Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2020) 000–000

Procedia
Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2020)
November 2-5, 2020, Madeira, Portugal

Ubiquitous Distributed Deep Reinforcement Learning at the Edge: Analyzing Byzantine Agents in Discrete Action Spaces

Wenshuai Zhao[a,*], Jorge Peña Queralta[a], Li Qingqing[a], Tomi Westerlund[a]

[a] Turku Intelligent Embedded and Robotic Systems Lab, University of Turku, Finland

Abstract

The integration of edge computing in next-generation mobile networks is bringing low-latency and high-bandwidth ubiquitous connectivity to a myriad of cyber-physical systems. This will further boost the increasing intelligence that is being embedded at the edge in various types of autonomous systems, where collaborative machine learning has the potential to play a significant role. This paper discusses some of the challenges in multi-agent distributed deep reinforcement learning that can occur in the presence of byzantine or malfunctioning agents. As the simulation-to-reality gap gets bridged, the probability of malfunctions or errors must be taken into account. We show how wrong discrete actions can significantly affect the collaborative learning effort. In particular, we analyze the effect of having a fraction of agents that might perform the wrong action with a given probability. We study the ability of the system to converge towards a common working policy through the collaborative learning process based on the number of experiences from each of the agents to be aggregated for each policy update, together with the fraction of wrong actions from agents experiencing malfunctions. Our experiments are carried out in a simulation environment using the Atari testbed for the discrete action spaces, and advantage actor-critic (A2C) for the distributed multi-agent training.

Keywords: Reinforcement Learning; Edge Computing; Multi-Agent Systems; Collaborative Learning; RL; Deep RL; Adversarial RL;

1. Introduction

The edge computing paradigm is bringing higher degrees of intelligence to connected cyber-physical systems across multiple domains. This intelligence is being in turn enabled by lightweight deep learning (DL) models deployed at the edge for real-time computation. Among the multiple DL approaches, reinforcement learning (RL) has been increasingly adopted in various types of cyber-physical systems over the past decade, and, in particular, multi-agent

[*] Corresponding author: Wenshuai Zhao.
E-mail address: wezhao@utu.fi

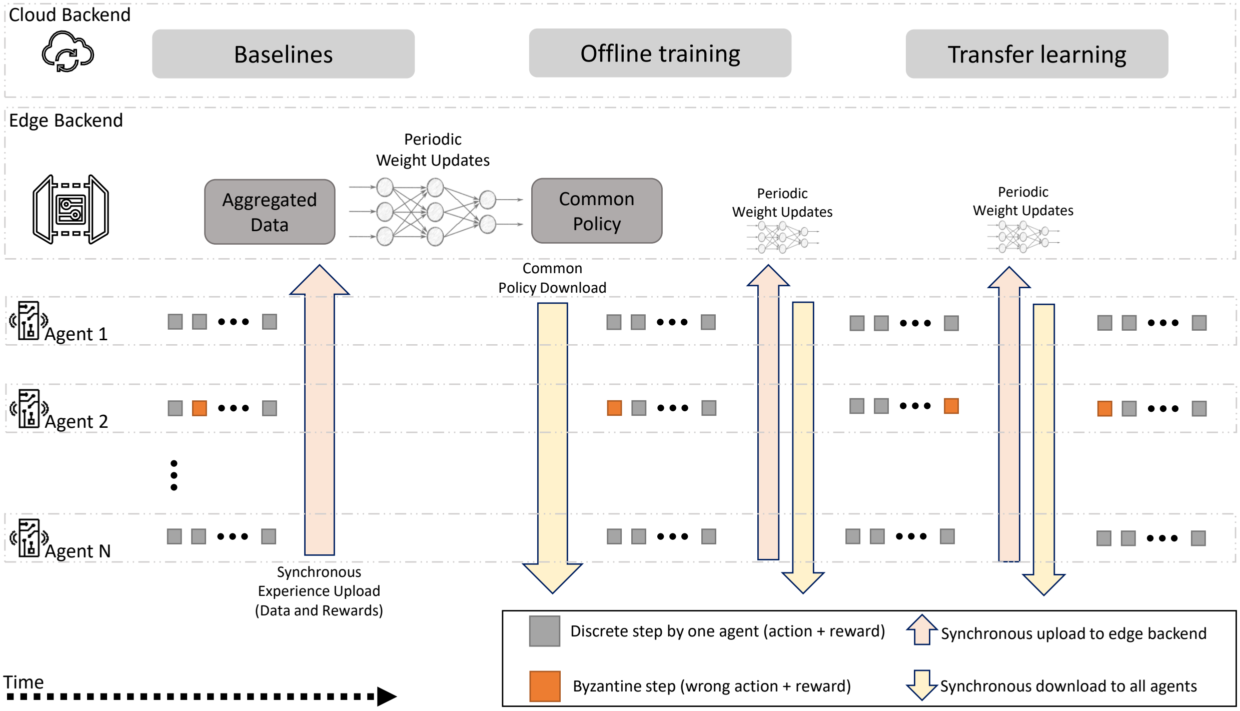


Fig. 1: Conceptual view of the system architecture proposed in this paper. Multiple agents are collaborating towards learning a common task, with the deep neural network updates being synchronized at the edge layer. Each agent, however, individually explores its environments gathering experience in a series of episodes, and calculating the corresponding rewards.

RL [1]. Deep reinforcement learning (DRL) algorithms are motivated by the way natural learning happens: through trial and error, learning from experiences based on the performance outcome of different actions. Among other fields, DRL algorithms have had success in robotic manipulation [2], but also in finding more optimal approaches to complex multi-dimensional problems involved in edge computing [3].

We are particularly interested on how edge computing can enable more efficient real-time distributed and collaborative multi-agent RL. When discussing multi-agent DRL, two different views emerge from the literature: those where multiple agents are utilized to improve the learning process (e.g., faster learning through parallelization [4], higher diversity by means of exploration of different environments [5], or increased robustness with redundancy [1]), and those where multiple agents are learning a policy emerging from an interactive behavior (e.g., formation control algorithms [6], or collision avoidance [7]).

This work explores an scenario where multiple agents are collaboratively learning towards the same task, which is often individual, as illustrated in Fig. 1. Deep reinforcement learning has been identified as one of the key areas that will see wider adoption within the Internet of Things (IoT) owing to the rise of edge computing and 5G-and-beyond connectivity [8]. Multiple application fields can benefit from this synergy in different domains, from robotic agents in the industrial IoT, to information fusion in wireless sensor networks, and including all types of connected autonomous systems and other types of cyber-physical systems.

Multiple challenges still exist in distributed multi-agent DRL. Among the most relevant ones within the scope of this paper are the development of novel techniques to increase robustness in the presence of adversarial agents or perturbed environments [9, 10, 11], as well as closing the simulation-to-reality gap [2, 12]. In relation to the former area, recent works have focused on exploring different types of noise or perturbations in the agents or environments to better understand how these potentially adversarial conditions affect the collaborative learning process. For example, Gu et al. study in [13] the effect of network delays and propose an asynchronous method for off-policy updates, while Yu et al. have studied the effect of adversarial conditions in the network connection between the agents [14]. We have seen a lack of research, however, on the analysis of adversarial conditions in discrete action spaces. These type of scenarios occur when agents need to make a decision from a finite and discrete set of actions.

In this paper, we study the effect of byzantine agents that perform the wrong action with certain probabilities and report initial results that let us understand the limitations of the state-of-the-art in multi-agent RL in the presence of byzantine agents for discrete action spaces. In particular, we utilize the synchronous advantage actor-critic (A2C) algorithm on two of the standard Atari environments typically used for benchmarking DRL methods. This is, to the best of our knowledge, the first paper analyzing the effects terms of policy convergence in collaborative multi-agent DRL caused by having different fractions of wrong actions in discrete action spaces. Our results show that in some environments with totally 16 distributed agents the training process is highly sensitive to having a single agent acting in the wrong manner over a relatively small fraction of its actions, and unstable convergence appears with just a single agent having 2.5% of its actions wrong. In other environments the threshold is higher, with the network still converging to a working policy at over 10% of wrong actions in a single byzantine agent.

The remainder of this document is organized as follows. Section 2 presents related works in the area of adversarial RL and applications combining RL with edge computing. Section 3 describes the basic theory behind A2C and the simulation environments. Section 4 then presents our results on the convergence of the system when a fraction of the actions is wrong, and Section 5 concludes the work and outlines our future work directions.

2. Related Works

Adversarial RL has attracted many researchers' interest in recent years. Multiple deep learning algorithms are known to be vulnerable to manipulation by perturbed inputs [9]. This problem also affects various reinforcement learning algorithms under different scenarios. In multi-agent environments, an attacker can significantly increase the adversarial observations ability [10]. Ilahi et al. review emerging adversarial attacks in DRL-based systems and the potential countermeasures to defend against these attacks [15]. The authors classify the attacks as attacks targeting (i) rewards, (ii) policies, (iii) observations, and (iv) the environment. In this paper, instead, we consider targeting the agents' actions, which can happen in real-world applications when agents interact with their environment.

Similarly considering how to better transfer learning from simulations to the real-world, multiple researchers have been working on a simulation-to-reality transfer for specific applications in different environments [12, 2, 16]. In this paper, we will analyze the effect of the adversarial of byzantine effects in multi-agents reinforcement learning, and introduce a fraction of byzantine actions, which has not been studied before.

Other researchers have explored the influence of noisy rewards in RL. Wang et al. present a robust RL framework that enables agents to learn in noisy environments where only perturbed rewards are observed, and analyzes different algorithms performance under their proposed framework, including PPO, DQN, and DDPG [11]. In this paper, the perturbances on the DRL process will be explored, but we focus on analyzing discrete action spaces and a fraction of byzantine actions performed by a small number of byzantine agents.

Multiple works have been presented in the convergence of DRL and edge computing. However, rather than focusing on exploiting edge computing for distributed RL, most of the current literature is exploiting RL for edge service. For instance, Ning et al. apply DRL for more efficient offloading orchestration [3], while Wang et al. have applied DRL to optimize resource allocation at the edge [17]. In our work, however, we focus on analyzing some of the challenges that can appear when edge computing is exploited for distributed multi-agent DRL in real-world applications.

3. Methodology

This section describes the methods and simulation environments utilized for the analysis in Section 4: the advantage actor-critic (A2C) algorithm for distributed DRL, and the simulation environment.

Actor-critic methods combine the advantages of value based and policy based methods, and has been regarded as the base of many modern RL algorithms. In A2C, two neural networks represent the actor and critic, where the actor controls the agent's behavior and the critic evaluate how good the action taken is. As value-based methods tend to high variability, an advantage function is employed to replace the raw value function, leading to advantage actor-critic (A2C). The main scheme of the policy gradient updates is shown in (1):

$$\theta^{new} \leftarrow \theta^{old} + \eta \nabla \overline{R}_\theta \tag{1}$$

where θ denotes the policy to be learned, η is the learning rate, and $\nabla\overline{R}_\theta$ represents the policy gradient, given by (2).

$$\nabla\overline{R}_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n)\nabla logp(a_t^n|s_t^n,\theta) \tag{2}$$

where N is the number of trajectories sampled under the policy τ, and $R(\tau^n)$ denotes the accumulated reward for each episode consisting of T_n steps. In the policy with weight θ, an action a_t^n under the state s_t^n is chosen with probability $p(a_t^n|s_t^n,\theta)$.

In this policy gradient method, the accumulated reward $R(\tau^n)$ is calculated by sampling the trajectories, which are computed when the whole episode is finished, and hence might bring high variability affecting the policy convergence. To avoid this, value estimation is introduced and merged into the policy gradient method. An advantage function is thus proposed to replace $R(\tau^n)$ according to (3), which is also the reason for the name of A2C.

$$R(\tau^n) = r_t^n + V^\pi(s_{t+1}^n) - V^\pi(s_t^n) \tag{3}$$

where r_t is the reward gained in the step t, and V^π denotes the value function to estimate the accumulated reward that will be gained. Additionally, in the implementation of this A2C algorithm, multiple agents are employed to produce the trajectories in parallel. Compared to A3C [4], in which each agent will update the network individually and asynchronously, A2C collects the whole data from each agents and then update the shared network. This is also illustrated in Fig. 1.

In order to analyze the effect of byzantine actions, we choose two typical gym-wrapped Atari games as our simulation environments: PongNoFrameskip-v4 (Fig. 2b) and BreakoutNoFrameskip-v4 (Fig. 3b). Both of them take video as an input, based on which the policy will be trained to produce the corresponding discrete actions to obtain higher rewards. The action spaces for Pong and Breakout have cardinality of 5 and 4, respectively. We set their corresponding byzantine agents to behave with the opposite actions (e.g., if the output action from the policy is $action = 2$ in Pong, then a wrong action by the byzantine agents will be $action = 3$). In this paper, we consider the effect of the presence of Byzantine agent in terms of their number and the frequency of wrong actions they perform. The patterns we observe in the experiments can be further utilized to detect Byzantine agents in distributed multi-agent DRL scenarios.

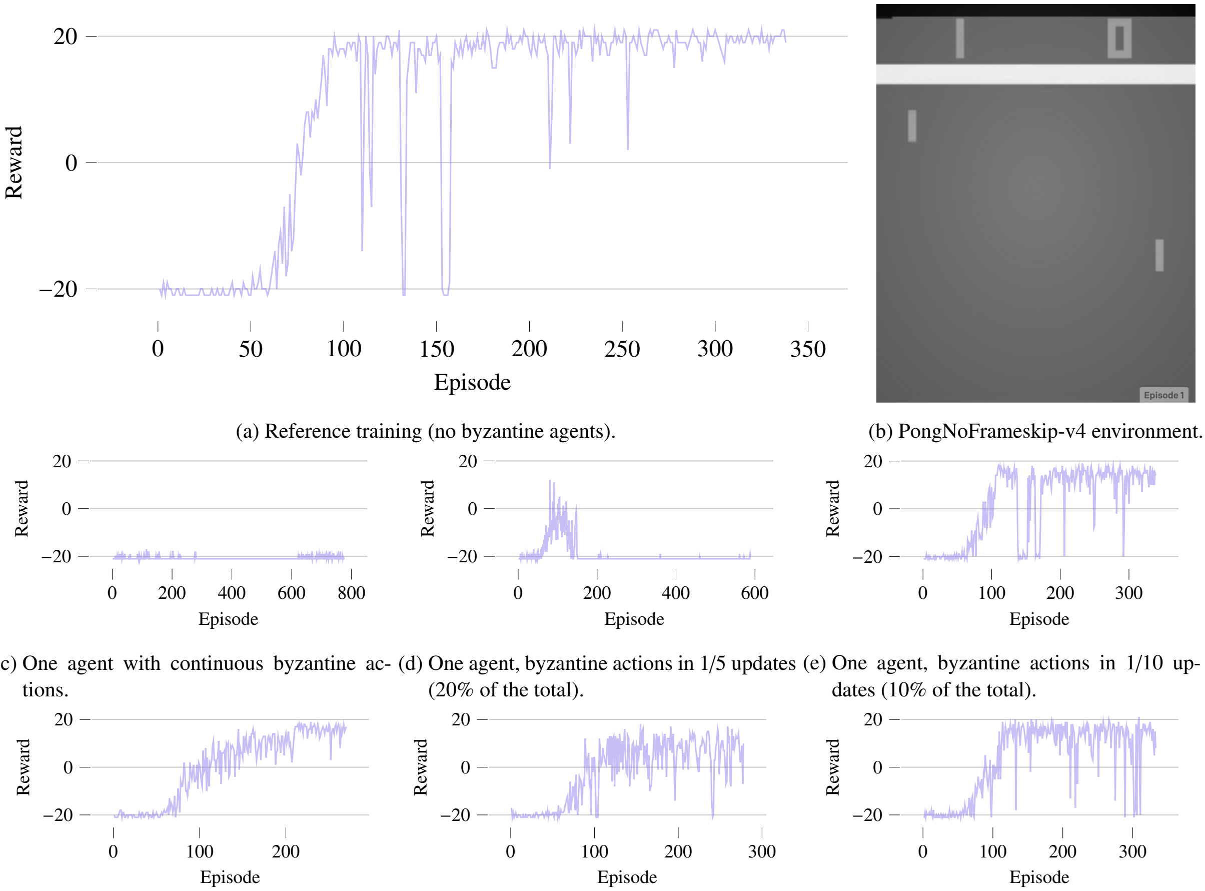
4. Simulations and Results

In this section, we describe the settings in our experiments and present the main conclusions of our analysis. There are totally 16 agents or workers employed to produce trajectory data both in Pong and Breakout environments. The experiences, actions and rewards from the different agents are then aggregated synchronously to calculate the policy gradients and update the policy towards a more optimal one. In the experiments, we analyze how byzantine agents that perform wrong actions unknowingly affect the collaborative learning effort. The reference training without byzantine agents for the Pong and Breakout environments are shown in Fig. 2a and Fig. 3a, respectively.

In the Pong environment, we first set a single agent continuously behaving wrongly, out of the total of 16 agents working in parallel (Fig. 2c). Compared to the reference training, we observe that the policy is unable to improve in order to obtain better rewards. Therefore, a single byzantine agent representing as little as 6.25% of the total is enough to completely disable the ability of the system to converge towards a working policy. Therefore, we have focused on analyzing the maximum fraction of wrong actions that byzantine agents can perform in order to ensure convergence of the system. Moreover, in order to test whether it is the total fraction what matters or the number of agents, we have considered the same total fraction of byzantine actions in different settings.

In the training, the policy is updated only when the agents perform a series of steps, collecting a certain amount of interaction data. In particular, agents perform 5 steps between updates of the policy. This leads to 80 steps between updates, and we set the total number of steps to 10^7 for the complete training process. The number of episodes depends on the performance of the agents (the better the performance, the longer the episodes are).

With this, we set different fractions of byzantine actions depending on (i) the number of agents, (ii) the number of wrong actions in between updates, and (iii) the fraction of updates affected by byzantine actions. The results for the Pong environment are shown in Figures 2d through 2h. From these, we conclude that 20% of byzantine actions are enough to deplete the systems' ability to converge, while the system is able to converge with slight unstabilities in the presence of a 10% of byzantine actions (Figures 2e, 2g and 2h). Finally, with just 5% of byzantine actions (Fig. 2f) the convergence is similar to the reference.



(a) Reference training (no byzantine agents).

(b) PongNoFrameskip-v4 environment.

(c) One agent with continuous byzantine actions.

(d) One agent, byzantine actions in 1/5 updates (20% of the total).

(e) One agent, byzantine actions in 1/10 updates (10% of the total).

(f) One agent, byzantine actions in 1/2 steps of 1/10 updates (5% of the total).

(g) Two agents, byzantine actions in 1/2 steps of 1/10 updates (2 × 5% of the total).

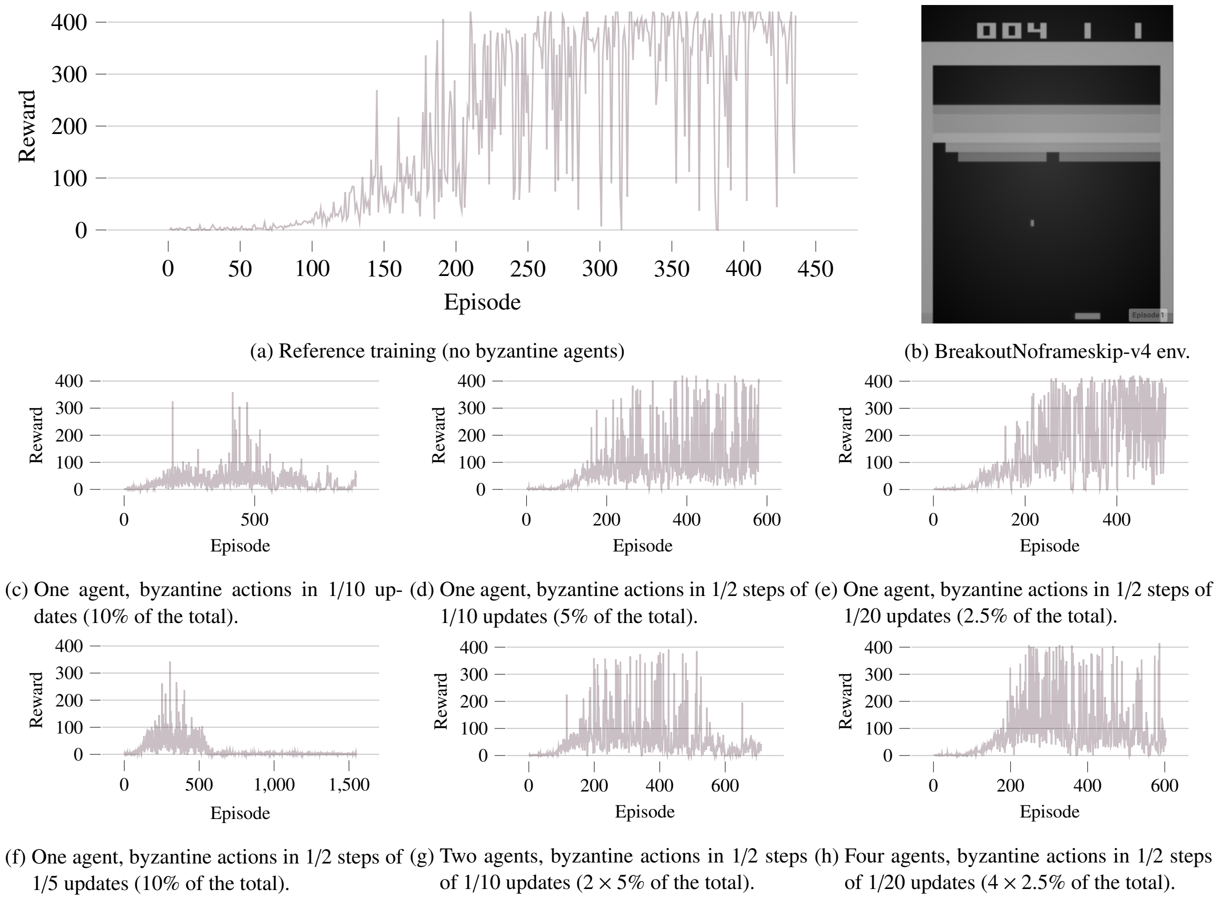
(h) Four agents, byzantine actions in 1/2 steps of 1/20 updates (4 × 2.5% of the total).

Fig. 2: Experiments in the PongNoFrameskip-v4 environment.

In addition, we also conduct similar experiments on another classical Atari game, BreakoutNoFrameskip-v4. The results with byzantine actions are shown in Figures 3c through Figure 3h. In this environment, 10% byzantine actions are enough to deter convergence (Fig. 3c, Figure 3f, and Figure 3h). Only by reducing the frequency to 2.5% can the training get acceptable convergence (Figure 3e). A general conclusion is also that the total fraction of byzantine actions is what matters the most, and not how they are introduced in the system.

5. Conclusion

Applying distributed multi-agent deep reinforcement learning in real-world scenarios requires further study of how adversarial conditions affect collaborative learning efforts. We have done this as an initial step towards distributed DRL applications falling under the umbrella of opportunities that the edge computing paradigm and next-generation mobile networks are bringing to the IoT and a myriad of cyber-physical systems. In this work, we have explored the performance of the state-of-the-art A2C algorithm in two different simulation environments and reported initial results analyzing the ability of the systems to converge in the presence of byzantine agents. In both environments, the agents were collaboratively learning a policy for discrete action spaces. Our study has focused on analyzing how different fractions of wrong actions performed unknowingly by byzantine agents affect the collaborative learning effort. These results will serve to prepare future research against such adversities. In our future works, we will focus on building methods to detect the byzantine agents and therefore provide a more robust collaborative learning framework.



(a) Reference training (no byzantine agents)

(b) BreakoutNoframeskip-v4 env.

(c) One agent, byzantine actions in 1/10 updates (10% of the total).

(d) One agent, byzantine actions in 1/2 steps of 1/10 updates (5% of the total).

(e) One agent, byzantine actions in 1/2 steps of 1/20 updates (2.5% of the total).

(f) One agent, byzantine actions in 1/2 steps of 1/5 updates (10% of the total).

(g) Two agents, byzantine actions in 1/2 steps of 1/10 updates (2 × 5% of the total).

(h) Four agents, byzantine actions in 1/2 steps of 1/20 updates (4 × 2.5% of the total).

Fig. 3: Experiments in the BreakoutNoFrameskip-v4 environment.

References

[1] T.T. Nguyen *et al.* Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 2020.

[2] J. Matas *et al.* Sim-to-real reinforcement learning for deformable object manipulation. *arXiv*, 2018.

[3] Z. Ning *et al.* Deep reinforcement learning for vehicular edge computing: An intelligent offloading system. *ACM TIST*, 10(6), 2019.

[4] V. Mnih *et al.* Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 2016.

[5] R. Raileanu *et al.* Modeling others using oneself in multi-agent reinforcement learning. *arXiv preprint arXiv:1802.09640*, 2018.

[6] R. Conde *et al.* Time-varying formation controllers for unmanned aerial vehicles using deep reinforcement learning. *arXiv*, 2017.

[7] P. Long *et al.* Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *ICRA*. IEEE, 2018.

[8] J. Peña Queralta *et al.* Enhancing autonomy with blockchain and multi-acess edge computing in distributed robotic systems. In *The Fifth International Conference on Fog and Mobile Edge Computing (FMEC). IEEE*, 2020.

[9] V. Behzadan *et al.* Vulnerability of deep reinforcement learning to policy induction attacks. In *MLDM*, 2017.

[10] A. Gleave *et al.* Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[11] J. Wang *et al.* Reinforcement learning with perturbed rewards. In *AAAI*, pages 6202–6209, 2020.

[12] B. Balaji *et al.* Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning. *arXiv:1911.01562*, 2019.

[13] S. Gu *et al.* Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA*. IEEE, 2017.

[14] Y. Yu *et al.* Multi-agent deep reinforcement learning multiple access for heterogeneous wireless networks with imperfect channels. *arXiv preprint arXiv:2003.11210*, 2020.

[15] I. Ilahi *et al.* Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *arXiv preprint arXiv:2001.09684*, 2020.

[16] K. Arndt *et al.* Meta reinforcement learning for sim-to-real domain adaptation. *arXiv*, 2019.

[17] J. Wang *et al.* Smart resource allocation for mobile edge computing: A deep reinforcement learning approach. *IEEE Transactions on emerging topics in computing*, 2019.