Random Actions vs Random Policies: Bootstrapping Model-Based Direct Policy Search

Elias Hanna H.ELIAS@HOTMAIL.FR
Sorbonne Université, CNRS,
Institut des Systèmes Intelligents et de Robotique, ISIR,
F-75005 Paris, France

Alex Coninx CONINX@ISIR.UPMC.FR
Sorbonne Université, CNRS,
Institut des Systèmes Intelligents et de Robotique, ISIR,
F-75005 Paris, France

Stéphane Doncieux DONCIEUX@ISIR.UPMC.FR
Sorbonne Université, CNRS,
Institut des Systèmes Intelligents et de Robotique, ISIR,
F-75005 Paris, France

Abstract

This paper studies the impact of the initial data gathering method on the subsequent learning of a dynamics model. Dynamics models approximate the true transition function of a given task, in order to perform policy search directly on the model rather than on the costly real system. This study aims to determine how to bootstrap a model as efficiently as possible, by comparing initialization methods employed in two different policy search frameworks in the literature. The study focuses on the model performance under the episode-based framework of Evolutionary methods using probabilistic ensembles. Experimental results show that various task-dependant factors can be detrimental to each method, suggesting to explore hybrid approaches.

Keywords: Initialization, Dynamics Model, Behavior Space

1. Introduction

Sparse reward tasks are frequent in robotics and call for data-greedy learning algorithms with strong exploration capabilities (Lehman and Stanley, 2011a; Pugh et al., 2016; Cully and Demiris, 2017; Kim and Doncieux, 2017). They have recently shown promising results on notoriously difficult tasks like grasping as they have discovered grasping movements without the need of demonstrations or gripper specific constraints (Morel et al., 2022). Such learning algorithms are based on direct policy search (Sigaud and Stulp, 2019) and cannot be used directly on the real robot as they need up to millions of policy evaluations. They thus heavily rely on simulations, but simulations are not perfect representations of the real robotic setup. It may create issues when such discrepancies are exploited by the learning algorithm, leading to issues either called reality gap (Jakobi et al., 1995) or simulation bias (An et al., 1988; Atkeson and Schaal, 1997).

To alleviate this problem, several techniques do exist. Some ignore the badly modeled parts of the simulator, potentially missing the most rewarding behaviours (Koos et al., 2012). Sim-to-real approaches, like domain randomization (Tobin et al., 2017) or domain adaptation (Jiang, 2008), are only efficient when the source domain (usually a simulator) dynamics are not too different from the real system dynamics. Model-based approaches do not suffer from those issues (Polydoros and Nalpantidis, 2017), as they directly learn a model of the system from data gathered on the target domain. The robot behaviour is then trained using the model in different fashions.

Policy search consists in defining a parameterized policy and in directly optimizing its parameters while it is controlling a robot (Sigaud and Stulp, 2019). Model-based policy search is a long standing research field made of very diverse methods ranging from methods inspired by model predictive control (MPC) (Camacho and Alba, 2013) to model-based reinforcement learning (RL) (Polydoros and Nalpantidis, 2017). On one side there is step-based policy search methods, mainly resulting from the reinforcement learning framework (Sutton and Barto, 2018), and on the other side, there is episode-based policy search methods, mainly resulting from Bayesian Optimization (Snoek et al., 2012) and from Evolutionary methods (Stanley et al., 2019). The interest of using a model in the RL framework is not new, and first promising results were obtained in PILCO by Deisenroth and Rasmussen (2011). Model-based techniques started by using Gaussian Processes (Deisenroth and Rasmussen, 2011; Gaier et al., 2018), but as problems arose (Huang et al., 2015), more methods started turning to Neural Networks architectures for dynamics modeling (Gal et al., 2016; Nagabandi et al., 2018; Chua et al., 2018; Sharma et al., 2019; Lim et al., 2021).

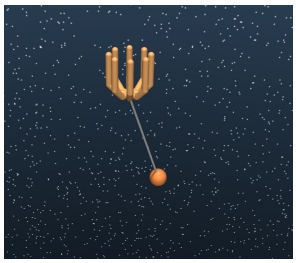
Model-based policy search is thus promising (Gaier et al., 2018; Keller et al., 2020; Lim et al., 2021, 2022), but it requires to learn a model that is accurate enough to predict the behavior of the policy over a complete rollout, *i.e.* with model predictions that are used in closed loop over a given horizon. In this case, error accumulates and makes predictions rapidly diverge from the ground truth, thus misleading any learning algorithm that would rely on it. The bootstrap phase in which data is acquired to train a first model is thus critical. As experiments on the real robot are to be minimized, it is important to define the most efficient approach to randomly gather initial data. Two different approaches have been used, either random actions (Nagabandi et al., 2018; Hafner et al., 2019; Sekar et al., 2020) or random policies (Lim et al., 2021, 2022). We compare their impact on the generated data distribution and on the obtained model quality, so that an active model-based learning approach is bootstrapped as efficiently as possible.

2. Method

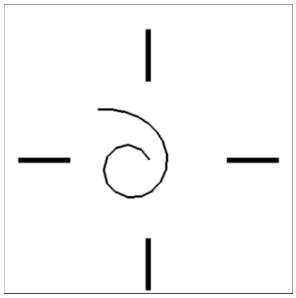
We place ourselves in the Reinforcement Learning framework (Sutton and Barto, 2018). In this context, we define the tuple $< \mathcal{S}, \mathcal{A}, f, \pi >$, where $\mathcal{S}$ is the state-space in which the agent can take states s and that contains all the needed information to determine the agent and environment dynamics, $\mathcal{A}$ is the action-space in which the action can draw actions from. f is the transition function that describes how the agent actions influence the state of the system such that $f : \mathcal{A} \times \mathcal{S} \to \mathcal{S}$, and finally π is a function that maps an action to a specific state $s \in \mathcal{S}$ such that $\pi : \mathcal{S} \to \mathcal{A}$.

The transition function f is supposed to be deterministic: $s_{t+1} = f(s_t, a_t)$. Learning the dynamics model is the same as learning explicitly the system's transition function f, by approximating it with a function $\hat{f}_\theta$ parameterized by a vector θ. The idea behind model based policy search is thus to fit a model $\hat{f}_\theta$ given limited measurements of the true transition function f in the form of N data samples $\mathcal{D} = \{(s_n, a_n), s_{n+1}\}_{n=1}^{N}$. The approximated function $\hat{f}$ is then used recursively to predict a policy behaviour without interacting with the costly real system. Policies are then rolled out on the model on a given horizon H. We call episode a complete policy rollout on the learnt model. In this paper, the model that learns the dynamics of the task is represented by ensembles of probabilistic models (Chua et al., 2018; Hafner et al., 2019; Sekar et al., 2020). We refer the reader to Chua et al. (2018) for more details on ensembling for dynamics modeling.

As we want to study the influence of the model initial data gathering when coupled with evolutionary methods, we further detail the variables of interest we will be looking into for our experiments. Indeed, evolutionary methods, like Novelty Search (Lehman and Stanley, 2011a) or Quality-Diversity approaches (Lehman and Stanley, 2011b; Pugh et al., 2016), make use of what is called a *behavior space*, denoted $\mathcal{B}$, and of an *observer function* $o_\mathcal{B}$ which associates a behavior descriptor $b \in \mathcal{B}$ to a trajectory of states $\tau = \{s_0, s_1, ..., s_T\}$ of length T such that $o_\mathcal{B} : S^T \to \mathcal{B}$. This behavior descriptor b thus characterizes the agent's behavior in a way that is task-aligned and more compact than the whole trajectory τ. When using dynamics models, the part of most interest for evolutionary methods is thus the ability for the dynamics model to predict the agent's behavior b, as the agent's behavior is directly derived from the agent trajectory in the state-space during a rollout.



(a) Ball In Cup Environment



(b) Redundant Arm with walls environment

Figure 1: Benchmark environments

Model-based approaches are iterative and alternate between exploration on the real system and exploration in the model. The focus here is on the initial data gathering that bootstraps the process. Two initialization methods will be compared:

- Random policies: randomly parameterized policies, represented as a fully connected neural network with two hidden layers of size 10, rolled out on the task horizon H and taking as input the system current state s_t. An episode consists of rolling out such a policy on a newly reinitialized environment.

- Random actions: B uniformly drawn actions $a \in \mathcal{A}$, each applied for R step sequentially on the task horizon H such that $B = \frac{H}{R}$. An episode thus consists of rolling out B different actions, for R step each, on a newly reinitialized environment.

We also consider an hybrid approach that splits its budget evenly between the two aforementioned methods, which will be called Random Action Random Policies Hybrid (RARPH).

The only parameter we can influence to collect data is the way the agent acts in its environment. Ideally, we would need independent and identically distributed data to train the dynamics model. Indeed, this is a requirement so that the model is able to generalize well once it is used on unseen parts of the real-system state-action space. The issue is that *i.i.d.* actions do not guarantee *i.i.d.* states as the mapping between the two is a complex function. Random policies have been used to generate initial training data with convincing results although inducing a strong bias as all H samples collected during an episode come from a single trajectory and are thus not independent. In contrast, random actions should provide data closer to *i.i.d.* data, as the collected data samples are only dependant by subsets of size R and should be uniformly distributed in the whole action space $\mathcal{A}$.

The goal of this study is thus to determine if such an induced bias is detrimental or profitable to learning a dynamics model of the task, and to what extent it is task-dependant. We hypothesize that random policies might be advantageous in environments with sparse interactions, as they tend to explore a broader part of the state-space, while random actions should provide a safe initialization method, whatever the environment is. The naive hybrid approach proposed is expected to bring better performance than random actions in environments with sparse interactions, while remaining competitive in others as well.

3. Experiments

3.1 Experimental setups

The first environment considered is the Ball-In-Cup environment (Figure 1a). It consists of a ball hanging below a cup by a string. The cup is controlled in position in the 3D space. The task consists of putting the ball inside the cup. This problem is interesting as it involves sparse interactions, as giving the ball an upward velocity requires to swing the ball. The state-space consists of the 3D relative position and velocity of the ball to the cup, and the considered outcome space is the relative position of the ball to the cup.

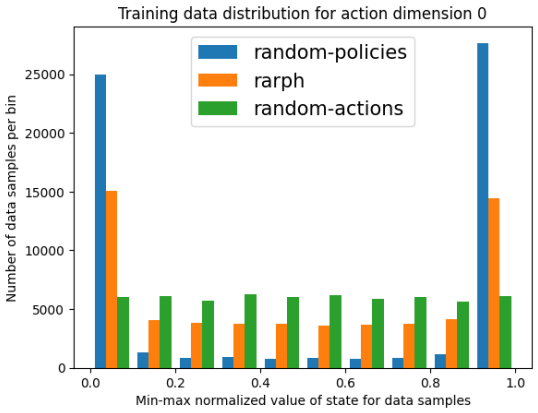
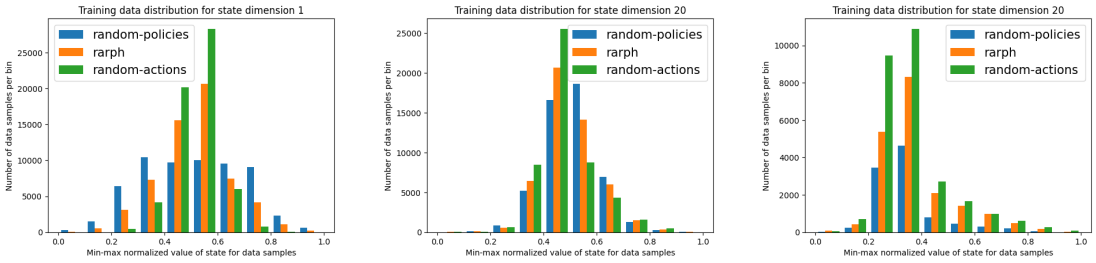


Figure 2: Histogram of training data distribution on test environments for actions

The second environment is the Redundant Arm environment (Figure 1b). It consists of a 20-DOF robotic arm controlled in an environment with or without obstacles (walls). Each articulation of the robotic arm is torque-controlled. The tasks consist in reaching certain parts of the space with the robot end-effector. The problem is hard to gather data from as hitting a wall or self-colliding stops the episode. Having the two scenarios, with and without walls helps us to distinguish the effect of early stopping on the data gathering of the initialization method. The state-space consists in the position of each of the twenty joints and of the x-y position of the end-effector. The considered outcome space is the end-effector position.



(a) Ball In Cup (b) RA No Walls (c) Redundant Arm

Figure 3: Histograms of training data distribution on test environments for RARPH

3.2 Data distribution analysis

As stated before, the key interest on the model learning aspect when coupling a learnt model with an evolutionary method is the capacity the model has to predict correctly the agent behavior in its outcome space. Focusing our analysis on the outcome space dimensions of the state-space is thus relevant for this study, as this will highlight which method explores best the state-space dimensions of interest and theoretically lead to less prediction error when using gathered data to train a dynamics model. In order to do so we estimate the discrepancies between the data distributions using histograms of the data distribution on the dimensions of the outcome space for each initialization method.

Ball In Cup				
Initialization episodes	Prediction horizon	Random policies	Random actions	RARPH
5	1	0.022 ± 0.049	0.054 ± 0.126	0.035 ± 0.061
	20	0.349 ± 0.685	0.601 ± 1.178	0.634 ± 1.052
	300	4.140 ± 2.413	26.56 ± 57.86	7.149 ± 2.591
10	1	0.018 ± 0.043	0.043 ± 0.093	0.030 ± 0.053
	20	0.293 ± 0.554	0.600 ± 1.175	0.521 ± 0.807
	300	3.981 ± 2.210	7.110 ± 12.36	7.003 ± 2.217
15	1	0.018 ± 0.042	0.037 ± 0.076	0.026 ± 0.050
	20	0.284 ± 0.569	0.547 ± 1.043	0.444 ± 0.755
	300	2.880 ± 1.668	4.395 ± 11.75	3.796 ± 1.603
20	1	0.017 ± 0.042	0.033 ± 0.070	0.024 ± 0.047
	20	0.266 ± 0.537	0.520 ± 1.044	0.404 ± 0.668
	300	4.463 ± 3.080	5.467 ± 4.736	4.500 ± 3.984

Table 1: Mean prediction error on Ball In Cup

All the results are gathered on each environment with 20 episodes of each method, repeated 10 times. Only one dimension of the outcome space is shown each time, as it is representative of the other outcome space dimensions. As observed on figure 3a, the data distribution of random policies on Ball In Cup is broader, as the agent observes transitions in a wider portion of the outcome space. On contrary, on the Redundant Arm environment we observe on figure 3c that the random actions data distribution is broader and has many more samples due to early stopping. This is explained by the fact that the random policies take action that are more often at the limit of the action space, as shown in figure 2 (figure obtained on the Ball In Cup environment, but representative of random policies action distribution), which in the case of a torque-controlled robotic arm can lead faster to the joint limits, self-collision or obstacle hitting and thus to episode termination. Removing the walls as shown on Figure 3b indeed qualitatively brings the two data distributions very close one to another. As expected, RARPH data distributions, both for actions and states, is in-between the ones obtained with random policies and random actions, as shown on figure 3.

3.3 Prediction error analysis

The results of this section are obtained by averaging the mean prediction error over 10 repetitions on NS examples trajectories. Model is either used for 1-step predictions or recursively for 20 or H steps, H being the task horizon.

Redundant Arm				
Initialization episodes	Prediction horizon	Random policies	Random actions	RARPH
5	1	0.011 ± 0.029	0.002 ± 0.002	0.002 ± 0.002
	20	0.236 ± 0.673	0.047 ± 0.042	0.046 ± 0.041
	250	16.34 ± 5.855	1.084 ± 0.350	1.071 ± 0.366
10	1	0.002 ± 0.002	0.002 ± 0.002	0.002 ± 0.002
	20	0.041 ± 0.035	0.044 ± 0.037	0.044 ± 0.039
	250	1.002 ± 0.314	1.066 ± 0.342	1.051 ± 0.359
15	1	0.002 ± 0.002	0.002 ± 0.002	0.002 ± 0.002
	20	0.038 ± 0.034	0.041 ± 0.034	0.040 ± 0.035
	250	0.917 ± 0.301	1.050 ± 0.322	1.002 ± 0.328
20	1	0.002 ± 0.002	0.002 ± 0.002	0.002 ± 0.002
	20	0.033 ± 0.032	0.040 ± 0.034	0.039 ± 0.034
	250	0.918 ± 0.287	1.051 ± 0.340	1.019 ± 0.334

Table 2: Mean prediction error on Redundant Arm

Looking at Table 1, we observe that the mean prediction errors and standard deviations are smaller by a factor of around two on all initialization budgets and all prediction horizons between random policies and random actions. Qualitative results are available in appendix A. This validates previous results that seemed to show that random policies are better at gathering initial data for training a dynamics model on the Ball In Cup. Moreover, RARPH initialization performs better than random actions, but does not run up to random policies prediction quality.

Redundant Arm No Walls				
Initialization episodes	Prediction horizon	Random policies	Random actions	RARPH
5	1	0.038 ± 0.286	0.004 ± 0.005	0.004 ± 0.004
	20	1.145 ± 7.778	0.082 ± 0.078	0.075 ± 0.076
	250	20.37 ± 50.97	2.558 ± 0.613	2.445 ± 0.177
10	1	0.003 ± 0.004	0.003 ± 0.004	0.003 ± 0.004
	20	0.063 ± 0.063	0.065 ± 0.069	0.067 ± 0.062
	250	2.864 ± 0.770	2.491 ± 0.153	2.521 ± 0.236
15	1	0.003 ± 0.003	0.003 ± 0.003	0.003 ± 0.004
	20	0.059 ± 0.059	0.056 ± 0.056	0.063 ± 0.068
	250	2.534 ± 0.304	2.532 ± 0.175	2.405 ± 0.144
20	1	0.003 ± 0.003	0.003 ± 0.003	0.003 ± 0.003
	20	0.058 ± 0.059	0.057 ± 0.055	0.060 ± 0.062
	250	2.485 ± 0.291	2.476 ± 0.198	2.458 ± 0.116

Table 3: Mean prediction error on Redundant Arm without walls

On the Redundant Arm environment, we observe that random actions and RARPH have a better prediction performance (Table 2), RARPH even slightly outperforming random actions. This is especially true for small episodes budget, where random policies suffer most from the lack of data due to early stopping. Indeed, when removing the walls, we observe that random policies and random actions perform equally better, except for the smallest budget scenario where self-collision must play an important part in the early stopping of random policies.

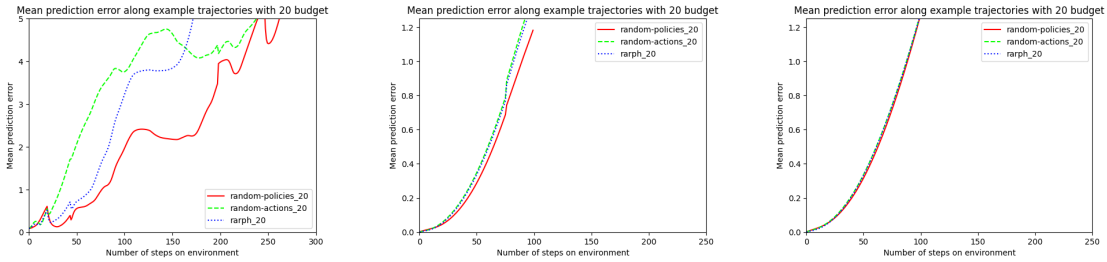
4. Conclusion

In this paper we have shown the importance of comprehending the task dynamics and of selecting the data gathering initialization method beforehand. We compared two initialization methods employed in the state of the art and compared their performance on three different experimental setups. Results show that various factors like sparse interactions or early-stopping criterion can be detrimental to one method or another, suggesting hybrid approaches. The hybrid approach does not yield best results on all tasks, but its performances are more regular than other techniques.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no 869855 (Project 'SoftManBot').

Appendix A.



(a) Ball In Cup (b) Redundant Arm (c) RA no walls

Figure 4: Initialization methods mean prediction error over whole state-space on considered environments on NS archive trajectories on a 20 episode budget

References

Chae H An, Christopher G Atkeson, and John M Hollerbach. *Model-based control of a robot manipulator*. MIT press, 1988.

Christopher G Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, volume 97, pages 12–20, 1997.

Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472. Citeseer, 2011.

Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Data-efficient design exploration through surrogate-assisted illumination. *Evolutionary computation*, 26(3):381–410, 2018.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, page 25, 2016.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

Wenbing Huang, Deli Zhao, Fuchun Sun, Huaping Liu, and Edward Chang. Scalable gaussian process regression using deep neural networks. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer, 1995.

Jing Jiang. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey*, 3(1-12):3, 2008.

Leon Keller, Daniel Tanneberg, Svenja Stark, and Jan Peters. Model-based quality-diversity search for efficient robot learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9675–9680. IEEE, 2020.

Seungsu Kim and Stéphane Doncieux. Learning highly diverse robot throwing movements through quality diversity search. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1177–1178, 2017.

Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Transactions on Evolutionary Computation*, 17(1):122–145, 2012.

Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011a.

Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218, 2011b.

Bryan Lim, Luca Grillotti, Lorenzo Bernasconi, and Antoine Cully. Dynamics-aware quality-diversity for efficient learning of skill repertoires. *arXiv preprint arXiv:2109.08522*, 2021.

Bryan Lim, Alexander Reichenbach, and Antoine Cully. Learning to walk autonomously via reset-free quality-diversity. *arXiv preprint arXiv:2204.03655*, 2022.

Aurélien Morel, Yakumo Kunimoto, Alex Coninx, and Stéphane Doncieux. Automatic acquisition of a repertoire of diverse grasping trajectories through behavior shaping and novelty search. In *IEEE International Conference on Robotics and Automation 2022*, 2022.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

Olivier Sigaud and Freek Stulp. Policy search in continuous action domains: an overview. *Neural Networks*, 113:28–40, 2019.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019.

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

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.