Multi-Agent Distributed Deep Deterministic Policy Gradient for Partially Observable Tracking

Dongyu Fan¹, Haikuo Shen¹,²,∗ and Lijing Dong¹,²,³

Abstract: In many existing multi-agent reinforcement learning tasks, each agent observes all the other agents from its own perspective. In addition, the training process is centralized, namely the critic of each agent can access the policies of all the agents. This scheme has certain limitations since every single agent can only obtain the information of its neighbor agents due to the communication range in practical applications. Therefore, in this paper, a multi-agent distributed deep deterministic policy gradient (MAD3PG) approach is presented with decentralized actors and distributed critics to realize multi-agent distributed tracking. The distinguishing feature of the proposed framework is that we adopted the multi-agent distributed training with decentralized execution, where each critic only takes the agent’s and the neighbor agents’ policies into account. Experiments were conducted in the distributed tracking tasks based on multi-agent particle environments where 𝑁 (𝑁 = 3, 𝑁 = 5) agents track a target agent with partial observation. The results showed that the proposed method achieves a higher reward with a shorter training time compared to other methods, including MADDPG, DDPG, PPO, and DQN. The proposed novel method leads to a more efficient and effective multi-agent tracking.

Keywords: multi-agent systems; deep reinforcement learning; actor–critic; partial observability

1. Introduction

Multi-agent systems (MASs) comprise intelligent entities (robots, vehicles, machines, etc.) that interact within a common environment [1,2]. The tracking of a target in MASs often plays a pivotal part in a variety of applications, including convoys, surveillance, and navigation [3–6].

In the scientific literature on MASs, the theory and application related to reinforcement learning (RL) have attracted considerable interest [7–9]. RL is an area of machine learning, providing a formalism for behavior and concerning how agents are supposed to adopt effective actions in an environment so as to maximize the long-term reward [10]. The classical RL methods are limited to the tasks with a small state space and action space that are generally discrete. However, more complex tasks, which are closer to practical applications, often have a large state space and continuous action space. The classical RL methods have difficulty dealing with high-dimensional input. With the improvement of computing and storage capacity, deep reinforcement learning (DRL), a combination of deep learning (DL) and RL, has achieved human-level or higher performances on challenging games [11–14]. Multi-agent designs are more robust than single monolithic designs, since agent failures can be compensated, which makes the system more scalable and complex in the sense of [15]. Multi-agent deep reinforcement learning (MADRL) applies the idea and algorithm of DRL to the learning of multi-agent systems [16–20], which is an important
method to develop swarm intelligence \[21,22\] and has received increasing attention in recent years \[23–30\].

The basic assumption of RL is to increase the accumulated reward as much as possible; thus, the design of reward function is certainly a key step. Several recent studies investigating difference reward \[31\], coordinated learning without exploratory action noise (CLEAN) reward \[32\], and counterfactual reward \[33\] have been carried out on MADRL. Independent Q-learning (IQL) \[34\] simply and roughly performs a Q-learning algorithm for each agent, which has a good effect in some applications. Value decomposition networks (VDNs) \[35\] learn to divide the joint value function into pieces, which are agentwise and additive. Monotonic value function factorization (QMIX) \[36\] can combine the value of each agent based on a local observation in a complex and nonlinear way to estimate the joint action value. Unlike VDNs and QMIX, the factorisable pieces in learning to factorize with transformation (QTRAN) \[37\] is not constrained by additivity or monotonicity, but has a more general factorization method of joint action–value functions. However, the multi-agent deep deterministic policy gradient (MADDPG) \[38\] extends the deep deterministic policy gradient (DDPG) method for multi-agent environments, where the critic takes the information about the policies of all the agents into account while the actor has access only to the information of the agent itself and the associated information of the other agents from the perspective of the agent.

The methods \[35–38\] proposed in recent years took the decentralized partially observable Markov decision process (Dec-POMDP) \[39,40\] as a de facto standard to model cooperative multi-agent tasks, but there is much less research on partially observable stochastic games (POSGs) \[41\] in multi-agent systems. The agents in POSG do not use a joint reward signal, which is suitable for describing decision-making problems in scenarios that consist of multiple agents with partial observations.

While MADRL is a growing field, relatively little research has been carried out on distributed tracking problems in the field. Distributed mode is applicable to multi-agent tracking, which requires less computation and fewer costs. However, in the literature mentioned above, each agent observes all the other agents, and the training process of multiple agents is centralized \[35–38\]. The generalizability of these results is subject to certain limitations. In practical applications, due to the limitations of the communication technique, the communication range is constrained. It is unpractical for each agent to observe all other agents’ information. Besides, the MASs with centralized training require a large amount of computation and training costs. To alleviate these problems, the research on distributed problems is of significance, where each agent can only obtain the observation of the neighbor agents in distributed tracking cases. There remains a paucity of studies focusing specifically on the issue of multi-agent reinforcement learning for distributed partially observable tracking, which is the main content of our research.

In brief, the aforementioned methods ignored several problems as follows. In practical tracking processes, each agent could hardly obtain the observation of all the other agents. Besides, the right reward function should be given to each agent to satisfy the goal of the system. In addition, the centralized training mode is not suitable for distributed tracking of multi-agent learning.

Motivated by the above discussions, this paper explores scenarios in a distributed topology, in which a number of agents with partial observations of neighbor agents cooperate to track the target. To realize the multi-agent distributed tracking, we provide a novel method of multi-agent reinforcement learning, as shown in Figure 1. This paper investigates multi-agent distributed tracking problems with the framework of distributed training and decentralized execution.
The major contributions are addressed as follows:

- A new multi-agent tracking environment based on the multi-agent particle environment (MPE) is built. Different from receiving observations of all agents, in our scenarios, each agent only obtains information from the neighbor agents, which is more practical under the condition of limited communication;
- In consideration of the distributed partially observable condition, this study establishes a framework of a private reward mechanism for the multi-agent tracking system, providing a kind of incentive method in distributed scenarios. This is the fundament of further research;
- A novel methodology named the multi-agent distributed deep deterministic policy gradient (MAD3PG) for the tracking system is proposed. In this method, a distributed critic and a decentralized actor are designed for the distributed scenario. The importance and originality of the MAD3PG is that it adopts distributed training with decentralized execution rather than centralized or decentralized training, which are widely used in current approaches. Compared with recent relevant algorithms, the MAD3PG has shorter time cost and better training performance.

The remaining part is organized as follows. Section 2 introduces the necessary background regarding graph theory, similarities and differences between the Dec-POMDP and POSG, and the current methods related to our research. Section 3 addresses the main contributions, including the essential settings of the distributed tracking model and the structure of the MAD3PG. The results of empirically evaluating the MAD3PG and compared methods are discussed in Section 4. Section 5 concludes this work and briefly describes the direction for future research.

2. Background

2.1. Graph Theory

In the multi-agent tracking scenario, agents in different positions aim to be consistent with the target agent through learning, which can often be represented as a set of different agents with an inherent high-level structure [42]. Every single agent could be abstracted as a node. Then, the communication topology of the target tracking system could be described as a graph, which is denoted by $G = (\mathcal{N}, \mathcal{E}, \mathcal{A})$, where $\mathcal{N} = \{1, 2, \ldots, N\}$, $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$, and $\mathcal{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$ represent a node set, an edge set, and a weighted adjacency matrix, respectively. In a directed graph, $(i, j)$ and $(j, i)$, which belong to $\mathcal{E}$, are regarded differently. The nodes that can be accessed by node $i$ are judged as its neighbors. A set of neighbors of node $i$ is described by $\mathcal{N}_i = \{j | j \in \mathcal{N}, (i, j) \in \mathcal{E}\}$. With $a_{ij} > 0$, otherwise $a_{ij} = 0$. The connections between tracking agents and the target are expressed by a weighted matrix $\mathcal{B} = \text{diag}\{\beta_1, \beta_2, \ldots, \beta_N\}$. If the $i$-th tracking agent can obtain the target’s information, $\beta_i > 0$, otherwise $\beta_i = 0$. 

![Figure 1](image_url). The proposed multi-agent graph on the left and a detailed look at the internal architecture of each agent on the right. Other observable neighbor agents of each agent are depicted by gray edges, for instance the $e$-th agent and $j$-th agent are neighbor agents of the $i$-th agent.
2.2. Multi-Agent Decision Process

An agent of full observability directly observes the environment state, formally as a Markov decision process (MDP). An agent of partial observability indirectly observes the environment, formally as a partially observable Markov decision process (POMDP) [43]. In MASs, the process that a bunch of agents are working together for a common reward is called a decentralized POMDP (Dec-POMDP) [39,40]. In general, the Dec-POMDP can be expressed by a tuple $< I, S, \{A_i\}, \mathcal{P}, \{\Omega_i\}, O, R, h >$, where the meaning of each symbol can be obtained by referring to Table 1. This model contains only one reward function, meaning that the agents operate together toward the same objective.

| Symbol | Meaning | Dec-POMDP | POSG |
|--------|---------|-----------|------|
| $I$    | a finite set of agents indexed $\{1, 2, \ldots, N\}$ | ✓ | ✓ |
| $S$    | a finite set of states | ✓ | ✓ |
| $A_i$  | a finite set of actions available to agent $i$ | ✓ | ✓ |
| $\mathcal{P}$ | a Markovian transition function | ✓ | ✓ |
| $\Omega_i$ | a finite set of observations available to agent $i$ | ✓ | ✓ |
| $R$ | a reward function | ✓ | ✓ |
| $R_i$ | an individual reward function of the $i$-th agent | ✓ | ✓ |
| $h$ | a finite horizon | ✓ | ✓ |

The Dec-POMDP is useful when all agents have a common goal. However, the Partially Observable Stochastic Game (POSG) [41] allows each agent to have a different objective encoded by a private reward function, which is represented by a tuple $< I, S, \{A_i\}, \mathcal{P}, \{\Omega_i\}, O, \{R_i\}, h >$, where $R_i$ refers to the individual reward function of the $i$-th agent. The difference between the POSG and Dec-POMDP can be easily distinguished from Table 1. Except the reward function, all the components in the POSG are the same as those in the Dec-POMDP.

2.3. Policy Gradient

The agent’s goal is to learn a policy, which is a mapping of state to the optimal action. The parametric representation of policy is $\pi_\theta(s, a) = \mathbb{P}[a|s, \theta]$. In order to learn the policy parameter $\theta$, policy gradient methods seek to maximize performance according to the gradient of some scalar performance measure $J(\theta)$ of policy parameters. The update process can be regarded as a gradient ascent in $J$: $\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t)$. $\nabla J(\theta)$ is the policy gradient [10], as shown in Equation (1).

$$\nabla \theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla \theta \log \pi_\theta(a_{i,t}|s_{i,t}) Q^\pi_{i,t}$$ (1)

where $\hat{Q}^\pi_{i,t}$ is an estimate of the return if the action $a$ is taken in the state $s$.

As one of the variants of the policy gradient, proximal policy optimization (PPO) [13] adopts importance sampling to realize on-policy learning, which is the default RL algorithm in OpenAI. Either an adaptive KL penalty coefficient or a clipped surrogate objective is available in PPO. In [13], with the setting of clipping, PPO was simpler to implement and had better performance. The multi-agent scenario in this paper required some lines of code change to a vanilla PPO implementation for the comparison in the experiments.

2.4. Value Function Methods

$Q(s, a)$ denotes an array estimate of action–value function $q_\pi(s, a)$ or $q_+ (s, a)$. $Q^\pi(s, a)$ denotes an estimate of the value when action $a$ is taken in state $s$ under policy $\pi$, which is used in Q-learning. One of the representations of Q-function is $Q^\pi(s, a) = \mathbb{E}_\pi[r(s, a) + \gamma \mathbb{E}_{a' \sim \pi} \left[ Q^\pi(s', a') \right]]$. $Q^\pi(s, a)$ refers to the estimated action–value under the optimal policy.
\[ \mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s',t \sim D}[(Q^*(s,a|\theta) - y)^2] \]

where \( y = r + \gamma \max_{a'} Q^*(s', a'|\theta') \), and \( Q^* \) stands for the target action–value function \([44]\).

In a multi-agent environment, each agent \( i \) owns an independent optimal function \( Q_i \) that is updated by learning. Value function methods are key to most of the reinforcement learning algorithms, which are essential for the efficient search in the space of policies. There is a more general view about the family of Q-learning algorithms, which have three main processes consisting of data collection, target update, and Q-function regression \([45,46]\).

### 2.5. Deep Deterministic Policy Gradient

When facing a continuous vector action \( a \), Q-learning does not have general solutions to the optimization problem, which can be solved by a neural network \( \mu(s) \) named the actor that outputs the action \([47,48]\). Meanwhile, the Q function, as the critic, tells the actor what kind of behavior will obtain more value. They are updated in the deep deterministic policy gradient (DDPG) as follows.

The loss function of the critic is expressed by:

\[ L = \mathbb{E}_{s_i,a_i,r_i,s'_{i+1} \sim D}[Q(s_i,a_i|\phi) - y] \]

where:

\[ y = r_i + \gamma Q^*(s_{i+1},a_{i+1}) \]

By the sampled policy gradient, the actor can be updated as:

\[ \nabla_{\theta} \approx \mathbb{E}_{s_i \sim D}[\nabla_a Q(s,a|\phi)|_{s=s_i,a=\mu(s)} \nabla \mu(s|\theta)|_{s=s_i}] \]

where \( \theta \) refers to the parameters of the policy networks, which are updated using the chain rule. In addition, there exist a target Q-function \( Q' \) and a target actor \( \mu' \). \( \phi' \) and \( \theta' \) represent the parameters of \( Q' \) and \( \mu' \), respectively.

### 2.6. Multi-Agent Deep Deterministic Policy Gradient

The multi-agent deep deterministic policy gradient (MADDPG) \([38]\) is a common algorithm used in deep reinforcement learning in environments where multiple agents are interacting with each other. The MADDPG is based on a framework of centralized training and decentralized execution (CTDE). That is, agents can share experience during training, which is implemented by a shared experience replay buffer \( D \) containing the information of states, next states, actions, and rewards. Each agent uses only local observations at execution time. Consider a scenario with \( N \) agents with policies \( \mu_{\theta_i} \); the gradient is calculated by:

\[ \nabla_{\theta_i} \mu_{\theta_i}(a_i|o_i) = \mathbb{E}_{x,a \sim D} \nabla \mu_{\theta_i}(a_i|o_i) \nabla \mu_{\theta_i}(x,a_1,a_2,\ldots,a_N|a_i = \mu_{\theta_i}(o_i)) \]

where \( Q^i_t(x,a_1,a_2,\ldots,a_N) \) can be updated in centralized training by the following loss function:

\[ \mathcal{L}(\theta_i) = \mathbb{E}_{x,a,r,s' \sim D}[(Q^i_t(x,a_1,a_2,\ldots,a_N) - y)^2] \]

\[ y = r_i + \gamma Q^i_{t+1}(x,a_1,a_2,\ldots,a_N|a'_i = \mu'_i(o_i)) \]

where \( \mu'_i \) represents the target policies of the agents and \( (x,a,r,s') \) denotes the sample obtained from \( D \).

### 3. Proposed Method

#### 3.1. Distributed Observation

Consider a partially observable distributed scenario for multi-agent reinforcement learning, where the \( i \)-th \( (i \in N) \) agent receives its own local observation \( o_i^t \) rather than the
state $s_t$ at time $t$. For simplicity, time $t$ is dropped in the following notations. In this work, instead of containing the states of all the agents, $o_i$ consists of the positions and velocities of the $i$-th agent as well as its neighbors according to the topology of the multi-agent tracking system. Besides, each agent $i \in \mathcal{N}$ keeps its own reward function that is pertinent to the task.

The observation of the $i$-th tracking agent is:

$$o_i = [X_i, \alpha_{ij} \tilde{X}_j, \beta_i \tilde{X}_T] \quad (j \in \mathcal{N}_i)$$

where $X_i$ is a vector that contains the total information of the $i$-th agent, $\tilde{X}_j$, $\tilde{X}_T$ refers to a vector that contains the partial information of the $j$-th agent and the target, respectively, and $\alpha_{ij}$, $\beta_i$ are connected weights between the $i$-th tracking agent and the $j$-th tracking agent and the $i$-th tracking agent and the target agent, respectively.

### 3.2. Reward Function

In the scenario of multi-agent reinforcement learning, each agent aims at maximizing its expected accumulated reward. The global goal of the multi-agent tracking system is to minimize the distances between each tracking agent and the target as soon as possible, which are important indexes in the design of the reward function. In this paper, due to the requirement of the distributed tracking where each agent observes the information of the neighbor agents from its own perspective, using the reward function related to the distributed topology can effectively reduce the computation cost. Thus, the reward of the $i$-th tracking agent, as shown in (9), is supposed to be designed concerning the task demands, as well as the tracking topology.

$$r_i = -\sum_{j=1}^{N} k \alpha_{ij} \| P_i - P_j \|_2^2 - m \beta_i \| P_i - P_T \|_2^2 \quad (j \in \mathcal{N}_i)$$

where $P_i$, $P_j$, and $P_T$ refer to the position information of the $i$-th, the $j$-th agent, and the target agent, respectively. $k$ and $m$ denote reward evaluation factors that are task related, which can balance the weight between the $i$-th tracking agent and the $j$-th tracking agent and the $i$-th tracking agent and target agent.

### 3.3. Multi-Agent Distributed Deep Deterministic Policy Gradient

In order to deal with distributed tracking problems in multi-agent RL, a multi-agent distributed deep deterministic policy gradient (MAD3PG) method is proposed, as illustrated in Figure 2.

Our method improves the critic in the training process. Each blue dotted line with an arrow in Figure 2 is to judge whether the $j$-th or $i$-th agent is the neighbor agent of the $i$-th or $j$-th agent. If the judging result is true, the information can be transmitted; otherwise, the information cannot be transmitted. In the MADDPG, the information can be transmitted without any judgments. Thus, in the centralized training of the MADDPG, the information of all the agents is fed into each critic, while in distributed training of the MAD3PG, only the information of the agent and its neighbors can be fed into the corresponding critic.

More concretely, consider a system with $N$ tracking agents and a target, where each agent owns a decentralized actor and a distributed critic, the parameters of which are represented by $\theta$ and $\phi$, respectively. As outlined in the Algorithm 1, MAD3PG has two complementary strategies: decentralized policies, as well as distributed critics, the details of which are addressed as follows.
Algorithm 1 MAD3PG.

1: Initialize the environment with $N$ tracking agents and a target agent
2: Initialize an experience replay buffer $\mathcal{D}$
3: for episode=1 to $E$, where $E$ denotes the number of training episodes do
   4:   Reset the environment, and obtain initial state $x$; each agent $i$ obtains the initial observation $o_i$
   5:   for $t = 1$ to $T$, where $T$ refers to the maximum number of steps in an episode do
      6:      for agent $i = 1$ to $N$ do
         7:         Select action $a_i = \mu_i(o_i) + \mathcal{N}_i$ based on the actor, as well as exploration
         8:         Execute action $a_i$, then obtain reward $r_i$ and new observation $o_i'$
         9:         Store $(o_i, a_i, r_i, o_i')$ in $\mathcal{D}$
      10:     end for
   11:   State $x$ is replaced by the next state $x'$
   12:   for agent $i = 1$ to $N$ do
      13:      Randomly sample a mini-batch $B$ of $(o, a, r, o')$ from $\mathcal{D}$
      14:      Set the ground truth of the critic networks to be $y_i = r_i + \gamma Q_i(o_i', a_i', \theta_i')$ (where $i \in N_i$)
      15:      Update the distributed critic by minimizing the loss: $L(\theta_i) = \mathbb{E}_{o_i,a_i,r_i,o_i',\theta_i} \mathcal{D}[(Q_i(o_i, a_i, a_i', \theta_i) - y_i)^2$ (where $i \in N_i$)
      16:      where $\delta_i = [o_i, a_i]$ and $\delta_i = [a_i, a_i]$
      17:      Update the decentralized actor:
         18:      \[ \nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{o_i,a_i,r_i,o_i',\theta_i} \mathcal{D}[(\nabla_{\theta_i} \mu_i(a_i,o_i) \nabla_{a_1} Q_i(o_i,a_1,a_i,a_j(j \in N_i) | a_j = \mu_j(o_j))] \]
      19:      end for
   20:   if episode $\mod n = 0$, where $n$ denotes the target update interval then
      21:      Update the parameters of the target actor networks and target critic networks for each agent $i$:
      22:      $\theta_i' \leftarrow \tau \theta_i + (1 - \tau) \theta_i'$
      23:      $\phi_i' \leftarrow \tau \phi_i + (1 - \tau) \phi_i'$
   24:   end if
   25: end for

Figure 2. Frameworks of the MAD3PG.
As illustrated in Figure 2, the input of the i-th agent’s policy networks is the observation \( o_i \), which contains itself and its neighbor agents’ information, while the output is the action \( a_i \). The gradient for the policy of the i-th agent \( \mu_{\theta_i} \) with respect to parameters \( \theta_i \) is expressed as follows:

\[
\nabla_{\theta_i} J(\mu_{\theta_i}) = \mathbb{E}_{\delta_i,a_i} \sum_{l=1}^{L} \nabla_{\theta_i} \mu_{\theta_i}(a_i | o_i) \nabla_{a_i} Q_{\theta_i}^{a_i} (o_i, o_{j_l}, a_i, a_{j_l}) | a_i = \mu_{\theta_i}(o_i) 
\]

where \( \mathcal{D} \) refers to the experience replay buffer, containing the data from interaction with the environment \( (\delta_i, a_i, r_i, o_{i+1}) \), \( \mu_{\theta_i}^{a_i} = \{ \mu_{\theta_i}, \mu_{\theta_{[j_i]}(\in N_i)} \} \) is a set of policies, and \( \mu_{\theta_{[j_i]}(\in N_i)} \) denotes the policies of the i-th agent’s neighbors with parameters \( \theta_j \). \( \mu_{\theta_i} \) uses rewards to directly enhance and weaken the possibility of choice behavior. That is, the probability of positive behavior being selected next time will be enhanced, while the probability of negative behavior being selected next time will be reduced.

It can be seen from Figure 2 that the input of the i-th agent’s critic networks consists of \( o_i, a_i, o_j, \) and \( a_j \) only if the \( j \)-th agent is its neighbor agent \( (j \in N_i) \), while the output is the state-action value \( Q_i \). The distributed state-action value function \( Q_{\theta_i}^{a_i} \) can be updated by minimizing the loss:

\[
\mathcal{L}(\theta) = \mathbb{E}_{\delta_i,a_i,r_i,o_i} [(Q_{\theta_i}^{a_i}(o_i, o_j, a_i, a_j) - y_i)^2] \quad (j \in N_i) 
\]

\[
y_i = r_i + \gamma Q_{\theta_i}^{a_i} (o_i', o_j', a_i', a_j') | a_i' = \mu_{\theta_i}^{a_i}(o_i'), a_j' = \mu_{\theta_j}^{a_j}(o_j') 
\]

where \( \mu_{\theta_i}^{a_i} \) and \( \mu_{\theta_j}^{a_j} \) are the target policy of the i-th agent and its neighbor(s) that take \( \theta_i \) and \( \theta_j \) as delayed parameters, respectively.

Agents are learning in the environment and actually interacting with the environment. When collecting data, the agents capture the latest value function, with a little bit of exploration, and fetch some transitions that will be put into a buffer. The first in, first out (FIFO) queue could be adopted to evict the oldest entry and let new data in, when the buffer fills up. Then, the target parameters will be updated by the current parameters at set intervals.

The computational complexity of the proposed algorithm is \( O(2N( \sum_{l=1}^{L^a} \zeta_{l-1}^a \zeta_{l+1}^a + \sum_{l=1}^{L^c} \zeta_{l-1}^c \zeta_{l+1}^c ) ) \), where \( L^a \) and \( L^c \) stand for the depth of the neural networks in the actor and the critic, respectively. \( \zeta_{l-1}^a \) and \( \zeta_{l-1}^c \) refer to the input dimension and output dimension of the \( l \)-th layer in the actor. \( \zeta_{l-1}^c \) and \( \zeta_{l-1}^c \) represent the input dimension and output dimension of the \( l \)-th layer in the critic.

Ignoring the redundant information of non-neighbor agents in the environment, the distributed training requires less computation cost and a shorter elapsed time. With the framework of distributed training with decentralized execution, the proposed method may work well in scenarios such as collaborative target tracking, a convey of vehicles, defensive navigation, and surveillance of a specific area where each agent is only able to obtain the observation of some agents in the communication range.

4. Experimental Setup and Validation
4.1. Task Description

The proposed algorithm was evaluated through a multi-agent target tracking task. The experiments were performed based on the multi-agent particle environment [38], where the agents can take actions in a two-dimensional space following some basic simulated physics.

In this paper, a scenario where a maneuvering target is tracked by \( N \) tracking agents in two-dimensional space is presented. Agents are unaware of the states of all the other agents in the task. The MAD3PG and compared methods were trained with the settings of \( N = 3 \) and \( N = 5 \), where each tracking agent has an observation of the neighbor agents with the relative positions and velocities. At each time step, the reward of the i-th agent
follows Equation (9). In this tracking scenario, the reward evaluation factors were set as \( k = 0.1 \) and \( m = 0.1 \).

4.2. Implementation Specifications

This work focused on the distributed tracking problem in multi-agent systems, in which each agent receives a private observation. When the discount factor (\( \gamma \)) is closer to zero, the agent pays more attention to the short-term return. However, when \( \gamma \) is closer to one, the longer-term return becomes more important to the agent. The soft target update rate (\( \tau \)) is related to the update of target actors and target critics. For the sake of comparison fairness, we adopted the same hyperparameter settings as those in the MADDPG [38], which are specified in Table 2. The actor and critic have the same neural networks structure of a two-layer rectified linear unit (ReLU) multilayer perception (MLP) with 64 neurons per layer with the Adam [49] optimizer. In our tracking tasks, each episode took 100 steps in training and testing.

| Parameters                        | Value        |
|-----------------------------------|--------------|
| discount factor (\( \gamma \))    | 0.95         |
| soft target update rate (\( \tau \)) | 0.01       |
| capacity of the replay buffer     | \( 10^6 \)   |
| mini-batch size                   | 1024         |
| learning rate                     | 0.01         |
| number of total episodes          | 25,000       |

Table 2. Hyperparameters for learning.

To train the neural networks, we needed a dataset of transitions. Therefore, there was a data collection process that went out and fetched data. The agents are learning in the scenario mentioned above, interacting with each other. Then, they take some actions and obtain the transitions that are supposed to put into experience replay buffer \( D \). When the buffer fills up, another process will use a FIFO queue to evict the previous data. This is an eviction process to ensure that the memory requirements will not grow out of control.

Then, the training process is actually a process of \( Q \)-function regression. This process fetches a mini-batch of data from the buffer uniformly and fetches the parameters of target neural networks to calculate the target values and update the current \( Q \) networks. Then, the chain rule is adopted to update the parameters of current policy networks.

There is another process that updates the parameters of the target neural networks. It is a much slower process that will grab the parameters of current policies and critics, which are supposed to be copied into the parameters of their corresponding target neural networks, with the Polyak averaging method in every single step.

4.3. Experimental Evaluation Results of a System with Three Tracking Agents

This section presents the experimental evaluation results of the proposed MAD3PG algorithm compared with the MADDPG [38], DDPG [48], PPO [13], and DQN [44] mentioned in Section 2 in the scenarios based on the Dec-POMDP and POSG.

Every single agent cannot observe all the other agents, namely not every tracking agent can observe the target in the tracking process. Each tracking agent only has an observation of its neighbor agents. At the initial time of tracking, the neighbor set of each agent is determined from the connection of agents. The value of observation evolves through time, but the connection between agents do not evolve through time. The weighted graph matrices, which present the connection status of the multi-agent system with three tracking agents and a target agent, are represented by \( \mathcal{A} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \), \( \mathcal{B} = \text{diag}\{1, 0, 0\} \).

The numerical comparison of experimental results of the five methods regarding the time cost and the average tracking rewards in 1000 episodes is given in Table 3. The
MAD3PG in the POSG group took the shortest time (5177 s) and obtained the highest reward (−0.013). In addition, Figure 3 displays the experimental results intuitively. What is striking is that the performance of the MAD3PG (five-pointed star) in the POSG group was better than the other compared methods.

**Table 3.** Results of the tracking rewards and the time cost.

| Method   | Dec-POMDP Reward | Dec-POMDP Time (s) | POSG Reward | POSG Time (s) |
|----------|------------------|--------------------|-------------|---------------|
| MAD3PG   | −0.061           | 5532               | −0.013      | 5177          |
| MADDPG   | −0.046           | 6054               | −0.032      | 5692          |
| DDPG     | −0.165           | 5732               | −0.047      | 5381          |
| PPO      | −0.166           | 6283               | −0.166      | 6198          |
| DQN      | −0.125           | 5907               | −0.105      | 5578          |

![Figure 3. Comparisons of the tracking rewards and the time cost.](image)

The bar chart (Figure 4) compares the training time cost in two scenarios with the MAD3PG, MADDPG, DDPG, PPO, and DQN, respectively. It is noticeable that the training time of each method applied to the Dec-POMDP was longer than that applied to POSG. Overall, less training time was taken using the MAD3PG than the other compared methods in both scenarios. With the proposed MAD3PG, the shortest time cost was in the POSG training process.

Figure 5 presents the mean rewards in an episode that averaged over all tracking agents obtained by the proposed MAD3PG and compared methods. At the beginning of the training process, the rewards obtained by the five methods increased alternately and fluctuated greatly. At around the 15,000th episode, the reward of the MAD3PG converged to a relatively stable state with a reward close to zero, and it maintained at this level for the rest of the training process. However, the rewards of the MADDPG, DDPG, PPO, and DQN still fluctuated at the final stages, which were worse than the MAD3PG. Besides, the MAD3PG gained a higher convergence speed than the other compared methods. Overall, it is clear that the MAD3PG showed outstanding performance over the training period. We can see that the MAD3PG converged to a higher mean reward with a higher convergence rate and better stability than the compared methods.
We tested the models of the tracking system at different episodes saved in the training process with the MAD3PG algorithm. The tracking trajectories at the episodes of $n$ ($n \in \{1k, 5k, 10k, 15k, 20k, 25k\}$) are shown in Figure 6. In these figures, red lines represent the trajectories of the target. At the $1k$-th episode, agents do not know how to follow the target. During the $1k$-th episode to $20k$-th episode, agents gradually have the trend of convergence to the target. Finally, the agents can achieve the learning goal. In general, with the increase of the training episodes, tracking agents gradually learn to track the target.
4.4. Experimental Evaluation Results of System with Five Tracking Agents

The basic performance of the proposed algorithm was tested in the scenario with three tracking agents. In order to evaluate the flexibility of proposed algorithm, the models of the MAD3PG and compared methods were also trained in the task where a target agent is tracked by five tracking agents with the partial observation of different agents. We tested the models with four different graphs. The weighted graph matrices of this scenario are
represented by $\mathcal{A}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $\mathcal{A}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, $\mathcal{A}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$, and $\mathcal{B}_0 = \text{diag}\{0, 0, 0, 1\}$.

The mean episode reward ($E(r)$) and variance of the reward ($D(r)$) obtained by the MAD3PG and compared methods are listed in Table 4. Our method (MAD3PG) gained the highest mean episode reward with the smallest variance among the compared methods.

| Method   | $E(r)$        | $D(r)$        |
|----------|---------------|---------------|
| MAD3PG   | $1.17 \times 10^{-2}$ | $2.11 \times 10^{-6}$ |
| MADDPG   | $1.94 \times 10^{-2}$ | $4.23 \times 10^{-5}$ |
| DDPG     | $1.93 \times 10^{-2}$ | $4.63 \times 10^{-6}$ |
| PPO      | $1.53 \times 10^{-1}$ | $1.30 \times 10^{-3}$ |
| DQN      | $1.60 \times 10^{-1}$ | $3.55 \times 10^{-3}$ |

Figure 7 displays the trajectories of five tracking agents and a target agent, which were tested by the model of the MAD3PG. During the tracking process, the five agents approached the target and achieved successful tracking, which evidently illustrated the flexibility and effectiveness of the proposed algorithm.

5. Conclusions

In this paper, a novel distributed training method with decentralized execution was proposed for multi-agent settings by partial observations of neighbor agents. The distributed algorithmic framework of the multi-agent distributed deep deterministic policy gradient (MAD3PG) reduced the interference of the nonstatic environment in multi-agent systems to the critic, which outperformed MADDPG, DDPG, PPO, and DQN in the tested tracking environment. With a shorter training time and better performance, the experimental results demonstrated that this presented algorithm is efficient and effective. In future work, it will be worth extending it to multi-agent systems with switching topologies since
the information flow connection between neighbor agents may be broken or fail with an uncertain probability in practical target tracking applications.

**Author Contributions:** Conceptualization, D.F. and L.D.; methodology, D.F. and L.D.; software, D.F.; validation, D.F.; formal analysis, H.S.; investigation, D.F. and L.D.; resources, H.S.; data curation, D.F.; writing—original draft preparation, D.F.; writing—review and editing, L.D.; visualization, D.F.; supervision, H.S.; project administration, H.S.; funding acquisition, H.S., and L.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National Natural Science Foundation of China under Grant 61903022 and funded by the Beijing Advanced Innovation Center for Intelligent Robots and Systems under Grant 2019IRS11.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; nor in the decision to publish the results.

**Abbreviations**

The following abbreviations are used in this manuscript:

- **MAss** Multi-agent systems
- **RL** Reinforcement learning
- **DRL** Deep reinforcement learning
- **MADRL** Multi-agent deep reinforcement learning
- **CLEAN** Coordinated learning without exploratory action noise
- **IQL** Independent Q-learning
- **VDN** Value decomposition networks
- **QMIX** Monotonic value function factorization
- **QTRAN** Learning to factorize with transformation
- **MADDPG** Multi-agent deep deterministic policy gradient
- **DDPG** Deep deterministic policy gradient
- **Dec-POMDP** Decentralized partially observable Markov decision process
- **POSG** Partially observable stochastic game
- **MPE** Multi-agent particle environment
- **MAD3PG** Multi-agent distributed deep deterministic policy gradient
- **MDP** Markov decision process
- **POMDP** Partially observable Markov decision process
- **Dec-POMDP** Decentralized POMDP
- **PPO** Proximal policy optimization
- **CTDE** Centralized training and decentralized execution
- **ReLU** Rectified linear unit
- **MLP** Multilayer perceptron
- **FIFO** First in, first out
- **DQN** Deep Q-learning

**References**

1. Shoham, Y.; Leyton-Brown, K. *Multiagent Systems—Algorithmic, Game-Theoretic, and Logical Foundations*; Cambridge University Press: New York, NY, USA, 2009.
2. Mahmoud, M.S. *Multiagent Systems: Introduction and Coordination Control*; CRC Press: Boca Raton, FL, USA, 2020.
3. Hasan, Y.A.; Garg, A.; Sugaya, S.; Tapia, L. Defensive Escort Teams for Navigation in Crowds via Multi-Agent Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* 2020, 5, 5645–5652. [CrossRef]
4. Kappel, K.S.; Cabreira, T.M.; Marins, J.L.; de Brisolara, L.B.; Ferreira, P.R. Strategies for Patrolling Missions with Multiple UAVs. *J. Intell. Robot. Syst.* 2020, 99, 499–515. [CrossRef]
5. Wang, Y.; Wu, Y.; Shen, Y. Cooperative Tracking by Multi-Agent Systems Using Signals of Opportunity. *IEEE Trans. Commun.* 2020, 68, 93–105. [CrossRef]

6. Yu, X.; Andersson, S.B.; Zhou, N.; Cassandras, C.G. Scheduling Multiple Agents in a Persistent Monitoring Task Using Reachability Analysis. *IEEE Trans. Auton. Control* 2020, 65, 1499–1513. [CrossRef]

7. Busoniu, L.; Babuska, R.; Schutter, B.D. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.*) 2008, 38, 156–172. [CrossRef]

8. Busoniu, L.; Babuška, R.; De Schutter, B. Multi-agent Reinforcement Learning: An Overview. In *Innovations in Multi-Agent Systems and Applications—1*; Srinivasan, D., Jain, L.C., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 183–221.

9. Schwartz, H.M. *Multi-Agent Machine Learning: A Reinforcement Approach*; John Wiley & Sons: Hoboken, NJ, USA, 2014.

10. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: London, UK, 2018.

11. Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, L.; Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* 2016, 529, 484–489. [CrossRef] [PubMed]

12. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of Go without human knowledge. *Nature* 2017, 550, 354–359. [CrossRef]

13. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. 2018. Available online: https://arxiv.org/pdf/1707.06347v2 (accessed on 13 August 2018).

14. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An Introduction to Deep Reinforcement Learning. *Found. Trends® Mach. Learn.* 2018, 11, 219–354. [CrossRef]

15. Weiss, G. *Multiagent Systems*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2013.

16. Lanctot, M.; Zambaldi, V.F.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Póropical, J.; Silver, D.; Graepel, T. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 4190–4203.

17. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A Survey and Critique of Multiagent Deep Reinforcement Learning. *Auton. Agents-Multi-Agent Syst.* 2019, 33, 750–797. [CrossRef]

18. Zhang, K.; Yang, Z.; Başar, T. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. In *Handbook of Reinforcement Learning and Control*; Vamvoudakis, K.G., Wan, Y., Lewis, F.L., Cansever, D., Eds.; Springer: Cham, Switzerland, 2021; pp. 321–384.

19. Gronauer, S.; Diepold, K. Multi-agent Deep Reinforcement Learning: A Survey. *Artif. Intell. Rev.* 2021, 1–49. [CrossRef]

20. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Trans. Cybern.* 2020, 50, 3826–3839. [CrossRef] [PubMed]

21. Shoham, Y.; Powers, R.; Grenager, T. If multi-agent learning is the answer, what is the question? *Artif. Intell.* 2007, 171, 365–377. [CrossRef]

22. Allbrecht, S.V.; Stone, P. Autonomous Agents Modelling Other Agents: A Comprehensive Survey and Open Problems. *Artif. Intell.* 2018, 258, 66–95. [CrossRef]

23. Hung, S.M.; Givigi, S. A Q-Learning Approach to Flocking With UAVs in a Stochastic Environment. *IEEE Trans. Cybern.* 2016, 47, 186–197. [CrossRef] [PubMed]

24. Leibo, J.Z.; Zambaldi, V.F.; Lanctot, M.; Marecki, J.; Graepel, T. Multi-agent Reinforcement Learning in Sequential Social Dilemmas. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2017), São Paulo, Brazil, 8–12 May 2017; pp. 464–473.

25. Hong, Z.; Su, S.; Shann, T.; Chang, Y.; Lee, C. A Deep Policy Inference Q-Network for Multi-agent Systems. In Proceedings of the 17th International Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2018), Richland, SC, USA, 10–15 July 2018; pp. 1388–1396.

26. Prasad, A.; Dusparic, I. Multi-agent Deep Reinforcement Learning for Zero Energy Communities. In *2019 IEEE PES Innovative Smart Grid Technologies Europe, ISGT-Europe*; IEEE: Bucharest, Romania, 2019; pp. 1–5.

27. Liang, L.; Ye, H.; Li, G.Y. Spectrum Sharing in Vehicular Networks Based on Multi-agent Reinforcement Learning. *IEEE Trans. Autom. Control* 2020, 65, 484–489. [CrossRef] [PubMed]

28. Jaderberg, M.; Czarnecki, W.; Dunning, I.; Marris, L.; Lever, G.; Castañeda, A.; Beattie, C.; Rabinowitz, N.; Morcos, A.; Ruderman, A.; et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* 2019, 364, 859–865. [CrossRef]

29. Menda, K.; Chen, Y.; Grana, J.; Bono, J.W.; Tracey, B.D.; Kochenderfer, M.J.; Wolpert, D. Deep Reinforcement Learning for Event-Driven Multi-Agent Decision Processes. *IEEE Trans. Intell. Transp. Syst.* 2019, 20, 1259–1268. [CrossRef]

30. Wu, F.; Zhang, H.; Wu, J.; Song, L. Cellular UAV-to-Device Communications: Trajectory Design and Mode Selection by Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Commun.* 2020, 68, 4175–4189. [CrossRef]

31. Agogino, A.; Turner, K. Multi-Agent Reward Analysis for Learning in Noisy Domains. In Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, The Netherlands, 25–29 July 2005; pp. 81–88.
32. Holmes-Parker, C.; Taylor, M.E.; Agogino, A.K.; Tumer, K. CLEANing the Reward: Counterfactual Actions to Remove Exploratory Action Noise in Multiagent Learning. In Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), Paris, France, 5–9 May 2014; pp. 1353–1354.

33. Foerster, J.N.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual Multi-Agent Policy Gradients. In Proceedings of 32nd AAAI Conference on Artificial Intelligence/30th Innovative Applications of Artificial Intelligence Conference/8th AAAI Symposium on Educational Advances in Artificial Intelligence (AAAI 2018), New Orleans, LA, USA, 2–7 February 2018; pp. 2974–2982.

34. Tan, M. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In Proceedings of the 10th International Conference of Machine Learning (ICML 1993), Amherst, MA, USA, 27–29 June 1993; pp. 330–337.

35. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.F.; Jaderberg, M.; Lancot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-Decomposition Networks for Cooperative Multi-Agent Learning Based on Team Reward. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018), Stockholm, Sweden, 10–15 July 2018; pp. 2085–2087.

36. Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; Whiteson, S. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In Proceedings of the 35th International Conference on Machine Learning (ICML 2018), Stockholm, Sweden, 10–15 July 2018; pp. 4295–4304.

37. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.; Yi, Y. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In Proceedings of the 36th International Conference on Machine Learning (ICML 2019), Long Beach, CA, USA, 9–15 June 2019; pp. 5887–5896.

38. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 6379–6390.

39. Bernstein, D.S.; Givan, R.; Immerman, N.; Zilberstein, S. The Complexity of Decentralized Control of Markov Decision Processes. Math. Oper. Res. 2002, 27, 819–840. [CrossRef]

40. Oliehoek, F.A.; Amato, C. A Concise Introduction to Decentralized POMDPs, 1st ed.; Springer: Berlin/Heidelberg, Germany, 2016.

41. Hansen, E.A.; Bernstein, D.S.; Zilberstein, S. Dynamic Programming for Partially Observable Stochastic Games. In Proceedings of the 19th National Conference on Artificial Intelligence/16th Conference on Innovative Applications of Artificial Intelligence, San Jose, CA, USA, 25–29 July 2004; pp. 709–715.

42. Dong, L.; Chai, S.; Zhang, B.; Nguang, S.K.; Savvaris, A. Stability of a Class of Multiagent Tracking Systems With Unstable Subsystems. IEEE Trans. Cybern. 2017, 47, 2193–2202. [CrossRef]

43. Doshi, P.; Gmytrasiewicz, P. A Framework for Sequential Planning in Multi-Agent Settings. J. Artif. Intell. Res. 2005, 24, 49–79.

44. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.A.; Fidjeland, A.; Ostrovski, G.; et al. Human-level Control through Deep Reinforcement Learning. Nature 2015, 518, 529–533. [CrossRef] [PubMed]

45. Foerster, J.N.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P.H.S.; Kohli, P.; Whiteson, S. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Sydney, NSW, Australia; 6–11 August 2017; pp. 1146–1155.

46. Omidshafiei, S.; Pazis, J.; Amato, C.; How, J.P.; Vian, J. Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability. In Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Sydney, NSW, Australia; 6–11 August 2017; pp. 2681–2690.

47. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M.A. Deterministic Policy Gradient Algorithms. In Proceedings of the 31th International Conference on Machine Learning (ICML 2014), Beijing, China; 21–26 June 2014; pp. 387–395.

48. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. In Proceedings of the 4th International Conference on Learning Representations (ICLR 2016), San Juan, PR, USA, 2–4 May 2016.

49. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.