Attacking c-MARL More Effectively: A Data Driven Approach

Nhan H. Pham, Lam M. Nguyen  
IBM Research, Thomas J. Watson Research Center  
{nhp, lamnguyen.mltd}@ibm.com

Jie Chen  
MIT-IBM Watson Al Lab, IBM Research  
chenjie@us.ibm.com

Hoang Thanh Lam  
IBM Research Europe  
t.l.hoang@ie.ibm.com

Subhro Das  
MIT-IBM Watson Al Lab, IBM Research  
subhro.das@ibm.com

Tsui-Wei Weng  
Halıcıoğlu Data Science Institute, University of California San Diego  
lweng@ucsd.edu

Abstract—In recent years, a proliferation of methods were developed for cooperative multi-agent reinforcement learning (c-MARL). However, the robustness of c-MARL agents against adversarial attacks has been rarely explored. In this paper, we propose to evaluate the robustness of c-MARL agents via a model-based approach, named c-MBA. Our proposed formulation can craft much stronger adversarial state perturbations of c-MARL agents to lower total team rewards than existing model-free approaches. In addition, we propose the first victim-agent selection strategy and the first data-driven approach to define targeted failure states where each of them allows us to develop even stronger adversarial attack without the expert knowledge to the underlying environment. Our numerical experiments on two representative MARL benchmarks illustrate the advantage of our approach over other baselines: our model-based attack consistently outperforms other baselines in all tested environments.

Index Terms—adversarial attack, MARL, robustness

I. INTRODUCTION

Deep neural networks are known to be vulnerable to adversarial examples, where a small and often imperceptible adversarial perturbation can easily fool the state-of-the-art deep neural network classifiers [1, 2, 3, 4]. Since then, a wide variety of deep learning tasks have been shown to also be vulnerable to adversarial attacks, ranging from various computer vision tasks to natural language processing tasks [5, 6, 7, 8].

Perhaps unsurprisingly, deep reinforcement learning (DRL) agents are also vulnerable to adversarial attacks, as first shown in [9] for atari games DRL agents. In [10], the authors further investigate a strategically-timing attack when attacking victim agents on Atari games at a subset of the time-steps. Meanwhile, [11] use the fast gradient sign method (FGSM) [3] to generate adversarial perturbation on the A3C agents [12] and explore training with random noise and FGSM perturbation to improve resilience against adversarial examples. While the above research endeavors focus on actions that take discrete values, another line of research tackles a more challenging problem on DRL with continuous action spaces [13, 14].

However, all the above works focused on single DRL setting.

In this work we propose to study the vulnerability of multi-agent DRL, which has been widely applied in many safety-critical real-world applications including swarm robotics [15], electricity distribution, and traffic control [16]. In particular, we focus on the collaborative multi-agent reinforcement learning (c-MARL) setting, where a group of agents is trained to generate joint actions to maximize the team reward. We note that c-MARL is a more challenging yet interesting setting than the single DRL agent setting, as now one also needs to consider the interactions between agents, which makes the problem becomes more complicated.

Our contribution can be summarized as follows:

- We propose the first model-based adversarial attack framework on c-MARL called c-MBA (Model-Based Attack on c-MARL). We formulate the attack into a two-step process and solve for adversarial state perturbation efficiently by existing proximal gradient methods. We show that our model-based attack is stronger and more effective than all of existing model-free baselines. Besides, we propose a novel adaptive victim selection strategy and show that it could further increase the attack power of c-MBA by decreasing the team reward even more.

- To alleviate the dependence on the knowledge of the c-MARL environment, we also propose the first data-driven approach to define the targeted failure state based on the collected data for training the dynamics model. Our numerical experiments illustrate that c-MBA with the data-driven failure state is comparable and even outperforms c-MB with the expert-defined failure state in many cases.
Therefore, our data-driven approach is a good proxy to the optimal failure state when we have little or no knowledge about the state space of the c-MARL environments.

- We demonstrate that c-MBA consistently outperforms the SOTA baselines. We show that c-MBA reduces the team reward up to $8 - 9\times$ when attacking the c-MARL agents. In addition, c-MBA with the proposed victim selection scheme matches or even outperforms other c-MBA variants in all environments with up to 80% of improvement on reward reduction.

## II. RELATED WORK

Most of existing adversarial attacks on DRL agents are on single agent [9, 10, 11, 13, for examples] while there are limited works that focus on the c-MARL setting. [17] considers a different problem than ours where they want to find an optimally "sparse" attack by finding an attack with minimal attack steps. [18] focuses on adversarial attacks against c-MARL system under time-delayed data transmission setting which is not considered in this paper. [19] proposes a new robustness testing framework for c-MARL which considers state, action, and reward robustness. The design of state adversarial attack in [19] is based on gradient-based attack. [20] proposes a two-step attack procedure to generate state perturbation for c-MARL setting which is the most relevant to our work. However, there are two major differences between our work and [20]: (1) their attack is only evaluated under the StarCraft Multi-Agent Challenge (SMAC) environment [21] where the action spaces are discrete; (2) their approach is model-free as they do not involve learning the dynamics of the environment and instead propose to train an adversarial policy for a fixed agent to minimize the the total team rewards. The requirement on training an adversarial policy is impractical and expensive compared to learning the dynamics model. To the best of our knowledge, there has been no any work considering adversarial attacks on the c-MARL setting using model-based approach on continuous action spaces. In this paper, we perform adversarial attacks on agents trained using MADDPG [22] on two multi-agent benchmarks including multi-agent MuJoCo and multi-agent particle environments. Note that in the setting of adversarial attacks, once the agents are trained, policy parameters will be frozen and we do not require any retraining of the c-MARL agents during our attack.

## III. C-MBA: MODEL-BASED ATTACK FOR C-MARL

### A. Problem formulation and c-MBA attack

Our goal is to generate adversarial perturbations imposed to the victim agents’ input (state) in order to deteriorate the total team reward. The added perturbations encourages the victim agents’ state to be close to a desired failure state corresponding to low reward. To avoid sampling from the environment, we use a pre-trained dynamics model to predict the next state given the perturbed state and current action, then find the suitable perturbation that minimizes the distance between the predicted next state and a predefined target state. For now, we assume the target state is given and we show in Section III-B that this target state can actually be learned directly from the data.

Formally, we consider a multi-agent setting with $|\mathcal{N}| = n$ agents, each agent $i \in \mathcal{N}$ receives state $s^i_t$ locally and takes action $a^i_t$ following the pre-trained c-MARL policy $\pi^i(s^i_t)$. Let $s_t = (s^1_t, \ldots, s^n_t) \in \mathcal{S}$ be the joint global state at time step $t$ which is concatenated from local states $s^i_t$ for each agent $i \in \mathcal{N}$. Denote the joint action as $a_t = (a^1_t, \ldots, a^n_t)$ concatenated from each agent’s action $a^i_t$. Let $\mathcal{V} \subseteq \mathcal{N}$ be the set of victim agents at time step $t$, i.e, the set of agents that can be attacked. Let $f : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ be a parameterized function that approximates the dynamics of the environment, where $\mathcal{A}$ is the set of concatenated actions, one from each $\mathcal{A}_i$. Let $s_{fail}$ be the targeted failure state which corresponds to poor performance of the agent. We denote $\epsilon$ as an upper bound on budget constraint w.r.t some $\ell_p$-norm $\|\cdot\|_p$. The state perturbation $\Delta s = (\Delta s^1, \ldots, \Delta s^n)$ (we suppress the dependence on $t$ of $\Delta s$ to avoid overloading the notation) to $s_t$ is the solution to the following problem:

$$\begin{align*}
\min_{\Delta s} & \quad d(\hat{s}_{t+1}, s_{fail}) \\
\text{s.t.} & \quad \hat{s}_{t+1} = f(s_t, a_t) \\
& \quad a^i_t = \pi^i(s^i_t + \Delta s^i) , \quad \forall i \in \mathcal{N} \\
& \quad \Delta s^i = 0 , \quad \forall i \notin \mathcal{V}_t \\
& \quad \ell_S \leq \Delta s + \Delta s^i \leq u_S \\
& \quad \|\Delta s^i\|_p \leq \epsilon , \quad \forall i \in \mathcal{V}_t
\end{align*}$$

where $\mathbf{0}$ is a zero vector, and the state vector follows a box constraint specified by $\ell_S$ and $u_S$.

Let us first provide some insights for the formulation (1). For each agent $i$, using the trained policy $\pi^i$, we can compute the corresponding action $a^i_t$ given its (possibly perturbed) local state $s^i_t$ or $s^i_t + \Delta s^i$. From the concatenated state-action pair $(s^i_t, a^i_t)$, we can predict the next state $\hat{s}_{t+1}$ via the learned dynamics model $f$. Then by minimizing the distance between $\hat{s}_{t+1}$ and the targeted failure state $s_{fail}$ subject to the budget constraint, we encourage the victim agents to move closer to a damaging failure state in the next time step which consequently leads to low team reward.

Finally, the full attack algorithm of c-MBA at timestep $t$ can be summarized in Alg. 1.

**Algorithm 1** c-MBA algorithm at timestep $t$

1. **Initialization**: 
   2. Given $s_t$, $s_{fail}$, $\pi$, $f$, $\mathcal{V}_t$; initialize $\Delta s = \epsilon \ast \text{sign}(x)$ for $x \sim N(0, 1)$, attack budget $\epsilon$, $p$; choose learning rate $\eta > 0$
   3. For $k = 0, \ldots, K - 1$ do
      4. Compute $a^i_t = (a^1_t, \ldots, a^n_t)$ where $a^i_t = \pi^i(s^i_t + \Delta s^i)$ if $i \in \mathcal{V}_t$ and $a^i_t = \pi^i(s^i_t)$ otherwise.
      5. Compute $\hat{s}_{t+1} = f(s_t, a_t)$.
      6. Update $\Delta s_k$ using PGD.
   7. End For

**Learning dynamics model.** One of the key enabler to solve (1) is the availability of the learned dynamics model $f$. We
can learn the dynamics model via some function approximator such as neural networks. The learning of dynamics model can be formulated as solving the following optimization problem

$$\min_{\phi} \sum_{t \in \mathcal{D}} \| f(s_t, a_t; \phi) - s_{t+1} \|^2,$$

where \( \mathcal{D} \) is a collection of state-action transitions \( \{(s_t, a_t, s_{t+1})\}_{t \in \mathcal{D}} \) and \( s_{t+1} \) is the actual state that the environment transitions to after taking action \( a_t \) determined by a given policy. In particular, we separately collect transitions using the pre-trained policy \( \pi_{tr} \) and a random policy \( \pi_{rd} \) to obtain \( \mathcal{D}_{\text{train}} \) and \( \mathcal{D}_{\text{random}} \). The motivation of using the random policy to sample is to avoid overfitting the dynamics model to the trained policy. Then the dataset \( \mathcal{D} \) is built as \( \mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{random}} \). Since (2) is a standard supervised learning problem, the dynamics model \( f \) can be solved by existing gradient-based methods.

B. Learning failure states from data

In order to solve (1), we need to specify the failure state \( s_{\text{fail}} \) which often requires prior knowledge of the state definition of the c-MARL environment. To make our method more flexible, we propose the first data-driven approach to learn the failure state. As our c-MBA attack involves training a dynamics model of the environment by collecting the transition data, we can learn the failure state from directly the pre-collected dataset without extra overhead. Based on the observation that the failure state should be a resulting state where the reward corresponding to that transition is low, we sort the collected transition \( (s_t, a_t, s_{t+1}) \) by ascending order of the reward \( r_t \) and choose the failure state to be \( s_t^{\text{fail}} \) that corresponds to the lowest \( r_t \) in the dataset. This process is described in Algorithm 2.

**Algorithm 2** Learning failure state from collected data

1. **Input**: a dataset \( \mathcal{D} = \{(s_t, a_t, s_{t+1})\}_{t \in \mathcal{D}} \) as a collection of transitions.
2. Sort the transition by ascending order of reward \( r_t \)
3. Determine \((s_t^{(1)}, a_t^{(1)}, s_{t+1}^{(1)})\) as the transition corresponding to the minimum reward.
4. Set \( s_{\text{fail}} = s_t^{(1)} \).

Using this data-driven strategy, we show that performing c-MBA with the learned failure state either matches or performs better than the expert-defined failure state for multi-agent MuJoCo environments. In addition, the data-driven approach demonstrates its advantage in the multi-agent particle environment where we do not have expert knowledge of the state space. Overall, c-MBA using the learned failure state shows its superior performance over other model-free baselines in most cases in the experiments.

C. Stronger attack with c-MBA – victim agent selection strategy

We propose a novel strategy to select most vulnerable victim agents with the goal to further increase the power of our c-MBA. We note that this scenario is unique in the setting of multi-agent DRL. To the best of our knowledge, our work is the first to consider the victim selection strategy as [20] only use one fixed agent to perform the attack. As a result, we can craft a stronger attack by selecting appropriate set of “vulnerable” agents. This strategy can be effective in the “sparse attack” setting when only a few agents in the team are attacked [17]. To start with, we first formulate a mixed-integer program to perform the attack on a set of victim agents as

$$\min_{\Delta s, w} d(s_{t+1}, s_{\text{fail}})$$

s.t. 
- \( s_{t+1} = f(s_t, a_t) \)
- \( a_i = \pi(s_t^i + w_i \cdot \Delta s^i), \quad \forall i \in \mathcal{N} \)
- \( u^s \leq s_t^i + \Delta s^i \leq u_S, \quad \forall i \in \mathcal{N} \)
- \( \| \Delta s^i \|_p \leq \varepsilon, \quad \forall i \in \mathcal{N} \)
- \( w_i \in [0, 1], \quad \forall i \in \mathcal{N} \)
- \( \sum w_i = n_v \)

where we introduce a new set of binary variables \( \{w_i\} \) to select the suitable agent to attack. Due to the existence of the new binary variables, problem (3) is much harder to solve. Therefore, we instead propose to solve a proxy of (3) as follows

$$\min_{\Delta s, \theta} d(s_{t+1}, s_{\text{fail}})$$

s.t. 
- \( s_{t+1} = f(s_t, a_t) \)
- \( a_i = \pi((s_t^i + W_i(s_t; \theta) \cdot \Delta s^i)), \quad \forall i \in \mathcal{N} \)
- \( u^s \leq s_t^i + \Delta s^i \leq u_S, \quad \forall i \in \mathcal{N} \)
- \( \| \Delta s^i \|_p \leq \varepsilon, \quad \forall i \in \mathcal{N} \)
- \( 0 \leq W_i(s_t; \theta) \leq 1, \quad \forall i \in \mathcal{N} \)

where \( W(s; \theta) : \mathcal{S} \rightarrow \mathbb{R}^n \) is a function parametrized by \( \theta \) that takes current state \( s \) as input and returns the weight to distribute the noise to each agent. Suppose we represent \( W(s; \theta) \) by a neural network, we can rewrite the formulation (4) as (5) because the last constraint in (4) can be enforced by using a softmax activation in the neural network \( W(s; \theta) \).

$$\min_{\Delta s, \theta} d(f(s_t, \pi((s_t^i + W_i(s_t; \theta) \cdot \Delta s^i)_i)), s_{\text{fail}})$$

s.t. 
- \( \Delta s \in \mathcal{C}_{p,c,t} \)

where the notation \( \pi((s_t^i + W_i(s_t; \theta) \cdot \Delta s^i)_i) \) denotes the concatenation of action vectors from each agent to form a global action vector \( \pi(s_t^i + W_i(s_t; \theta) \cdot \Delta s^i) \), \( \ldots, \pi(s_t^n + W_n(s_t; \theta) \cdot \Delta s^n) \). As a result, (5) can be efficiently solved by PGD. We present the pseudo-code of the attack in Alg. 3. After \( K \) steps of PGD update, we define the \( l(n_{i+1}) \) as index of the \( j \)-th largest value within \( W(s_t; \theta_K) \in \mathbb{R}^n \), i.e., \( W_{i,n_{l(i+1)}}(s_t; \theta_K) \geq W_{i,n_{l(i)}}(s_t; \theta_K) \geq \cdots \geq W_{i,1}(s_t; \theta_K) \).

Let \( I_j \) be the index set of top-\( j \)-largest outputs of the \( W(s_t; \theta_K) \) network. The final perturbation returned by our victim agent selection strategy will be \( \Delta s = (\Delta s_1, \ldots, \Delta s^n) \) where \( \Delta s_i = 0 \) if \( i \notin I_{n_{l(i+1)}} \) and \( \Delta s^i = (\Delta s_i^1, \ldots, \Delta s_i^n) \) if \( i \notin I_{n_{l(i+1)}} \).

**Remark III.1.** For this attack, we assume each agent \( i \) has access to the other agent’s state to form the joint state \( s_i \). If the set of victim agents is pre-selected, we do not need this assumption and the adversarial attack can be performed at each agent independently of others.

IV. EXPERIMENTS

We perform the attack on four multi-agent MuJoCo (MA-MuJoCo) environments [23] including Ant(4x2), HalfCheet-
Algorithm 3 cMBA with victim agent selection at time-step \( t \)

1. **Initialization:** Given \( s_t, s_{fail}, \pi, f, n_v \); initialize \( \Delta s_0 \);
   choose learning rate \( \eta, \lambda > 0 \).
2. **For** \( k = 0, \ldots, K-1 \) **do**
   3. Compute \( a_k = \pi(\{s^*_i + W_i(s_t; \theta) \cdot \Delta s^1\}_i) \).
   4. Compute \( s_{t+1} = f(s_t, a_k) \).
   5. Update \( \Delta s_{k+1} = \text{proj}_{c_{\mathcal{C}}}(\Delta s_k - \eta \nabla \Delta s d(s_{t+1}, s_{fail})) \).
   6. Update \( \theta_{k+1} = \theta_k - \lambda \nabla \theta d(s_{t+1}, s_{fail}) \).
   7. **End For**
8. Compute \( I_{n_v} = \{i_{(n)} \cdots , i_{(n-v)}\} \) such that \( W_{i_{(n)}}(s_t, \theta_K) \geq \cdots \geq W_{i_{(1)}}(s_t, \theta_K) \).
9. Return \( \Delta t = ((\Delta s)^1, \cdots , (\Delta s)^v) \) where \( (\Delta s)^i = (\Delta s_K)^i \) if \( i \in I_{n_v} \) and \((\Delta s)^i = 0\) otherwise.

\( \text{tah}(2x3), \text{HalfCheetah}(6x1), \) and \( \text{Walker2d}(2x3) \). We also demonstrate the effectiveness of the learned failure state approach using the multi-agent particle environment, denoted as \( \text{MPE}(3x5) \) [22, 24] where we do not have expert knowledge of the failure state. For all environments, we use MADDPG [22] to obtain trained MARL agents. To perform the attack, we consider the following **model-free** baselines:

1) **Uniform**: the perturbation follows the Uniform distribution \( U(-\varepsilon, \varepsilon) \).
2) **Gaussian**: the perturbation follows the Normal distribution \( \mathcal{N}(0, \varepsilon) \).
3) **Lin et al. (2020) + iFGSM**: Since there is no other work performing adversarial attack for continuous action space in c-MARL, we adapt the approach in [20] to form another baseline.

In our experiments, we consider two variants of c-MBA:

1) **c-MBA-F**: we perform c-MBA using an expert-defined failure state where we have the knowledge of the state definition of the c-MARL environment.
2) **c-MBA-D**: we perform c-MBA attack using failure state learned from the collected data as described in Section III-B.

We defer the details about experiment setups and complete experimental results to [25].

**Experiment (I)** – **model-free baselines vs model-based attack c-MBA on \( \ell_\infty \) perturbation.** We compare the 3 baseline attacks with two variants of our model-based attack on various MA-MuJoCo environments with one victim agent \( (n_v = 1) \) in coherence with [20]. Fig. 2 illustrates the performance when we perform these attacks on each agent with different attack budget using \( \ell_\infty \)-norm. Our c-MBA yields much lower rewards under relatively low budget constraints (when \( \varepsilon \in [0.05, 0.2] \)) compared to other baselines. We also observe that c-MBA-D either matches or performs better than c-MBA-F with the expert-defined failure state.

**Experiment (II)** – **effectiveness of learned adaptive victim selection.** We consider the following variants of our model-based attack:

1) **c-MBA(fixed agents)**: attack a fixed set of victim agents with Alg. 1. We use c-MBA (best fixed agents) to denote the best result among fixed agents.
2) **c-MBA(random agents)**: uniformly randomly select victim agents to attack with Alg. 1.
3) **c-MBA(greedy agents selection)**: for each time step, sweep all possible subsets of agents with size \( n_v \) and perform Alg. 1. Select the final victim agents corresponding to the objective value (distance between predicted observation and target failure observation).
4) **c-MBA(learned agents selection)**: use Alg. 3 to attack the most vulnerable agents.
5) **c-MBA(learned agents selection + Alg. 1)**: use Alg. 3 to select the most vulnerable agents to attack then perform the attack with the selected agents with Alg. 1.

Fig. 3 illustrates the results on MA-MuJoCo environments. c-MBA(learned agents selection) and c-MBA(learned agents selection + Alg. 1) appear to be better than the random or
greedy strategy and is comparable with the randomly selected one in HalfCheetah(2x3) environment. It is interesting to observe that c-MBA(learned agents selection + Alg. 1) is either comparable or better than c-MBA(learned agents selection) which shows that running Alg. 1 using agents selected by Alg. 3 can be beneficial. We show that the victim selection technique is really important as randomly choosing the agents to attack (i.e. c-MBA(random agents)) or even choosing agents greedily (i.e. c-MBA(random agents)) cannot form an effective attack and are in fact, even worse than the attacking on fixed agents (i.e c-MBA(best fixed agents)) in most cases.

Experiment (III) – attacking two agents using model-free baselines vs model-based attack c-MBA using $\ell_\infty$ constrained: We use model-free and model-based approaches to simultaneously attack two agents in Ant(4x2) environment. Fig. 4 illustrate the performance of various attacks. We observe that our c-MBA attack outperforms other baselines.

Fig. 4. c-MBA vs baselines in Ant(4x2) when attacking 2 agents - Exp. (III).

Experiment (IV) – model-free baselines vs model-based attack c-MBA in MPE(3x5) environment: We perform the same procedure as in Experiment (III) to attack different agents in the MPE(3x5) environments where we attack two or three agents at the same time. Since we do not have expert knowledge about the failure state in this environment, we compare c-MBA-D with other model-free baselines. The results are depicted in Figure 5 where c-MBA-D perform slightly better than other random noise baselines when attacking 2 agents and significantly outperforms other baselines when attacking three agents simultaneously. We note that [20] does not perform well in this experiment as we observe during training the adversarial policy that it could not lower the team reward effectively.

Experiment (V): model-free baselines vs model-based attack c-MBA on $\ell_1$ perturbation. In addition to the $\ell_\infty$-norm budget constraint, we also evaluate adversarial attacks using the $\ell_1$-norm constraint. Note that using $\ell_1$-norm for budget constraint is more challenging as the attack needs to distribute the perturbation across all observations while in the $\ell_\infty$-norm the computation of perturbation for individual observation is independent. Fig. 6 illustrate the effect of different attacks on HalfCheetah(6x1) and Walker2d(2x3) environments, respectively. Our c-MBA is able to outperform other approaches in almost all settings.

Fig. 5. c-MBA vs baselines in MPE(3x5) when attacking two agents (3 leftmost) and three agents (rightmost) simultaneously using $\ell_\infty$ constraint - Exp. (IV).

Fig. 6. c-MBA vs baselines in HalfCheetah(6x1) and Walker2d(2x3) - Exp. (V).

In summary, our c-MBA attack is able to shows its advantage in all tested multi-agent environment where it achieves lower reward with smaller budget level. Moreover, c-MBA with the
victim agent selection has been shown to constantly performs better than the original c-MBA variant as seen in Figure 3.

V. Conclusions

We propose a new attack algorithm named c-MBA for evaluating the robustness of c-MARL environment. Our algorithm is the first model-based attack to craft adversarial observation perturbations and we have shown that c-MBA outperforms existing model-free attack baselines by a large margin under both MA-MuJoCo and MPE benchmarks. Unique to multi-agent setting, we also propose a new victim-selection strategy to select the most vulnerable victim agents given the attack budget, which has not been studied in prior work. We show that with the victim-agent selection strategy, c-MBA attack becomes stronger and more effective. We also propose the first data-driven approach to determine the failure state based on the pre-collected data without extra overhead making our attack more flexible.

REFERENCES

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” arXiv:1312.6199, 2013.
[2] A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 427–436.
[3] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” arXiv:1412.6572, 2014.
[4] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” in 2016 IEEE European symposium on security and privacy (EuroS&P). IEEE, 2016, pp. 372–387.
[5] R. Jia and P. Liang, “Adversarial examples for evaluating reading comprehension systems,” arXiv:1707.07328, 2017.
[6] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, “Adversarial attacks on deep-learning models in natural language processing: A survey,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 11, no. 3, pp. 1–41, 2020.
[7] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, “Is bert really robust? a strong baseline for natural language attack on text classification and entailment,” in Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 05, 2020, pp. 8018–8025.
[8] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, “Generating natural language adversarial examples,” arXiv:1804.07998, 2018.
[9] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, “Adversarial attacks on neural network policies,” arXiv:1702.02284, 2017.
[10] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, “Tactics of adversarial attack on deep reinforcement learning agents,” arXiv:1703.06748, 2017.
[11] J. Kos and D. Song, “Delving into adversarial attacks on deep policies,” arXiv:1705.06452, 2017.
[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in International conference on machine learning. PMLR, 2016, pp. 1928–1937.
[13] T.-W. Weng, K. D. Dvijotham, J. Uesato, K. Xiao, S. Gowal, R. Stanforth, and P. Kohli, “Toward evaluating robustness of deep reinforcement learning with continuous control,” in International Conference on Learning Representations, 2019.
[14] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, “Adversarial policies: Attacking deep reinforcement learning,” arXiv:1905.10615, 2019.
[15] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, “A taxonomy for swarm robots,” in Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS’93), vol. 1. IEEE, 1993, pp. 441–447.
[16] A. Oroojlooyjadid and D. Hajinezhad, “A review of cooperative multi-agent deep reinforcement learning,” arXiv:1908.03963, 2019.
[17] Y. Hu and Z. Zhang, “Sparse adversarial attack in multi-agent reinforcement learning,” arXiv:2205.09362, 2022.
[18] N. Elhami Fard and R. R. Selmic, “Adversarial attacks on heterogeneous multi-agent deep reinforcement learning system with time-delayed data transmission,” Journal of Sensor and Actuator Networks, vol. 11, no. 3, p. 45, 2022.
[19] J. Guo, Y. Chen, Y. Hao, Z. Yin, Y. Yu, and S. Li, “Towards comprehensive testing on the robustness of cooperative multi-agent reinforcement learning,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 115–122.
[20] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, “On the robustness of cooperative multi-agent reinforcement learning,” in 2020 IEEE Security and Privacy Workshops (SPW). IEEE, 2020, pp. 62–68.
[21] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, “The StarCraft Multi-Agent Challenge,” CoRR, vol. abs/1902.04043, 2019.
[22] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” arXiv:1706.02275, 2017.
[23] C. S. de Witt, “Multi-agent mujoco,” https://github.com/schroederdewitt/multiagent_mujoco, 2020.
[24] I. Mordatch and P. Abbeel, “Emergence of grounded compositional language in multi-agent populations,” arXiv:1703.04908, 2017.
[25] N. H. Pham, L. M. Nguyen, J. Chen, H. T. Lam, S. Das, and T.-W. Weng, “Attacking c-marl more effectively: A data driven approach,” arXiv:2202.03558, 2023.