of RL responsible for discovery of new behaviors. It is typically achieved by executing a stochastic behavior policy [Sutton and Barto, 2018]. In continuous control domain, for instance, policies with a parametrized Gaussian distribution have been commonly used [Schulman *et al.*, 2015; Mnih *et al.*, 2016; Schulman *et al.*, 2017; Haarnoja *et al.*, 2018]. The samples from such policies are temporally coherent only through the distribution mean. In most environments this coherence is not sufficient to provide consistent and effective exploration. In early stages of learning in particular, with randomly initialized policy parameters, exploration essentially relies on a white noise process around zero mean. In environments where actions represent low-level motion control, e.g. velocity or torque, such exploration rarely produces a consistent motion that could lead to discovery of rewarding behaviors. This contributes to low sample efficiency of learning algorithms [Wawrzynski, 2015; Hoof *et al.*, 2017; Plappert *et al.*, 2018]. For real world robotic applications a short reaction time is often desirable, however, as action rate increases, a white noise exploration model becomes even less viable, effectively locking the robot in place [Plappert *et al.*, 2018]. In addition, temporally incoherent exploration results in a jerky movement on physical robots leading to hardware damage and safety concerns [Peters and Schaal, 2007].

In this work we observe that the parameters of policy distribution typically exhibit high temporal coherence, particularly at higher action rates. Specifically, in the case of Gaussian policy distribution, the action can be represented as a sum of deterministic parametrized mean and a scaled white noise component. The mean part typically changes smoothly between subsequent states. It is the white noise part that results in inconsistent exploration behavior. We propose to replace the white noise component with a stationary autoregressive Gaussian process that has stationary standard normal distribution, while exhibiting temporal coherence between subsequent observations. We derive a general form of these processes of an arbitrary order and show how the degree of temporal coherence can be continuously adjusted with a scalar parameter. We demonstrate an advantage of higher orders processes compared to the first order ones used in prior work. Further, we propose an agent's policy structure that directly implements the autoregessive computation in such processes. Temporal action smoothing mechanism therefore is not hidden from the agent but is made explicit through its policy function. In order to achieve this, we require a fixed length history of past states and actions. However, the set of resulting history-dependent policies contains the set of Markov deterministic policies, and those contain the optimal policies in many tasks [Puterman, 2014, Section 4.4]. We find that,

in practical applications, *the search* for such optimal policies can be more efficient and safe in a space of history-dependent stochastic policies with special structure, compared to conventional search in a space of Markov stochastic policies.

Empirically we show that proposed autoregressive policies can be used with off-the-shelf learning algorithms and result in superior exploration and learning in sparse reward tasks compared to conventional Gaussian policies, while achieving similar or slightly better performance in tasks with dense reward. We also show that the drop in learning performance due to increasing action rate can be greatly mitigated by increasing the degree of temporal coherence in the underlying autoregressive process. In the real world robotic experiments we demonstrate that the autoregressive policies result in a smoother and safer movement. [1]

2 Related Work

The problem of exploration in Reinforcement Learning has been studied extensively. One approach has been to modify the environment by changing its reward function to make it easier for an agent to obtain any rewards or to encourage the agent to visit new environment states. This approach includes work on reward shaping [Ng *et al.*, 1999] and auxiliary reward components such as curiosity [Oudeyer *et al.*, 2007; Pathak *et al.*, 2017; Burda *et al.*, 2018]. Note that regardless of chosen reward function temporally consistent behavior would still be beneficial in most tasks as it would discover rewarding behaviors more efficiently. A randomly initialized agent is unaware of the reward function and for example will not exhibit curiosity until after some amount of learning, which already requires visiting new states and discovering rewarding behaviors in the first place.

A second approach, particularly common in practical robotic applications, has been to directly enforce temporal coherence between subsequent motion commands. In the most simple case a low-pass filter is applied, e.g. the agent actions are exponentially averaged over the fixed or infinite length window [Benbrahim and Franklin, 1997]. A similar alternative is to employ a derivative control where agent's actions represent higher order derivatives of the control signal [Mahmood *et al.*, 2018a]. Both of these approaches correspond to acting in a modified MDP with different state and action spaces and result in a less direct connection between agent's action and its consequence in the environment, which can make the learning problem harder. They also make the process less observable unless the agent has access to the history of past actions used in smoothing and to the structure of a smoothing mechanism itself, which is typically not the case. As in the case with modified reward function, the optimal policies in the new MDP generally may not correspond to the optimal policies in the original MDP.

A third approach has been to learn parameters of predefined parametrized controllers, such as motor primitives, instead of learning control directly in the actuation space [Peters and Schaal, 2008]. This approach is attractive, as it allows to ensure safe robot behavior and often results in an easier learning problem [Hoof *et al.*, 2017]. However, it requires expert knowledge to define appropriate class of controllers and limits possible policies to those, representable within the selected class. In complex tasks [Andrychowicz *et al.*, 2018; Kalashnikov *et al.*, 2018] it may be non-trivial to design a sufficiently rich primitives set.

Several studies have considered applying exploration noise to policy distribution parameters such as network weights and hidden units activations. Plappert *et al.* [2017] applied Gaussian exploration noise to policy parameters at the beginning of each episode, demonstrating a more coherent and efficient exploration behavior compared to only adding Gaussian noise to the action itself. Fortunato *et al.* [2017] similarly applied independent Gaussian noise to policy parameters, where the scale of the noise was also learned via gradient descent. Both of these works demonstrated improved learning performance compared to baseline Gaussian action space exploration, in particular in tasks with sparse rewards. Our approach is fully complimentary to auxiliary rewards and parametric noise ideas, as both still rely on exploration noise in the action space in addition to other noise sources and can benefit from consistent and temporally smooth exploration trajectories provided by our method.

In the context of continuous control deep RL our work is most closely related to the use of temporally coherent Gaussian noise during exploration. Wawrzynski [2015] used moving average process for exploration where temporal smoothness of exploration trajectories was determined by the integer size of an averaging window. They showed that learning with such process results in a similar final performance as with standard Gaussian exploration, while providing smoother behavior suitable for physical hardware applications. Hoof *et al.* [2017] proposed a stationary first order AR exploration process in parameters space. Lillicrap *et al.* [2015] and Tallec *et al.* [2019] used Ornstein–Uhlenbeck (OU) process for exploration in off-policy learning. The latter work showed that adjusting process parameters according to the time step duration helps to maintain exploration performance at higher action rates. It can be shown that in a discrete time form OU process is a first order Gaussian AR process, which makes it a particular case of our model. AR processes derived in this work generalize the processes used in these studies, providing a wider space of possible exploration trajectories. In addition, the current work proposes policy structure that directly implements autoregressive computation, in contrast to the above studies, where the agent was unaware of the noise structure. Due to this explicit policy formulation, autoregressive exploration can be used in both, on-policy and off-policy learning.

Autoregressive architectures have been proposed in the context of high-dimensional discrete or discretized continuous action spaces [Metz *et al.*, 2017; Vinyals *et al.*, 2017] with regression defined over action components. The objective of such architectures was to reduce dimensionality of the action space. In contrast, we draw on autoregressive stochastic processes literature, and define regression over time steps directly in a multidimensional continuous action space with the objective of enforcing temporally coherent behavior.

[1] See accompanying video at https://youtu.be/NCpyXBNqNmw

3 Background

3.1 Reinforcement Learning

Reinforcement Learning (RL) framework [Sutton and Barto, 2018] describes an agent interacting with an environment at discrete time steps. At each step t the agent receives the environment state $s_t \in \mathcal{S}$ and a scalar reward signal $r_t \in R$. The agent selects an action $a_t \in \mathcal{A}$ according to a policy defined by a probability distribution $\pi(a|s) \coloneqq P\left\{a_t = a|s_t = s\right\}$. At the next time step $t+1$ in part due to the agent's action, the environment transitions to a new state s_{t+1} and produces a new reward r_{t+1} according to a transition probability distribution $p(s', r|s, a) \coloneqq \Pr\left\{s_{t+1} = s', r_{t+1} = r|s_t = s, a_t = a\right\}$. The objective of the agent is to find a policy that maximizes the expected return defined as the future accumulated rewards $G_t \coloneqq \sum_{k=t}^{\infty} \gamma^{k-t} r_{k+1}$, where $\gamma \in [0, 1]$ is a discount factor. In practice, the agent observes the environment's state partially through a real-valued observation vector o_t.

3.2 Autoregressive Processes

An autoregressive process of order $p \in \mathbb{N}$ (AR-p) is defined as

$$X_t \quad = \sum_{k=1}^{p} \phi_k X_{t-k} + Z_t, \tag{1}$$

where $\phi_k \in R,\ k = 1, \ldots, p$ are real coefficients, and Z_t is a white noise with zero mean and finite variance, $Z_t \sim \text{WN}(0, \sigma_Z^2),\ \sigma_Z^2 < \infty$.

An autoregressive process $\{X_t\}$ is called weakly *stationary*, if its mean function $\mu_X(t) = \mathbb{E}[X_t]$ is independent of t and its covariance function $\gamma_X(t+h, t) = \text{cov}(X_{t+h}, X_t)$ is independent of t for each h. In the future we will use the term stationary implying this definition. The process (1) is stationary if the roots $G_i, i = 1, \ldots, p$ (possibly complex) of its *characteristic polynomial*

$$P(z) \coloneqq z^p - \sum_{i=1}^{p} \phi_i z^{p-i}$$

lie within a unit circle, e.g. $|G_i| < 1, i = 1, \ldots, p$ (see e.g. [Brockwell *et al.*, 2002, Section 3.1]).

An autocovariance function γ_k is defined as $\gamma_k = \text{cov}(X_t, X_{t-k}),\ k = 0, \pm 1, \pm 2, \ldots$. From definition, $\gamma_0 = \text{var}(X_t) = \sigma_X^2$. For a stationary AR-$p$ process a linear system of Yule-Walker equations holds:

$$\begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_p \end{bmatrix} = \begin{bmatrix} \gamma_0 & \gamma_1 & \cdots & \gamma_{p-1} \\ \gamma_1 & \gamma_0 & \cdots & \gamma_{p-2} \\ \gamma_2 & \gamma_1 & \cdots & \gamma_{p-3} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{p-1} & \gamma_{p-2} & \cdots & \gamma_0 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_p \end{bmatrix} \tag{2}$$

and

$$\gamma_0 = \sum_{i=1}^{p} \phi_i \gamma_i + \sigma_Z^2.$$

The system (2) has a unique solution with respect to the variables $\{\gamma_k\},\ k = 0, \ldots, p$.

4 Stationary Autoregressive Gaussian Processes

In this section we derive a family of stationary AR-p Gaussian processes for any $p \in \mathbb{N}$, such that $X_t \sim \mathcal{N}(0, 1)\ \forall t$, meaning X_t has a marginal standard normal distribution at each t. We also show how the degree of temporal smoothness of trajectories formed by subsequent observations of such processes can be continuously tuned with a scalar parameter.

Proposition 4.1. *For any $p \in \mathbb{N}$ and for any $\alpha_k \in [0, 1)$, $k = 1, \ldots, p$ consider a set of coefficients*

$$\{\tilde{\phi}_k\}_{k=1}^{p} = (-1)^{k+1} \sum_{1 \le i_1 < i_2 \cdots < i_k \le p} \alpha_{i_1} \alpha_{i_2} \ldots \alpha_{i_k}. \tag{3}$$

The Yule-Walker system (2) with coefficients $\{\tilde{\phi}_k\}$ has a unique solution with respect to $(\tilde{\gamma}_0, \tilde{\gamma}_1, \ldots, \tilde{\gamma}_p, \tilde{\sigma}_Z^2)$, such that $\tilde{\gamma}_0 = 1$ and $\tilde{\sigma}_Z^2 > 0$. Furthermore, the autoregressive process

$$\begin{aligned} X_t &= \sum_{k=1}^{p} \tilde{\phi}_k X_{t-k} + Z_t \\ Z_t &\sim \mathcal{N}(0, \tilde{\sigma}_Z^2), \end{aligned} \tag{4}$$

is a stationary Gaussian process with zero mean and unit variance, meaning $X_t \sim \mathcal{N}(0, 1)\ \forall t$.

Proof. The proof follows from the observation that $\{\tilde{\phi}_k\}$ are coefficients of a polynomial $P(z) = (z - \alpha_1)(z - \alpha_2) \ldots (z - \alpha_p)$ with roots $\{\alpha_k\}$ that all lie within a unit circle. Since $P(z)$ is a characteristic polynomial of a process (4), the process is stationary. The existence of a unique solution to the system (2) with $\tilde{\gamma}_0 = 1,\ \tilde{\sigma}_Z^2 > 0$ follows from the observation that for any $\tilde{\sigma}_Z^2 > 0$ the system (2) has a unique solution with respect to $(\tilde{\gamma}_0, \tilde{\gamma}_1, \ldots, \tilde{\gamma}_p)$, while it is homogeneous with respect to $(\tilde{\gamma}_0, \tilde{\gamma}_1, \ldots, \tilde{\gamma}_p, \tilde{\sigma}_Z^2)$. The result then follows from observing that $\tilde{\gamma}_0 > 0$, and scaling the solution $(\tilde{\gamma}_0, \tilde{\gamma}_1, \ldots, \tilde{\gamma}_p, \tilde{\sigma}_Z^2)$ by $1/\tilde{\gamma}_0$. A complete proof can be found in Supplementary Materials A available at https://bit.ly/2BQqWMx. □

Corollary 4.1.1. *For any $p \in \mathbb{N}$ and for any $\alpha \in [0, 1)$ let $\{X_t\}$ be the autoregressive process:*

$$\begin{aligned} X_t &= \sum_{k=1}^{p} \tilde{\phi}_k X_{t-k} + Z_t \\ \tilde{\phi}_k &= (-1)^{k+1} \binom{p}{k} \alpha^k \\ Z_t &\sim \mathcal{N}(0, \tilde{\sigma}_Z^2), \end{aligned} \tag{5}$$

where $\binom{p}{k} = \frac{p!}{k!(p-k)!}$ is a binomial coefficient, $\tilde{\sigma}_Z^2$ is a solution to the system (2) with $\{\phi_k = \tilde{\phi}_k\}_{k=1}^{p}$ and $\gamma_0 = 1$. Then $X_t \sim \mathcal{N}(0, 1)\ \forall t$.

Proof. Set $\alpha_k = \alpha\ \forall k$ in (3) and apply Proposition 4.1. □

An example of a third order process AR-3 defined by (4) is given in Supplementary Materials B.

Proposition 4.1 allows to formulate stationary autoregressive processes of an arbitrary order p for arbitrary values

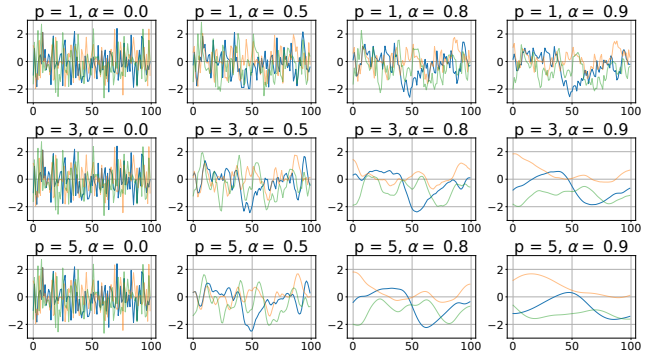

Figure 1: Realizations of processes (5) for different p and α and the same set of 3 random seeds.

$\alpha_k \in [0, 1), k = 1, \dots, p$, such that the marginal distributions of realizations of these processes are standard normal at each time step. This gives us great flexibility and power in defining properties of these processes, such as the degree of temporal coherence between process realizations at various time lags. If we were to use these processes as a source of exploration behavior in RL algorithms, this flexibility would translate into a flexibility in defining the shape and smoothness of exploration trajectories. Note, that the process (4) trivially generalizes to a vector form by defining Z_t a multivariate white noise with a diagonal covariance.

Autoregressive processes in the general form (4) possess a number of interesting properties that can be utilized in reinforcement learning. However, for the purposes of the discussion in the following sections, from now on we will consider a simpler subfamily of processes, defined by (5). Notice, that $\alpha = 0$ results in $\tilde{\phi}_k = 0 \; \forall k$, and X_t becomes a white Gaussian noise. On the other hand, $\alpha \to 1$ results in $\sigma_Z^2 \to 0$, and X_t becomes a constant function. Therefore, tuning a single scalar parameter α from 0 to 1 continuously adjusts temporal smoothness of X_t ranging from white noise to a constant function. Figure 1 shows realizations of such processes at different values of p and α. The realizations are initialized from the same set of 3 random seeds for each p, α pair.

5 Autoregressive Policies

In continuous control RL a policy is often defined as a parametrized diagonal Gaussian distribution:

$$p_\theta(a_t|s_t) = \mathcal{N}(\mu_\theta(s_t), \sigma_\theta^2(s_t) \cdot I), \tag{6}$$

where s_t is a state at time t, $\mu_\theta(s_t)$ and $\sigma_\theta^2(s_t)$ are vectors parametrized by deep neural networks. The actions, sampled from such distribution, can be represented as $a_t = \mu_\theta(s_t) + \sigma_\theta(s_t)\varepsilon_t$, where $\varepsilon_t \sim \mathcal{N}(0, I)$ is a white Gaussian noise. We propose to replace ε_t with observations of an AR-p process $\{X_t\}$ defined by (5) for some $p \in N$ and $\alpha \in [0, 1)$:

$$a_t = \mu_\theta(s_t) + \sigma_\theta(s_t)X_t. \tag{7}$$

Both ε_t and X_t follow marginal standard normal distribution at each step t, therefore such substitution does not change the network output to noise ratio in sampled actions, however for $\alpha > 0$ the sequence $\{X_t\}$ possesses temporal coherence and can provide a more consistent exploration behavior.

We would like to build an agent that implements stochastic policy with samples, defined by (7). From definition (5) of the process $\{X_t\}$, (7) can be expanded as

$$\begin{aligned} a_t &= \mu_\theta(s_t) + \sigma_\theta(s_t) \sum_{k=1}^{p} \tilde{\phi}_k X_{t-k} + \sigma_\theta(s_t)\tilde{\sigma}_Z\varepsilon_t, \\ \varepsilon_t &\sim \mathcal{N}(0, I). \end{aligned} \tag{8}$$

From (7) also, $X_t = (a_t - \mu_\theta(s_t))/\sigma_\theta(s_t) \; \forall t$, hence (8) can be rewritten as

$$\begin{aligned} a_t = \mu_\theta(s_t) &+ \sigma_\theta(s_t) \sum_{k=1}^{p} \tilde{\phi}_k \frac{a_{t-k} - \mu_\theta(s_{t-k})}{\sigma_\theta(s_{t-k})} + \\ &+ \sigma_\theta(s_t)\tilde{\sigma}_Z\varepsilon_t, \\ \varepsilon_t &\sim \mathcal{N}(0, I). \end{aligned} \tag{9}$$

Denote $f_{\theta,t} = \sum_{k=1}^{p} \tilde{\phi}_k \frac{a_{t-k} - \mu_\theta(s_{t-k})}{\sigma_\theta(s_{t-k})}$ the auto-regressive "history" term in (9). Note that $f_{\theta,t}$ is a function of past p states and actions, $f_{\theta,t} = f(\{s_{t-k}, a_{t-k}\}_{k=1}^{p}, \theta)$. Then a_t follows the distribution:

$$\begin{aligned} &a_t \sim \mathcal{N}(\mu_\theta(s_t) + \sigma_\theta(s_t)f_{\theta,t}, \sigma_\theta^2(s_t)\tilde{\sigma}_Z^2 \cdot I), \\ &\tilde{\phi}_k, \; \tilde{\sigma}_Z^2 \text{ are defined by (5).} \end{aligned} \tag{10}$$

In order to implement such action distribution, we need to define a *history-dependent* policy $\pi(a_t|s_t, h_t^p)$, where $h_t^p = (s_{t-p}, a_{t-p}, \dots, s_{t-1}, a_{t-1})$ is a history of past p states and actions. In general, history-dependent policies do not induce Markov stochastic processes, even if the environment transition probabilities are Markovian [Puterman, 2014, Section 2.1.6]. However when the dependence is only on a history of a fixed size, such policy induces a Markov stochastic process in an extended state space, where states are defined as pairs (h_t^p, s_t). In order to be able to lean on existing theoretical results, such as Policy Gradient Theorem [Sutton *et al.*, 2000], and to use existing learning algorithms, we will talk about learning policies in this extended MDP.

More formally, let $M = (S, A, P(\cdot|a, s), r(s, a))$ be a given MDP with S and A denoting state and action sets, and P and r denoting transition probability and reward functions respectively. Let p be an arbitrary integer number. We define a modified MDP $\tilde{M}^p = (\tilde{S}, \tilde{A}, \tilde{P}(\cdot|a, \tilde{s}), \tilde{r}(\tilde{s}, a))$ with the elements $\tilde{S} = \{S \times A\}^p \times S$, where $\{C\}^p$ denotes Cartesian product of set C with itself p times, $\tilde{A} = A$, and $\tilde{P}$ and $\tilde{r}$ defined as follows:

$$\begin{aligned} &\forall \tilde{s}, \tilde{s}' \in \tilde{S}: \\ &\quad \tilde{s} = (s_1, a_1, \dots, s_p, a_p, s_{p+1}), a_k \in A, s_k \in S \; \forall k \\ &\quad \tilde{s}' = (s'_1, a'_1, \dots, s'_p, a'_p, s'_{p+1}), a'_k \in A, s'_k \in S \; \forall k \\ &\tilde{P}(\tilde{s}'|a, \tilde{s}) = \begin{cases} P(s'_{p+1}|a, s_{p+1}), & \text{if } s'_k = s_{k+1}, k \le p \\ & \quad a'_k = a_{k+1}, \; k < p \\ & \quad a'_p = a \\ 0 & \text{otherwise} \end{cases} \\ &\tilde{r}(\tilde{s}, a) = r(s_{p+1}, a) \end{aligned}$$

In other words, transitions in a modified MDP $\tilde{M}^p$ correspond to transitions in the original MDP M with states in $\tilde{M}^p$ containing also the history of past p states and actions in M. The interaction between the agent and the environment, induced by $\tilde{M}^p$, occurs in the following way. At each time t the agent is presented with the current state $\tilde{s}_t = (s_{t-p}, a_{t-p}, \dots, s_{t-1}, a_{t-1}, s_t)$. Based on this state and its policy, it chooses an action a_t from the set A and sends it to the environment. Internally, the environment propagates the action a_t to the original MDP M, currently in state s_t, which responds with a reward value r_{t+1} and transitions to a new state s_{t+1}. At this moment, the MDP $\tilde{M}^p$ transitions to a new state $\tilde{s}_{t+1} = (s_{t-p+1}, a_{t-p+1}, \dots, s_t, a_t, s_{t+1})$ and presents it to the agent together with the reward r_{t+1}. Let s_0 be an element of the set of the initial states of M. A corresponding initial state of $\tilde{M}^p$ is defined as $\tilde{s}_0 = (\underbrace{s_0, a_0, \dots, s_0, a_0}_{\text{p repetitions}}, s_0)$, where a_0 is any element of an action set A, for example zero vector in the case of continuous space. The particular choice of a_0 is immaterial, since it does not affect future transitions and rewards (more details in Supplementary Materials F).

We define an **autoregressive policy** (ARP) over $\tilde{M}^p$ as:

$$\begin{aligned} \forall \tilde{s}_t &= (s_{t-p}, a_{t-p}, \dots, s_{t-1}, a_{t-1}, s_t) : \\ \pi_\theta(a_t|\tilde{s}_t) &= \mathcal{N}(\mu_\theta(s_t) + \sigma_\theta(s_t) f_\theta(\tilde{s}_t),\ \sigma^2_\theta(s_t)\tilde{\sigma}^2_Z I), \\ f_\theta(\tilde{s}_t) &= \sum_{k=1}^{p} \tilde{\phi}_k \frac{a_{t-k} - \mu_\theta(s_{t-k})}{\sigma_\theta(s_{t-k})}, \\ \tilde{\phi}_k,\ \tilde{\sigma}^2_Z &\text{ are defined by (5),} \end{aligned} \tag{11}$$

where $\mu_\theta(\cdot)$ and $\sigma_\theta(\cdot)$ are parametrized function approximations, such as deep neural networks. For notation brevity, we omitted dependence of policy π_θ on $\{\tilde{\phi}_k\}$ and $\tilde{\sigma}_Z$, since these values are constant once the autoregressive model $\{X_t\}$ is selected. In this parametrization, $\mu_\theta(s_{t-k}),\ k = 0, \dots, p$ should be thought of as the same parametrized function $\mu_\theta(\cdot)$ applied to different parts of the state vector $\tilde{s}_t$, therefore each occurence of $\mu_\theta(\cdot)$ in (11) contributes to the gradient w.r.t. parameters θ. Similarly, each occurence of $\sigma_\theta(\cdot)$ contributes to the gradient w.r.t. θ. Note, that including history of states and actions does not affect the dimensionality of the input to the function approximations, as both $\mu_\theta(\cdot)$ and $\sigma_\theta(\cdot)$ accept only states from the original space as inputs.

The history-dependent policy (11) results in the desired action distribution (10) in the original MDP M, at the same time with respect to $\tilde{M}^p$ it is just a particular case of a Gaussian policy (6). Formally, we will perform learning in $\tilde{M}^p$, where π_θ is Markov, and therefore all the related theoretical results apply, and any off-the-shelf learning algorithm, applicable to policies of type (6), can be used. In particular, the value function in e.g. actor-critic architectures is learned with usual methods. Empirically we found that conditioning value function only on a current state s_t from the original MDP instead of an entire vector $\tilde{s}_t$ gives more stable learning performance. It also helps to maintain the critic network size invariant to the AR process order p.

By design, for each sample path $(\tilde{s}_0, a_0, \tilde{s}_1, a_1, \dots)$ in $\tilde{M}^p$ there is a corresponding sample path $(s_0, a_0, s_1, a_1, \dots)$ in M with identical rewards. Therefore, improving the policy and the obtained rewards in $\tilde{M}^p$ results in identical improvement of a corresponding history-dependent policy in M. Notice also, that if $\sigma_\theta(s_t) \to 0$ in (11), then π_θ reduces to a Markov deterministic policy $a_t = \mu_\theta(s_t)$ in M. Therefore, the optimal policy in the set of ARPs defined by (11) is at least as good, as the best deterministic policy in the set of policies $a_t = \mu_\theta(s_t)$. This is in contrast with action averaging approaches, where temporal smoothing is typically imposed on the entire action vector and not just on the exploration component, limiting the space of possible deterministic policies.

It is important to point out that for any history-dependent policy there exists an equivalent Markov stochastic policy with identical expected returns [Puterman, 2014, Theorem 5.5.1]. For the policy (11), for example, it can be constructed as $\pi^M_\theta(a|s) = \sum_{h^p \in H^p} \pi_\theta(a|s, h^p) p(h^p|s, \pi_\theta)\ \forall (a, s)$, where H^p is a set of all histories of size p. However, $\pi^M_\theta(a|s)$ is a non-trivial function of a state s, unknown to us at the beginning of learning. It is certainly not given by a random initialization of (6), while a random initialization of (11) already provides consistent and smooth behavior. $\pi^M(a|s)$ also cannot be derived analytically from (11), since computing $p(h^p|s, \pi_\theta)$ requires knowledge of environment transition probabilities, which we cannot expect to have for each given task. From these considerations, the particular form of policy parametrization defined by (11) can also be thought of as an additional structure, enforced upon the general class of Markov policies, such as policies defined by (6), restricting possible behaviors to temporally coherent ones.

Although autoregressive term $f_\theta(\tilde{s}_t)$ in (11) is formally a part of the distribution mean, numerically it corresponds to a stationary zero mean random process $F_t = \sum_{k=1}^{p} \tilde{\phi}_k X_{t-k}$, where $\{X_t\}$ is an underlying AR process defined by (5). Therefore, $f_\theta(\tilde{s}_t)$ can be thought of as a part of an action exploration component around the deterministic mean, given by $\mu_\theta(s_t)$. It is this part that ensures a consistent and smooth exploration, as will be demonstrated in the next section.

In principle, one could define π_θ in (11) using arbitrary values of coefficients $\{\tilde{\phi}_k\}$ and $\tilde{\sigma}^2_Z$. The role of particular values of $\{\tilde{\phi}_k\}$ computed according to (5) is to make sure, that the underlying AR process $\{X_t\}$ is stationary and the autoregressive part $f_\theta(\tilde{s}_t)$ does not explode. The role of $\tilde{\sigma}^2_Z$ computed by solving (2) with coefficients $\{\phi_k = \tilde{\phi}_k\}$ and $\gamma_0 = 1$ is to make sure, that the variance of the underlying process $\{X_t\}$ is 1. The total variance around $\mu_\theta(s_t)$ is then conveniently defined by an agent controlled $\sigma_\theta(s_t)$.

Since linear system (2) with coefficients (3) and $\gamma_0 = 1$ has a unique solution according to the Proposition 4.1, its matrix has a full rank, and therefore the system is well-determined and can be solved numerically to an arbitrary precision. In practice we solve it with numpy.linalg.solve function.

6 Experiments

We compared conventional Gaussian policy with ARPs on a set of tasks with both, sparse and dense reward functions, in simulation and the real world. In the following learning experiments we used the Open AI Baselines PPO al-

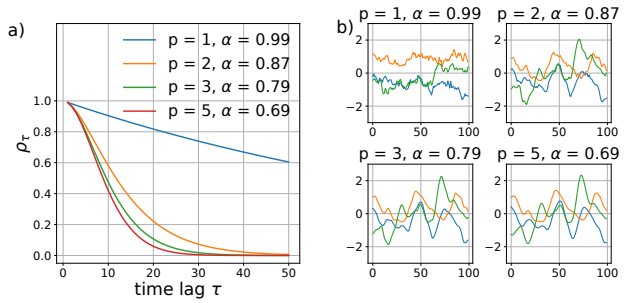

Figure 2: a) Autocorrelation function ρ_τ for autoregressive processes (5) with different orders p but the same value of $\rho_1 = 0.99$. b) Realizations of processes (5) with the same $\rho_1 = 0.99$.

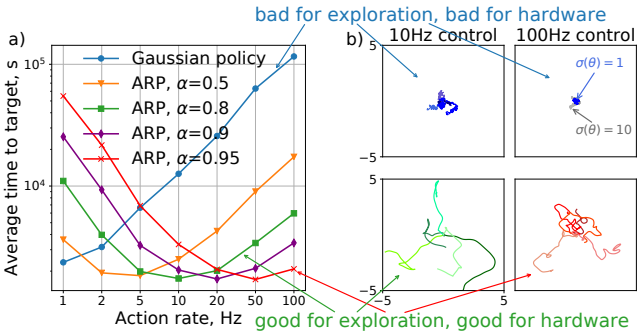

Figure 3: a) Average time to target in Square environment as a function of an action rate for Gaussian policy and ARPs with varied α. b) 10 seconds long exploration trajectories at 10Hz (left column) and 100Hz (right column) action rate using Gaussian policy (top row) and ARPs with $p = 3$ and α values 0.8 and 0.95 (bottom row).

gorithm implementation [Schulman *et al.*, 2017]. The results with Baselines TRPO [Schulman *et al.*, 2015] are provided in Supplementary Materials C available at https://bit.ly/2BQqWMx. For each experiment we used identical algorithm hyper-parameters and neural network structures to parametrize μ_θ, σ_θ and the value networks for both Gaussian and ARP policies. We used the same set of random seeds to initialize neural networks and the same set of random seeds to initialize environments that involve uncertainty. Detailed parameters for each task are included in Supplementary Materials E. We did not perform a hyper-parameter search to optimize for ARP performance, as our primary objective is to demonstrate the advantage of temporally coherent exploration even in the setting, tuned for a standard Gaussian policy. The video of agent behaviors can be found at https://youtu.be/NCpyXBNqNmw. The code to reproduce experiments is available at https://github.com/kindredresearch/arp.

6.1 The Order of an Autoregressive Process

From Figure 1 one can notice that the temporal smoothness of realizations of AR processes (5) empirically increases with both, parameter α and order p. Why do we need higher order processes if we can simply increase α to achieve a higher degree of temporal coherence? To answer this question it is helpful to consider an autocorrelation function (ARF) $\rho_\tau = \text{cov}(X_t, X_{t+\tau})/\text{var}(X_t) = \gamma_\tau/\gamma_0$ of these processes. White Gaussian noise by definition has autocorrelation function equal to zero at any τ other than 0. An autoregressive process with non-zero coefficients generally has non-zero values of autocorrelation function at all τ.

One of the reasons we are interested in autoregressive processes for exploration is that they provide smooth trajectories that do not result in jerky movement and do not damage physical robot hardware. Intuitively, the smoothness of the process realization is defined by a correlation between subsequent observations $\text{corr}(X_t, X_{t+1}) = \rho_1$, which for a given p increases with increasing α. However, given the same value ρ_1, processes of different orders p behave differently. Figure 2a shows ARFs for different processes defined by (5) and their corresponding values of α with the same value of $\rho_1 = 0.99$, while Figure 2b shows realizations of these processes. ARF values at higher orders p decrease much faster with increasing time lag τ compared to the 1st order process, where correlation between past and future observations lingers over long periods of time. As shown on Figure 2b, the 1st order AR process produces nearly a constant function, while the 5th order process exhibits a much more diverse exploratory behavior. Given the same value of correlation between subsequent realizations, higher order autoregressive processes exhibit lower correlation between observations distant in time, resulting in trajectories with better exploration potential. In robotics applications where smoothness of the trajectory can be critical, higher order autoregressive processes may be a preferable choice. Empirically we found that the 3-rd order processes provide sufficiently smooth trajectories while exhibiting a good exploratory behavior, and used $p = 3$ in all our subsequent learning experiments varying only the smoothing parameter α.

6.2 Toy Environment with Sparse Reward

To demonstrate the advantage of temporally consistent exploration, in particular at high action rates, we designed a toy *Square* environment with a 2D continuous state space bounded by a 10x10 square arena. The agent controls a dot through a continuous direct velocity control. The agent is initialized in the middle of the arena at the start of each episode and receives a -1 reward at each time step scaled by time step duration. The target is generated at a random location on a circle of diameter 5 centered at the middle of the arena to make episodes homogenous in difficulty. The episode is over when the agent approaches the target to within a distance of 0.5. The action space is bounded within a two-dimensional $[-1, 1]^2$ interval. The observation vector contains the agent's position, velocity, and the vector difference between agent position and the target position.

To compare exploration efficiency we ran random ARP ($p = 3$) and Gaussian agents with $\mu_\theta(\cdot)$, σ_θ initialized to $\vec{0}$ and $\vec{1}$ respectively for 10 million simulated seconds at different action rates. Figure 3a shows average time to reach the target as a function of an action rate. The results show that the optimal degree of temporal coherence depends on the environment properties, such as action rate. At low control frequency a white Gaussian exploration is more effective than

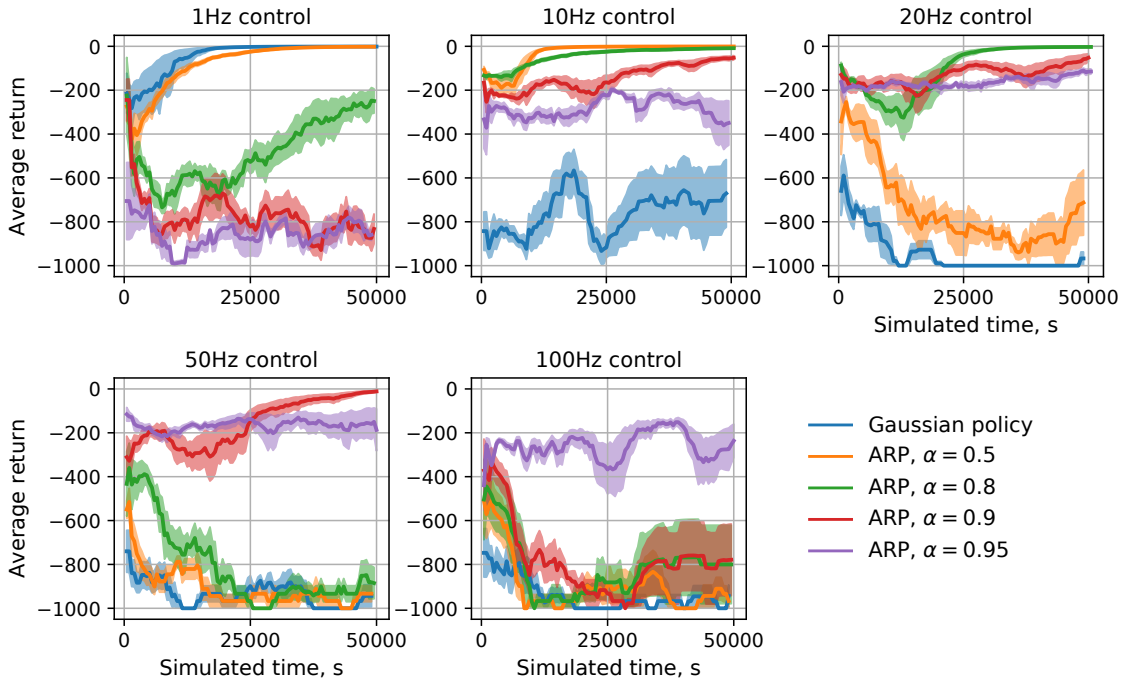

Figure 4: Learning curves in a toy 2D environment with sparse reward. White noise exploration (Gaussian policy) leads to ineffective learning at higher action rates. Temporally smoother processes are capable to learn an effective behavior at wide range of action rates.

ARPs with high α, as in the latter the agent quickly reaches the boundary of the state space and gets stuck there. The efficiency of Gaussian exploration drops dramatically with the increase of action rate. However it is possible to maintain a consistent exploration performance in ARP across wide range of action rates by increasing accordingly the α parameter. This effect is visualized on Figure 3b which shows five 10 seconds long exploration trajectories at 10Hz and 100Hz control for Gaussian and ARP policies. Although run for the same amount of simulated time, Gaussian exploration at 100Hz covers substantially smaller area of state space compared to 10Hz control, while increasing α from 0.8 to 0.95 (the values were chosen empirically) results in ARP trajectories covering similar space at both action rates. Note that the issue with Gaussian policy can not be fixed by simply increasing the variance, as most actions will just be clipped at the $[-1, 1]^2$ boundary, resulting in a similarly poor exploration. Figure 3b top right plot shows exploration trajectories for $\sigma(\theta) = \vec{1}$ (blue) and $\sigma(\theta) = 10\vec{}$ (gray). To the contrary of the common intuition, in bounded action spaces Gaussian exploration with high variance does not produce a diverse state-action visitation.

The advantage of ARPs in exploration translates into an advantage in learning. Figure 4 shows learning curves (averaged over 5 random seeds) on Square environment at different action rates ran for 50,000 seconds of total simulated time with episodes limited to 1000 simulated seconds. Not only ARPs exhibit better learning at most action rates, but the initial random behavior gives much higher returns compared to initial Gaussian agent behaviour. At higher action rates ARPs with higher α produce better results. At the same time at the lowest action rate Gaussian policy (corresponding to ARP with $\alpha = 0$) outperforms policies with higher α values. In a *Square* environment the fast reaction time is not essential and an effective policy can be achieved at low action rates, since the target is stationary and there are no safety hazards. In practical applications however this is rarely the case. The robots often share workspace with other robots and humans and the environment changes dynamically. A fast reaction time of the agent is often necessary to have both safe and effective behaviour. In Supplementary Materials D we provide learning results in a modified *Square* environment with a mobile target where an effective behaviour cannot be achieved at low action rate.

In the formulation of the AR-1 process used in Lillicrap *et al.* [2015] and in Tallec *et al.* [2019], parameter α corresponds to $1 - \kappa dt$, where dt is a time step duration. Hence, in that formulation α naturally approaches 1 as dt approaches zero. Note, that in order to achieve the best performance on each given task, parameter κ still needs to be tuned, just as parameter α needs to be tuned in our formulation. The optimal values of these parameters depend not only on action rate, but also on environment properties, such as a size of a state space relative to the typical size of an agent step. For example, in the defined here *Square* environment increasing the size of the arena and the initial distance to the target k-fold is equivalent to keeping the arena size the same but decreasing time step durartion k-fold. Therefore, given the same value of dt, bigger arena would require a larger value of α or a smaller value of κ parameters respectively.

Notice that ARP, although having better performance than a Gaussian policy, still degrades at higher action rates (e.g.

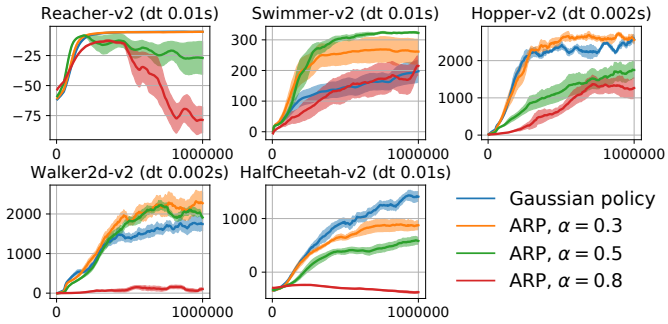

Figure 5: Learning curves in Mujoco-based environments.

$\alpha = 0.5$ at 10Hz vs $\alpha = 0.95$ at 100Hz control). There are two main challenges in learning at high action rates: one being an effective exploration and the other being a problem of credit assignment (i.e. at higher action rates the information about returns needs to be propagated through a larger number of steps and the effect of each individual action is less perceptible to e.g. neural network approximations). Our method addresses the first challenge but not the second one, and therefore its performance still suffers from very high action rates.

6.3 Mujoco Experiments

Figure 5 shows the learning results on standard OpenAI Gym Mujoco environments [Brockman *et al.*, 2016]. These environments have dense rewards, so consistent exploration is less crucial here compared to tasks with sparse rewards. Nevertheless, we found that ARPs perform similarly or slightly better, than a standard Gaussian policy. On a Swimmer-v2 environment ARP resulted in a much better performance compared to Gaussian policy, possibly because in this environment smooth trajectories are highly rewarded. Note that different Mujoco environments are defined with different simulated time step durations (shown on Figure 5), which partially define the best value of α for each task. In addition, as we discussed in the previous subsection, the optimal value of alpha is defined by other aspects of the simulation, such as torque limits, joint space limits, and the reward function. The results suggest that fot the best performance α generally needs to be tuned separately on each given domain.

6.4 Physical Robot Experiments

On a UR5 robotic arm we were able to obtain results similar to those in the toy environment. We designed a sparse reward version of a UR5 Reacher 2D task introduced in [Mahmood *et al.*, 2018b]. In a modified task at each time step the agent receives a -1 reward scaled by a time step duration. The episode is over when the agent reaches the target within a distance of 0.05. In order to provide sufficient time for exploration in a sparse reward setting we doubled the episode time duration to 8 seconds. Figure 6 shows the learning curves for 25Hz and 125Hz control. Each curve is an average across 4 random seeds. The Gaussian policy fails to learn in a 125Hz control setting within a 5 hours time limit, while ARP was able to find an effective policy in 50% of the runs, and the effectiveness at higher α increased at a higher action rate.

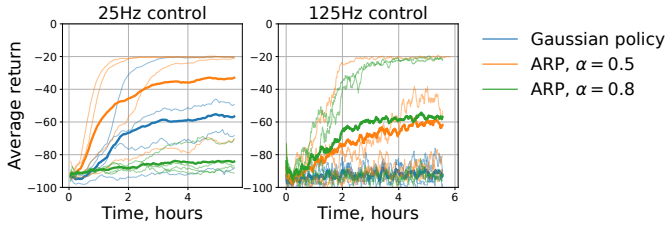

Figure 6: Learning curves on a UR5 Reacher 2D environment with sparse reward at 25Hz and 125Hz velocity control.

7 Conclusions

We introduced autoregressive policies (ARPs) for temporally coherent exploration in continuous control deep reinforcement learning. The policy form is grounded in the theory of stationary autoregressive stochastic processes. We derived a family of stationary Gaussian autoregressive stochastic processes for an arbitrary order p with continuously adjustable degree of temporal coherence between subsequent observations. We derived an agent policy that implements these processes with a standard agent-environment interface. Empirically we showed that ARPs result in a superior exploration and learning in sparse reward tasks and perform on par or better compared to standard Gaussian policies in dense reward tasks. On physical hardware, ARPs result in smooth trajectories that are safer to execute compared to the trajectories provided by conventional Gaussian exploration.

References

[Andrychowicz *et al.*, 2018] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[Benbrahim and Franklin, 1997] Hamid Benbrahim and Judy A Franklin. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, 22(3-4):283–302, 1997.

[Brockman *et al.*, 2016] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[Brockwell *et al.*, 2002] Peter J Brockwell, Richard A Davis, and Matthew V Calder. *Introduction to time series and forecasting*, volume 2. Springer, 2002.

[Burda *et al.*, 2018] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[Fortunato *et al.*, 2017] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[Haarnoja *et al.*, 2018] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie

Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[van Hoof *et al.*, 2017] Herke van Hoof, Daniel Tanneberg, and Jan Peters. Generalized exploration in policy search. *Machine Learning*, 106(9-10):1705–1724, 2017.

[Kalashnikov *et al.*, 2018] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR, 29–31 Oct 2018.

[Lillicrap *et al.*, 2015] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[Mahmood *et al.*, 2018a] A Rupam Mahmood, Dmytro Korenkevych, Brent J Komer, and James Bergstra. Setting up a reinforcement learning task with a real-world robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4635–4640. IEEE, 2018.

[Mahmood *et al.*, 2018b] A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots. In *CoRL*, 2018.

[Metz *et al.*, 2017] Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete sequential prediction of continuous actions for deep rl. *arXiv preprint arXiv:1705.05035*, 2017.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[Ng *et al.*, 1999] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

[Oudeyer *et al.*, 2007] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[Pathak *et al.*, 2017] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

[Peters and Schaal, 2007] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750. ACM, 2007.

[Peters and Schaal, 2008] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.

[Plappert *et al.*, 2017] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

[Plappert *et al.*, 2018] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[Puterman, 2014] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

[Sutton *et al.*, 2000] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[Tallec *et al.*, 2019] Corentin Tallec, Léonard Blier, and Yann Ollivier. Making deep q-learning methods robust to time discretization. *arXiv preprint arXiv:1901.09732*, 2019.

[Vinyals *et al.*, 2017] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[Wawrzynski, 2015] Pawel Wawrzynski. Control policy with autocorrelated noise in reinforcement learning for robotics. *International Journal of Machine Learning and Computing*, 5(2):91, 2015.