Towards Comprehensive Testing on the Robustness of Cooperative Multi-agent Reinforcement Learning

Jun Guo[1], Yonghong Chen[2], Yihang Hao[2], Zixin Yin[1], Yin Yu[3], Simin Li[1*]
[1] State Key Lab of Software Development Environment, Beihang University, Beijing, China
[2] Yangzhou Collaborative Innovation Research Institute CO., LTD
[3] No. 38 Research Institute of CETC
{junguo, lisiminsimon, yzx835}@buaa.edu.cn,
13514262035@163.com, hyh19951114@gmail.com, tony_yu_0210@126.com

Abstract

While deep neural networks (DNNs) have strengthened the performance of cooperative multi-agent reinforcement learning (c-MARL), the agent policy can be easily perturbed by adversarial examples. Considering the safety critical applications of c-MARL, such as traffic management, power management and unmanned aerial vehicle control, it is crucial to test the robustness of c-MARL algorithm before it was deployed in reality. Existing adversarial attacks for MARL could be used for testing, but is limited to one robustness aspects (e.g., reward, state, action), while c-MARL model could be attacked from any aspect. To overcome the challenge, we propose MARLSafe, the first robustness testing framework for c-MARL algorithms. First, motivated by Markov Decision Process (MDP), MARLSafe consider the robustness of c-MARL algorithms comprehensively from three aspects, namely state robustness, action robustness and reward robustness. Any c-MARL algorithm must simultaneously satisfy these robustness aspects to be considered secure. Second, due to the scarceness of c-MARL attack, we propose c-MARL attacks as robustness testing algorithms from multiple aspects. Experiments on SMAC environment reveals that many state-of-the-art c-MARL algorithms are of low robustness in all aspect, pointing out the urgent need to test and enhance robustness of c-MARL algorithms.

1. Introduction

With the success of deep neural networks (DNNs), tremendous success have been made in cooperative multi-agent reinforcement learning (c-MARL), which powers numerous real-world applications, including traffic management [5, 12], power management [8, 39] and unmanned aerial vehicle control [6,7], *etc*. Recently, it has been shown

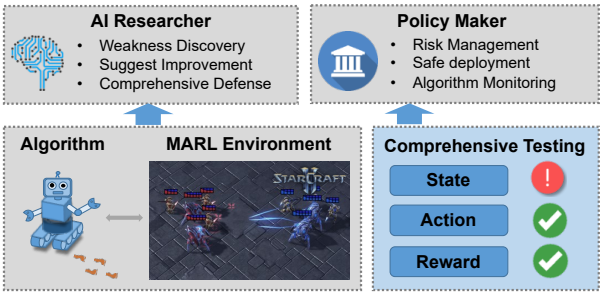


Figure 1. An overview of our work. c-MARL algorithm could be attacked from three aspects, namely state, action and reward. We test the robustness of c-MARL from these aspects.

that adversarial examples [2, 10, 19, 21, 23, 40] are capable to perturb these safety-critical application with high-confidence, raising a serious concern on the robustness of c-MARL algorithms.

Testing the robustness has been a promising solution for DNN models. Being able to thoroughly test the robustness of DNN models will benefit researchers to discover weakness in DNN models and policy makers to ensure safe deployment in many sensitive scenarios. Many highly-influential works have been published in computer vision communities to test robustness and interpret adversarial examples [9, 15, 17, 18, 24, 25, 37, 42] using multiple algorithms, metrics and attack settings. Recently, Behzadan et al. [1] also benchmarked the robustness of reinforcement learning (RL) algorithms towards different state perturbations.

However, to the best of our knowledge, no work exists to test the robustness of c-MARL algorithm. Besides, from the perspective of multi-agent MDP, its possible for hackers to attack from the aspect of reward [13], state [22] and action [10]. While existing attack could be used as testing tool, they all focus on only one aspect (state, action, reward), making the test limited since c-MARL algorithm

might be robust in one aspect, but hacker can attack from all possible aspects.

To tackle the problem, we propose MARLSafe, a multi-aspect testing framework of c-MARL algorithms. The motivation of our paper is summarized in Fig. 1. First, motivated by multi-agent Markov Decision Process (MMDP), multi-agent reinforcement learning contains 5 *elements*: state, action, reward, environment dynamic, and discount factor. Consider existing literature and the feasibility of perturbation, we assume hacker might perturb the state, action and reward in MMDP. Then, we view the robustness of c-MARL as the ability to resilience attacks from multi aspects. Note that since hackers might attack from any aspect, an algorithm must be simultaneously robust in all aspects in order to be considered robust. Second, due to the scarcity in c-MARL attack literature, to the best of our knowledge, [21] is the only paper to attack c-MARL in state aspect, while no work exists to attack other aspects. We propose c-MARL attacks based on the aspect of state, action and reward. To figure out the performance and characteristic of these attacks in c-MARL tasks, we conduct experiments of attacks on the *StarCraftII Multi Agent Challenge* (*SMAC*) environment [32]. Our contributions can be listed as follows:

- To the best of our knowledge, MARLSafe is the first to test the robustness of c-MARL algorithms from multiple aspects, namely state, action and reward.
- Technically, MARLSafe propose adversarial attack for c-MARL algorithms from multiple aspects. Some aspects are first proposed in c-MARL literature.
- Empirically, we find the testing method in MARLSafe could attack state-of-the-art c-MARL algorithm with high confidence (*i.e.*, towards 0% winning rate).

2. Related Work

2.1. Adversarial Attacks

Szegedy *et al*. [34] first defined adversarial attacks and proposed L-BFGS attack to generate adversarial examples. By leveraging the gradient of the target model, Goodfellow *et al*. [11] proposed the Fast Gradient Sign Method (FGSM) to quickly generate adversarial examples. Since then, many types of adversarial attacks have been proposed, such as gradient-based attacks (PGD, C&W) [4, 27], boundary-based attack (DeepFool) [29], saliency-based attack (JSMA) [30]. Brown *et al*. [3] first proposed advesarial patch, which adds a local patch with impressive textures to the input image. Liu *et al*. [26] proposed a patch attack towards automatic check-out in physical world. Wang *et al*. [38] proposed Dual Attention Suppression attack to make the adversarial patches both malign and beautiful. Adversarial attacks on machine learning models have been adequately investigated, showing the potential risk of neural networks when it comes to practical application.

2.2. Multi-Agent Deep Reinforcement Learning

Deep reinforcement learning methods tend to train a policy network which maps state observations to action probabilities. DRL algorithms can be roughly categorized into two types: policy-based and value-based algorithm. Policy based algorithms often rely on policy gradient, such as DDPG [20] and PPO [33]. Value based algorithms often predict the Q-value, such as Deep Q Network (DQN) [28]. In MARL tasks, the most straightforward way to acquire a policy is to train individual agents, which is called Independent Q-Learning (IQL) [35, 36]. However, this strategy is not efficient in MARL environments requiring cooperation. Recent works adopted CTDE framework, such as QMIX [31] and MAPPO [41], can enhance the cooperation of agents and achieve better performance. However, those algorithms also suffer from robustness problem, which have not been properly evaluated.

2.3. Adversarial Attacks on DRL

Huang *et al*. [16] evaluated the robustness of DRL policies by perturbing the observations through FGSM attack on Atari Games. Liu *et al*. [23] proposed a spatiotemporal attack for embodied agents, which generates adversarial textures in the navigation environment. Lin *et al*. [22] proposed an attack method which perturbs the observation at some crucial frames, and they achieved targeted attack for DRL policies. Behzadan and Munir [2] propose a black-box attack by introducing a surrogate policy to minimize the return. Han *et al*. [13] proposed reward flipping attack at train time in software-defined networking tasks. Gleave *et al*. [10] proposed the adversarial policy in competitive multi-agent settings, which trains an opponent agent while fix parameters of the victim policy to attack the victim model. To the best of our knowledge, [21] is the only paper to attack c-MARL by perturbing the input state of agent.

3. Methodology

3.1. Formulation of Attacks

A multi-agent Markov Decision Process (MMDP) is defined as a tuple $(\mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, R, \mathcal{P}, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A} := \mathcal{A}^1 \times ... \times \mathcal{A}^N$ denotes the joint action space with N agents, $R : \mathcal{S} \times \{\mathcal{A}\} \times \mathcal{S} \to \mathbb{R}$ is the joint reward function for all c-MARL agents, and $\mathcal{P} : \mathcal{S} \times \{\mathcal{A}\} \to \mathcal{S}$ is the transition probability of the environment, also known as environment dynamics. The next state is determined by environmental dynamics, current state and actions taken by agent: $\mathcal{P}(s', r|s, \{a^i\}) = p(s_{t+1} = s', r_{t+1} = r|s_t = s, \{a_t^i\} = \{a^i\})$, where t is the time step. $\gamma \in [0, 1]$ is the discount factor. Most c-MARL

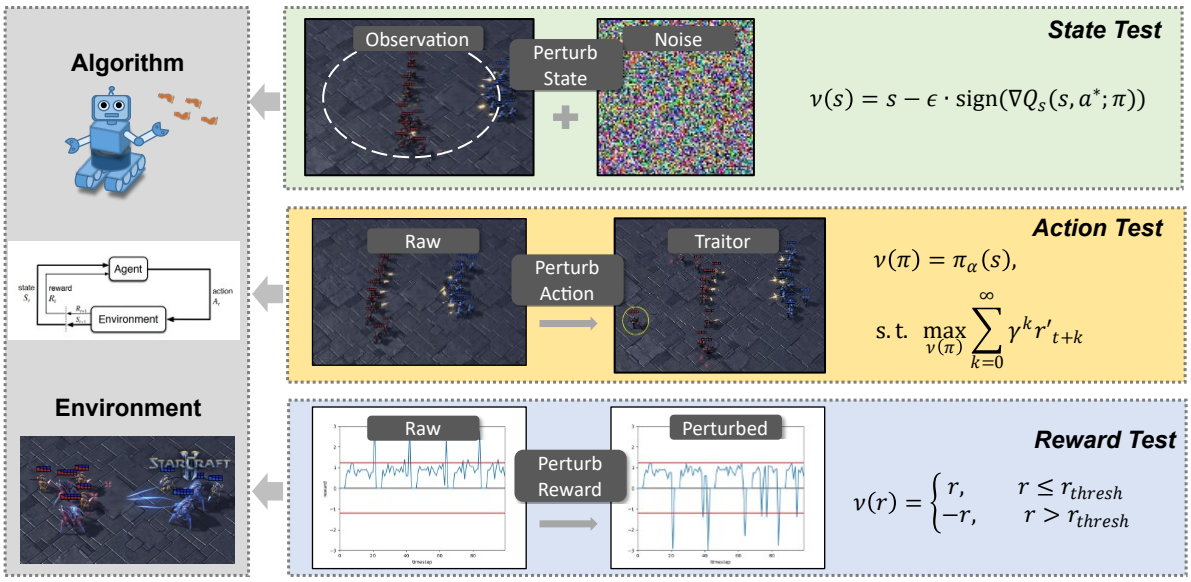


Figure 2. The framework of MARLSafe. Motivated by MMDP framework, we propose state, action and reward test to test its robustness.

tasks can be regarded as a MMDP, and agents hope to learn a stationary policy $\pi(\cdot|s)$ to maximize the discounted return $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$.

When it comes to adversarial attacks, we introduce an adversary $\nu(\cdot) \in B(\cdot)$ to perturb the elements in MMDP, where $B(\cdot)$ is the available perturbation set. $(\cdot)$ can be state s, action a or reward r. The goal of adversarial attack is to minimize the discounted return G by perturbing elements in MMDP. Generally, due to the limited perturbation budget and attack feasibility, the adversaries are encouraged to perturb the elements in MDP as small as they could, while achieving attack as strong as they could. Thus, attackers have the full control of victim model and in most works, they choose to attack on one of the three elements which is suitable for their specific conditions.

3.1.1 Attacks towards States

The goal of attacks towards states aims to perturb the state observed by agents, such that agents will perform erroneous actions that harm the final reward. The goal of attacks towards state can be formulated as below:

$$\min_{\nu(s)} G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k},$$
$$\text{s.t. } \nu(s) \in B(s), s' \sim P(s'|s, \{a^i\}), \{a^i\} \sim \pi(\cdot|\nu(s)), \tag{1}$$

Note that state-based attack is very similar to adversarial attacks in computer vision, or natural language processing, which perturb model input at test time to mislead the output at test time. Therefore, methods perturbing states usually apply gradient-based attack (*e.g.* FGSM, PGD, *etc.*) to generate the adversarial perturbation in a white-box setting. For black-box settings, the transferability of adversarial examples renders it possible to train a surrogate model, generate white-box perturbations, then transfer them to the black-box policy. Attacks based on query is also feasible.

3.1.2 Attacks towards Rewards

Applied in training time, attacks towards rewards modify rewards given to agents to interfere the policy, such that the attacked agent cannot achieve its desired goal. Given adversary ν that perturbs the reward, the process can be formulated as:

$$\min_{\nu(r)} G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k},$$
$$\text{s.t. } \nu(r) \in B(r), s' \sim P(s'|s, \{a^i\}), \{a^i\} \sim \pi(\cdot|s, \nu(r)). \tag{2}$$

where $\pi(a|s, \nu(r))$ means the policy π is learned from perturbed reward $\nu(r)$. This type of attack usually do not need to have any knowledge of the model. Instead, it misleads the policy by flipping the sign of rewards at certain time. Previous works [13] have shown that only a certain number (*e.g.*, 5% of all experiences) of flipping operation can destroy the training process. This can be extremely harmful when the reward signal is corrupted or hijacked.

3.1.3 Attacks towards Actions

Attacks towards actions directly perturb selected actions without modifying rewards or observations. Specifically, attacks towards actions substitute the action given by original policy to the action given by an adversarial policy, or

add noise to the original policy to minimize the total goal. The formulation of attacks towards actions can be listed as follows:

$$\min_{\theta} G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k},$$
$$\text{s.t. } \nu(\pi) \in B(\pi), s' \sim P(s'|s, \{a^i\}), \{a^i\} \sim \nu(\pi(\cdot|s)). \tag{3}$$

There are several approaches about how to perturb the action. For continuous action space, adversary can add proper perturbations on the original action. Additionally, it is effective to train an adversarial policy π_{adv} to replace the original policy π. Attacks towards actions require the control of model outputs, and accordingly, it can degrade the model performance by a large margin.

3.2. Proposed Method

Based on the formulation above, we propose our MARLSafe method with the help of three types of attacks. These attacks can (1) comprehensively cover the elements of MMDP and test the robustness from multiple aspects, (2) test the policy at training time and test time, and (3) can run in white-box and black-box settings. The overall framework of MARLSafe is given in Fig. 2.

3.2.1 State Test

To test the robustness in the dimension of state, we apply gradient based attack to the observation of agents. In c-MARL settings, agents follow CTDE framework and communicate with each other only by their observations at test time. The perturbed observations mislead the attacked agent to deviate from the cooperation. To simplify the attack, we add perturbations by using fast gradient sign method (FGSM) [11], where we execute untargeted attack with the objective of minimizing the probability of original optimal action. As a result, our adversarial perturbations reduce the output logits of optimal actions, and induce the policy to choose worse actions. Define $a^* = \arg\max_a \pi(s)$ as the optimal action selected by policy, and $Q(s, a^*; \pi)$ refers to the output logit of policy network π for s and a^*. The attack in state test can be formulated as below:

$$\nu(s) = s - \epsilon \cdot \text{sign}(\nabla Q_s(s, a^*; \pi)). \tag{4}$$

3.2.2 Reward Test

Reward poisoning is a threatening attack method for RL policies at training time. In c-MARL settings, the training process is centralized, and the environment usually returns a total reward rather than a group of rewards. Motivated by [13], we flip the sign of a certain percent $k\%$ of rewards during training time, poisoning rewards to prevent the model from acquiring a good policy. As the training data is collected one episode a time, we choose the max $k\%$ reward in an episode to flip their sign. When $k = 0$, it becomes a normal training process, and when $k = 100$, the policy will aim at minimizing the team reward. Define r_{thresh} as the threshold at the $k\%$ rewards of all time steps, and we can formulate our testing method as below:

$$\nu(r) = \begin{cases} r, & r \le r_{thresh} \\ -r, & r > r_{thresh} \end{cases} \tag{5}$$

3.2.3 Action Test

In c-MARL settings, we use the powerful black-box attack method *adversarial policy* [10] as our testing method. However, vanilla adversarial policy is formulated as a zero-sum Markov game between two opposing agents, which is not suitable for c-MARL settings. For MARLSafe, we propose the testing method that we get the control of one agent as a "traitor", and train the "traitor" to maximally perturb other agents with their policies fixed. The reward of the adversarial policy is set to the opposite number of original team reward (*i.e.*, the traitor seeks to minimize the reward of collaborated agents with fixed policy). Define the reward of adversarial policy as r', the formulation of action test is listed in Eq. (6).

$$\nu(\pi) = \pi_\alpha(s),$$
$$\text{s.t. } \max_{\nu(\pi)} \sum_{k=0}^{\infty} \gamma^k r'_{t+k} \tag{6}$$

4. Experiments

In this section, we conduct experiments to demonstrate the effectiveness of MARLSafe. We choose *StarCraftII Multi-Agent Challenge* (*SMAC*) [32] as our experimental environment. We use *EPyMARL* framework as our testbed. All of our experiments are conducted on a server with 3 NVIDIA RTX 2080Ti GPUs and a 26-core 2.10GHz Intel(R) Xeon(R) Gold 6230R CPU.

4.1. Experiment Settings

MARL Algorithm We choose QMIX [31] and MAPPO [41] as algorithms to test. QMIX and MAPPO are popular algorithms with CTDE framework in c-MARL tasks, thus deserving a test. QMIX is based on Q-learning, while MAPPO is based on policy gradient. In QMIX, agents share a deep Q-network for decentralized execution, and a Q-value mixer is applied for centralized training. In MAPPO, agents select actions via an actor network, and actions of all agents are evaluated by a critic network. The hyperparameters of MARL algorithms is consistent with *EPyMARL*.

Map	Algorithm	WR	TR	mDA	mDE
2s3z	QMIX	96.88%	19.74	0.84	4.94
	MAPPO	100.00%	20.00	0.84	5.00
11m	QMIX	100.00%	20.00	2.00	11.00
	MAPPO	100.00%	20.00	2.81	11.00

Table 1. Performance of QMIX and MAPPO without attack. Both algorithms were able to outperform hardest build-in AI of Starcraft with a very high win rate.

Map	Algorithm	WR	TR	mDA	mDE
2s3z	QMIX	9.38%	11.85	4.72	1.69
	MAPPO	65.62%	17.91	3.34	4.25
11m	QMIX	0.00%	9.76	10.94	5.53
	MAPPO	31.25%	14.62	9.69	8.66

Table 2. Performance of QMIX and MAPPO under state test. Both algorithms shows weak robustness, while MAPPO is relatively robust.

Map	Algorithm	WR	TR	mDA	mDE
2s3z	QMIX	0.00%	5.32	4.41	0.06
	MAPPO	0.00%	0.00	3.72	0.00
11m	QMIX	0.00%	0.00	2.09	0.00
	MAPPO	0.00%	5.89	11.00	0.16

Table 3. Performance of QMIX and MAPPO under reward test. None of the algorithms are robust under reward-based attack.

Map	Algorithm	WR	TR	mDA	mDE
2s3z	QMIX	0.00%	10.07	4.97	1.34
	MAPPO	0.00%	11.59	5.00	2.13
11m	QMIX	6.25%	9.88	10.69	4.81
	MAPPO	0.00%	10.53	11.00	6.31

Table 4. Performance of QMIX and MAPPO under action test. None of the algorithms are robust under action-based attack.

SMAC Maps We choose 2s3z (2 Stalkers and 3 Zealots) and 11m (11 Marines) as our experiment maps, where red team controlled by agents and blue team controlled by computer have same units. Note that the original *SMAC* does not contain the map "11m". The 11m map is modified from the original map "10m_vs_11m", to balance the number of units for two players and give fair comparison in action test. The goal of normal agents in *SMAC* maps is to kill enemy units as much as possible, and the game ends when one team lose all units or the time step reach the limit. In our setting of action test, the first agent will be controlled as a "traitor". In 2s3z, the "traitor" is a Stalker. We select the difficulty of computer as level 7 (hardest possible).

Hyperparameters of Attack In state test, we perform FGSM attack in ℓ_∞-norm and set $\epsilon = 0.05$. In reward test, we set filp rate $k\% = 10\%$. In action test, we fix other agent's policy and train the "traitor" agent with deep recursive Q-network (DRQN) [14]. The traitor do not have access to global state or observation of other agents. As for the reward of traitor, it receives a positive reward when allies get damaged or die, and receives a negative reward when when enemies get damaged or die. Winning the game will receive negative rewards, and losing the game will receive positive rewards. The reward is normalized to $[-20, 20]$.

Evaluation Metrics The performance of a policy in *SMAC* environment can be evaluated by metrics below: win rate (WR), team reward (TR), mean number of dead allies (mDA), and mean number of dead enemies (mDE). In our experiments, we calculate and show these 4 metrics to evaluate the robustness of MARL policies. We test 32 episodes of games for each experiment to calculate these metrics.

4.2. Experimental Results

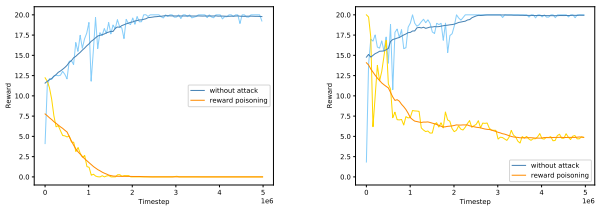
Performance without Attack The performance of QMIX and MAPPO without attack is listed in Tab. 1. Without attack, the performance of QMIX and MAPPO greatly surpasses the hardest build-in AI inside the StarCraftII game. Both in 2s3z and 11m, the winning rate reaches 100% with a maximum reward. The great performance proves the effectiveness of these MARL algorithms.

Performance under State Test We apply our state test on QMIX and MAPPO, whose results are showed in Tab. 2. The results show the vulnerability of these two algorithms, drastically reducing the winning rate of QMIX and MAPPO from 100% to lower than 15%. State-based attacks came as a "natural" attack. The replay demonstrates that all agents behaves as they are on battle with their opponent naturally, but ends up losing the game. When comparing two algorithms, we find that agents trained with MAPPO algorithm is more robust in the dimension of state than those trained with QMIX. Considering that the decentralized agent network structures of QMIX and MAPPO are identical, we hypothesis the robustness came from the training process, and the centralized training network (*i.e.* mixer network or critic network) might play an important role in model robustness.

Performance under Reward Test The results of reward test are listed in Tab. 3. We notice that win rates in these experiments are equally 0%. However, the mean number of dead allies does not reach the number of all allies. According to the detailed results, not all allies are killed when the time step reaches the limit, and the game ends with nobody winning. The trained policy behaves as all agents fleeing from their enemies. They spare no effort to fight against enemies or quickly surrender and get killed. It possibly results from the characteristic of reward attack, which only flips the

(a) attack towards states (b) attack towards rewards (c) attack towards actions

Figure 3. Illustration of different behaviors of MARL agents under three types of attacks. Agents under state-based attack act relatively normal, but failed to cooperate. Agents under reward-based attack jointly flee from opponents. Traitor under action-based attack first flee away, then exert influence on normal agents.



(a) MAPPO, 2s3z (b) MAPPO, 11m

Figure 4. Reward curve of normal agents and reward-attacked agents in an episode, during training.

sign of the maximum part of rewards. When enemies get damaged or killed, the reward increases rapidly and thus is likely to be perturbed, thus avoided. When agents died, the reward decreases, and is unlikely to get perturbed, As a result, agents get punished by being killed and develop the behaviour of avoid getting killed, but not killing opponents. Fig. 4 shows the total rewards by time step during training. Despite perturbing only 10% of the total reward, the algorithm learns a corrupted policy, and the real reward continues to go down. Due to the high effectiveness of reward test, we conclude that robustness at the dimension of reward is usually overlooked yet vulnerable. It is necessary to pay more attention to reward poisoning attacks.

Performance under Action Test Tab. 4 shows the performance of QMIX and MAPPO under action test. We can draw a conclusion that the "traitor" controlled by adversarial policy leads to a great failure of battles. Interestingly, in the original *SMAC* map "10m_vs_11m", 10 Marines controlled by MARL policies can easily defeat 11 Marines controlled by the computer with almost 100% win rate. However, when the traitor was added, the agents performed much worse with stronger ally numbers, stongly proofing the vulnerability of MARL policies. On the other hand, agents controlled by QMIX can sometimes defeat the computer in 11m map, which implies its better robustness towards action test.

Different Behaviors of Agents under Attacks Fig. 3 presents the behavior of MARL agents under different types of attacks. We can clearly see the various behavior of agents and infer the characteristic of these attacks. Under the attack toward states, agents seem to try to behave as normal agents, but their actions become chaotic and non-cooperative. Clearly, attack toward states only suppresses the probability of the optimal action, so agents tend to choose suboptimal actions, which are sometimes effective but cannot achieve 100% win rate since not being optimal. Under the attack towards rewards, agents seem to flee from their enemies. As the sign of greater reward flipped, agents avoid causing severe damages to their enemies, but they also avoid death because they cannot get any reward after they die. These limitations result in the fleeing behavior of agents. Under the attack towards actions, the adversarial policy hide behind the team while others attack as normal. However, the action of the adversarial policy affects the decision of its teammates, leading them to be defeated. After all normal agents die, the adversarial agent moves forward and quickly get killed. The adversarial policy learned by reinforcement learning acquire the optimal action to lose the game.

5. Conclusion

In this paper, we propose MARLSafe, a robustness testing framework for c-MARL algorithms. First, we formulate the existing attack method in the formulation of MDP, and categorize them by MDP elements: state, reward and action. Moreover, we propose a robustness testing method from multi aspects, and propose several adversarial attack method c-MARL settings to test the robustness of c-MARL algorithms. To the best of our knowledge, MARLSafe is the first paper to test the robustness of c-MARL algorithms from multiple aspects, and our method could attack state-of-the-art c-MARL algorithm with high performance degradation. The results of our experiments indicate that c-MARL algorithms are facing severe robustness problems, and it is necessary to explore comprehensive defense methods that jointly covers the aspect of state, action and reward.

References

[1] Vahid Behzadan and William Hsu. Rl-based method for benchmarking the adversarial resilience and robustness of deep reinforcement learning policies. In *International Conference on Computer Safety, Reliability, and Security*, pages 314–325. Springer, 2019. 1

[2] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017. 1, 2

[3] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. 2

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (S&P)*, pages 39–57. IEEE, 2017. 2

[5] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1086–1095, 2019. 1

[6] Jingjing Cui, Yuanwei Liu, and Arumugam Nallanathan. The application of multi-agent reinforcement learning in uav networks. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2019. 1

[7] Jingjing Cui, Yuanwei Liu, and Arumugam Nallanathan. Multi-agent reinforcement learning-based resource allocation for uav networks. *IEEE Transactions on Wireless Communications*, 19(2):729–743, 2019. 1

[8] Xiaohan Fang, Jinkuan Wang, Guanru Song, Yinghua Han, Qiang Zhao, and Zhiao Cao. Multi-agent reinforcement learning approach for residential microgrid energy scheduling. *Energies*, 13(1):123, 2019. 1

[9] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1147–1158. IEEE, 2020. 1

[10] 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. 1, 2, 4

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2, 4

[12] Martin Gregurić, Miroslav Vujić, Charalampos Alexopoulos, and Mladen Miletić. Application of deep reinforcement learning in traffic signal control: An overview and impact of open traffic data. *Applied Sciences*, 10(11):4011, 2020. 1

[13] Yi Han, Benjamin IP Rubinstein, Tamas Abraham, Tansu Alpcan, Olivier De Vel, Sarah Erfani, David Hubczenko, Christopher Leckie, and Paul Montague. Reinforcement learning for autonomous defence in software-defined networking. In *International Conference on Decision and Game Theory for Security*, pages 145–165. Springer, 2018. 1, 2, 3, 4

[14] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015. 5

[15] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58, 2011. 1

[16] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017. 2

[17] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019. 1

[18] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 1

[19] Siyuan Liang, Baoyuan Wu, Yanbo Fan, Xingxing Wei, and Xiaochun Cao. Parallel rectangle flip attack: A query-based black-box attack against object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7697–7707, 2021. 1

[20] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 2

[21] Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot. On the robustness of cooperative multi-agent reinforcement learning. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 62–68. IEEE, 2020. 1, 2

[22] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017. 1, 2

[23] Aishan Liu, Tairan Huang, Xianglong Liu, Yitao Xu, Yuqing Ma, Xinyun Chen, Stephen J Maybank, and Dacheng Tao. Spatiotemporal attacks for embodied agents. In *European Conference on Computer Vision*, pages 122–138. Springer, 2020. 1, 2

[24] Aishan Liu, Xianglong Liu, Jun Guo, Jiakai Wang, Yuqing Ma, Ze Zhao, Xinghai Gao, and Gang Xiao. A comprehensive evaluation framework for deep model robustness. *arXiv preprint arXiv:2101.09617*, 2021. 1

[25] Aishan Liu, Xianglong Liu, Hang Yu, Chongzhi Zhang, Qiang Liu, and Dacheng Tao. Training robust deep neural networks via adversarial noise propagation. *IEEE Transactions on Image Processing*, 2021. 1

[26] Aishan Liu, Jiakai Wang, Xianglong Liu, Bowen Cao, Chongzhi Zhang, and Hang Yu. Bias-based universal adversarial patch attack for automatic check-out. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 395–410. Springer, 2020. 2

[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 2

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 2

[29] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 2

[30] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016. 2

[31] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018. 2, 4

[32] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. *CoRR*, abs/1902.04043, 2019. 2, 4

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 2

[34] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 2

[35] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017. 2

[36] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993. 2

[37] Shiyu Tang, Ruihao Gong, Yan Wang, Aishan Liu, Jiakai Wang, Xinyun Chen, Fengwei Yu, Xianglong Liu, Dawn Song, Alan Yuille, et al. Robustart: Benchmarking robustness on architecture design and training techniques. *arXiv preprint arXiv:2109.05211*, 2021. 1

[38] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. Dual attention suppression attack: Generate adversarial camouflage in physical world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8565–8574, 2021. 2

[39] Jianhong Wang, Wangkun Xu, Yunjie Gu, Wenbin Song, and Tim Green. Multi-agent reinforcement learning for active voltage control on power distribution networks. *Advances in Neural Information Processing Systems*, 34, 2021. 1

[40] Zhipeng Wei, Jingjing Chen, Micah Goldblum, Zuxuan Wu, Tom Goldstein, and Yu-Gang Jiang. Towards transferable adversarial attacks on vision transformers. *ArXiv*, abs/2109.04176, 2021. 1

[41] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021. 2, 4

[42] Chongzhi Zhang, Aishan Liu, Xianglong Liu, Yitao Xu, Hang Yu, Yuqing Ma, and Tianlin Li. Interpreting and improving adversarial robustness with neuron sensitivity. *IEEE Transactions on Image Processing*, 2020. 1