Towards Closing the Sim-to-Real Gap in Collaborative Multi-Robot Deep Reinforcement Learning

Wenshuai Zhao[1], Jorge Peña Queralta[1], Li Qingqing[1], Tomi Westerlund[1]

[1] Turku Intelligent Embedded and Robotic Systems Lab, University of Turku, Finland
Emails: [1]{wezhao, jopequ, qingqli, tovewe}@utu.fi

***Abstract*—Current research directions in deep reinforcement learning include bridging the simulation-reality gap, improving sample efficiency of experiences in distributed multi-agent reinforcement learning, together with the development of robust methods against adversarial agents in distributed learning, among many others. In this work, we are particularly interested in analyzing how multi-agent reinforcement learning can bridge the gap to reality in distributed multi-robot systems where the operation of the different robots is not necessarily homogeneous. These variations can happen due to sensing mismatches, inherent errors in terms of calibration of the mechanical joints, or simple differences in accuracy. While our results are simulation-based, we introduce the effect of sensing, calibration, and accuracy mismatches in distributed reinforcement learning with proximal policy optimization (PPO). We discuss on how both the different types of perturbances and how the number of agents experiencing those perturbances affect the collaborative learning effort. The simulations are carried out using a Kuka arm model in the Bullet physics engine. This is, to the best of our knowledge, the first work exploring the limitations of PPO in multi-robot systems when considering that different robots might be exposed to different environments where their sensors or actuators have induced errors. With the conclusions of this work, we set the initial point for future work on designing and developing methods to achieve robust reinforcement learning on the presence of real-world perturbances that might differ within a multi-robot system.**

***Index Terms*—Reinforcement Learning; Multi-Robot Systems; Collaborative Learning; Deep RL; Adversarial RL; Sim-to-Real;**

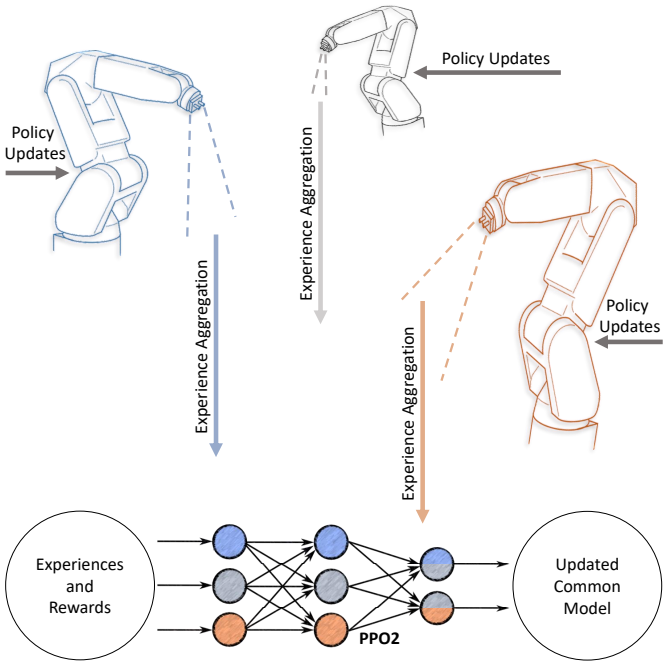


Fig. 1: Conceptual view of the proposed scenario, where multiple robotic agents are collaboratively learning the same task. While the task is common, and the agents are a priori identical, we study how different alterations in the agents or their environments affects the performance of the collaborative learning process.

I. Introduction

Reinforcement learning (RL) algorithms for robotics and cyber-physical systems have seen an increasing adoption across multiple domains over the past decade [1], [2]. Deep reinforcement learning (DRL) enables agents to be trained in realistic environments without the need for large amounts of data to be gathered and labeled a priori. Specifically, reinforcement learning has enjoyed significant success in robotic control tasks involving manipulation [3], [4], [5]. Motivated by the way humans and animals learn, DRL algorithms work on a *trial and error* basis, where an agent interacts with its environment and receives a reward based on its performance. When complex agents or environments are involved, the learning process can be relatively slow. This has motivated the design and development of multi-agent DRL algorithms. In this paper, we are interested in exploring some of the challenges remaining in multi-robot collaborative DRL.

Reinforcement learning applied to multi-agent systems has two dimensions: DRL algorithms that model policies for multi-agent control and interaction, and DRL approaches that rely on multiple agents to parallelize the learning process or explore a wider variety of experiences. Within the former category, we can find examples of DRL for formation control [6], obstacle and collision avoidance [7], [8], collaborative assembly [9], or cooperative multi-agent control in general [10]. In the latter category, most existing approaches refer to the utilization of multiple agents to learn in parallel, but from the point of view of a multi-process or multi-threaded application [3]. We are interested in works exploring the possibilities of using multiple robotic agents that collaborate on learning the same task. This has been identified as one of the future paradigms with 5G-and-beyond connectivity and edge computing [11]. For instance, in [12] an asynchronous method for off-policy updates between robots was presented. Other works also consider network conditions and propose frameworks for multi-agent collaborative DRL over imperfect network channels [13]. This type of scenario is illustrated in Fig. 1, where three robotic arms are collaboratively learning the same task and sharing their experiences to update a common policy. Hereinafter, we refer to these types of scenarios as multi-agent or multi-robot

collaborative RL tasks, where multiple agents collaborate to learn the same task but might be exposed to different environments, or work under different conditions.

Among the multiple challenges in DRL, recent years have seen a growing research interest in closing the simulation-to-reality gap [5], [14], and on the design and development of robust algorithms with resilience against adversarial conditions [15], [16], [17]. This latter topic is also of paramount relevance in distributed or multi-agent DRL, where adversarial agents can hinder the collaborative learning process [18]. When multiple agents are learning a collaborative or coordinated behavior, byzantine agents can significantly reduce the performance of the system as a whole.

We aim at studying how adversarial conditions can help to bridge the simulation-to-reality gap. In [5] and [14], the authors analyze perturbances in the rewards towards the applicability of DRL in real-world applications. In [5], the focus is on learning how to manipulate deformable objects, with agents trained in a simulation environment but directly deployable in the real-world. In [19], the authors present a meta-learning approach for domain adaption in simulation-to-reality transfers. Our objective in this paper is not to design a specific sim-to-real method for a given algorithm or task, but instead to analyze the performance of collaborative multi-robot DRL in the presence of disturbances in the environment as a step towards more effective sim-to-real transfers where real noises, errors or perturbances are accounted for also in the simulation environment. This includes variability in the operation of the robots, as robots might be operating in slightly different environments, or operate in different ways under the same environment. In particular, we are interested in studying how exposing multiple collaborative robots to different environments from the point of view of sensing and actuation can affect the joint learning effort.

In this paper, therefore, we focus on introducing perturbances inspired by real-world cases in a multi-agent DRL simulation. We expose different subsets of agents to slightly modified environments and study how different types of disturbances affect the collaborative learning process and the ability of the multi-robot system to converge to a common policy. The main contribution of this paper is the analysis of how input and output disturbances affect a collaborative deep reinforcement learning process with multiple robot arms. In particular, we simulate real-world perturbations that can occur on robotic arms, from the sensing and actuation perspectives. This is, to the best of our knowledge, the first study to consider the evaluation of both sensing and actuation disturbances in a multi-robot collaborative learning scenario, with different robots being exposed to different environments.

The remainder of this document is organized as follows. In Section II we review the literature in distributed RL, adversarial RL, and robust multi-agent RL in the presence of byzantine agents. Then, Section III introduces the DRL algorithm, and the methodology and simulation environment utilized in our experiments. The agent training methods and environment disturbances introduced to emulate real-world operational variability, together with the simulations results, are presented in Section IV. Section V concludes the work.

II. Related Works

In this work, we study adversarial conditions in a simulation environment to emulate real-world conditions in terms of variability of the environment across a set of multiple agents collaborating in learning the same task. With most of the literature in simulation-to-reality transfer aiming at specific applications or adaptation to different environments [14], [5], [19], in this section we focus instead on previous works analyzing the effect of adversarial of byzantine effects in multi-agent reinforcement learning, as well as considering other perturbations in the environment to better emulate real-world conditions. The literature in adversarial conditions for collaborative multi-agent learning is, nonetheless, sparse.

Adversarial RL has been a topic of extensive study over the past years. Multiple deep learning algorithms have been shown to be vulnerable when subject to adversarial input perturbations, being able to induce certain policies [15]. This is a general problem of reinforcement learning that affects different types of algorithms and scenarios. In multi-agent environments, the ability of an attacker to create adversarial observations increases significantly [16]. A comprehensive survey on the main challenges and potential solutions for adversarial attacks on DRL is available in [20]. The authors classify attacks in four categories: attacks targeting (i) rewards, (ii) policies, (iii) observations, and (iv) the environment. Among these, those targeting observations and the environment are the most relevant within the scope of this survey. In most of these cases, however, the literature only considers single-agent learning (or multiple agents being affected in the same way). Moreover, previous works focus on malicious perturbations aimed at decreasing the performance of the learning agent. In this paper, nonetheless, we induce perturbations that are inspired by real-world issues including changes in accuracy or calibration errors.

Other authors have explored the effects of having noisy rewards in RL. In this direction, Wang et al. presented an analysis of perturbed rewards for different RL algorithms, including PPO but also DQN and DDPG, among others [17]. Compared to their approach, we also consider perturbances on the RL process but focus on those that model real-world noises and errors. Moreover, we specifically put an emphasis on multi-robot collaborative learning, and consider situations in which the perturbances that affect different robots are also different. We also focus on the PPO algorithm as the state-of-the-art in three-dimensional locomotion. In fact, PPO has been identified as one of the most robust approaches against reward perturbances in [17]. Also within the study of noisy rewards, a method to improve performance in such scenarios is proposed in [21].

In general, we see a gap in the literature in the study of noisy or perturbed environments that do not affect equally across multiple agents collaborating towards learning the same task. This paper thus tries to address this issue with an initial assessment of how perturbations in the environment influencing a subset of agents affect a global common model where experiences from different agents are aggregated.

III. Methodology

In this section, we define our problem of distributed reinforcement learning with a subset of perturbed agents, as well as the simulation environment and modifications applied to it.

A. Multi-agent RL

In multi-agent reinforcement learning, approaches can be roughly divided into two parallel modes, asynchronous and synchronous. A3C (Asynchronous Advantage Actor-Critic)[3] is one of the most widely adopted methods for multi-agent reinforcement learning, representing the asynchronous paradigm. A3C consists of multiple independent agents with their own networks. These agents interact with different copies of the environment in parallel and update a global network periodically and asynchronously.After each update, the agents reset their own weights to those of the global network and then resume their independent exploration.Because some of the agents will be exploring the environment with an older version of the network weights, A3C results in relatively suboptimal use of computational resources as well as more noisy updates. An alternative is A2C (Advantage Actor-Critic), which utilizes synchronous parallel mode. In this case, there are only two networks in the system. One is used by all agents equally to interact with the environment in parallel, and outputs a batch of experiences. With this data, the second network is trained and updates the former network.

In this paper, we utilize a synchronous multi-agent reinforcement learning algorithm: proximal policy optimization (PPO). PPO and has been adopted as the default method of OpenAI owing to its excellent performance. The PPO algorithm takes advantage of the A2C ideas in terms of having multiple workers, and gradient policy ideas from TRPO (Trust Region Policy Optimization) to improve the actor performance by utilizing a trust region. PPO seeks to find a balance between the ease of implementation, sample complexity, and ease of adjustment, trying to update at each step to minimize the cost function while assuring that the new policies are not far from last policies. The scheme follows these steps:

1) Set the initial policy parameters θ^0.
2) In each iteration, use θ^k to interact with the environment, collect experience data (a tuple of state and action $\{s_t, a_t\}$), and compute their advantage $A^{\theta^k}(s_t, a_t)$ [3].
3) Find the optimal θ by optimizing $J_{PPO}(\theta)$:

$$J^{\theta^k}_{PPO}(\theta) = J^{\theta^k}(\theta) - \beta KL\left(\theta, \theta^k\right) \tag{1}$$

where β is a hyperparameter and will be adapted according to the value of KL. $J^{\theta^k}(\theta)$ is calculated by:

$$J^{\theta^k}(\theta) \approx \sum_{(s_t,a_t)} \frac{p_\theta(a_t|s_t)}{p_{\theta^k}(a_t|s_t)} A^{\theta^k}(s_t, a_t) \tag{2}$$

where $p_{\theta^k}(a_t|s_t)$ is the probability of (s_t, a_t) under θ^k.

B. Simulation Environment

Our simulation environment is built based on top one of the Bullet physics simulators, specifically the PyBullet Kuka arm for grasping [22]. In order to simplify the training of our RL algorithm, we modify the original grasping task into a reaching task, which allows us to focus on observing the effect of adversarial agents in training distributed reinforcement learning algorithms, rather on optimizing the training itself.

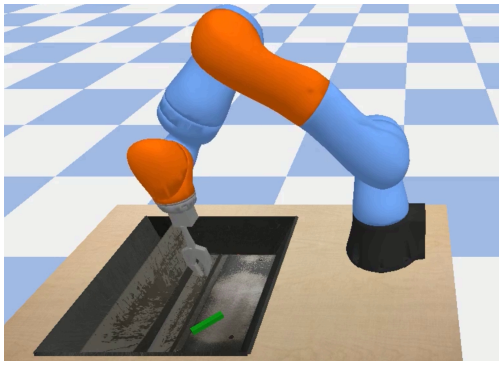


Fig. 2: Kuka arm reaching environment based on Bullet simulator.

The simulation environment is shown in Figure 2. The robotic arm in this environment attempts to reach the object in the bin. It takes the Cartesian coordinates of the gripper and the relative position of the object as input instead of the on-shoulder camera observation. This input can be seen as a list with nine elements:

$$Input = [x_g, y_g, z_g, yaw_g, pit_g, rol_g, x_{og}, y_{og}, rol_{og}] \tag{3}$$

where x_g, y_g, z_g denote the Cartesian coordinates of the center of the gripper, and yaw_g, pit_g, rol_g refers to its three Euler angles, while x_{og}, y_{og}, rol_{og} indicate the relative x, y position and the roll angle of the object in the gripper space.

Our RL algorithm receives the input and then gives a Cartesian displacement:

$$Output = [dx, dy, dz, d\phi] \tag{4}$$

in which ϕ is the rotation angle of the wrist around the z-axis. An inverse kinematics method is then employed to calculate the real motor control values of the joints. Note that all the units used for the position are in meters, and the angles are in radians. This environment with our training code is now open-source on Github[1].

C. Collaborative Learning under Real-World Perturbations

In real robots, some of the most characteristic sources of perturbations within a homogeneous multi-robot team come from the calibration of the robots in terms of sensing and actuation. In this paper, we thus study how these two types of input (sensing) and output (actuation) perturbations affect a collaborative learning process:

Input perturbations: we consider both fixed and variable errors in the input to the network regarding the position of the object to be reached. This emulates the error that might result from identifying the position of the object from a camera or another sensor on the robot arm. The fixed noise represents, for instance, installation or calibration errors on the position of the camera, which might have an offset in position or orientation.

[1]https://github.com/TIERS/NoisyKukaReacher

Variable errors, on the other hand, try to emulate the sensing errors that come, for example, from the vibration of the arm or local odometry errors describing its orientation and position.

Output perturbations: we simulate both fixed and variable perturbations in the actuation of the robotic arm, emulating calibration errors (e.g., a constant offset in one direction), or changes in accuracy or repeatability across different robots.

Through multiple simulations, we study how these types of perturbations affect the collaborative learning effort when they are not common across the entire set of agents.

IV. Experimentation and Results

In this section, we describe the training parameters utilized through our simulations, and the ways in which the environments have been modified to introduce disturbances in both sensors and actuators. We then present the results of multiple simulations where different numbers of agents have been trained in different environments but treated equally from the point of view of the collaborative learning process.

A. Training Method

The maximal number of steps in one episode is set to 1200, and the maximum number of steps for the whole training process is set to 4 million. If the gripper contacts the object or approaches it at a very small distance (0.008 m), the episode will be terminated. The final score for this episode is thus calculated by summing all the rewards obtained in all the steps until termination.

The initial reward is set as -1000 for each step if the distance is larger than 1 m. However, if the distance between the finger on the gripper and the object is smaller than 1 m, the reward is computed as $reward_{raw} = -10 \cdot distance$. Moreover, we also add the cost of each step, in order to encourage the gripper to approach the target as soon as possible. The cost of each step is set as 1. Therefore, the final reward for this step is hence: $reward_{final} = -10 \cdot distance - 1$, where the distance is given in meters. If the gripper finally contacts its target or approach it in a threshold, we give it a significantly larger reward (1000) to help the model learn faster and clearly.

In total, in our simulations, we utilize 30 agents parallelized on the GPU processes to produce experience data based on a vectorized environment. Therefore, these agents can represent a multi-robot system learning a collaborative RL task. We give different settings on individual environments to manually simulate the possible perturbations that robots find in real-world scenarios.

B. Calibration and Accuracy Noises

To emulate the practical noises and errors that robots could encounter when training an RL algorithm, we consider the following four types of perturbations, for each of which we generate a different environment to expose a variable number of robots to: fixed input errors on all the nine elements by $0.005\,m$, uniformly distributed sensing errors in the interval $[0.005\,m, 0.01\,m]$, fixed output errors modifying the gripper actuators by an offset of $0.005\,m$ on the x axis, and uniformly distributed output errors in the interval $[0.005\,m, 0.01\,m]$. It should be noted that the uniform distributed errors on input and output could be different in each step, which can be regarded as inaccurate sensing errors, or reduced repeatability in the actuation of real robots.

Moreover, in order to further analyze how more extreme cases affect the collaborative learning process, we also consider fixed disturbances on larger magnitude (0.015 m on all the values for the sensing error and 0.015 m on the x-axis for the actuation error) as well as scenarios where the noise is different for each of the agents exposed to the modified environment (in the interval 0.005 m to 0.025 m for 25 agents).

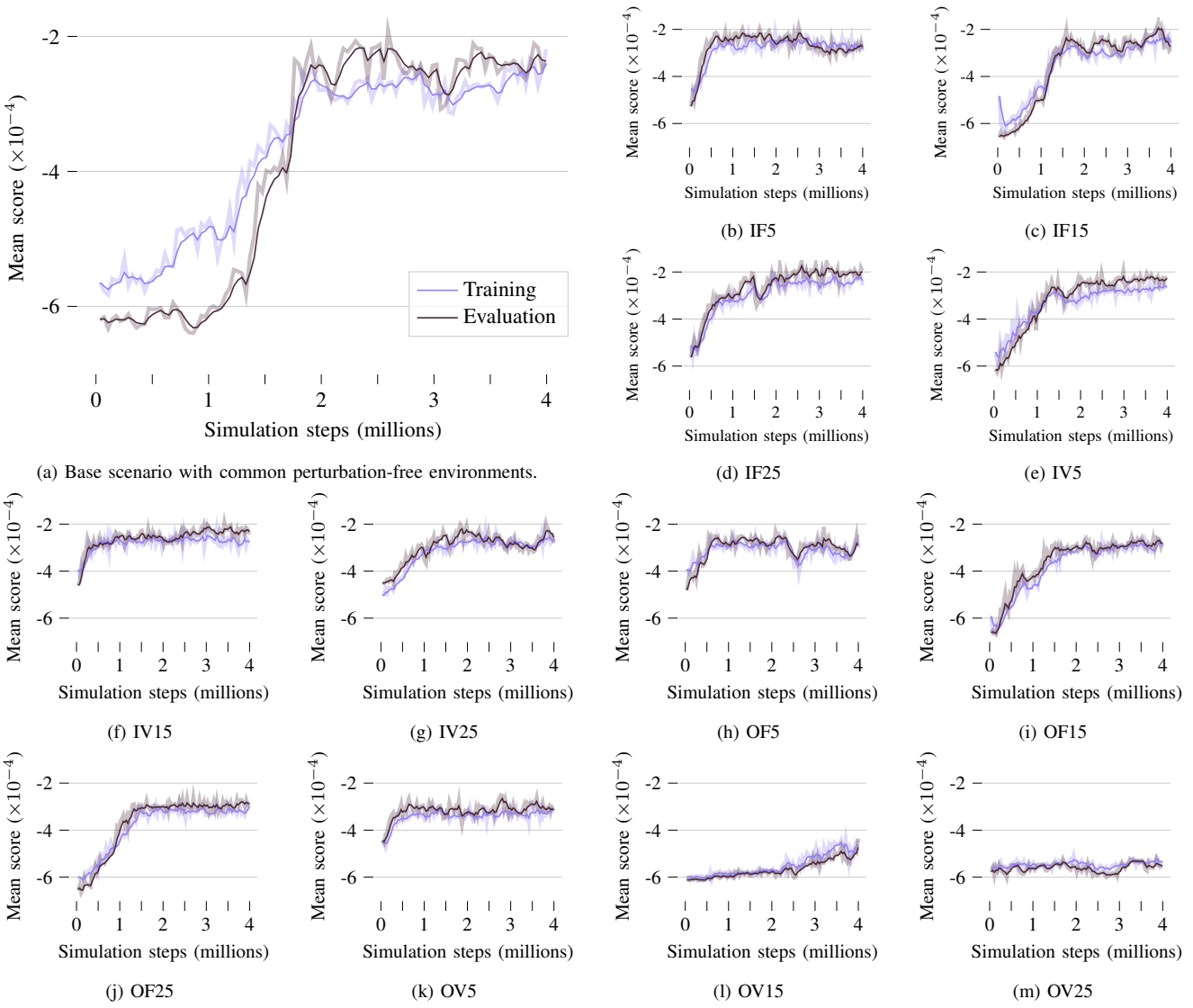
C. Simulation Results

Figure 3 shows the results of our simulations. The notation describing each subfigure is as follows: {I,O}: representing the input (sensing) and output (actuation) perturbances, {F,V}: representing fixed and variable perturbances, and {5, 15, 25}: representing the number of agents exposed to the modified environment where the perturbances occur.

Comparing perturbations in the sensors versus perturbations in the actuators, we see an overall more robust performance against adversarial elements in the sensing part. In Figures 3b to 3d, we see that the network always converges and we only see more unstable behaviour when there is a large fraction of agents suffering of variable sensing errors (50% and 83%). When we compare the effect of constant or fixed perturbations against variable ones, we notice that variable perturbations induce less stable convergence. This can be to some extent explained by the fact that there are no large subsets of agents being exposed to a common environment.

For small fixed perturbations affecting actuation (output disturbances), we have seen that the agents are able to converge towards a working policy. In the cases where 5 or 25 of the agents are affected, this was expected as there is a majority (25) of agents, in both cases, that work in exactly the same way, and a small subset (5) that work in a slightly different way (but still the same within that subset). When this fixed perturbation is introduced to half of the agents, then we have two subsets of the same size operating in different ways, but again consistently across each of the subsets. In this case, we have seen that for a small magnitude in the perturbation, the agents still converge on a policy that works for both subsets. As the difference between the operation of the agents in these two subsets diverges, the performance of the system as a whole drops significantly. Nonetheless, we have observed that the case were half of the agents have a common fixed perturbation of small magnitude the system is able to converge even when the initial conditions are disadvantageous.

In order to analyze the effect of perturbations with larger magnitude as well as fixed perturbations in both sensing and actuation that vary across the robots exposed to a modified environment, we have analyzed four more cases shown in Fig. 4. In Figures 4a and 4b, we analyze how perturbations with larger magnitude affect the learning process, with half of the agents being affected as the worst-case scenario. We see that the trend from the previous results is followed, with the



(a) Base scenario with common perturbation-free environments.

(b) IF5 (c) IF15 (d) IF25 (e) IV5

(f) IV15 (g) IV25 (h) OF5 (i) OF15

(j) OF25 (k) OV5 (l) OV15 (m) OV25

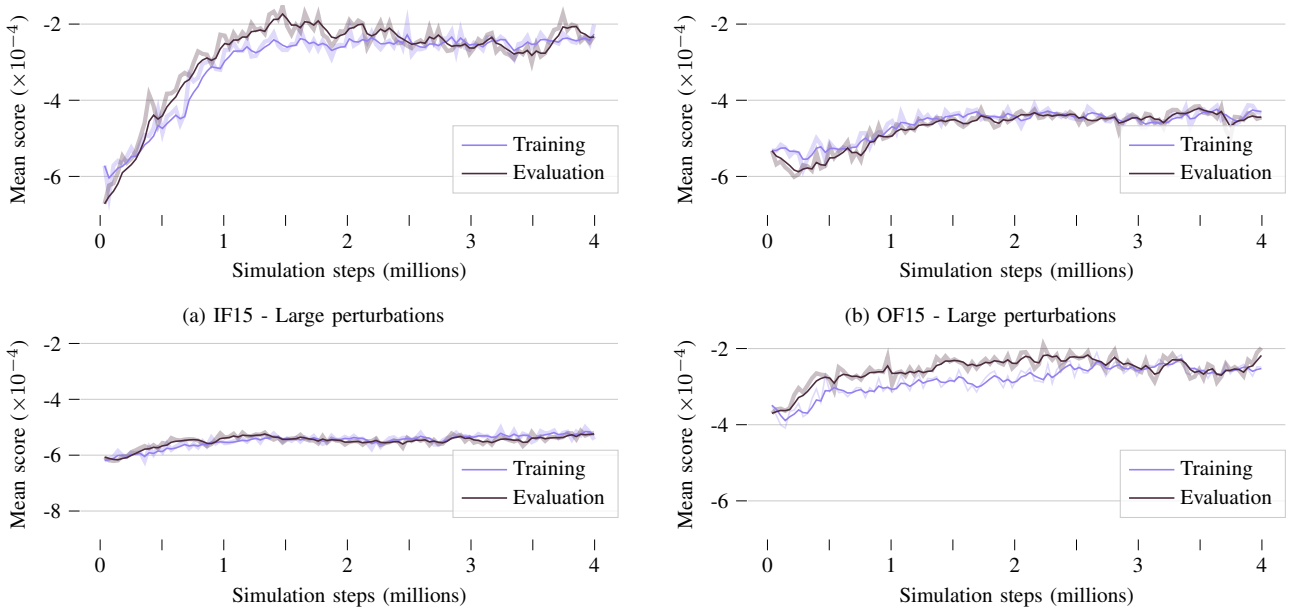
Fig. 3: Simulation results where we show the training in perturbance-free environment, and 12 cases where we analyze the effect of a modified environment (fixed and variable perturbances on both sensing and actuation) on 5, 15 and 25 agents. The total number of agents is 30 in all cases. The legend is common across all graphs and has been omitted in subfigures (b) through (m) to improve readability.

network being able to converge to a common policy when a constant error is added to the sensors interface, but not when the disturbances affect to the actuators. Finally, Figures 4c and 4d show that when there are no differentiated subsets of agents with a common behaviour and the perturbations are different across a large number of agents, then the system is not able to converge.

V. Conclusion and Future Work

Adversarial agents and closing the simulation-to-reality gap are among the key challenges preventing wider adoption of reinforcement learning in real-world applications. In this paper, we have addressed the latter one from the perspective of the former: by introducing adversarial conditions inspired by real-world perturbances to a subset of agents in a multi-robot system during a collaborative reinforcement learning process, we have been able to identify points where the robustness of distributed multi-agent DRL algorithms needs to be improved. In this paper, we have considered multiple robotic arms in a simulation environment collaborating towards learning a common policy to reach an object. In order to emulate more realistic conditions and understand how perturbances in the environment affect the learning process, we have considered variability across the agents in terms of their ability to sense and actuate accurately. We have shown how different types of disturbances in the model's input (sensing) and output (actuation) affect the robustness and ability to converge towards an effective policy. We have seen how variable perturbances have the most effect on the ability of the network to converge, while disturbances in the ability of the robots to actuate properly have had a comparatively worse effect than those in their ability to sense the position of the object accurately.

The conclusions of this work serve as a starting point towards the design and development of more robust methods able to identify and take into account these disturbances in the environment that do not occur across all robots equally.



(a) IF15 - Large perturbations

(b) OF15 - Large perturbations

(c) OF25 - Each agent exposed to a different environment

(d) IF25 - Each agent exposed to a different environment

Fig. 4: Simulation results with an extra 4 scenarios analyzed: two of them where we consider perturbations of larger magnitude, and two more where we consider that each of the agents in a modified environment is affected differently.

This will be the subject of our future work, as well as the study of other types or combinations of disturbances in the environment. We will also work towards modeling more accurately real-world errors for RL simulation environments.

Acknowledgements

This work was supported by the Academy of Finland's AutoSOS project with grant number 328755.

References

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[2] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 2020.

[3] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 2016.

[4] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[5] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. *arXiv preprint arXiv:1806.07851*, 2018.

[6] Ronny Conde, José Ramón Llata, and Carlos Torre-Ferrero. Time-varying formation controllers for unmanned aerial vehicles using deep reinforcement learning. *arXiv preprint arXiv:1706.01384*, 2017.

[7] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *ICRA*, pages 285–292. IEEE, 2017.

[9] Dorothea Schwung, Fabian Csaplar, Andreas Schwung, and Steven X Ding. An application of reinforcement learning algorithms to industrial multi-robot stations for cooperative handling operation. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pages 194–199. IEEE, 2017.

[8] Pinxin Long, Tingxiang Fanl, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *ICRA*. IEEE, 2018.

[10] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS*, pages 66–83. Springer, 2017.

[11] Jorge Peña Queralta, Li Qingqing, Zhuo Zou, and Tomi Westerlund. 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.

[12] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA*, pages 3389–3396. IEEE, 2017.

[13] Yiding Yu, Soung Chang Liew, and Taotao Wang. Multi-agent deep reinforcement learning multiple access for heterogeneous wireless networks with imperfect channels. *arXiv preprint arXiv:2003.11210*, 2020.

[14] Bharathan Balaji, Sunil Mallya, Sahika Genc, Saurabh Gupta, Leo Dirac, Vineet Khare, Gourav Roy, Tao Sun, Yunzhe Tao, Brian Townsend, et al. Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning. *arXiv:1911.01562*, 2019.

[15] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *MLDM*, pages 262–275. Springer, 2017.

[16] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[17] Jingkang Wang, Yang Liu, and Bo Li. Reinforcement learning with perturbed rewards. In *AAAI*, pages 6202–6209, 2020.

[18] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 7461–7472, 2018.

[19] Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. *arXiv preprint arXiv:1909.12906*, 2019.

[20] Inaam Ilahi, Muhammad Usama, Junaid Qadir, Muhammad Umar Janjua, Ala Al-Fuqaha, Dinh Thai Hoang, and Dusit Niyato. Challenges and countermeasures for adversarial attacks on deep reinforcement learning. *arXiv preprint arXiv:2001.09684*, 2020.

[21] Aashish Kumar et al. *Enhancing performance of reinforcement learning models in the presence of noisy rewards*. PhD thesis, 2019.

[22] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.