Proceedings of the 38th Chinese Control Conference
July 27-30, 2019, Guangzhou, China

# Towards Efficient Mapless Navigation Using Deep Reinforcement Learning with Parameter Space Noise

Xiaoyun Liu[1], Qingrui Zhou[1], Hui Wang[1], Ying Yang[1]

1. Qian Xuesen Laboratory of Space Technology, Beijing 100094, China
E-mail: liuxiaoyun@qxslab.cn, zhouqingrui@qxslab.cn

**Abstract:** This paper presents a deep reinforcement learning framework that is capable of training the mapless motion planner end-to-end by taking the laser range findings as input and the continuous steering commands as output. Major improvements are introduced in our Deep Deterministic Policy Gradient algorithm (DDPG): parameter space noise is used to encourage exploration and an efficient exploration strategy is designed to boost navigation performance. The proposed learning framework is implemented to train the mobile robot in the Gym-gazebo simulation environment. The simulation study shows that the proposed mapless motion planner can navigate the nonholonomic mobile robot effectively without collisions.

**Key Words:** Reinforcement learning, neural networks, parameter space noise for exploration, mobile robot, path planning

## 1 Introduction

Many robotic applications, such as target tracking [1], search and rescue [2] require the mobile robots to explore an unknown environment without collisions. Generally, a collision-free path from the current location of the robot to the goal position can be planned if an accurate map of the environment is provided, such as simultaneous localization and mapping (SLAM) [3], approaches to recover 3D environment structures and/or estimate robot motion models from vision information [4, 5] and other map-based navigation techniques imposing a human-guided pre-training phase [6], which all rely on precise sensor for the mapping work and local cost-map prediction, as well as requiring considerable computation overhead. Mapless navigation offers considerable promise for mobile robot path planning in unknown environments without map available in advance. Several prior works, such as Optical-flow-based solutions [7] that estimate the motion of objects or features within a sequence of images, but can cause failure in algorithms due to scattering effects and bad image quality. Appearance-based matching techniques [8] are based on the storage of images in a previous recording phase, having difficulties with creating the environment representation and defining the on-line matching criteria.

Increasing amounts of researchers are also interested in investigating the potential of implementing Deep Reinforcement Learning (DRL) [9] methods to allow mobile robots to navigate in complex environments. Inspired by the insights from the recent success of Deep Q Network (DQN) [10], Deep Deterministic Policy Gradient (DDPG) makes use of experience replay and separate target networks to reduce variance and increase stability, which allow it to use neural network function approximators to learn in large state and action spaces. [11] presented a learning-based mapless motion planner using an asynchronous DDPG [12] which can be trained end-to-end without any manually designed features and prior demonstrations. However, navigation capability of exploration efficiency was constrained or even ignored. In contrast, our work takes into account of this factor under the framework of Deep Deterministic Policy Gradient algorithm (DDPG) proposed in [13] to train the mapless motion planner. Moreover, as hinted by [14], we encourage exploration through parameter space noise and investigate exploration strategies of DRL methods to learn navigation policies more efficiently.

In this work, we equip our robot with low-cost laser range finders and utilize the DDPG algorithm with parameter space noise for better exploration to train the mapless motion planner from raw sensor input. Our work has the following contributions: (1) We propose a more efficient exploration strategy by implementing the parameter space noise and introducing an exploration rate, which would decay with the episodes running to a minimum threshold value, to improve the exploration of DDPG compared to the classical algorithm proposed in [13]. (2) We simulate the proposed learning framework using gym-gazebo [15] simulation environment that integrates Robotic Operating System (ROS) [16] and Gazebo. To best of our knowledge it's the first work to implement this method to the robot navigation under this integrated platform.

The rest of paper is structured as follows. The principles of the DDPG is presented in section 3. Our proposed improvement of exploration strategy for training DDPG is elaborated and implemented in section 4. The simulation setup and results are presented in section 5, followed by final conclusions in section 6.

## 2 Related Work

There is extensive prior work on learning-based navigation methods that train the mapless path planner from data. Kretzschmar et al. [17] used inverse reinforcement learning method to model the cooperative navigation behavior of humans, which was able to replicate a specific behavior that was taught by tele-operation in the target environment of the robot. However, learning from demonstration and methods based on human supervision are inherently dependent on the demonstration information and limited by the amount of human data available.

[*]This work is supported by National Natural Science Foundation of China(NSFC) of China under Grant 61833009 and National Defense Science and Technology Innovation Special Zone Project.

Recently, deep learning techniques integrated with RL methods have shown promising results. While DQN [10] solves problems with high-dimensional observation spaces, it can only handle discrete and low-dimensional action spaces, which makes it impossible to be straightforwardly applied to continuous domains. Minh et al. [18] utilized the insights of DQN to estimate the value function with deep neural networks. In [19], Zhu et al. proposed an actor-critic model whose policy is a function of the goal as well as the current state, allowing better generalization. However, their control commands are simply discrete actions which may lead to rough navigation behaviors. Tai et al. [11] trained a mapless motion planner using an asynchronous Deep Deterministic Policy Gradients (ADDPG) method, which takes the sparse laser range findings as input and the continuous steering commands as output. Nevertheless, the path generated from this planner lacks exploration efficiency. Thus, our work is under the framework of DDPG, and we propose a more efficient exploration strategy by implementing the parameter space noise and introducing an exploration rate, which would decay with the episodes running to a minimum threshold value, to improve the exploration of DRL.

## 3 Preliminaries

### 3.1 Reinforcement Learning

In Reinforcement Learning (RL) [20], an agent interacts with an environment $E$ in discrete timesteps. We model the environment as a Markov Decision Process (MDP) with a state space $S$, action space $A = IR^N$, an initial state distribution $p(s_1)$, transition dynamics $p(s_{t+1} \mid s_t, a_t)$, and reward function $r(s_t, a_t)$. At each timestep $t$ the agent receives an observation $x_t$, takes an action $a_t$ and receives a reward $r_t$. Here, we assume the environment is fully-observed so the state $s_t = x_t$. An agent's behavior is defined by a policy $\pi$, which maps states to a probability distribution over the actions: $\pi : S \to P(A)$. The return from a state is defined as the sum of discounted future reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r(s_i, a_i)$ with a discounting factor $\gamma \in [0,1]$. The goal of reinforcement learning is to learn a policy which maximizes the expected return from the start distribution $J = E_{r_i, s_i \sim E, a_i \sim \pi}[R_1]$.

In RL, we use the action-value function to describe the expected return after taking an action $a_t$ in state $s_t$ and thereafter following policy $\pi$:

$$Q^{\pi}(s_t, a_t) = E_{r_i \geq t, s_i > t \sim E, a_i > t \sim \pi}[R_t \mid s_t, a_t] \tag{1}$$

For a deterministic policy $\mu : S \to A$, the recursive relationship known as the Bellman equation is:

$$Q^{\mu}(s_t, a_t) = E_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q^{\mu}(s_{t+1}, \mu(s_{t+1}))] \tag{2}$$

The central optimization of RL can be then expressed as:

$$\mu^* = \arg\max_{\mu} Q^{\mu}(s_t, a_t) \tag{3}$$

with $\mu^*$ being the optimal policy.

### 3.2 Deep Deterministic Policy Gradient Algorithm

Deep Deterministic Policy Gradient (DDPG) [13] is an off-policy algorithm which concurrently learns a Q-function and a policy. Since it is not possible to straightforwardly apply Q-learning to continuous action spaces, in which we can't exhaustively evaluate the space, and solving the optimization problem is highly non-trivial. DDPG uses an actor-critic approach based on the DPG algorithm [21], which consists of a parameterized actor function $\mu(s \mid \theta^{\mu})$ specifying the current policy by deterministically mapping states to a specific action. The actor is updated by applying the following policy gradient:

$$\nabla_{\theta^{\mu}} J \approx E_{s_t \sim \rho^{\beta}}[\nabla_{\theta^{\mu}} Q(s, a \mid \theta^{Q})|_{s_t, \mu(s_t)} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu})|_{s_t}] \tag{4}$$

The critic $Q(s, a)$ is learned using the Bellman equation as in Q-learning. Suppose the approximator is a neural network parameterized by $\theta^{Q}$, and that we have collected a set $R$ of transitions $(s_t, a_t, r_t, s_{t+1}, d_t)$, then we can optimize the approximator by minimizing the loss:

$$L(\theta^{Q}) = E_{s_t \sim \rho^{\beta}, a_t \sim \beta, r_t \sim E}[(Q(s_t, a_t \mid \theta^{Q}) - y_t)^2] \tag{5}$$

where $\beta$ represents a stochastic behavior policy, and

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1} \mid \theta^{Q})) \tag{6}$$

DDPG also employs the merits from its predecessor algorithms to make it robust and efficient in learning when introducing nonlinear function approximators to learn the value functions. As in DQN, DDPG uses a replay buffer to train the deep neural networks. The replay buffer is a finite sized cache $R$, in which the tuple $(s_t, a_t, r_t, s_{t+1})$ is stored. The replay buffer should be large, allowing the algorithm to benefit from learning across a set of uncorrelated and identically distributed transitions. Another addition in DDPG is the use of a second network, called the target network, and "soft" target updates. There are a copy of the actor and critic networks, $Q'(s, a \mid \theta^{Q'})$ and $\mu'(s \mid \theta^{\mu'})$ respectively, that are used for calculating the target values in order to consistently train the critic without divergence. The weights of these target networks are then updated by having them slowly track the learned networks:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{7}$$

with $\tau << 1$.

For exploration, DDPG uses an action space stochastic policy of the form $\mu(s_t) = \mu(s_t \mid \theta_t^{\mu}) + w$, where $w$ is either $w \sim N(0, \sigma^2 I)$ (uncorrelated) or $w \sim OU(0, \sigma^2)$ (correlated).

## 4 Mapless Motion Planner Implementation

Our work aims to provide a mapless motion planner for mobile ground robots by leveraging the DDPG algorithm with adaptive noise to the parameters to boost performance. This ensures consistency in actions, and directly introduces a dependence between the state and the exploratory action taken.

### 4.1 DDPG with Adaptive Parameter Noise for Exploration

Denote policies that are realized as parameter functions as $\mu_\theta$, with $\theta$ being the parameter vector. Parameter noise adds additive Gaussian noise to the parameter vector of the current policy: $\tilde{\theta} = \theta + N(0, \sigma^2 I)$, rather than to its action space. Denote this perturbed policy as: $\tilde{\mu} := \mu_{\tilde{\theta}}$ and analogously define $\mu := \mu_\theta$. $\sigma$ is the adaptive noise scaling and can be presented as:

$$\sigma_{k+1} = \begin{cases} \partial \sigma_k, & d(\mu, \tilde{\mu}) \le \delta \\ \frac{1}{\partial} \sigma_k, & otherwise \end{cases} \tag{8}$$

where $\partial \in R_{>0}$ is a scaling factor and $\delta \in R_{>0}$ is a threshold value. The concrete realization of $d(\mu, \tilde{\mu})$ in DDPG is:

$$d(\mu, \tilde{\mu}) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} E_s[(\mu(s)_i - \tilde{\mu}(s)_i)^2]} \tag{9}$$

Where $E_s[\cdot]$ is estimated from a batch of states from the replay buffer and $N$ denotes the dimension of the action space.

We use layer normalization, which ensures that the output of a perturbed layer is still within a similar distribution. This deals with the problem that different layers of the network have different sensitivities to perturbations.

Our DDPG implementation also introduces an exploration rate $\varepsilon$, which would decay with the episodes running to a minimum threshold value. In the first episode, $\varepsilon$ is set to 1, and the agent takes actions which are sampled from a uniform random distribution over valid action spaces. After that, it gradually returns to normal DDPG exploration as $\varepsilon$ decays. A pseudocode of our DDPG algorithm is detailed in Algorithm 1.

### 4.2 Problem Definition

We train a mapless motion planner for mobile ground robots to navigate in environments without collisions. We model the problem as establishing the following translation function:

$$\omega_t = f(x_t) \tag{10}$$

Where $x_t$ is the observation from the raw sensor information, which is the 100-dimensional laser range findings and can be regarded as the instant state $s_t$. The model directly maps the state to the action $\omega_t$ which is the command angular velocity.

### 4.3 Network Structure

We design the Actor, as shown in Fig.1, as a 2 fully-connected neural network. The input is the 100-dimensional laser range findings that are sampled from the raw laser findings between -90 and +90 degrees in a trivial and fixed angle distribution. The output of the Actor network is the angular velocity commands of the mobile robot after 2 fully-connected layers of neural network with 300 neurons with Relu. The tanh activation function is to limit the output range within [-1,1]. The output actions are multiplied with a hyper parameter to decide the final angular velocity directly executed by the mobile robot. Considering the real dynamics of a Turtlebot, we choose 0.3rad/s as the max angular velocity.

Algorithm 1 Modified Q-learning Algorithm

```
Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor network $\mu(s|\theta^\mu)$;
Initialize target networks $Q^{'}$, $\mu^{'}$;
Initialize replay buffer $R$
1. For $episode = 1:N$ do
2.   Initialize a random process and the exploration rate $\varepsilon$ for action exploration
3.   While not done:
4.     Generate $\rho \in (0,1)$ randomly
5.     If $(\rho < \varepsilon)$
6.       $a_t = random.uniform(-act\_range, act\_range)$
7.     Else: $a_t = \mu(s_t|\tilde{\theta}^\mu)$ according to the current policy and parameter space noise
8.     Execute action $a_t$, observe reward $r_t$ ,new state $s_{t+1}$ and if it's terminal $d_t$
9.     Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
10.    Sample a random minibatch of $m$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
11.    Set $y_i = r_i + \gamma Q^{'}(s_{i+1}, \mu^{'}(s_{i+1}|\theta^{\mu^{'}})|\theta^{Q^{'}})$, and update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
12.    Update the actor policy using the policy gradient:
         $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$
13.    Update the target networks:
         $\theta^{Q^{'}} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q^{'}}$
         $\theta^{\mu^{'}} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu^{'}}$
14.    $\varepsilon = 0.095 \cdot \varepsilon, and \varepsilon = \min(0.05, \varepsilon)$
15. End for
```

The Critic, as shown in Fig. 1 shares the structure of the Actor and predicts the Q-value of the state and action pair. The action is merged in the second fully-connected neural network layers. The Q-value is finally activated through a linear activation function.

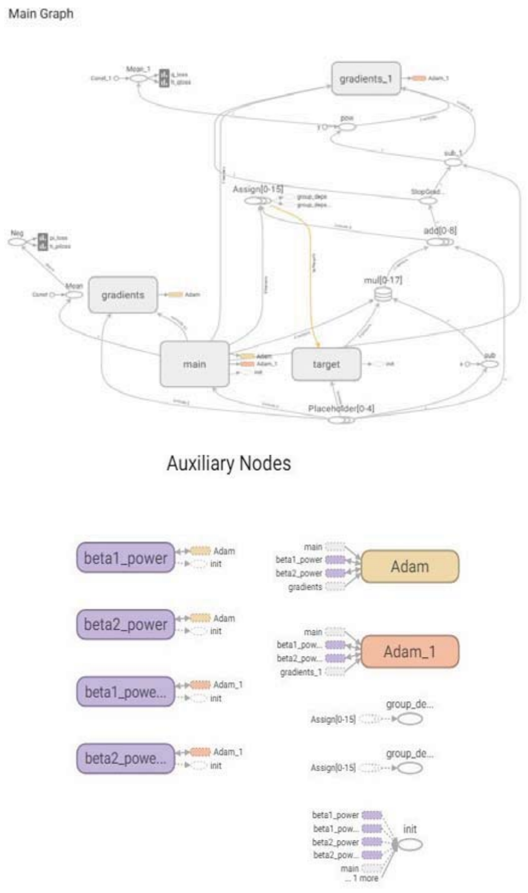

Fig. 1: The structure of neural networks

## 5 Simulation Study

This section illustrates the training results of our model in an virtual environment simulated by gym-gazebo, where the Turblebot is trained to navigate in the 2D environment *GazeboCircuit2TurtlebotLidar-v0* without collisions.

### 5.1 Simulation Environment Setup

Gym-gazebo is a toolkit for developing and comparing RL algorithms using Robotic Operating System (ROS) and Gazebo. We simulated our algorithm in one of the 2D environments called *GazeboCircuit2TurtlebotLidar-v0*, as shown in Fig.2, which is a circuit with straight tracks and 90 degree turns. The simulation environment provides 100-dimensional discretized LIDAR readings to train the Turtlebot.

The goal of the task is that the mobile robot can navigate along the tracks and make turns when necessary without collisions. The environment runs episodically, and an episode terminates if: the robot collides to the walls around tracks, or the maximum runtime (2000 episodes) is achieved without collisions.

The reward function is designed as the following:

$$r(s_t,a_t)=\begin{cases}15\times[\max(\omega)-|\omega|+0.0335] \\ -200, & if\ x_t<c_o\end{cases} \tag{11}$$

Where $x_t$ and $c_o$ represent the laser range findings and a minimum range checking for collision, respectively. If the robot collides with the walls, a negative reward is arranged. Otherwise, the reward is given in Eq.(11), where $\omega$ is the command angular velocity learned by the actor neural network. Other hyper-parameters are listed in Table 1.

Table 1: Hyper Parameters Used in Simulation

| Hyper Parameters | |
|---|---|
| Replay Buffer Size | 50000 |
| Discounted Factor | 0.99 |
| $\tau$ for Soft Update | 0.001 |
| Learning Rate for the Actor | 0.0001 |
| Learning Rate for the Critic | 0.0001 |
| Batch size | 32 |
| Act noise | 0.01 |

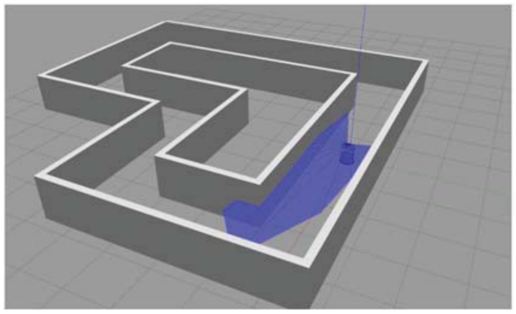

Fig. 2: GazeboCircuit2TurtlebotLidar-v0

### 5.2 Simulation Results

We trained the model from scratch with an Adam [22] optimizer for 2000 episodes which took almost 8 hours. The Q loss in the critic network and policy gradient in the actor network are shown in Fig.3 and Fig.4, respectively. The Q loss that estimates how closely the predicted Q value comes to satisfying the Bellman equation in Eq.(5) converges, which means the algorithm can learn the policy to finish the task.

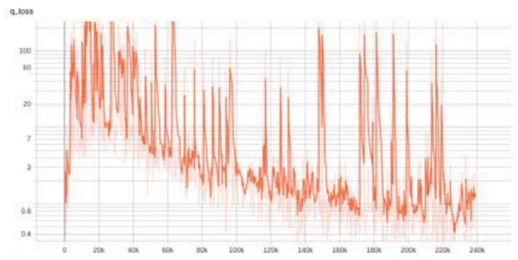

Fig. 3: The Q loss in the critic network

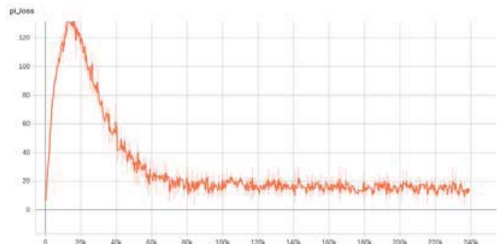

Fig. 4: The policy gradient in the actor network

## 6 Conclusions

In this paper, a mapless motion planner is trained end-to-end through a continuous deep-RL method from scratch. We investigate a more efficient exploration strategy using parameter space noise rather the traditional action space noise, which can enhance the navigation performance. The motion planner can be directly implemented in unseen environments such as the 2D simulation environment in Gym-gazebo, taking the 100-dimensional laser range findings as input and the steering command as the output. Simulation proves that our proposed method can achieve collision-free navigation efficiently. Our future work would implement the method to real world experiment, and improve the performance by using Recurrent Neural Networks (RNN) and LSTM.

## References

[1] A.C. Woods and H.M. La. Dynamic target tracking and obstacle avoidance using a drone. *International Symposium on Visual Computing. Springer*, 2015, pp.857-866.

[2] T.Tomic, K. Schmid, P. Lutz, A. Domel et al. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE robotics & automation magazine*, vol.19, no. 3, pp. 46-56, 2012.

[3] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part I. *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006

[4] B. Smith. An approach to graphs of linear forms, accepted.

[5] D. Cheng, On logic-based intelligent systems, in *Proceedings of 5th International Conference on Control and Automation*, 2005: 71–75.

[6] Yengera, Gaurav , et al. Less is More: Surgical Phase Recognition with Less Annotations through Self-Supervised Pre-training of CNN-LSTM Networks, *Computer Vision and Pattern Recognition*, *arXiv:1805.08569,* 2018.

[7] Talukder, Ashit , and L. Matthies. Real-time Detection of Moving Objects from Moving Vehicles Using Dense Stereo and Optical Flow. *IEEE/RSJ International Conference on Intelligent Robots & Systems IEEE*, 2004.

[8] Zhou, Chao , Y. Wei , and T. Tan. Mobile robot self-localization based on global visual appearance features. *IEEE International Conference on Robotics & Automation IEEE*, 2003.

[9] Krizhevsky, Alex , I. Sutskever , and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems* 25.2 (2012).

[10] Volodymyr Mnih, et al. Human-level control through deep reinforcement learning. *Nature* 518.7540 (2015):529-533.

[11] Tai, Lei , G. Paolo , and M. Liu. Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation. *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on IEEE*, 2017.

[12] S. Gu, E. Holly, T. Lillicrap, and S. Levine. Deep reinforcement learningfor robotic manipulation with asynchronous off-policy updates. *arXiv preprint arXiv:1610.00633*, 2016.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[14] Plappert, Matthias, et al. Parameter Space Noise for Exploration. *International Conference on Learning Representations*, 2017.

[15] Fairchild, Carol, and T. L. Harman. *ROS Robotics By Example*. 2016.

[16] Zamora, Iker, et al. Extending the OpenAI Gym for robotics: a toolkit for reinforcement learning using ROS and Gazebo. 2016.

[17] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, vol. 35, no. 11, pp.1289–1307, 2016.

[18] Levine S , Finn C , Darrell T , et al. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 2015, 17(1):1334-1373.

[19] Zhu, Yuke, et al. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *IEEE International Conference on Robotics & Automation (ICRA)*, 2016.

[20] R. S. Sutton and A. G. Barto. Reinforcement learning: An Introduction. *MIT press Cambridge*, 1998, vol. 1, no. 1.

[21] Silver, David, et al. Deterministic Policy Gradient Algorithms. *International Conference on International Conference on Machine Learning* 2014.

[22] D. Kingma and J. Ba, Adam: A method for stochastic optimization. *arXiv preprint arXiv*:1412.6980, 2014.