drones

MDPI

Article

Deep Reinforcement Learning Based Computation Offloading in UAV-Assisted Edge Computing

Peiying Zhang [1,2], Yu Su [1], Boxiao Li [3,4], Lei Liu [2,5,*], Cong Wang [6], Wei Zhang [7,*] and Lizhuang Tan [7]

1 Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China; zhangpeiying@upc.edu.cn (P.Z.)
2 State Key Laboratory of Integrated Services Networks, Xidian University, Xi' an 710071, China
3 Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
4 China Academy of Electronics and Information Technology, Beijing 100041, China
5 Xidian Guangzhou Institute of Technology, Xidian University, Guangzhou 510555, China
6 School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, Qinhuangdao 066004, China
7 Shandong Provincial Key Laboratory of Computer Networks, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250013, China
* Correspondence: leiliu@xidian.edu.cn (L.L.); wzhang@sdas.org (W.Z.)

Abstract: Traditional multi-access edge computing (MEC) often has difficulty processing large amounts of data in the face of high computationally intensive tasks, so it needs to offload policies to offload computation tasks to adjacent edge servers. The computation offloading problem is a mixed integer programming non-convex problem, and it is difficult to have a good solution. Meanwihle, the cost of deploying servers is often high when providing edge computing services in remote areas or some complex terrains. In this paper, the unmanned aerial vehicle (UAV) is introduced into the multi-access edge computing network, and a computation offloading method based on deep reinforcement learning in UAV-assisted multi-access edge computing network (DRCOM) is proposed. We use the UAV as the space base station of MEC, and it transforms computation task offloading problems of MEC into two sub-problems: find the optimal solution of whether each user's device is offloaded through deep reinforcement learning; allocate resources. We compared our algorithm with other three offloading methods, i.e., LC, CO, and LRA. The maximum computation rate of our algorithm DRCOM is 142.38% higher than LC, 50.37% higher than CO, and 12.44% higher than LRA. The experimental results demonstrate that DRCOM greatly improves the computation rate.

Keywords: multi-access edge computing; deep reinforcement learning; computation offloading

check for updates

Citation: Zhang, P.; Su, Y.; Li, B.; Liu, L.; Wang, C.; Zhang, W.; Tan, L. Deep Reinforcement Learning Based Computation Offloading in UAV-Assisted Edge Computing. *Drones* **2023**, *7*, 213. https://doi.org/10.3390/drones7030213

Academic Editor: Shiva Raj Pokhrel

Received: 24 February 2023
Revised: 13 March 2023
Accepted: 17 March 2023
Published: 19 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the rapid development of mobile communication technology, the number of Internet users has exploded, and people have a higher pursuit: everything is connected, which is the Internet of Things (IoT) [1]. The IoT is essentially an extension on the basis of the Internet that combines various information sensing devices with the network to form a huge network to realize the interconnection of people, machines, and things at any time and any place. Nowadays, IoT has been applied to intelligent transportation, smart homes, and other fields [2]. However, as users have higher and higher requirements for IoT devices, the computing power of some IoT devices have difficulty processing a large amount of data. At the same time, there are also hidden dangers in real-time, energy consumption, and data security. In order to solve these problems, mobile edge computing comes into being.

In 2017, mobile edge computing officially changed its name to multi-access edge computing (MEC) [3]. It can deploy tasks that require high-density computing, large traffic, and low latency nearby to meet customers' multiple requirements for security, speed, and

reliability [4]. Additionally, it can generally be regarded as a cloud server running at the edge of the mobile network and running a specific task. Since the MEC is very close to the user or information source geographically, the delay of the network responding to the user request is greatly reduced, and the possibility of network congestion in the transmission network and core network is also reduced. In traditional multi-access edge computing, servers are mostly deployed in ground base stations [5]. In some remote areas with complex landforms, the cost of deploying servers is often higher than the revenue. When we carry out wilderness rescue and geological exploration, it is hard to apply MEC technology.

The cost of deploying servers is often high when providing edge computing services in remote areas or some complex terrains. An unmanned aerial vehicle (UAV) is a kind of unmanned aerial vehicle controlled by radio remote control equipment and program control device. In the present era, it has been applied to highway surveillance, post-disaster search and rescue, military, and other fields [6]. As shown in Figure 1, because of its small size, low cost, ease of use, and high mobility, it can be applied to MEC as an aerial base station, which can receive computationally intensive tasks offloaded from user devices. In this paper, we consider that the offloading policy of user devices is binary, either computing locally or offloading computing tasks to adjacent edge servers [7]. In general, the quality of service of the system depends, to a certain extent, on the optimal offloading policy. Due to its binary characteristics, this kind of problem is usually expressed as a mixed integer programming (MIP) problem, which is an NP-hard problem. It is generally solved by the branch and bound method [8], cut plane method [9], and heuristic algorithm [10]. The first two methods have extremely high computational complexity in the face of large-scale MEC networks, and the results obtained by the heuristic algorithm are sometimes unsatisfactory. In order to solve these problems, scholars have introduced deep reinforcement learning into MEC to make offloading policies in recent years. The contributions of this paper are as follows:

- UAV is introduced into traditional multi-access edge computing, due to the characteristics of being easy to use and having low cost. We can use it as a server to provide edge computing services in remote areas or some complex terrains.
- A computation offloading method based on deep reinforcement learning in UAV-assisted multi-access edge computing network (DRCOM) is proposed. In order to maximize the computing rate, deep reinforcement learning is used to find the optimal solution of whether each user device performs computation offloading.
- We transform the computation offloading problem into two sub-problems: computing offloading policy problem and resource allocation problem. It transforms a MIP problem into a complex problem, which reduces computational complexity.

The rest paper is structured as follows: related work is introduced in Section 2, including multi-access edge computing and UAV-assisted multi-access edge computing. Our model and problem description is introduced in Section 3. We first introduce local computing and computation offloading and then propose the optimization objective. In Section 4, we introduce the overall deep reinforcement learning model, followed by the generation and updating of offloading action. Additionally, we carry out the experiment and analyze the experimental results in Section 5. In Section 6, we summarize the conclusions and propose directions for future work.

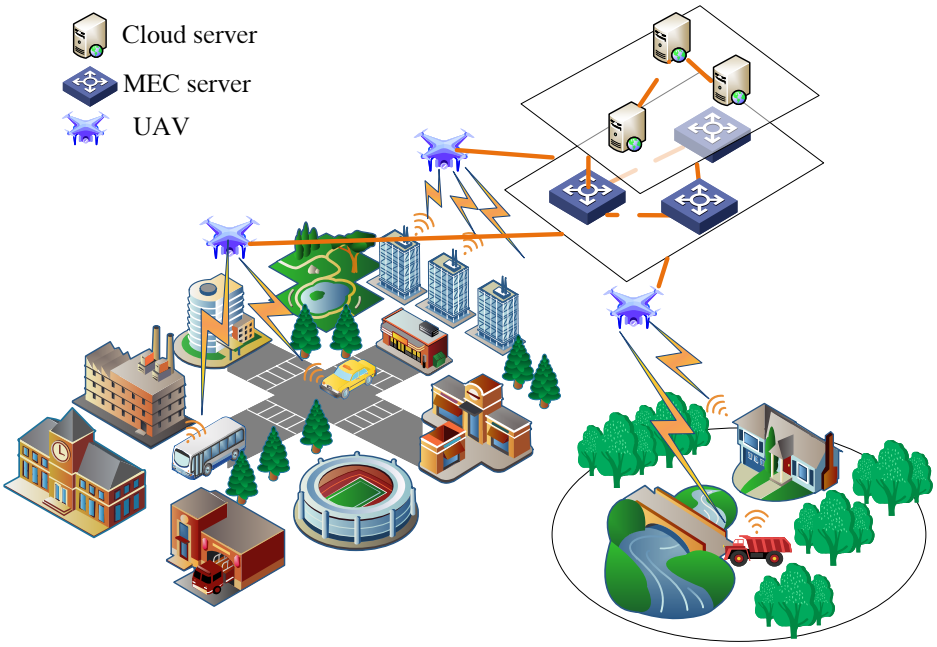

Figure 1. UAV-assisted multi-access edge computing model.

2. Related Works

In this section, the research related to multi-access edge computing and UAV-assisted mobile edge computing is introduced.

Multi-access edge computing is at the intersection of the wireless network edge and the infrastructure edge, where the mobile network and the Internet meet and deliver data traffic [11]. It shares the same goal as edge computing: offloading the computing power closer to where data is generated and decisions are made for faster, more efficient responses. By placing traditional digital infrastructures close to mobile networks, MEC enables telecom operators to provide substantial improvements in performance and latency reduction for mobile communications, games, video streaming, and the Internet of Things [12]. All of these are achieved through the combination of wireless networks and traditional Internet infrastructure. MEC technologies mainly include four types: server placement technology, resource allocation, computation offloading technology, and mobility management technology [13]. The server placement technology mainly studies how to select the best placement position of the server to make the system performance optimal. The resource allocation and computation offloading technology mainly focuses on the joint management of the system's computing resources, storage resources, and communication resources and makes offloading decisions to achieve the expected goal. In the aspect of mobility management technology, more consideration is given to the mobility of the user device and whether to perform task offloading when the user device moving in different cells [14].

This paper mainly focuses on the computation offloading problem in MEC, which is usually formulated as a convex optimization problem in most research. Ren et al. [15] focused on the problem of minimizing the delay in the mobile-edge computation offloading system. For the partial compression offloading model, they transformed the original problem into a piecewise convex problem, and proposed an optimal resource allocation scheme based on the subgradient algorithm. Chang et al. [16] proposed a joint computation offloading and radio and computation resource allocation algorithm based on Lyapunov optimization. By minimizing the derived upper bound of the Lyapunov drift plus penalty function, the main problem was divided into several sub-problems, which were addressed separately in pursuit of the minimization of the energy consumption of the system under consideration for each time slot. Zhao et al. [17] proposed a privacy-preserving computation offloading method based on privacy entropy. First, privacy entropy was proposed as a quantitative analysis metric because of the task offloading frequency characteristics of user

computation tasks. Then, privacy constraints were introduced into the existing offloading decision model, and a privacy-preserving offloading decision model was established. Finally, they solved the offloading decision, which satisfied the privacy constraints and optimal energy consumption goals based on the genetic algorithm, used the optimal decision to train the artificial neural network parameters. Additionally, the trained neural network was used to adjust the impact of privacy, energy consumption, and delay constraints.

Jeong et al. [18] studied a UAV-based mobile cloud computing system that deployed computation offload capability for UAV. For the uplink and downlink transmissions required for the offloading procedure, two types of access schemes are considered, namely orthogonal access and non-orthogonal access. They introduced successive convex approximation strategies to solve the problem of jointly optimizing the bit allocation for uplink and downlink communications, as well as the computation at the UAV, which achieves total mobile energy consumption minimization. Kim et al. [19] proposed an optimal task-UAV-mobile edge server matching algorithm based on the Hungarian algorithm, which took into account the energy consumption and processing, latency times in the mobile edge server, and the location of the UAVs, tasks, and mobile edge server. In order to reduce the energy consumption of the UAVs when they were moving, they not only considered the current position of the UAVs, but also the position the UAVs returned. Zhang et al. [20] proposed a new optimization problem formulation, which aims to minimize the total energy consumption by optimizing bit allocation, slot scheduling, power allocation, and UAV trajectory design. The energy included communication-related energy, computing-related energy, and drone flight energy. Since the formulation problem was non-convex, it was difficult to find the optimal solution. They solved the problem in two parts and obtained an approximate optimal solution by the Lagrangian dual method and the successive convex approximation technique, respectively. Liu et al. [21] proposed an optimization problem to jointly optimize the CPU frequencies, the offloading amount, the transmit power, and the UAV's trajectory, in order to minimize the total energy required for UAVs in UAV-assisted MEC systems. Since the problem was non-convex, they used an algorithm based on successive convex approximation to solve the problem. However, it had high computational complexity, so an algorithm based on decomposition and iteration was proposed.

3. Problem Description and Modeling

In this paper, we focus on the problem of computation task offloading policies for UAV-assisted MEC. We need to pay attention to the fact that the UAV is equipped with a server, and our research content is the MEC assisted by a single UAV. In fact, there is more than one user device connected to a UAV in an area. We use a set U to represent the set of user devices: $U = \{1, 2, ..., N\}$, where the user devices are all connected to the UAV through the wireless network. The UAV can broadcast radio frequency energy to user devices, each of which is powered and can store the received energy. The wireless power transmission (WPT) and computation offloading are carried out in the same frequency band. In order to avoid mutual interference between the WPT and computation offloading, all circuits use time-division multiplexing circuits. Assuming that the computation power of the UAV is higher than that of the connected user devices, the user devices can offload its computation tasks to the UAV facing with computationally intensive tasks. We represent the user device's offloading policy as a binary policy, either offloading to the UAV or performing local computations. We use the symbol f_i to represent the policy of the i^{th} user device, where $i \in N$. If $f_i = 1$, it means that the computation task is offloaded to the UAV. If $f_i = 0$, the computation task of the user devices is computed locally. The system time is divided into continuous times of equal length T. We will introduce the whole process from two aspects: local computing and computation offloading.

3.1. Local Computing

The user devices may choose to perform local computing, which can harvest energy and perform computation tasks concurrently [22]. The energy harvested by the i^{th} user device E_i can be given by:

$$E_i = eh_i PxT, \tag{1}$$

where e is the efficiency of energy harvesting and $e \in (0, 1)$. P is the transmit power of the UAV. h_i is the wireless channel gain between the UAV and the i^{th} device within T time. xT is the length of time for WPT within T time, and $x \in (0, 1)$ [23].

The energy consumption of the user device during T time is kc^3T, where k is the energy efficiency coefficient [24] and c refers to the computing speed of the processor. In general, it should be less than or equal to the harvested energy, so we have:

$$kc^3T \leq E_i. \tag{2}$$

In order to maximize the local computing power of the user devices, we assume that they used up all the harvested energy. So we have:

$$kc^3T = E_i, \tag{3}$$

so we can deduce $c = (\frac{E_i}{kT})^{\frac{1}{3}}$.

Thus, the local computation rate $R_{L,i}(x)$ can be given by:

$$\begin{aligned} R_{L,i}(x) &= \frac{ct_i}{\phi T} \\ &= \frac{(\frac{E_i}{kT})^{\frac{1}{3}} T}{\phi T} \\ &= \frac{(E_i)^{\frac{1}{3}}}{k^{\frac{1}{3}} T^{\frac{1}{3}} \phi} \\ &= \frac{(eP)^{\frac{1}{3}}}{\phi} (\frac{h_i}{k})^{\frac{1}{3}} x^{\frac{1}{3}} \\ &= \mu (\frac{h_i}{k})^{\frac{1}{3}} x^{\frac{1}{3}}, \end{aligned} \tag{4}$$

where $\mu = \frac{(eP)^{\frac{1}{3}}}{\phi}$, and $\frac{ct_i}{\phi}$ refers to the number of bits processed by the user device. What we consider is that the user device makes full use of the T time period, so $t_i = T$.

3.2. Computation Offloading

We define the time taken for i^{th} user device computation offloading as $y_i T$, where $y_i \in [0, 1]$ and $i \in N$. Due to the limitation of its circuit, the user devices can only offload the task to the UAV after harvesting energy. Assuming that the computing speed and transmission power of the UAV are much greater than that of the user device [25], the time of the UAV on computation offloading and downloading can be ignored, and each time frame is only occupied by WPT and computation offloading. So we have:

$$\sum_{i=1}^{N} y_i + x \leq 1. \tag{5}$$

In order to maximize the computing speed, the user devices will try to use up the energy that is harvested during offloading, so the offloading power of the user devices p_i is denoted as:

$$p_i = \frac{E_i}{y_i T}. \tag{6}$$

Accordingly, the computing rate of the user device is equal to its data offloading capability, so we have:

$$R_{O,i}(x, y_i) = \frac{By_i}{v_u} \log_2(1 + \frac{ePx(h_i)^2}{y_i P_N}), \tag{7}$$

where $R_{O,i}(x, y_i)$ is the data offloading capability, B is the bandwidth of the link, and P_N is the noise power [26].

3.3. Optimization Objective

We assume that only h_i among the previously mentioned parameters changes over time, and the other parameters are fixed. Therefore, the weighted sum computation rate of MEC in T time frame is expressed as:

$$C(h, f, y, x) = \sum_{i=1}^{N} w_i((1 - f_i)R_{L,i}(x) + f_i R_{O,i}(x, y_i)), \tag{8}$$

where w_i denotes the weight assigned to the computing rate of the i^{th} device.

During the task offloading process, a faster computation rate means the higher the efficiency. Therefore, the optimization objective OPT is expressed as follows:

$$OPT(h) = max\ C(h, f, y, x) \tag{9}$$

$$\text{subject to } \sum_{i=1}^{N} y_i + x \leq 1, \tag{10}$$

$$x \geq 0, \tag{11}$$

$$y_i \geq 0, \tag{12}$$

$$f_i \in \{0, 1\}, \tag{13}$$

$$\forall i \in N. \tag{14}$$

The optimization problem is a mixed integer programming non-convex problem, which is NP-hard. However, once f_i is given, OPT can be transformed into a convex problem, as follows:

$$OPT(h, f) = max\ C(h, f, y, x) \tag{15}$$

$$\text{subject to } \sum_{i=1}^{N} y_i + x \leq 1, \tag{16}$$

$$x \geq 0, \tag{17}$$

$$y_i \geq 0, \tag{18}$$

$$\forall i \in N. \tag{19}$$

Therefore, we can divide $OPT(h)$ into two sub-problems: computation offloading policy problem and resource allocation problem:

- *Computation offloading policy*. Each user device has two options: local computing or offloading the computation task to the UAV, so there are 2^N cases. In this paper, we use the deep reinforcement learning algorithm to decide the computation offloading policy, which can achieve better results in a shorter time.
- *Resource Allocation*. After solving the computation offloading policy problem, f_i is already a fixed value, and the problem is transformed into a convex problem, which can be solved efficiently. Table A1 summarizes the notations commonly used in this section.

4. Deep Reinforcement Learning Based Computation Offloading in UAV-Assisted MEC

In this section, we describe the computation offloading algorithm based on deep reinforcement learning in UAV-assisted MEC in detail. Our optimization objective is to maximize the weighted sum computation rate of MEC. From the above derivation, we divide the optimization objective into two sub-problems. First, deep reinforcement learning is used to solve the computation offloading decision-making problem of user devices and determine f_i, and then we allocate resource to determine the time spent on WPT and computation offloading in each time frame.

4.1. Offloading Action Generation

As shown in Figure 2, we observe the channel gain at time t. When $t = 0$, the weights in the deep neural network are randomly initialized according to the normal distribution with mean 0. Due to the approximation principle of neural network, ReLu, whose output is expressed as $f(x) = max(0, x)$, is used here as the activation function of hidden layer [27]. Sigmod activation function is used in output layer, whose output is expressed as $S(x) = \frac{1}{1+e^{-x}}$, and it can make offloading action between (0,1) [28]. Then, we perform decision concretization on the trained v_t, mapping it into a binary decision:

$$f_t = \begin{cases} 1, \ v_t > 0.5, \\ 0, \ v_t \leq 0.5. \end{cases} \tag{20}$$

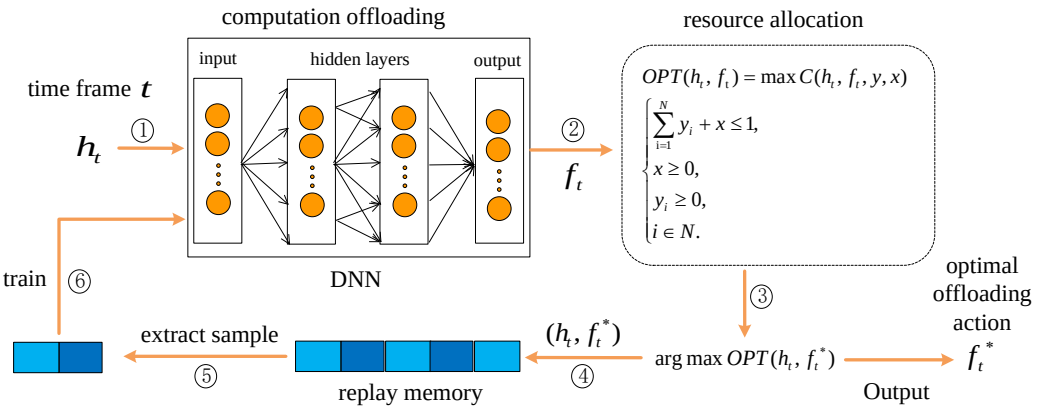

Figure 2. Deep reinforcement learning model.

We bring the obtained offloading policy into the $OPT(h, f)$ problem, solve the complex problem, and find its optimal value. The offloading action f_t^* at this time is expressed as:

$$f_t^* = argmax \ OPT(h_t, f_t). \tag{21}$$

4.2. Deep Reinforcement Learning

Deep reinforcement learning combines deep learning and reinforcement learning to demonstrate powerful learning capabilities in the field of artificial intelligence, which is mainly used to solve high-dimensional spatial tasks [29]. It mainly consists of five elements: agent, environment, reward, state, and action. The agent, which is generally composed of a neural network, performs new actions in the environment after training. The environment will feed back rewards to the agent, and the agent will update its state accordingly [30]. As shown in Figure 2, our computation offloading policy employs deep reinforcement learning to make decisions.

The algorithm mainly consists of two processes: offloading action generation and offloading action update. The generation of offloading action depends on the deep neural network. At the time point t, the deep neural network takes the channel gain h_t as input, obtains the current offloading policy, and outputs the offloading action v_t after training.

After that, we quantify v_t relaxation action into binary offloading actions f_t, bring it into the problem $OPT(h_t, f_t)$, and select an optimal offloading action in the optimization problem. We use $argmax(h_t, {f_t}^*)$ to represent this step. MEC performs the offloading action ${f_t}^*$ to get the reward $OPT(h_t, {f_t}^*)$ and add the newly acquired state action $(h_t, {f_t}^*)$ to the replay memory. Then, a batch of training samples is extracted from the replay memory to train the deep neural networks (DNN) in the policy update phase of the t time frame [31], and the DNN updates the parameters accordingly. We use the new offloading policy in the next time frame to generate new offloading decisions based on the new channel h_{t+1} observed. We iterate the model repeatedly, and the policy of the DNN is gradually improved.

4.3. Offloading Action Updates

We use the replay memory technology to train the deep neural network, which reduces the correlation of samples and greatly improves the performance and stability of network training. At time t, we put the obtained state action $(h_t, {f_t}^*)$ into the replay memory. Since the memory is limited, the newly generated sample data will replace the oldest data sample when the memory is full. We define ξ as the training interval. Every ξ time frames, we randomly select a batch of training data samples from the memory for training. The Adam algorithm [32] is used to optimize the parameters (weights of neurons in the hidden layer) in the deep neural network. The pseudo code of the DRCOM algorithm is shown in Algorithm 1.

Algorithm 1 Deep reinforcement learning training process

Input: Wireless channel gain h_t;
Output: Offloading policy ${f_t}^*$, optimal resource allocation at time t;

```
Initialize the parameters of the DNN, set the number of iterations K and training interval ξ, empty memory;
for t = 1, 2, ..., K do
    Randomly generate an offloading strategy v_0 according to a normal distribution with a mean of 0;
    Generate binary offloading decision f_t by (20);
    Put f_t into OPT(h), calculate OPT(h_t, f_t) by (15);
    Find the optimal allocation and output f_t*;
    Add (h_t, f_t*) to memory;
    if K mod ξ = 0 then
        Uniformly sample (h_t, f_t*) from the memory;
        Train the DNN and update the parameters with the Adam algorithm;
    end if
end for
return the candidate substrate nodes;
```

5. Performance Evaluation

In this section, we will introduce the details of our simulation experiments to demonstrate the performance of the proposed algorithm.

5.1. Simulation Environment and Parameters

All experiments were performed on Core i5-8300H 2.3 GHz CPU and 16GB computers. In this experiment, we use Powercast TX91501-3W as an energy emitter to simulate the user device and set the energy harvesting efficiency to 0.51. The distance from the UAV to the user device is represented by a uniform distribution. The average channel gain follows the free-space path loss $h = A_d(\frac{3 \cdot 10^8}{4\pi f_c d_i})^{d_e}$, where $A_d = 4.11$ is the antenna gain, $f_c = 915$ MHZ is the carrier frequency, and $d_e = 2.8$ is the path loss exponent. It is assumed that the channel gain remains constant within one time frame and changes independently between different time frames. Our deep neural network consists of an input layer, two hidden layers, and an output layer.

We implement our algorithm with tensorflow, setting the number of user devices to 30, the training batch size to 128, memory size to 1024, and the learning rate and training interval of the Adam optimizer to 0.001 and 5 respectively. Here, we explain that all the parameters are set according to the experiments we conducted. In order to make the article not too long, we did not describe the experiments about memory size and training batch size in detail in the article. We explained in Section 5.2 why the learning rate and training interval are set to 0.001 and 5, respectively. There are a total of 30,000 sets of data, which were split into separate training and testing datasets with a ratio of 8:2.

5.2. Experimental Results and Analysis

In order to see the training situation of the computation rate more clearly, we define the normalized calculation rate $R \in [0, 1]$ as follows:

$$R = \frac{OPT(h_t, f_t)}{max_{f_t \in \{0,1\}^N} OPT(h_t, f_t^*)}. \tag{22}$$

The training time is generally between 2 and 3 minutes. As shown in Figure 3, the training loss gradually decreases and stabilizes at 0.05 as time goes by. With the continuous training of our deep neural network, the training loss decreases, which also means that our model training effect is getting better and better. At the same time, it means our algorithm automatically updates its offloading policy and converges to the optimal solution.

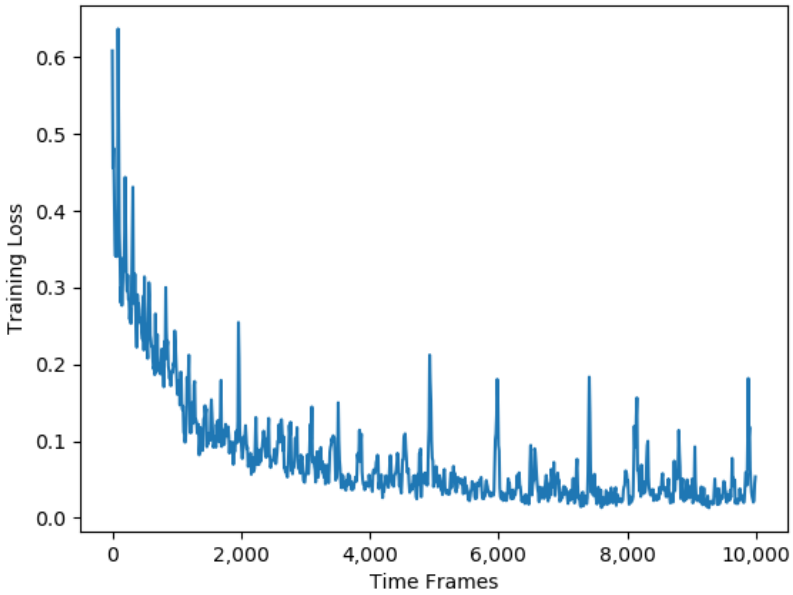

Figure 3. Training losses for DRCOM algorithm.

The normalized computation rate R in the training process is shown in Figure 4, and the light blue part in the background is the maximum and minimum normalized computation rate at this time point. It can be seen from Figure 4 that the normalized computation rate gradually rises and converges to 1 when time frame is large.

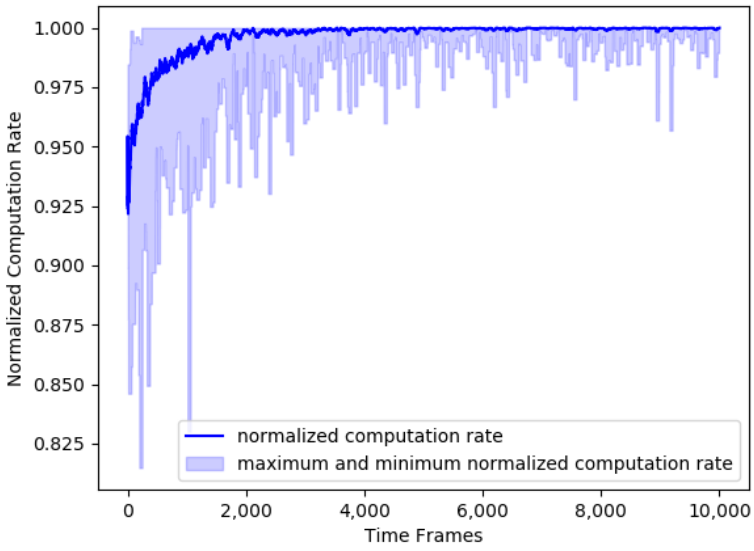

Figure 4. Normalized computation rates for DRCOM algorithm.

We further studied the effect of different algorithm parameters for the Adam optimizer on the experimental results, including the learning rate and training intervals. We set the number of iterations to 10,000 and calculate the average of the normalized computation rate every 400 times. First, we set the learning rates to 0.1, 0.01, 0.001, and 0.0001, respectively. As shown in the Figure 5, the curve oscillates and then stabilizes. We can see that the curve with the learning rate of 0.001 is more stable and becomes stable earlier than the other curves. Therefore, our experiment used an Adam optimizer with the learning rate of 0.001.

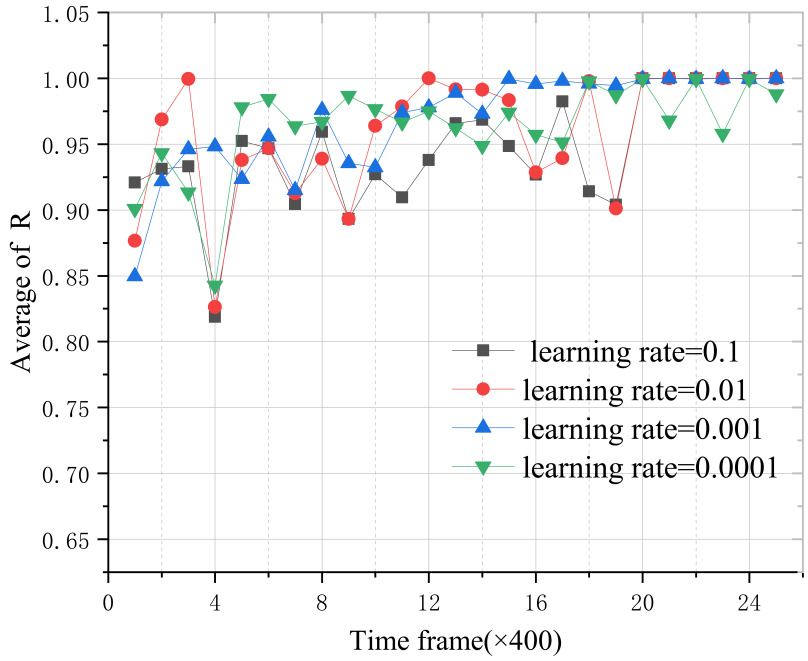

Figure 5. Average of *R* at different learning rates.

As shown in the Figure 6, we study the effect of different training intervals on the experimental results. We set the training intervals to 5, 10, 50, and 100, respectively. The experimental results show that the average of the normalized calculation rate increases

gradually and then tends to 1 gradually. When the training interval is 5, the rate tends to 1 faster and is more stable, while the average of the normalized computation rate is not stable when the training interval is 10 and 50. When the training interval is 100, the curve tends to 1 at a slower rate. Therefore, we set the training interval in the experiment to 5, in order to accelerate our algorithm to converge better and faster.

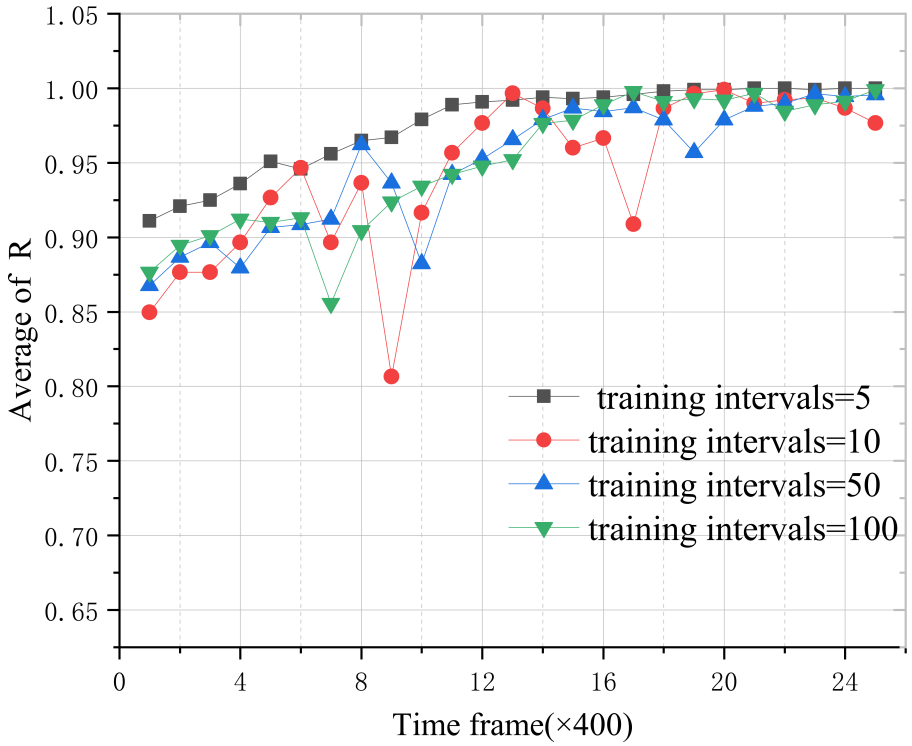

Figure 6. Average of R at different training intervals.

To demonstrate the superiority of our algorithm, we compare our algorithm DRCOM with the following three offloading methods:

- Local computing (LC). All user devices perform local computation, i.e., each $f_i = 0$.
- Computation offloading (CO). All user devices offload computation tasks to UAV, i.e., each $f_i = 1$.
- Linear relaxation algorithm (LRA). f_i is relaxed to be a real number between 0 and 1, $f_i \in [0, 1]$. Then, the problem OPT changes into a convex problem related to f_i, and we use the CVXPY convex optimization toolbox to solve it [24]. After obtaining the solution of f_i, we have:

$$f_i^* = \begin{cases} 1, & f_i > 0.5, \\ 0, & f_i \leq 0.5. \end{cases} \tag{23}$$

As shown in the Figure 7, we compared the maximum computation rate of our algorithm with the other three methods. According to quantitative analysis from the experimental results, the maximum computation rate of our algorithm DRCOM is 142.38% higher than method LC, 50.37% higher than method CO, and 12.44% higher than method LRA. We can see that our algorithm performs the best, followed by LRA. These two methods are far better than LC and CO, and the experimental results of LC are the worst. It further proves the importance of computation task offloading, which aims to increase the computation rate.

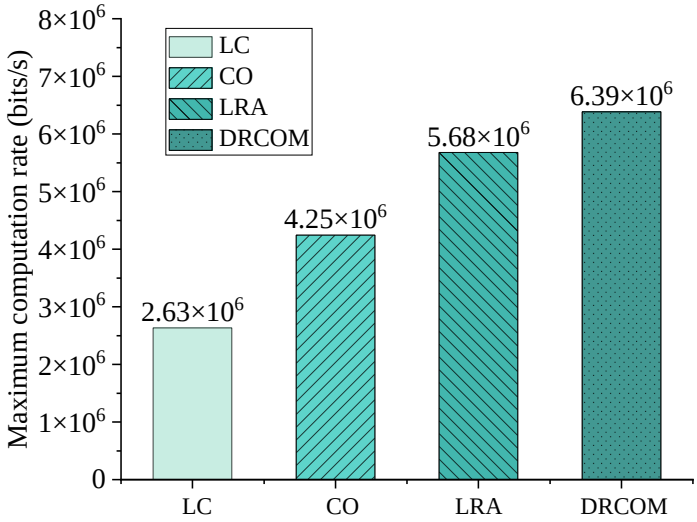

Figure 7. Maximum computation rate of four methods.

6. Conclusions and Future Work

In this paper, a computation offloading method based on deep reinforcement learning in UAV-assisted multi-access edge computing network (DRCOM) is proposed, which uses UAVs as aerial base stations. Faced with the problem of high computation-intensive tasks, which are often difficult to handle in the multi-access edge computing network, we carry out computation policy offloading in the network. Since this problem is NP-hard, we divide it into two sub-problems. Deep reinforcement learning is used to solve the problem of the computation offloading policy, and then we allocate resources to pursue high computing rate. Another three offloading methods are used to compared with DRCOM, and the experimental results demonstrated that the maximum computation rates of DRCOM are 142.38%, 50.37%, and 12.44% higher than the other three methods LC, CO, and LRA, respectively. It means that all user devices that perform local computing or computation offloading have poor performance. Additionally, the algorithm DRCOM using deep reinforcement learning has the highest computation rate.

In the future, we also need to consider the mobility of user devices in UAV-assisted multi-access edge computing networks. In some practical situations, the user device is mobile. So, our proposed method may not be suitable for mobile user device scenarios. For mobile user device, we can propose a new framework and use mobility management technology to solve it. In this paper, we regard the computation offloading policy of the user device as a binary, which can only perform local computing or offloading. When the number of user devices is large, this binary policy will greatly bring bandwidth pressure to the network, so we can consider partial computation offloading in future work.

Author Contributions: Conceptualization, P.Z. and L.L.; methodology, Y.S.; software, C.W.; validation, L.L., C.W., W.Z. and L.T.; formal analysis, P.Z.; investigation, B.L.; resources, B.L.; data curation, Y.S.; writing—original draft preparation, Y.S.; writing—review and editing, P.Z.; visualization, L.L.; supervision, C.W.; project administration, B.L.; funding acquisition, P.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the Natural Science Foundation of Shandong Province under grants ZR2022LZH015, ZR2020MF006, ZR2019LZH013, and ZR2020LZH010, partially supported by the Natural Science Foundation of Hebei Province under grant F2022501025, partially supported by the Pilot International Cooperation Project for Integrated Innovation of Science, Education, and Industry of Qilu University of Technology (Shandong Academy of Sciences) under

grant 2022GH007, partially supported by the Jinan Scientific Research Leader Studio Project under grant 2021GXRC091, partially supported by the One Belt One Road Innovative Talent Exchange with Foreign Experts under grant DL2022024004L, partially supported by the Industry-university Research Innovation Foundation of Ministry of Education of China under grant 2021FNA01001, partially supported by the Major Scientific and Technological Projects of CNPC under grant ZD2019-183-006, and partially supported by the Open Foundation of State Key Laboratory of Integrated Services Networks (Xidian University) under grant ISN23-09.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Notations.

Notation	Definition
U	the set of user devices
f_i	offloading policy of the i^{th} device
E_i	the energy harvested by the i^{th} user device
e	the efficiency of energy harvesting
P	the transmit power of the UAV
h_i	the wireless channel gain between the UAV and the i^{th} device
xT	the length of time for WPT within T time
k	the energy efficiency coefficient
c	the computing speed of the processor
$R_{L,i}(x)$	the local computation rate
$\frac{ct_i}{\phi}$	the number of bits processed by the user device
y_iT	the time taken for i^{th} user device computation offloading
p_i	the offloading power of the user devices
$R_{O,i}(x, y_i)$	the data offloading capability
B	the bandwidth of the link,
P_N	the noise power
$C(h, f, y, x)$	the weighted sum computation rate
w_i	the weight assigned to the computing rate of the i^{th} device.
OPT	the optimization objective

References

1. Xu, L.D.; He, W.; Li, S. Internet of Things in Industries: A Survey. *IEEE Trans. Ind. Inform.* **2014**, *10*, 2233–2243. [CrossRef]
2. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for Smart Cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]
3. Zhang, P.; Wang, C.; Jiang, C.; Benslimane, A. UAV-Assisted Multi-Access Edge Computing: Technologies and Challenges. *IEEE Internet Things Mag.* **2021**, *4*, 12–17. [CrossRef]
4. Anwar, M.R.; Wang, S.; Akram, M.F.; Raza, S.; Mahmood, S. 5G-Enabled MEC: A Distributed Traffic Steering for Seamless Service Migration of Internet of Vehicles. *IEEE Internet Things J.* **2022**, *9*, 648–661. [CrossRef]
5. Lakew, D.S.; Tran, A.T.; Dao, N.N.; Cho, S. Intelligent Offloading and Resource Allocation in HAP-Assisted MEC Networks. In Proceedings of the 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 20–22 October 2021; pp. 1582–1587. [CrossRef]
6. Utsav, A.; Abhishek, A.; Suraj, P.; Badhai, R.K. An IoT Based UAV Network For Military Applications. In Proceedings of the 2021 Sixth International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 25–27 March 2021; pp. 122–125. [CrossRef]
7. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tut.* **2017**, *19*, 2322–2358. [CrossRef]
8. Narendra, P.M.; Fukunaga, K. A branch and bound algorithm for feature subset selection. *IEEE Trans. Comput.* **1977**, *26*, 917–922. [CrossRef]
9. Sun, Z.; Zhang, Z.; Wang, H.; Jiang, M. Cutting Plane Method for Continuously Constrained Kernel-Based Regression. *IEEE Trans. Neural Netw.* **2010**, *21*, 238–247. [CrossRef]

10. Logenthiran, T.; Srinivasan, D.; Shun, T.Z. Demand Side Management in Smart Grid Using Heuristic Optimization. *IEEE Trans. Smart Grid* **2012**, *3*, 1244–1252. [CrossRef]
11. Ju, Y.; Chen, Y.; Cao, Z.; Liu, L.; Pei, Q.; Xiao, M.; Ota, K.; Dong, M.; Leung, V.C.M. Joint Secure Offloading and Resource Allocation for Vehicular Edge Computing Network: A Multi-Agent Deep Reinforcement Learning Approach. *IEEE Trans. Intell. Transp. Syst.* **2023**, *in press*. [CrossRef]
12. Ma, L.; Wang, X.; Wang, X.; Wang, L.; Shi, Y.; Huang, M. TCDA: Truthful Combinatorial Double Auctions for Mobile Edge Computing in Industrial Internet of Things. *IEEE Trans. Mob. Comput.* **2022**, *21*, 4125–4138. [CrossRef]
13. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Commun. Surv. Tut.* **2017**, *19*, 1657–1681. [CrossRef]
14. Guan, S.; Boukerche, A. A Novel Mobility-Aware Offloading Management Scheme in Sustainable Multi-Access Edge Computing. *IEEE Trans. Sustain. Comput.* **2022**, *7*, 1–13. [CrossRef]
15. Ren, J.; Yu, G.; Cai, Y.; He, Y. Latency Optimization for Resource Allocation in Mobile-Edge Computation Offloading. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 5506–5519. [CrossRef]
16. Chang, Z.; Liu, L.; Guo, X.; Chen, T.; Ristaniemi, T. Dynamic Resource Allocation and Computation Offloading for Edge Computing System. In Proceedings of the Artificial Intelligence Applications and Innovations. AIAI 2020 IFIP WG 12.5 International Workshops, Neos Marmaras, Greece, 5–7 June 2020; pp. 61–73.
17. Zhao, X.; Peng, J.; Li, Y.; Li, H. A Privacy-Preserving Computation Offloading Method Based on Privacy Entropy in Multi-access Edge Computation. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 1016–1021. [CrossRef]
18. Jeong, S.; Simeone, O.; Kang, J. Mobile Edge Computing via a UAV-Mounted Cloudlet: Optimization of Bit Allocation and Path Planning. *IEEE Trans. Veh. Technol.* **2018**, *67*, 2049–2063. [CrossRef]
19. Kim, K.; Hong, C.S. Optimal Task-UAV-Edge Matching for Computation Offloading in UAV Assisted Mobile Edge Computing. In Proceedings of the 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), Matsue, Japan, 18–20 September 2019; pp. 1–4. [CrossRef]
20. Zhang, T.; Xu, Y.; Loo, J.; Yang, D.; Xiao, L. Joint Computation and Communication Design for UAV-Assisted Mobile Edge Computing in IoT. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5505–5516. [CrossRef]
21. Liu, Y.; Xiong, K.; Ni, Q.; Fan, P.; Letaief, K.B. UAV-Assisted Wireless Powered Cooperative Mobile Edge Computing: Joint Offloading, CPU Control, and Trajectory Optimization. *IEEE Internet Things J.* **2020**, *7*, 2777–2790. [CrossRef]
22. Wang, F.; Xu, J.; Wang, X.; Cui, S. Joint offloading and computing optimization in wireless powered mobile-edge computing systems. *IEEE Trans. Wirel. Commun.* **2017**, *17*, 1784–1797. [CrossRef]
23. Bi, S.; Ho, C.K.; Zhang, R. Wireless powered communication: Opportunities and challenges. *IEEE Commun. Mag.* **2015**, *53*, 117–125. [CrossRef]
24. Guo, S.; Xiao, B.; Yang, Y.; Yang, Y. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In Proceedings of the IEEE INFOCOM 2016—The 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, 10–14 April 2016; pp. 1–9. [CrossRef]
25. You, C.; Huang, K.; Chae, H. Energy Efficient Mobile Cloud Computing Powered by Wireless Energy Transfer. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 1757–1771. [CrossRef]
26. Huang, L.; Bi, S.; Zhang, Y.J.A. Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks. *IEEE Trans. Mob. Comput.* **2020**, *19*, 2581–2593. [CrossRef]
27. Kirana, K.C.; Wibawanto, S.; Hidayah, N.; Cahyono, G.P.; Asfani, K. Improved Neural Network using Integral-RELU based Prevention Activation for Face Detection. In Proceedings of the 2019 International Conference on Electrical, Electronics and Information Engineering (ICEEIE), Hangzhou, China, 16–18 October 2019; Volume 6, pp. 260–263. [CrossRef]
28. Kaloev, M.; Krastev, G. Comparative Analysis of Activation Functions Used in the Hidden Layers of Deep Neural Networks. In Proceedings of the 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 11–13 June 2021; pp. 1–5. [CrossRef]
29. Liu, L.; Feng, J.; Mu, X.; Pei, Q.; Lan, D.; Xiao, M. Asynchronous Deep Reinforcement Learning for Collaborative Task Computing and On-Demand Resource Allocation in Vehicular Edge Computing. *IEEE Trans. Intell. Transp. Syst.* **2023**, *in press*. [CrossRef]
30. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. *IEEE Commun. Surv. Tut.* **2019**, *21*, 3133–3174. [CrossRef]
31. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
32. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.