Noname manuscript No.
(will be inserted by the editor)

Collaborative Multi-Agents in Dynamic Industrial Internet of Things using Deep Reinforcement Learning

Ali Raza · Munam Ali Shah · Hasan Ali Khattak · Carsten Maple · Fadi Al-Turjman · Hafiz Tayyab Rauf

Received: September 15, 2021/ Accepted: date

Abstract Sustainable cities are envisioned to have economic and industrial steps towards reducing pollution. Many real-world applications such as autonomous vehicles, transportation, traffic signals, and industrial automation can now be trained using Deep Reinforcement Learning (DRL) techniques. These applications are designed to take benefit of DRL in order to improve the monitoring as well as measurements in Industrial Internet of Things for Automation Identification System. The complexity of these environments mean that it is more appropriate to use multi-agent systems rather than a single-agent. However, in non-stationary environments multi-agent systems can suffer from increased number of observations, limiting the scalability of algorithms. This study proposes a model to tackle the problem of scalability in DRL algorithms in transportation domain. A partition-based approach is used in the

A. Raza · MA. Shah
Department of Computer Science, COMSATS University Islamabad, Pakistan
E-mail: razaali4939@gmail.com,mshah@comsats.edu.pk

HA. Khattak
School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabd Pakistan
E-mail: hasan.alikhattak@seecs.edu.pk

C. Maple
Security Cyber Systems Research Group, WMG, University of Warwick, Coventry, CV4 7AL, UK
E-mail: cm@warwick.ac.uk

F. Al-Turjman
Artificial Intelligence Dept., Research center for AI and IoT, Near East University, Nicosia, Mersin 10, Turkey.
E-mail: Fadi.alturjman@neu.edu.tr

HT. Rauf
Department of Computer Science, Faculty of Engineering & Informatics, University of BRADFORD, United Kingdom.
E-mail: h.rauf4@bradford.ac.uk

proposed model to reduce the complexity of the environment. This partition-based approach helps agents to stay in their working area. This reduces the complexity of the learning environment and the number of observations for each agent. The proposed model uses Generative Adversarial Imitation Learning (GAIL) and Behavior Cloning (BC), combined with a Proximal Policy Optimization (PPO) algorithm, for training multiple agents in a dynamic environment. We present a comparison of PPO, Soft Actor-Critic (SAC), and our model in reward gathering. Our simulation results show that our model outperforms SAC and PPO in cumulative reward gathering and dramatically improved training multiple agents.

Keywords Deep Reinforcement Learning · Multi Agents · Behavior Cloning · Dynamic Environment · Scalability

1 Introduction

Reinforcement Learning (RL) draws upon observations from experiments examining the behaviour of cats by Thorndike at the turn of 20th century [48] and has become a popular and effective technique in machine learning in recent years. It involves a process of learning what action yields the most reward [30]. In RL, an agent interacts with the learning environment, takes action from the current state, and moves to the next state. In RL, an agent is anything that takes some actions observing the environment, and receives some reward. The purpose of the agent is to maximize the value of the reward. Deep Reinforcement Learning (DRL) is the combination of Deep Learning and RL[2,31,39], developed to tackle high dimensional environments [3].

Multi-agent systems (MAS) are a technique developed to overcome the limitations of single-agent systems [51]. In a MAS, agents interact with the environment, collaborate and communicate with each other, and maximize the cumulative numerical reward [9,19]. The use of MAS provides many advantages like distribution, scalability, robustness, and parallelism [36]. However, MAS also has some disadvantages. It makes the environment nonstationary and more complex than single-agent environments. In many real-world applications, such as traffic control, task allocation, autonomous vehicles and drones, multi-agents are necessary to create a representative model. Due to the environment's complexity, a hard-coded agent (an agent that does not learn from the environment using DRL) cannot be performed well in these high dimensional environments. The agents must be capable of finding optimal solutions to their learning. In this context optimal solution means the best solution according to the given task. Multi-Agent Deep Reinforcement Learning (MADRL) is a powerful technique for modeling the above mentioned applications. Using MADRL, agents seek optimal solutions and policies.

In MADRL, agents try to estimate optimal actions, using neural networks of large layers and approximators. The agent's policies depend on the action with other agents [9]. This is due to the fact that when an agent takes action, a change occurs in the environment, thereby affecting the policy of other agents.

Multi-agents can be useful in many real-world applications, including digital factories. Digital factories are abstractions of real factories. The use of intelligent agents in these digital factories can provide several advantages i.e. adaption, autonomy, decentralization and robustness [5].

Many real-world problems are so complex that single-agent systems cannot cope. It is known that Unmanned Surface Vehicles (USV), robot soccer, autonomous vehicles, and controlling of traffic signals cannot be tackled through a single agent [40]. MAS, on the other hand has the advantage of being scalable. Recent DRL techniques perform well in single-agent environments, but they do not perform well in multi-agents environments [54] [41]. In multi-agent environments, the number of observation increased, and the environment becomes non-stationary [9] [20]. Due to the increased number of observations and non-stationary environment, many DRL algorithms are not scalable. They do not perform well in multi-agent environments.

In this paper, a multi-agent system has been proposed where multiple agents are trained in a digital factory environment. We use a partition-based approach in our proposed model, where each agent stays in its working area. This proposed partition approach reduces the number of observations for each agent, as each agent has to observe only its active neighbor agents. This also makes the environment less non-stationary. The use of GAIL and BC makes the policy convergence fast. Moreover, these techniques also helps agents to make right partitions. By reducing the number of observations and making the environment less non-stationary, our proposed system performs well in single-agent settings as well as multi-agent settings. We compare our model with SAC [17] and PPO [44]. Our model performs well compared to PPO and SAC in terms of reward gathering applied to moving heavy materials in the factory environment.

The remainder of this paper, is organized as follows: In section 2, the literature is reviewed; Section 3 presents preliminaries for this work; and in Section 4, we introduce our proposed model. Section 5 contains implementation details, and results are discussed in Section 6, before the conclusions in Section 7.

2 Literature Review

Mobile robots are the machines that sense the environment and move around the environment. They use sensors to sense the environment. Mobile robots are becoming popular in different domain. They are used to assist different process and to accomplish different tasks.

In different real-world applications, autonomous navigation is necessary for mobile robots [4]. The mobile robots should be able to navigate through various obstacles to reach their destination. In multi-agent scenarios, collision avoidance between agents (mobile robots) is a crucial feature. The non-stationary environment and high dimensional characteristics are the most significant reasons that algorithms are non-scalable in MAS contexts.

A rich literature is available concerning single-agent DRL. In [52], the author proposed the Autonomous Navigation and Obstacle Avoidance (ANOA) algorithm. ANOA is an extension of the DQN algorithm [35]. ANOA has been shown to perform better than DQN and Deep sarsa[55]. Hodge et al. introduced an algorithm for navigation of drones in infrastructure and buildings in [24]. The author used the PPO algorithm [44] along with curriculum learning. In this work, the author trained a single agent. Learning to navigate in complex dynamic environments is a significant area of current research in robotics. Various algorithms such as Potential Field Method (PFM) [29], and Simultaneous Localization and Mapping (SLAM) [10] are proposed in the literature to address obstacle avoidance while path planning. However, these approaches, like PFM, can get trapped into local minima. Another method, Vector Field Histogram (VFH) [6], works by continually constructing a map while the robot is moving in a environment.

SLAM deals with navigation problems by constructing and updating a map of the environment while localizing itself using the same map. However, SLAM suffers from high computational costs and a large equipment for memory. A deep modular RL architecture is proposed in [49] to address these issues. Autonomous navigation in mobile robots is a significant research problem to which RL has been applied. Earlier map-based techniques do not perform well when global path planning is required. In contrast with these techniques, heuristic-based approaches have been used to enable mobile robots to perform collision-free and effective path planning using a laser range finder.

In [34], an incremental learning and DRL based scheme is proposed to provide an effective path planning scheme for mobile robots. An incremental learning adapted ACKTR algorithm and a soft actor-critic neural network for practical navigation policies are presented. The proposed scheme ensures the effectiveness of an adapted approach in collision-free path planning to reach the destination. The discrete action space crowd navigation in an unfamiliar environment suffers from inefficient path planning and obstacle avoidance in the navigation domain. To overcome these issues, a DRL based approach is proposed in [47] that incorporates memory characteristics of the human being, specifically Long-short Term Memory (LSTM) [23].

In [28], the author introduces a path planning simulator for the vehicles. The author used the PPO algorithm of DRL in a single-agent settings. Recently, autonomous vehicles and intelligent transport systems have been studied using artificial intelligence techniques. In [26], a DRL technique was used to achieve UAV navigation utilizing a massive multiple-input-multiple-output (MIMO) technique. By constructing a DQN, an optimal location selection policy, based on the received signal strengths, is obtained. Unlike previous works, which mainly present the speed or geographic position for UAV navigation, the proposed method converges quickly.

To address decision-making problems in a dynamic robotic soccer game, a decision making strategy has been proposed in [46]. An Improved Support Vector Machine (ISVM) is used to collect and classify environmental, situational information constituted by the defined evaluation factors, and the Adaptive

Decision Making Algorithm with RL (ADMA-RL) chooses the strategy adaptively. There are multiple agents in the proposed scheme each having their own roles. Roles can be changed based on a given scenario. RL-based approaches have been used to provide precise and timely decisions for robots. The authors of [42] introduce a meta game adoption process, introducing a MAS to train agents against a master agent by extending the traditional RL approach.

Table 1 Literature Review Table, × represents the feature is not tackled in previous work and √ represents that feature is tackled in previous work.

Ref	Multi Agents	Agents make joint actions	Collaborative Agents	Partially observable environment	Real world complex environment
[52]	×	×	×	×	√
[24]	×	×	×	×	×
[49]	×	×	×	×	×
[46]	√	×	√	×	×
[42]	×	×	×	×	×
[53]	×	×	×	×	×
[34]	√	×	×	×	×
[54]	√	×	√	×	×
[8]	√	×	√	×	×
[25]	√	×	√	√	×
[50]	√	×	×	√	√
[9]	√	√	√	×	×
Proposed Model	√	√	√	√	√

Many real-world applications require MAS to perform tasks. MAS suffers from many challenges as communication and collaboration between agents. Non-stationary environments and high dimensional observations create issues such as scalability. To address these issues, the authors of [16] extend three single-agent RL algorithms based on policy gradient, temporal difference, and actor-critic methods. To address the problem of high input dimensionality, in [54] the authors adopt centralized training with a decentralized execution framework. To reduce the curse of dimensionality, they employ observation embedding. Nonstationarity is not compatible with the replay buffer of deep Q learning.

Deep Q learning depends on a replay buffer that stores the experience replays to use them repeatedly during the training. Nonstationarity changes the current dynamics, due to change in dynamics the experience replays stored in replay buffer are no longer efficient. To tackle this issue, the authors of [11] introduce two approaches. In the first approach, the experience is saved in a replay buffer as off policy environment data. In the second approach, each agent learns a policy as an estimate of the other agent's policy, observing their behaviors. The authors of [8] introduced an advantage actor-critic algorithm for stabilizing local agents' learning process. Two methods are proposed in [8]

and learning is stabilized by reducing the learning difficulty and increasing the observability.

MAS can be either centralized or decentralized systems and can be evaluated based on robustness, scalability, fairness, and stability [43]. In [43], various techniques used so far in MAS, have been critically reviewed and a comparison of their surveys conducted on MAS is presented. This demonstrates that none of them targeted deep learning-based methods in MAS. In [43,14], it is argued that MAS suffers from scalability, heterogeneity of agents, and continuous action spaces. This survey aims to highlight MAS's challenges in terms of decision-making in smart city environments and provides future directions to develop such systems that can be scalable and robust in terms of performance using DRL-based approaches [15,1].

Nowadays many researchers are working on single-agent and multi-agent domains. In [21] the authors present an overview of recent challenges and open research direction in MADRL. In the end, the authors also present that how RL and multi-agent learning adapted in the field of MADRL. In current studies on autonomous vehicles, datasets of experts are used by considering their behavior on different events. The main issue in this approach is the narrowness of human drivers and insufficient samples of data [37].

To overcome this issue, the authors of [50] apply DRL techniques on vehicles to take data samples. They used DQN to reduce the curse of dimensionality. To increase the computational speed, the authors used a fuzzy calculation process which is completed on MATLAB. In a multi-agent domain where the environment is partially observable and agents are cooperative, agents take actions jointly and increase the total cumulative reward [12]. In this kind of environment, it is very hard to increase the individual reward for the agents. To overcome this issue, the authors of [45] propose an algorithm called CoRe.

CoRe uses a counterfactual reward mechanism and computes the difference between the total reward and individual reward. Defensive escorts help to navigate payloads in a coordinated manner. In [18] authors introduce an approach for these escorts to navigate payloads. They use MADRL for the coordination and cooperation between multiple defensive escorts. A centralized trained policy is used in a distributed way for these escorts.

Multiple robots jointly navigate a path or explore an unknown environment in autonomous navigation and exploration of an unknown environment, which is an important use of mobile robotics. Fault tolerance, rescue, monitoring, and mapping are just a few of the benefits of this technique. Because of these benefits, this technique has attracted a lot of research attention. A unique exploration strategy is proposed in [25]. In [25], the authors describe a Voronoi-based strategy for exploration.

The intelligent control of traffic signals is essential for transportation system optimization. Recent research has concentrated on intersection coordination to attain global optimal traffic efficiency in massive road networks, with impressive results. In [33], the authors introduced a framework for the coordination of multi-agents. The authors defined a reward structure for the traffic control signals. Using spatial-temporal information, a differentiation method

is introduced for the coordination of agents. In [53] the authors consider the safety constraints for various robotics applications such as manipulation and self-driving vehicles. An algorithm was proposed to ensure the safety robustness, evaluated based on three different policies for a cable-driven parallel robot.

Table 1 presents the characteristics of different research papers. Further we present the characteristics of our proposed model.

3 Preliminaries

3.1 Multi Agent Deep Reinforcement Learning (MADRL)

MADRL is rapidly expanding and covers a wide variety of approaches to learning in multi-agent settings. In MADRL, multiple agents interact with the learning environment and try to maximize the total cumulative reward. The main issues in MADRL are dimensionality, nonstationarity, communication, and coordination. As the number of agents increases, the dimensionality of the observation increases, and each agent takes action and tries to maximize the total cumulative reward. Each agent's action creates some change in the environment, thereby making the environment non-stationary for other agents in the environment. Most current DRL algorithms perform well in single-agent settings but do not perform well in multi-agent environments. MADRL can be defined in the form a of Markov Decision Game (MDG). The generalization of the Markov decision process to the multi-agent case is the Markov game. It is called MDG in the literature. It is Markov Decision Process (MDP) which is called MDG for multi-agents in literature. The tuple of the game is written as (N, S, A, R, T). Here N represents number of agents, N= $n_1, n_2, ...n_n$; S represents finite states, S= $s_1, s_2, ...s_n$ and A= $A_1 \times A_2 ... \times A_n$ is a finite joint action space. R: S×A → (some real number). T: S×A×S→[0,1] [9] where R is the common expected reward and T is the transition function.

3.2 Proximal Policy Optimization (PPO)

Policy gradient methods are the foundation of many recent breakthroughs in control, 3D Games, Locomotion and Go using deep neural networks. PPO [44] performs better than other state of the art learning approaches and is very simple to implement and tune. The objective function of PPO is given by

$$L^{clip}(\theta) = \hat{E_t}[min(r_t(\theta)\hat{A_t}, clip(r_t(\theta, 1-\epsilon, 1+\epsilon)\hat{A_t})]$$

In this paper we use the PPO algorithm as a trainer in the continuous action space.

3.3 Imitation Learning

Imitation learning is a process of learning in which an agent tries to mimic a human's behavior in a given task. In RL, imitation learning is that an agent is trained to learn the behavior from recorded demonstrations and utilises a mapping between observations and actions to learn a policy.

4 Proposed Model

The problem considered in this work is the Partially Observable Markov Decision Process (POMDP). We introduced a partition-based approach in this work.

4.1 Partition-based Approach

Using partition-based approach, the working area can be broken into partitions dynamically between the agents based on their current location. This partition-based approach can be implemented in a decentralized manner through communication between agents.

Let T be a topological space of R^2 for the agents. This topological space is homomorphic to our area of learning environment. Suppose A_i $\forall i \in$ {1,2,3 . . . N} are the agents and N_A_i is an element from a set of neighbor agents of A_i. p_i represents the position of A_i and partition for agent A_i is defined as

$$P(A\_i) = \{\tau \in T| \ \ ||\tau – p\_i|| \leq ||\tau - p\_j||, \forall A\_j \in N\_A\_i\}$$

which means the partition of T for an agent A_i is included all those points of T which are more closer to A_i than any other points in T.

To make partitions, an agent A_i only needs to know the boundary of T and the position p_j of its neighbor agent. In this way, each agent can generate a partition by knowing only the locations of the neighbor agents. If any agent $A_j \in N_{A_i}$ of A_i stops working due to any fault, the agent A_i will automatically increase its partition to the edge of the partition of the next agent p_{j+1}. This helps agents to complete the given task collaboratively.

4.2 Proposed DRL

In our multi-agent settings, we use the PPO algorithm for training purposes. We use imitation learning and Behavior Cloning (BC) to scale the training process to several agents. Generative Adversarial Imitation Learning (GAIL) gives better results in complex environments because it learns through demonstrations and works well with sparse reward. BC is helpful to mimic the behavior of expert. These methods helps agents to make right partitions in our scenarios and increase the performance of agents. Policy convergence with

PPO is fast that why we use it as a trainer. In the learning environment, the agents are fully cooperative. Each agent has limited information about the environment. All agents try to maximize the total discounted reward by taking actions cooperatively. The tuple can be defined as

$$(I, S, A_i, Z_i, T, R, O)$$

where I is the set of agents, S is the set of states, A_i is a set of actions for agent i, Z_i is the set of observations for the agent i, and T, R, O is the joint transition, reward and observations model.

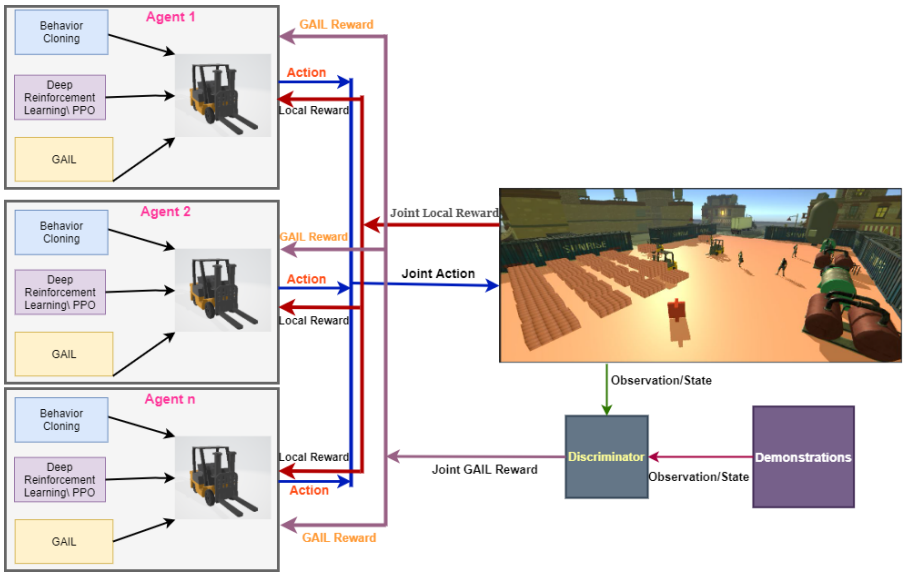

Fig. 1 Our Proposed System Model

As the learning environment is partially observable in this work and each agent has limited information about the environment on a specific state S_t, to make the training more stable, agents need to remember some information about previous observations; for this purpose, we use LSTM [23]. We used the GAIL algorithm [22] to make the training more successful with sparse rewards. This algorithm uses a second neural network, a discriminator, to distinguish between actions and observations from demonstration and actions and observations produced by the agent [13]. The discriminator examines the new actions and observations, and determines whether these are close to the demonstration's actions and observations. On behalf of this examination, the discriminator gives a reward to the agents. We define the discriminator network as $D_\omega : S \times A \rightarrow (0, 1)$ with weight ω [22]. We also use local rewards with GAIL. An extrinsic reward signal is used as a local reward signal in this work. To make the convergence of a policy quicker, we use the BC technique with RL

and GAIL. BC is a supervised learning technique in which agents try to mimic the behavior from demonstrations directly.

Algorithm 1: Pseudo Code of Proposed Model

```
while episodes not ends do
    while Time steps not equal to 5000 do
        For all agents
        {
        select action a_i from actions according to Policy π_θ
        Apply action a_i
        Compute Local Reward
        Compute observations and state s_{t+1}
        Compute GAIL Reward using Discriminator.
        Compute Partition
        Update Policy π_θ on the bases of GAIL reward.
        Update Discriminator.
        }
    end
end
```

The proposed system model is shown in Figure 1. As shown in figure, each agent is connected with RL or PPO, GAIL, and BC. These techniques are applied to each agent. The agents take actions cooperatively and collaboratively and interact with the learning environment by taking joint action. As a result of that action, a local cumulative reward is given to the agents. Moreover, the BC technique is applied to each agent. This helps agents to directly mimic the behavior of demonstrations, thereby making the training process faster. Using GAIL technique ensures the policies converges quickly. This technique is used to train the second neural network, to distinguish the agent's observation/action and the action from the demonstrations [22]. The pseudo-code for the proposed model is shown in Algorithm 1.

4.3 Reward Structure

To achieve the desired behavior, we use a local reward signal that encourages consistent behavior. A good reward scheme means the learning process converges quicker in single-agent settings [38]. We proposed a reward structure according to our environment which helps agents to learn to stay in their partition. It also helps agents to generate partitions correctly. In this way, it increased the performance of agents.

The rewards given to each agent on different events are shown in Table 2. We gave a -0.1 reward when an agent collides with another agent in the environment. In this way, the issue of inter-collision of agents is addressed. Moreover, we gave -0.1 when the agent collides with a wall or any other obstacles. The reward of -0.1 is also given when the agent hits any moving human in the environment. When an agent succeeds in picking heavy material from

the collection point, a reward of +0.3 is given to that agent. Further, when the agent succeeds in dropping the heavy material at the destination, it receives a reward of +0.5. When agents succeed in shifting the whole collection, then a further reward of +0.5 is given to each agent.

Table 2 Rewards Given On Different Events

Event	Reward
Agent collide with wall	-0.1
Agent collide with other agent	-0.1
Agent collide with moving human	-0.1
Agent collide with truck	-0.1
Agent succeed to pick brick	+0.3
Agent succeed to drop brick at destination	+0.5
On shifting whole collection at destination	+1

4.4 Environment

Robots playing a significant role in many fields of life. Recently, robots are being deployed in digital factories more frequently as compare to previous years [5]. Robots can be used to hold, move and lift heavy materials in factories. They can reduce the labor required in factories, working more efficiently and rapidly than humans. Due to the importance of robots in digital factories, we choose a digital factory's environment as a learning environment. Moreover, many real-world environments need multi-agents, but to begin with, we choose digital factories. A real-world scenario is considered in this work, where agents try to load heavy material onto trucks.

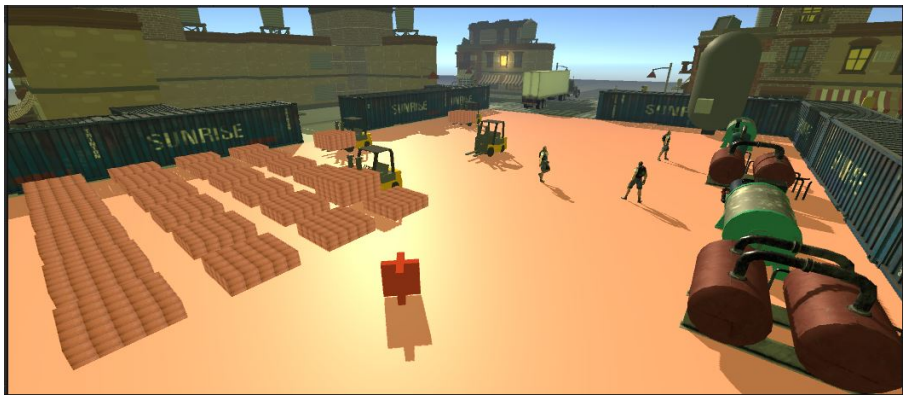

Fig. 2 Learning Environment

In the factory, agents pick, hold and move heavy material and drop them near a truck as shown in Fig. 2. The environment has obstacles and heavy

machinery. Moreover, the movement of the human is also considered in the learning environment. The agents can take four actions: left move, right move, forward, and backward. By taking these actions, agents move towards the heavy materials' stock by navigating through obstacles and moving humans.

4.5 Agents

In the learning environment, we use Unity 3D Game objects as agent as demonstrated in [52, 24, 7] and [32]. The agent is a lifter used to lift heavy materials, move them and drop them near to truck.

In the learning environment, agents generate the state, take actions from predefined actions (left, right, forward, and backward), and move from one state S_t to S_{t+1}. Each agent must be connected with one brain. The brain informs the agent about the action to be taken.

4.6 Pros and Cons of Proposed Model

The proposed model decreases the number of observations in the environment for each agent. It makes the environment less nonstationary. As a result of these advantages, the DRL becomes more scalable. A high number of agents can be trained in a complex and dynamic environment using the proposed model. The proposed model also has some disadvantages. The agents hesitate to pick the heavy material when it is placed at the boundary of the partition. This technique cannot be applicable in such an environment where two or more agents have to pick the same heavy material and move it together.

5 Implementation

This section contains the details of how we implement the proposed model. We also give details of the tools that are used for the implementation. Moreover, different behavior parameters, as well as training parameters used in this work, are discussed.

5.1 Unity 3D

Recently, the most common tool used for developing a learning environment is Unity 3D. Hodge et al [24] used the same tool for creating a learning environment, as did [52, 7] and [32]. We also used Unity 3D 2019.4.8.f1 to construct the learning environment.

5.2 ML-Agents toolkit

Machine Learning Agents Toolkit (ML-Agents toolkit) [27] is used to apply RL on Unity 3D learning environments. This toolkit provides many components including Brain, Academy, and Agent, for applying RL. Each agent is attached to a brain component and controlled by this brain. The Academy component is used to learn different environmental parameters. Furthermore, it also establishes a connection between the brain and Python Tensorflow.

The size of the working area of our learning environment is 200×200. There are different objects in the learning environment's functional area, such as machines, humans, obstacles, heavy materials, and agents. To make the learning environment more realistic and dynamic, we used moving humans and machines. In this way, we train our agents in a dynamic environment.

Table 3 Training Parameters Used For Our Training Setup

Parameters	Values
Batch size	1024
Buffer size	16384
Learning rate	0.0003
Max steps for each episode	5000
Number of epoch	3
Number 0f layers	2
Gamma in extrinsic Reward signal	0.99
GAIL strength	0.01
Behavior cloning strength	0.5

Table 3 shows some important training parameters used to train the agents through the proposed model. Empirical tests indicated that an appropriate value of batch size is 1024. The value of buffer size should be larger than the batch size. We set the buffer size value to be 16384. The proposed model consists of local RL reward signals, GAIL and BC; we use GAIL strength 0.01 and BC strength of 0.5.

6 Results and Discussion

Firstly, we performed experiments on our proposed model using a single-agent for comparison purposes. Moreover, we implement PPO and SAC according to our context. Our proposed model outperformed PPO and SAC in comparison to single-agent settings.

The agent's highest mean reward in single-agent settings of PPO is 0.26, SAC is 0.25 and the proposed model is 0.95, as shown in Figure 3. We train our agents with each algorithm up to 1.5 million time steps. At the start of the training, the agent gets a louder reward with the proposed model than the other two algorithms. As the training steps increase, the agent attains a better reward with the proposed model than other two methods.

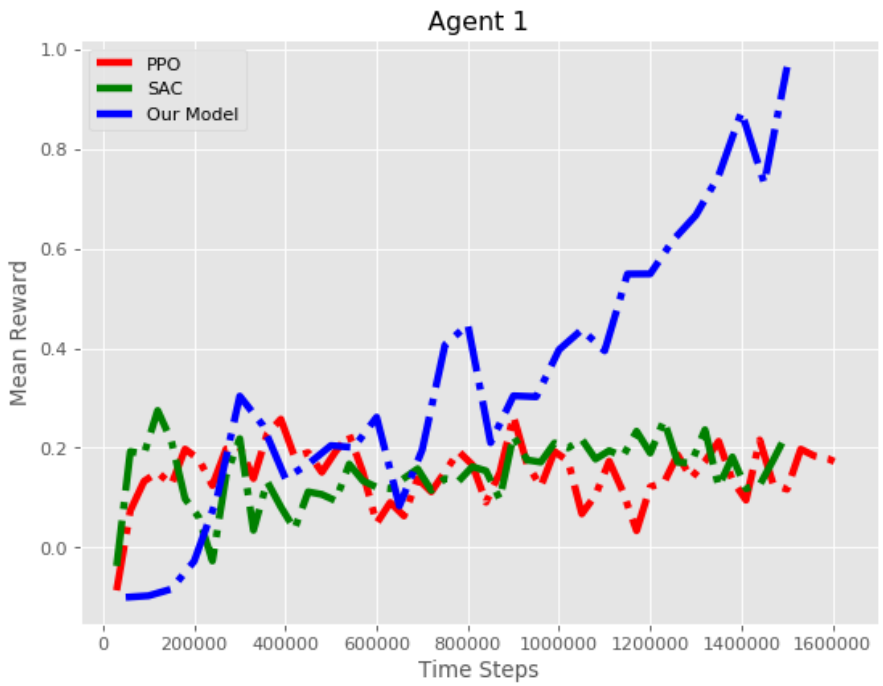

Fig. 3 Mean Reward Of PPO, SAC, and Proposed Model in Single Agent Settings

To demonstrate the scalability of our proposed model, we implemented our proposed model in multi-agent settings and compared the results with PPO and SAC. In the multi-agent setting, we performed experiments with four agents and six agents. Our proposed model also outperformed PPO and SAC in these settings.

Figure 4 shows the mean reward attained by agents in the four agents settings. It shows the comparison of the mean reward achieved using PPO, SAC, and the proposed model in four agent settings. This demonstrates that our proposed model performs better, with the agents gaining higher cumulative mean reward with our proposed model compared to the other two algorithms.

Furthermore, we compared our proposed model with PPO and SAC in six agent settings. The mean reward in six agents settings is shown in Figure 5. The results of six agents settings illustrate that the performance of our proposed model is better than both PPO and SAC. Figure 5 shows that the mean reward is negative for all algorithms at the beginning of the training. With more training, the agents explored the right state-action pairs, and the cumulative mean reward increased. Secondly, the discriminator neural network examines the observations and actions; with the increasing time steps, if an agent gains higher reward, it got stricter. In this way, the performance of the agents is improved.

Fig. 4 Mean Reward Of PPO, SAC, and Proposed Model in Four Agent Settings

We also compared the number of heavy materials moved during testing by agents in different settings. We consider the number of moving materials in the first five minutes of testing and then compared the PPO, SAC, and our proposed model.

Figure 6 expresses the number of heavy materials moved during the testing phase. It is clearly shown in the Figure 6 that the agent trained with the proposed model works quickly without any hesitation and moves more heavy materials than the agents trained with the other two algorithms; the agent trained through our proposed model moved twenty heavy materials in the first five minutes of the testing phase.

After comparing the results of moving heavy materials in single-agent settings, we compared the results of moving heavy materials in multi-agents settings. In multi-agents settings, first of all, we compare the results with four agents. The results shown in Figure 7 are the number of heavy materials moved by agents trained through the proposed model, PPO and SAC. In four agent settings, our proposed model performed well in the sense of moving heavy materials. Our agent moved total 155 heavy materials in the first five minutes of the testing phase.

Furthermore, we test each method in a six agent settings. Figure 8 shows the results of the number of heavy materials moved by agents trained through PPO, SAC, and our proposed model. It is clearly shown that our proposed

Fig. 5 Mean Reward Of PPO, SAC, and Proposed Model in six Agent Settings

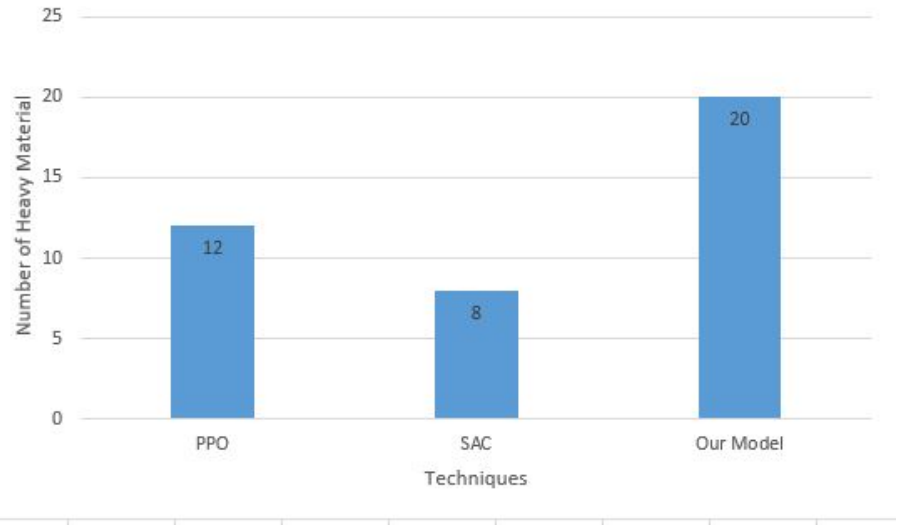

Fig. 6 Number of heavy materials moved during testing of PPO, SAC, and the proposed model in a single agent settings.

model performs better than SAC and PPO. Our proposed model moved total

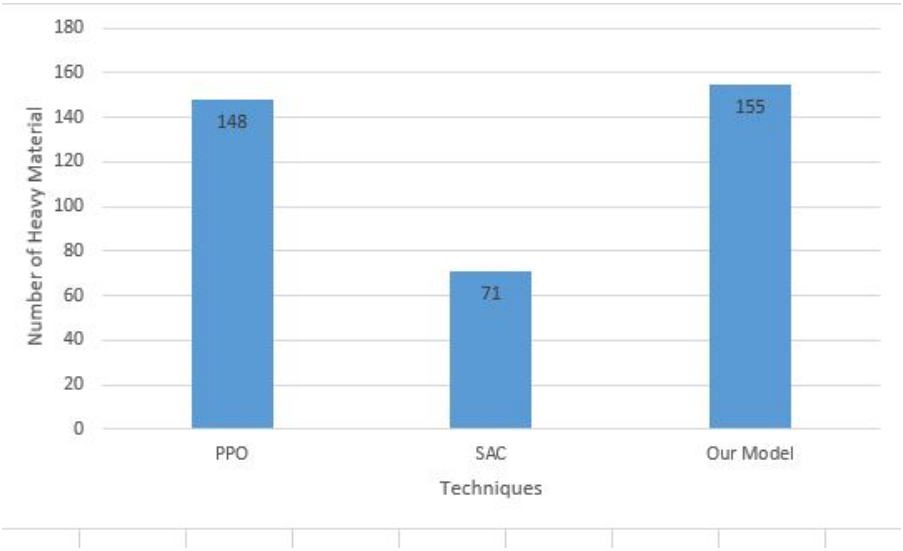

Fig. 7 Number of heavy materials moved during testing of PPO, SAC, and the proposed model in a four agents settings

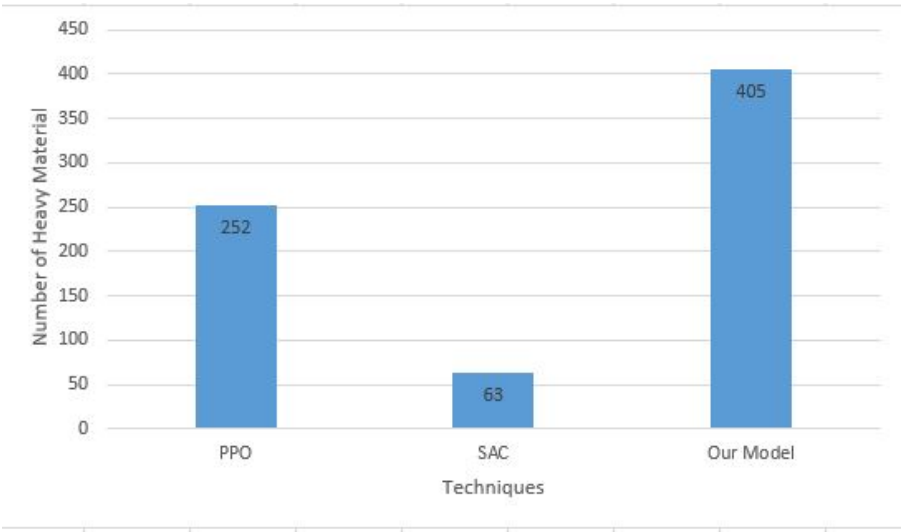

Fig. 8 Number of heavy materials moved during testing of PPO, SAC, and the proposed model in a six agents settings.

405 heavy materials in six agent settings while PPO and SAC moved total 252 and 63 heavy materials respectively.

Figure 9 shows the GAIL loss value showing our model's performance in different agent settings. For effective training of agents, the loss value should decrease with time. The graph shows that the loss value decreases with time in

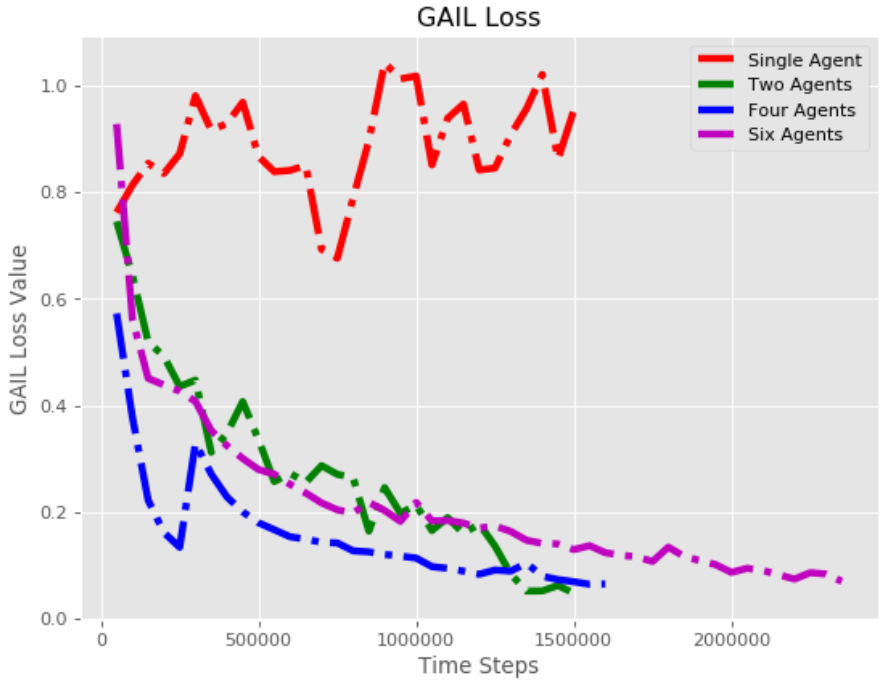

Fig. 9 GAIL Loss in terms of time steps

all agent settings. Thus it demonstrate the effectiveness of our model. In single-agent settings, the loss value is not continuously decreasing. This is because, the demonstrations in single-agent settings are low, and all state-action pairs are not covered in demonstrations. This increases the overall GAIL loss. We use GAIL Grad Mag Loss as another parameter to show the effectiveness of our training process. This loss decreases continuously with steps. This illustrates strong performance in all agent-settings. Figure 10 shows the GAIL Grad Mag Loss.

From these results, we conclude that using BC and GAIL techniques with PPO in dynamic environments is useful. Further it performs better in environments where rewards are sparse and scales the capability of training to more agents compare to use only PPO or SAC. BC helps agents to mimic the behavior from demonstrations, while GAIL teaches the agents to perform corrective actions as in the demonstrations. This method helps speed up the training process, and agents stay in their own area of responsibility which addresses the non-stationary issue of learning environment.

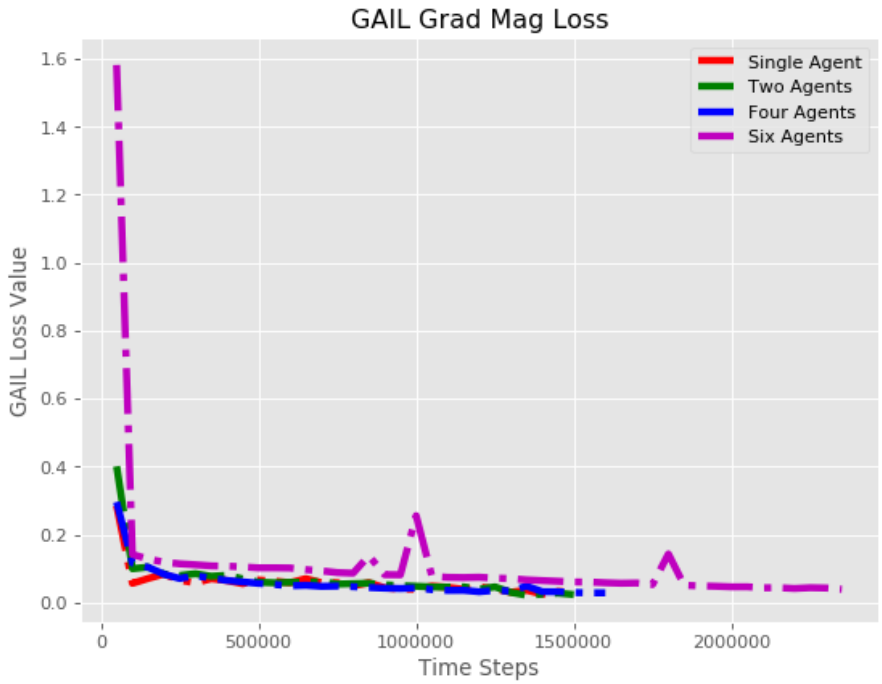

Fig. 10 GAIL Grad Mag Loss in terms of time steps

7 Conclusion

In this work, we have proposed a model to scale the DRL to several agents. We use GAIL and BC with PPO to train many agents in a complex dynamic environment. We use a partially observable learning environment. We also present a reward structure for teaching the multiple agents in a digital factory environment. Moreover, we compare the mean reward of our Proposed Model, PPO, and SAC. Our experimental results show that our proposed Model outperformed the PPO and SAC. In simulations, we offer the agents moving heavy materials in a digital factory. Agents navigate the stock of heavy materials and then move the heavy materials to the destination using DRL. In this work, we did not teach agents through DRL how agents pick heavy material and drop it. We use hard to code for this process. In the future, we will consider training agents on how to pick and drop heavy materials through DRL.

References

1. Arshad, S.R., Saeed, A., Akre, V., Khattak, H.A., Ahmed, S., Khan, Z.U., Khan, Z.A., Nawaz, A.: Leveraging traffic condition using iot for improving smart city street lights. In: 2020 IEEE International Conference on Communication, Networks and Satellite (Comnetsat), pp. 92–96. IEEE (2020)

2. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine **34**(6), 26–38 (2017)
3. Awan, K.A., Din, I.U., Almogren, A., Khattak, H.A., Rodrigues, J.J.: Edgetrust-a lightweight data-centric trust management approach for green internet of edge things. Wireless Personal Communications (2021)
4. Basit, M.A., Khattak, H.A., Ahmed, S., Nawaz, A., Habib, M., Zaman, K.: Driving behaviour analysis in connected vehicles. In: 2020 International Conference on UK-China Emerging Technologies (UCET), pp. 1–5. IEEE (2020)
5. Bicocchi, N., Cabri, G., Leonardi, L., Salierno, G.: Intelligent agents supporting digital factories. In: WOA, pp. 29–34 (2019)
6. Borenstein, J., Koren, Y., et al.: The vector field histogram-fast obstacle avoidance for mobile robots. IEEE transactions on robotics and automation **7**(3), 278–288 (1991)
7. Cao, Z., Lin, C.T.: Hierarchical critics assignment for multi-agent reinforcement learning. arXiv preprint arXiv:1902.03079 (2019)
8. Chu, T., Wang, J., Codecà, L., Li, Z.: Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Transactions on Intelligent Transportation Systems **21**(3), 1086–1095 (2019)
9. Diallo, E.A.O., Sugiyama, A., Sugawara, T.: Coordinated behavior of cooperative agents using deep reinforcement learning. Neurocomputing **396**, 230–240 (2020)
10. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part i. IEEE robotics & automation magazine **13**(2), 99–110 (2006)
11. Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P.H., Kohli, P., Whiteson, S.: Stabilising experience replay for deep multi-agent reinforcement learning. arXiv preprint arXiv:1702.08887 (2017)
12. Gao, J., Wang, H., Shen, H.: Machine learning based workload prediction in cloud computing. In: 2020 29th international conference on computer communications and networks (ICCCN), pp. 1–9. IEEE (2020)
13. Gao, J., Wang, H., Shen, H.: Smartly handling renewable energy instability in supporting a cloud datacenter. In: 2020 IEEE international parallel and distributed processing symposium (IPDPS), pp. 769–778. IEEE (2020)
14. Gao, J., Wang, H., Shen, H.: Task failure prediction in cloud data centers using deep learning. IEEE Transactions on Services Computing (2020)
15. Gheisari, M., Najafabadi, H.E., Alzubi, J.A., Gao, J., Wang, G., Abbasi, A.A., Castiglione, A.: Obpp: An ontology-based framework for privacy-preserving in iot-based smart city. Future Generation Computer Systems **123**, 1–13 (2021)
16. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multi-agent control using deep reinforcement learning. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 66–83. Springer (2017)
17. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290 (2018)
18. Hasan, Y.A., Garg, A., Sugaya, S., Tapia, L.: Defensive escort teams for navigation in crowds via multi-agent deep reinforcement learning. IEEE Robotics and Automation Letters **5**(4), 5645–5652 (2020)
19. Henna, S., Davy, A., Khattak, H.A., Minhas, A.A.: An internet of things (iot)-based coverage monitoring for mission critical regions. In: 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5. IEEE (2019)
20. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems **33**(6), 750–797 (2019)
21. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A very condensed survey and critique of multiagent deep reinforcement learning. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, pp. 2146–2148 (2020)
22. Ho, J., Ermon, S.: Generative adversarial imitation learning. arXiv preprint arXiv:1606.03476 (2016)
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
24. Hodge, V.J., Hawkins, R., Alexander, R.: Deep reinforcement learning for drone navigation using sensor data. Neural Computing and Applications pp. 1–19 (2020)

25. Hu, J., Niu, H., Carrasco, J., Lennox, B., Arvin, F.: Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. IEEE Transactions on Vehicular Technology **69**(12), 14413–14423 (2020)
26. Huang, H., Yang, Y., Wang, H., Ding, Z., Sari, H., Adachi, F.: Deep reinforcement learning for uav navigation through massive mimo technique. IEEE Transactions on Vehicular Technology **69**(1), 1117–1121 (2019)
27. Juliani, A., Berges, V.P., Vckay, E., Gao, Y., Henry, H., Mattar, M., Lange, D.: Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627 (2018)
28. Kang, J.M., Chang, H., Park, B.J.: A simulator for training path-following agents using reinforcement learning. International Journal of Applied Engineering Research **14**(23), 4325–4328 (2019)
29. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Autonomous robot vehicles, pp. 396–404. Springer (1986)
30. Koller, D., Friedman, N., Džeroski, S., Sutton, C., McCallum, A., Pfeffer, A., Abbeel, P., Wong, M.F., Meek, C., Neville, J., et al.: Introduction to statistical relational learning. MIT press (2007)
31. Li, Y.: Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274 (2017)
32. Li, Y., Dai, S., Shi, Y., Zhao, L., Ding, M.: Navigation simulation of a mecanum wheel mobile robot based on an improved a* algorithm in unity3d. Sensors **19**(13), 2976 (2019)
33. Liu, J., Zhang, H., Fu, Z., Wang, Y.: Learning scalable multi-agent coordination by spatial differentiation for traffic signal control. Engineering Applications of Artificial Intelligence **100**, 104165 (2021)
34. Luong, M., Pham, C.: Incremental learning for autonomous navigation of mobile robots based on deep reinforcement learning. Journal of Intelligent & Robotic Systems **101**(1), 1–11 (2021)
35. 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 **518**(7540), 529–533 (2015)
36. Moradi, M.H., Razini, S., Hosseinian, S.M.: State of art of multiagent systems in power engineering: A review. Renewable and Sustainable Energy Reviews **58**, 814–824 (2016)
37. Nawaz, A., Ahmed, S., Khattak, H.A., Akre, V., Rajan, A., Khan, Z.A.: Latest advances in interent of things and big data with requirments and taxonomy. In: 2020 Seventh International Conference on Information Technology Trends (ITT), pp. 13–19. IEEE (2020)
38. Ng, A.Y., Jordan, M.I.: Shaping and policy search in reinforcement learning. Ph.D. thesis, University of California, Berkeley Berkeley (2003)
39. Nguyen, N.D., Nguyen, T., Nahavandi, S.: System design perspective for human-level agents using deep reinforcement learning: A survey. IEEE Access **5**, 27091–27102 (2017)
40. Nguyen, T.T., Nguyen, N.D., Nahavandi, S.: Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE transactions on cybernetics **50**(9), 3826–3839 (2020)
41. Padakandla, S., Prabuchandran, K., Bhatnagar, S.: Reinforcement learning algorithm for non-stationary environments. Applied Intelligence **50**(11), 3590–3606 (2020)
42. Reis, S., Reis, L.P., Lau, N.: Game adaptation by using reinforcement learning over meta games. Group Decision and Negotiation pp. 1–20 (2020)
43. Rizk, Y., Awad, M., Tunstel, E.W.: Decision making in multiagent systems: A survey. IEEE Transactions on Cognitive and Developmental Systems **10**(3), 514–529 (2018)
44. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
45. Shao, K., Zhu, Y., Tang, Z., Zhao, D.: Cooperative multi-agent deep reinforcement learning with counterfactual reward. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
46. Shi, H., Lin, Z., Hwang, K.S., Yang, S., Chen, J.: An adaptive strategy selection method with reinforcement learning for robotic soccer games. IEEE Access **6**, 8376–8386 (2018)
47. Sun, L., Zhai, J., Qin, W.: Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning. IEEE Access **7**, 109544–109554 (2019)

48. Thorndike, E.L.: Animal intelligence: An experimental study of the associate processes in animals. American Psychologist **53**(10), 1125 (1998)
49. Wang, Y., He, H., Sun, C.: Learning to navigate through complex dynamic environment with modular deep reinforcement learning. IEEE Transactions on Games **10**(4), 400–412 (2018)
50. Wang, Z., Wan, Q., Qin, Y., Fan, S., Xiao, Z.: Research on intelligent algorithm for alerting vehicle impact based on multi-agent deep reinforcement learning. Journal of Ambient Intelligence and Humanized Computing **12**(1), 1337–1347 (2021)
51. Wooldridge, M.: An introduction to multiagent systems. John Wiley & Sons (2009)
52. Wu, X., Chen, H., Chen, C., Zhong, M., Xie, S., Guo, Y., Fujita, H.: The autonomous navigation and obstacle avoidance for usvs with anoa deep reinforcement learning method. Knowledge-Based Systems **196**, 105201 (2020)
53. Xiong, H., Diao, X.: Safety robustness of reinforcement learning policies: A view from robust control. Neurocomputing **422**, 12–21 (2021)
54. Zhang, J., Pan, Y., Yang, H., Fang, Y.: Scalable deep multi-agent reinforcement learning via observation embedding and parameter noise. IEEE Access **7**, 54615–54622 (2019)
55. Zhao, D., Wang, H., Shao, K., Zhu, Y.: Deep reinforcement learning with experience replay based on sarsa. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–6. IEEE (2016)