Toward a data efficient neural actor-critic

Matthieu Zimmer, Yann Boniface, Alain Dutech

To cite this version:

Matthieu Zimmer, Yann Boniface, Alain Dutech. Toward a data efficient neural actor-critic. EWRL 2016 - The 13th European Workshop on Reinforcement Learning, Dec 2016, Barcelona, Spain. hal-01413885

HAL Id: hal-01413885
https://hal.archives-ouvertes.fr/hal-01413885
Submitted on 11 Dec 2016

Journal of Machine Learning Research (2016) Submitted ; Published

Toward a data efficient neural actor-critic

Matthieu Zimmer, Yann Boniface, and Alain Dutech {FIRSTNAME.LASTNAME}@LORIA.FR
LORIA, UMR 7503, INRIA, UNIVERSITY OF LORRAINE
NANCY, F-54000, FRANCE

Editor: 13th European Workshop on Reinforcement Learning

Abstract

A new *off-policy*, *offline*, *model-free*, *actor-critic* reinforcement learning algorithm dealing with continuous environments in both states and actions is presented. It addresses discrete time problems where the goal is to maximize the discounted sum of rewards using *stationary* policies. Our algorithm allows to trade-off between *data-efficiency* and *scalability*. The amount of *a priori* knowledge is kept low by: (1) using neural networks to learn both the critic and the actor, (2) not relying on initial trajectories provided by an expert, and (3) not depending on known goal states. Experimental results show better *data-efficiency* than 4 state-of-the-art algorithms on two benchmark environments.

Keywords: Continuous Spaces, Actor-Critic, Neural Networks

1. Introduction

Reinforcement learning (RL) is a framework for solving sequential decision problems, in which an agent interacts with its environment and adapts its policy based on a scalar reward signal (Sutton and Barto, 1998). RL agents can autonomously learn difficult tasks, like playing video games (Mnih et al., 2015). While the basic setting of RL is currently well established, fully continuous environments for both state and action spaces need new algorithms to solve more real-world problems. In many realistic tasks, like robotics, it is time-consuming and costly to produce data. RL agents should thereby exhibit good data-efficiency, *i.e.* exploiting each sample as best as possible, even at the cost of a longer computational time.

The purpose of this work is to design an RL algorithm that: (1) tackles continuous state and action spaces, (2) is data-efficient, and (3) uses neural networks to be as generic as possible with minimal *a priori* knowledge.

Recently, several RL algorithms for fully continuous environments have been developed with neural networks control architectures (Lillicrap et al., 2015; Schulman et al., 2015). However, they were focused in the task performance rather than data-efficiency since they are *model-free* and data were not too costly to produce. Seeking for data-efficiency usually means to use *model-based* algorithms, like Probabilistic Inference for Learning COntrol (PILCO) (Deisenroth and Rasmussen, 2011). However, PILCO lacks scalability (Wahlström et al., 2015) and *model-based* algorithms does not always lead to straightforward improvements when using neural networks (Gu et al., 2016).

In this work, we present an *offline*, *model-free*, *off-policy*, *actor-critic* RL algorithm that allows a trade-off between scalability and data-efficiency. It is based on the *fitted actor-*

critic family (Antos et al., 2008; Zimmer et al., 2016) and benefits from the improvements proposed by Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015).

2. Background

We are interested in RL problems, modeled as *Markov Decision Processes* (MDP) $\langle S, A, T, R\rangle$, where the state space S and the action space A are continuous. The goal is to seek for an *optimal* policy π^* maximizing the expected discounted reward:

$$\pi^* = \underset{\pi}{\arg\max}\ J(\pi) = \underset{\pi}{\arg\max}\ \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \times R(s_t, \pi_t(s_t))\Big], \tag{1}$$

where t denotes a time step and $0 < \gamma < 1$ is the discount factor.

When the state space S is continuous, classical value-function methods like Least-Squares Temporal Difference (LSTD) (Bradtke et al., 1996) rely on an estimation of $Q : S \times A \to \mathbb{R}$, the sequential values of actions in each state:

$$Q^\pi(s,a) = \mathbb{E}_\pi\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \Big| s_t = s, a_t = a\Big], \tag{2}$$

where r_t is the reward obtained at time t from R following π. Being *data-efficient* means to search for the best policy given the collected samples. An example of a neural *data-efficient*, *critic-only* algorithm is Fitted Q Iteration (FQI) (Ernst et al., 2005; Riedmiller, 2005), which updates the Q function several times using the Bellman operator as an approximated version of Value Iteration (Howard, 1960). Instead of iterating over all states and actions, it relies only on the collected samples $(s_t, a_t, r_{t+1}, s_{t+1})$:

$$Q_{k+1} = \underset{Q\in\mathcal{F}}{\arg\min} \sum_{t=1}^{N} \Big[Q(s_t,a_t) - \big(r_{t+1} + \gamma \max_{a'\in A} Q_k(s_{t+1}, a')\big)\Big]^2. \tag{3}$$

When the action space A is continuous, the use of an *actor* (*i.e.* a parametric policy) becomes crucial to overcome the complexity of the argmax search. This often leads to *actor-only* methods like Policy Gradient (Sutton et al., 1999), Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001) or Trust Region Policy Optimization (TRPO) (Schulman et al., 2015). The major drawbacks of *actor-only* methods are the high variability of the cost J (because there is no *critic*) and, for gradient-based methods, the plateau effect and local minima that can lead to poor policies (Grondman et al., 2012; Konda and Tsitsiklis, 1999). On the other hand, *actor-critic* algorithms try to combine both the advantages of previous methods. The critic learns a value function thus reducing the variability of the approximation of J and the actor learns the parametric policy, allowing the use of continuous actions.

We now present three state-of-the-art *actor-critic* algorithms that we will use for comparison in our experiments (from least to most *data-efficient*). Continuous Actor Critic Learning Automaton (CACLA) is a successful *actor-critic* algorithm (Van Hasselt and Wiering, 2007) that uses neural networks for both the critic and the actor. Due to it's online nature and its *on-policy* updates, it cannot achieve good data efficiency (the collected data is used then forgotten). In some environments, CACLA performs better than CMA-ES (Van Hasselt, 2012). Neural Fitted Actor Critic (NFAC) may achieve a better data efficiency than CACLA since it uses FQI updates (Zimmer et al., 2016). However, the data is forgotten after each end of episode because the actor features *on-policy* update. Deep Deterministic Policy Gradient (DDPG) is also an *actor-critic* algorithm (Lillicrap et al., 2015). It accomplishes online updates of the policy and Q function, and it reuse previous samples through its *off-policy* update. Based on Neural Fitted Q with Continuous Actions (Hafner and Riedmiller, 2011), DDPG is more scalable due to online updates, targets networks (Mnih et al., 2015) and batch normalization (Ioffe and Szegedy, 2015). The target networks serve to slow down the weights updates to increase the stability of learning, by soft updating a copy of the policy and the value function.

Recently, two new methods have been proposed to increase the efficiency of some RL algorithms. When the dimensions of action space A are bounded, instead of limiting the output of the neural policy with a last layer (for instance with a hyperbolic tangent) that squashes the gradient obtained from the critic, it is preferable to have an unbounded last layer with an adapted gradient strategy (Hausknecht and Stone, 2016). Retrace(λ) is a new strategy to weight a sample for *off-policy* learning (Munos et al., 2016), it provides low-variance, safe and efficient updates.

3. Algorithm

Our algorithm, that we name Data Efficient Neural Fitted Actor Critic (DENFAC), can be seen as a neural version of a *fitted actor-critic* (FAC) algorithm (Antos et al., 2008). It contains both an approximated version of Value and Policy Iteration (for the critic and the actor respectively).

The critic is updated with a FQI update where the argmax operator is replaced by the policy choice. Moreover, the policy is able to change at each update to approximately fit what would be the argmax.

$$Q_{k+1} = \operatorname*{argmin}_{Q \in \mathcal{F}_c} \sum_{(s_t,a_t,r_{t+1},s_{t+1}) \in \mathcal{D}} c(s_t,a_t)\Big[Q(s_t,a_t) - \Big(r_{t+1} + \gamma Q_k(s_{t+1}, \pi_k(s_{t+1}))\Big)\Big]^2, \quad (4)$$

$$\pi_{k+1} = \operatorname*{argmax}_{\pi \in \mathcal{F}_a} \sum_{s_t \in \mathcal{D}} Q_{k+1}\big(s_t, \pi_k(s_t)\big), \quad (5)$$

where $c(s_t,a_t) = min\big(1, \frac{\pi_{k-1}(a_t|s_t)}{\pi_b(a_t|s_t)}\big)$ is the weight associated to a sample (Munos et al., 2016), and π_b is the policy that gathered the sample. This coupled optimization can be applied multiple times without acquiring new samples.

DENFAC is an *off-line* algorithm, therefore the execution part of one episode consists only of performing the policy choices and collecting the samples $(s_t, a_t, r_{t+1}, s_{t+1})$ that are added to $\mathcal{D}$ (the replay buffer). The *off-line* part is depicted in Algorithm 1. The algorithm is *data-efficient* because it performs a type of FQI. Furthermore, unlike DDPG, it performs updates over the largest set of data given a computational constraint. This might requires too much computational time so the *data-efficiency* vs *scalability* dilemma can be adjusted through the length of $\mathcal{D}$. If $\mathcal{D}$ is big enough to accurately represent the Q function, another meta-parameter of Algorithm 1, *reset_critic* that reset the weight of the critic, can lead to a even better *data-efficiency* by avoiding local minima.

```
Data: $\mathcal{D}$ replay buffer of $N$ samples, $Q_0$ value-function, $\pi_b$ previous policies, $K$
      number of fitted iteration, $G$ number of gradient descent for actor updates,
      inverting_gradient strategy, reset_critic strategy
Result: $\pi_K$ the next policy to play, $Q_K$ the next value function
for $k \leftarrow 1$ to $K$ do
    for $(s_t, a_t, u_t, r_{t+1}, s_{t+1}) \in \mathcal{D}$ do
        $q_{k,t} \leftarrow \begin{cases} r_{t+1}, & \text{if } s_{t+1} \in S^* \\ r_{t+1} + \gamma Q_{k-1}(s_{t+1}, \pi_{k-1}(s_{t+1})), & \text{otherwise} \end{cases}$
    end
    $Q_k \leftarrow$ randomly initialize critic network if reset_critic else $Q_{k-1}$
    Update critic by minimizing the loss :
        $L = \frac{1}{N} \sum_{t=1}^{N} min\big(1, \frac{\pi_{k-1}(a_t|s_t)}{\pi_b(a_t|s_t)}\big) \Big(q_{k,t} - Q_k(s_t, a_t)\Big)^2$
    Randomly initialize actor network $\pi_k$
    Update the actor policy using the batch gradient $G$ times:
        if inverting_gradient then
            $\nabla_a = \nabla_a \cdot \begin{cases} (a_{max} - a)/(a_{max} - a_{min}) & \text{if } \nabla_a < 0 \\ (a - a_{min})/(a_{max} - a_{min}), & \text{otherwise} \end{cases}$
        end
        $\nabla_{\theta^{\pi_k}} \pi_k = \frac{1}{N} \sum_{t=1}^{N} \nabla_a Q(s_t, a)|_{a=\pi_k(s_t)} \nabla_{\theta^{\pi_k}} \pi_k(s_t)$
end
```

Algorithm 1: Data Efficient Neural Fitted Actor Critic (DENFAC)

4. Experimental Setup

An experimental comparison of DENFAC, DDPG, CMA-ES, NFAC and CACLA is done into two environments: Acrobot (Spong, 1995) and Cartpole (Riedmiller et al., 2007).

In Acrobot (double swing-up), the reward function is defined as (1) +1 if the goal is reached (arm straight up), (2) the normalized max height of end effector if 500 steps are reached, and (3) 0 otherwise.

In Cartpole (inverted pendulum), the reward function is defined as (1) 0 when the cart position is between $[-0.05; 0.05]$ and the pole angle between $[-\frac{\pi}{60}, \frac{\pi}{60}]$, (2) $-2 \times (500 - last\_step)$ if it exits at $last\_step$ (pole angle $\notin [-\frac{\pi}{6}, \frac{\pi}{6}]$ or cart position $\notin [-2.4; 2.4]$), and (3) -1 otherwise.

The neural networks use (1) Adam learning algorithm (Kingma and Ba, 2015), (2) the leaky rectified non-linearity (ReLU) (Glorot et al., 2011), and (3) batch normalization (Ioffe

	FAC	DDPG	NFAC	DENFAC
Offline & Batch	×		×	×
Off-policy	×	×		×
Fitted Critic	×		×	×
Actor updated through ∇Q	×	×		×
Reset Networks			×	×
Retrace				×
Batch Normalization		×		×

Figure 1: Properties of the nearest actor-critic algorithms : FAC (Antos et al., 2008), DDPG (Lillicrap et al., 2015) and NFAC (Zimmer et al., 2016).

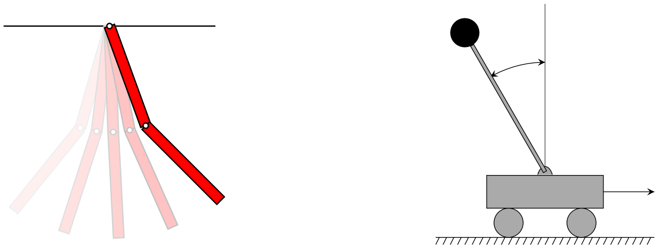

Figure 2: Illustration of Acrobot (left) and Cartpole (right) environments (reproduced from Wikipedia).

and Szegedy, 2015). Critic networks contain 2 hidden layers of 50 and 7 neurons. Actor networks contain only 1 hidden layer of 5 units (Acrobot) or 20 units (Cartpole). The last layer of the critic networks is linear while the actor's one is leaky ReLU. The actor policy is a truncated Gaussian policy between $[-1, 1]$ and $\sigma = 0.5$ with $\gamma = 0.9$ (Acrobot) or $\gamma = 0.99$ (Cartpole).

For each experimental setup, we first optimize all the meta-parameters of DDPG and then apply them to DENFAC. To obtain a fair comparison, we also optimized the number of updates performed by DDPG, and we applied the inverting gradient strategy (when it was better) to make it more data-efficient. We used Caffe as neural network library (Jia et al., 2014) and Open Dynamic Engine (ODE) as physic engine (Smith, 2005). Figure 3 shows that DENFAC quickly develops good policies on both tasks outperforming others algorithms. CMA-ES is not as good as DDPG since it does not store collected samples. Both CACLA and NFAC cannot reach the goal in only 1500 episodes on Acrobot.

We did not notice that adding a L2 regularization term in the critic improves DDPG in those environments. We found out that having an unbounded last layer for the actor is always better, even without an adapted gradient strategy (like inverting gradient). In some experiments, we also run our algorithm in an *online* setting or with target networks, but this did not improve the data-efficiency, while requiring more computations (results not shown here).

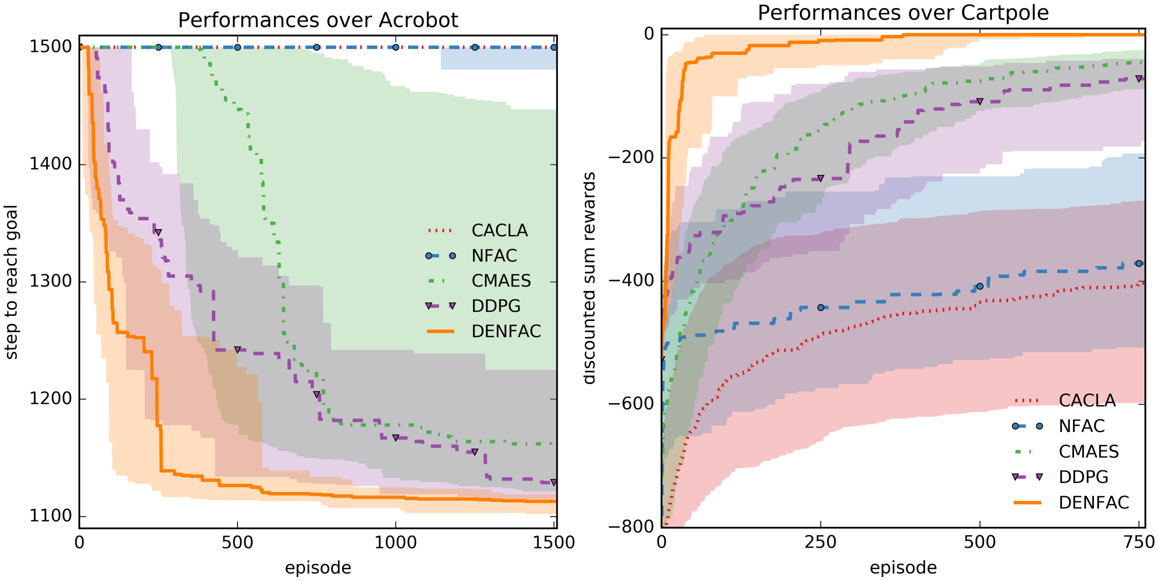

Figure 3: Median and quartiles of the best registered performance in Acrobot (the lower, the better) and Cartpole (the higher, the better) environments during RL learning with each algorithm. Each experiment has been run 40 times for statistical results.

		α_a α_c	inverting gradient	mini batch size	τ	additional updates	K	G	batch size $\mathcal{D}$	reset critic
Acrobot	DDPG	0.1	No	64	0.001	8				
	DENFAC	0.1	No				10	25	5000	Yes
Cartpole	DDPG	0.1	Yes	64	0.1	8				
	DENFAC	0.1	Yes				10	25	5000	No

Figure 4: Best meta-parameters found for DDPG and DENFAC.

5. Conclusions and further work

We investigated the *data-efficency* vs *scalability* dilemma in two fully continuous environments. *Data-efficency* often implies more computational time spent on each data impeding the scalability. In some cases, resetting the weights of the neural networks shows even more *data-efficency*. All those additional costs must be negligible compared to the cost of producing data in the environment otherwise such methods are not appropriate. DENFAC is more data-efficient than the current state-of-the-art *actor-critic* algorithms but comes at a higher computational cost. To further improve DENFAC, it should be analyzed if a First-In First-Out (FIFO) queue is the best choice for $\mathcal{D}$. Moreover, DENFAC lacks stability in learning, target networks did not helped, slowing down the change in the policy might increase his stability (Schulman et al., 2015).

Acknowledgments

We would like to acknowledge support of Bruno Scherrer for fructuous discussions and Iñaki Fernández for his useful comments on the manuscript. The data has been numerically analyzed with the free software package GNU Octave (John W. Eaton David Bateman and Wehbring, 2015). We used Caffe as neural network library (Jia et al., 2014) and Open Dynamic Engine (ODE) as physic engine (Smith, 2005). Experiments presented in this paper were carried out using the Grid5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

András Antos, Rémi Munos, and Csaba Szepesvari. Fitted Q-iteration in continuous action-space MDPs. 2008. ISBN 160560352X.

Steven J. Bradtke, Andrew G. Barto, and Pack Kaelbling. Linear least-squares algorithms for temporal difference learning, 1996. ISSN 0885-6125.

Marc Deisenroth and Carl E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, pages 465–472, 2011.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *Aistats*, volume 15, page 275, 2011.

Ivo Grondman, Lucian Buoniu, Gabriel AD. Lopes, and Robert Babuška. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man and Cybernetics*, 42(6):1291–1307, 2012. ISSN 10946977. doi: 10.1109/TSMCC.2012.2218595.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous Deep Q-Learning with Model-based Acceleration. *arXiv preprint arXiv:1603.00748*, 2016.

Roland Hafner and Martin Riedmiller. Reinforcement learning in feedback control. *Machine Learning*, 84(1-2):137–169, 2011. ISSN 0885-6125. doi: 10.1007/s10994-011-5235-x.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001. ISSN 1063-6560. doi: 10.1162/106365601750190398.

Matthew Hausknecht and Peter Stone. Deep Reinforcement Learning in Parameterized Action Space. *arXiv preprint arXiv:1511.04143*, 2016.

Ronard A. Howard. Dynamic Programming and Markov Processes. 1960.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv preprint arXiv:1502.03167*, 2015.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014.

S\oren Hauberg John W. Eaton David Bateman and Rik Wehbring. {*GNU Octave*} *version 4.0.0 manual: a high-level interactive language for numerical computations*. 2015. URL `http://www.gnu.org/software/octave/doc/interpreter`.

Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, pages 1–13, 2015.

Vijay R. Konda and John N. Tsitsiklis. Actor-Critic Algorithms. *Neural Information Processing Systems*, 13:1008–1014, 1999. ISSN 0363-0129. doi: 10.1137/S0363012901385691.

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.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, and Others. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G. Bellemare. Safe and Efficient Off-Policy Reinforcement Learning. *arXiv preprint arXiv:1606.02647*, 2016.

Martin Riedmiller. Neural fitted Q iteration - First experiences with a data efficient neural Reinforcement Learning method. In *Lecture Notes in Computer Science*, volume 3720 LNAI, pages 317–328, 2005. ISBN 3540292438. doi: 10.1007/11564096_32.

Martin Riedmiller, Jan Peters, and Stefan Schaal. Evaluation of Policy Gradient Methods and Variants on the Cart-Pole Benchmark. In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 254–261, apr 2007. ISBN 1-4244-0706-0. doi: 10.1109/ADPRL.2007.368196.

John Schulman, Sergey Levine, Michael Jordan, and Pieter Abbeel. Trust Region Policy Optimization. *International Conference on Machine Learning*, page 16, 2015. ISSN 2158-3226. doi: 10.1063/1.4927398.

Russell Smith. Open dynamics engine. 2005.

Mark W. Spong. Swing up control problem for the acrobot. *IEEE Control Systems Magazine*, 15(1):49–55, 1995. ISSN 02721708. doi: 10.1109/37.341864.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. A Bradford Book, 1998. ISBN 0262193981.

Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *In Advances in Neural Information Processing Systems 12*, pages 1057–1063, 1999. ISSN 0047-2875. doi: 10.1.1.37.9714.

Hado Van Hasselt. Reinforcement Learning in Continuous State and Action Spaces. In *Reinforcement Learning*, pages 207–251. Springer Berlin Heidelberg, 2012.

Hado Van Hasselt and Marco A. Wiering. Reinforcement learning in continuous action spaces. In *Proceedings of the IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 272–279, 2007. ISBN 1424407060.

Niklas Wahlström, Thomas B. Schön, and Marc Peter Deisenroth. From Pixels to Torques: Policy Learning with Deep Dynamical Models. *arXiv preprint arXiv:1502.02251*, 2015.

Matthieu Zimmer, Yann Boniface, and Alain Dutech. Neural Fitted Actor-Critic. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2016.