On the Robustness of Cooperative Multi-Agent Reinforcement Learning

Jieyu Lin*, Kristina Dzeparoska*, Sai Qian Zhang†, Alberto Leon-Garcia*, Nicolas Papernot*‡
*University of Toronto
†Harvard University
‡Vector Institute
{jieyu.lin, kristina.dzeparoska}@mail.utoronto.ca

***Abstract*—In cooperative multi-agent reinforcement learning (c-MARL), agents learn to cooperatively take actions as a team to maximize a total team reward. We analyze the robustness of c-MARL to adversaries capable of attacking one of the agents on a team. Through the ability to manipulate this agent's observations, the adversary seeks to decrease the total team reward.**

Attacking c-MARL is challenging for three reasons: first, it is difficult to estimate team rewards or how they are impacted by an agent mispredicting; second, models are non-differentiable; and third, the feature space is low-dimensional. Thus, we introduce a novel attack. The attacker first trains a policy network with reinforcement learning to find a wrong action it should encourage the victim agent to take. Then, the adversary uses targeted adversarial examples to force the victim to take this action.

Our results on the StartCraft II multi-agent benchmark demonstrate that c-MARL teams are highly vulnerable to perturbations applied to one of their agent's observations. By attacking a single agent, our attack method has highly negative impact on the overall team reward, reducing it from 20 to 9.4. This results in the team's winning rate to go down from 98.9% to 0%.

I. Introduction

Algorithms for cooperative Multi-Agent Reinforcement Learning (c-MARL) are driving progress in a wide range of applications such as traffic light control [1], autonomous driving [2], and cellular base station control [3]. Because they involve critical infrastructure, understanding the robustness of c-MARL agents to adversarial manipulations of their environment is a pre-requisite to their deployment in production.

The goal of Reinforcement Learning (RL) is to optimize the actions taken by agents in complex environments, by learning actions that effectively maximize a long-term reward. Techniques for c-MARL were introduced to adapt RL algorithms to environments in which a team of multiple RL agents interact. Agents that form a team collaborate towards a common goal. c-MARL presents a number of key challenges including non-stationary environment, and limited bandwidth for information sharing internal to a team once training is completed and agents are deployed [4]. Combined with partial observability (any given agent may not always have access to all observations made by other agents in the team), it is more difficult for an agent in c-MARL to select the best action and maximize the total team reward.

RL agents are known to be vulnerable to adversaries perturbing their observations with adversarial examples [5], as well as adversaries directly controlling the actions of one of the victim's opponents [6]. When exploited, this vulnerability results in agents taking unintended actions, often with adverse consequences. We hypothesize that cooperative aspects of c-MARL agents, which enable them to outperform classic RL agents when operating as a team, also increase the vulnerability of a team to failures of one of its constituent agents.

To the best of our knowledge, this work is the first to study c-MARL in adversarial settings and characterize the attack surface exposed by enabling agents to cooperate more effectively. Our goal is to understand the impact an adversary can have on the long-term reward of a team of agents. We consider an adversary who is capable of controlling the observations of a single *victim agent*. The adversary mounts their attack by presenting the victim agent with adversarial examples. As c-MARL systems are increasingly deployed in critical infrastructure, there is a pressing need to assess their robustness in order to develop tailored defense mechanisms that promote security and safety in c-MARL.

Several aspects of c-MARL preclude us from directly applying known attack algorithms for RL agents:

1) **Team Reward Estimation & Non-Differentiability**: Because joint value estimation networks used to estimate a team's total reward are trained with the assumption that agents will always choose the optimal action, they do not accurately predict the team reward when some of the agents misbehave (i.e., select non-optimal actions). Furthermore, estimating the team reward involves non-differentiable operations such as computing a maximum. This makes it difficult for an attacker to estimate (and thus decrease) the total team reward.
2) **Misclassification $\neq$ Reward**: High misclassification rates are not sufficient to lead to failure of RL. Wrong actions taken by agents at any point in time need to impact long-term rewards for the adversary to succeed.
3) **Low-dimensional Input Feature Space**: In many c-MARL settings, agents are processing feature vectors with lower dimensions than images; hence, this limits the ways an adversary can perturb these observations.

To address these challenges, we propose a novel two-step attack method (see Figure 2). The first step uses RL to train an adversarial policy. This policy helps the adversary select actions which, if taken by an agent, would minimize the total

© 2020, Jieyu Lin. Under license to IEEE.
DOI 10.1109/SPW50608.2020.00027

Authorized licensed use limited to: Mohammed VI Polytechnic University. Downloaded on January 07,2025 at 14:44:42 UTC from IEEE Xplore. Restrictions apply.

team reward. This method only requires black-box access to the agent. The second step perturbs the agent's observation with an adversarial example so as to lure it to perform this adversarial action. To accommodate for the specificities of c-MARL, we refine existing approaches for finding targeted adversarial examples. In particular, we address the low input dimensionality and lack of smoothness of the optimization objective by adaptively setting hyperparameters of the attack. Finally, we introduce a regularization penalty to the training objective of our adversarial policy (the first step of the attack) to ensure that this first step produces a target adversarial action that is more easily achievable by the second step of the attack. This ensures that adversarial examples found can consistently decrease the reward and team win rate simultaneously.

In our evaluation, we show that by perturbing the observation of a single agent, we can jeopardize the team almost as effectively as an adversary directly controlling this agent's actions. This significantly reduces the total team reward of the c-MARL system. Our results on the StartCraft II multi-agent benchmark [7] show that an attacker able to perturb with an average of 8.3 L1 norm (out of 96 features ranging from -1 to 1) in the victim's input observation can successfully degrade the system's team win rate from 98.9% to 0% (because its reward goes down from 20 to 9.4).

Considering we were able to produce such results by attacking a single c-MARL agent, we emphasize in Section V the necessity for suitable defenses, robust agents, and other protection measures against adversarial example attacks. Our contributions are listed as follows:

- We propose a novel two-step attack for c-MARL system that is capable of reducing the total team reward by only perturbing the observation of a single agent.
- We regularize the training of our adversarial policy to improve our two-step attack, demonstrating that reinforcement learning can be an effective adaptive strategy in the presence of difficult optimization landscapes when crafting adversarial examples (e.g., as is the case when defenses perform gradient masking [8], [9]).
- We extend existing adversarial example methods to create d-JSMA which is suitable for attacking RL model with low-dimensional feature space.
- We demonstrate the feasibility and effectiveness of our attack on c-MARL in the StarCraft II multi-agent benchmark. Our adversary brings down the team win rate from 98.9% to 0% by perturbing a single agent's observations.

II. Problem Statement

A. Centralized Training Decentralized Execution and QMIX

In c-MARL, due to the partial observability and limited communication among agents, learning decentralized policies is required to support decentralized execution. Current state-of-the-art MARL methods apply centralized training scheme, where agents can exchange their local information during training phase, and act independently during execution.

One approach for centralized training and decentralized execution is QMIX [10] (Figure 1), a multi-agent Deep Q-learning (DQN) algorithm, where each agent uses an RNN-based DQN network to estimate the action values based on their partial observation. DQN has recently achieved success in a variety of RL tasks. It uses experience replay and a target network to improve stability and convergence of RL training. To maximize the total team reward, QMIX estimates the joint action values through a mixing network that takes in each agent's selected action Q-value (i.e., action with max Q-value) and the current state to estimate the total team reward, Q_{total}. With an accurate estimation of Q_{total}, we can fine-tune the individual agent's Q network, so a greater total team reward can be achieved during execution.

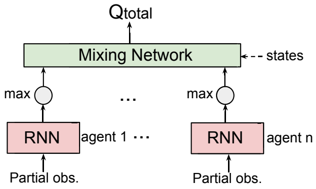


Fig. 1: High-level architecture of the QMIX algorithm.

B. Threat Model

We model the c-MARL game as a Dec-POMDP [11] with N agents cooperating to execute some tasks, with the goal of maximizing their total team reward. More specifically, at each time step t, each agent i ($0 < i < N$) takes the (partial) observation o_t^i as input, and performs an action a_t^i with the highest action value. Let $\boldsymbol{a_t} = \{a_t^i\}$ be a vector of actions taken by all the agents at time t. The goal is to find the optimal $\boldsymbol{a_t}$ that maximizes the total team reward R.

Our threat model considers a single vulnerable agent $\hat{i}$ whose observation $o_t^{\hat{i}}$ can be modified by the adversary. We assume the adversary has already predetermined this agent. This is realistic, as in a real-world distributed system environment, an adversary can potentially gain access to a single node, but may not be able to affect all nodes. The adversary's goal is to minimize R by making a minor modification to $o_t^{\hat{i}}$.

C. Attack Overview

Figure 2 shows a high-level diagram of our two-step attack. In the first step, we use RL to learn an adversarial policy that estimates which action, if taken by the victim, would result in the largest decrease in the total reward R. In the second step, we use gradient-based targeted adversarial example crafting to perturb the victim's observation such that it will take the action returned by the adversarial policy in the first step. Intuitively, using RL rather than a gradient-based strategy in the first step circumvents any non-differentiable operation found in how QMIX estimates the total reward R. Nevertheless, in the second step, the attacker can still apply gradient-based methods to craft adversarial examples—once it has identified the target action, these adversarial examples should conduce.

Our two-step attack can be implemented in realistic threat models. The first step only assumes black-box access: adversaries query the black-box agent policy and its environ-

Authorized licensed use limited to: Mohammed VI Polytechnic University. Downloaded on January 07,2025 at 14:44:42 UTC from IEEE Xplore. Restrictions apply.

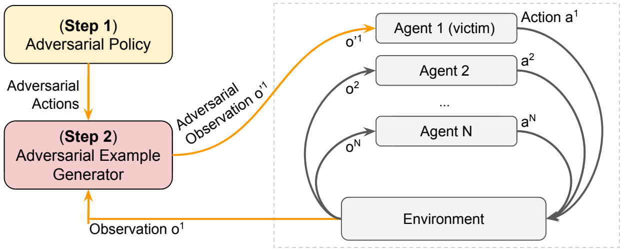


Fig. 2: Two-step attack on a single agent in a c-MARL environment, as depicted by the orange lines and boxes.

ment. The second step assumes the adversary has sufficient access to find adversarial examples. We instantiate a gradient-based strategy, which requires access to the victim's model parameters and observations. However, our attack could be extended to the black-box setting with finite-differences [12] or by creating a replica of the victim's model and leveraging the transferability property of adversarial examples.

III. Attack Methods

In this section, we discuss both steps of our attack in details.

A. *Learning adversarial policy*

In this part of the attack, the goal is to optimize the behavior of the victim agent so its actions will result in a decreased total team reward R. We first explore naïve ways of selecting target actions for the victim agent:

- **Random action**: Randomly selecting an available action.
- **Local worst action (LW)**: Selecting the action with the lowest Q-value based on the output of its Q network.
- **QMIX worst action**: Selecting the action leading to the min Q_{total} value output by the mixing network. We omit the max operation, and instead feed each available action's Q-value separately through the mixing network, and select the action leading to the worst Q_{total}.

Despite the success of the above methods in reducing R, the victim agent's negative effect upon the team performance remains limited. To help improve the robustness of c-MARL, it is important to understand the worst-case behavior of a single agent. We propose RL-based optimization to more effectively select an action for the victim agent. We design two variants:

- **Optimized worst adversarial policy (OW)**: Using RL to learn a victim agent policy that minimizes R.
- **OW with regularization (OWR)**: Adding regularization to OW, to be more efficiently combined with the adversarial example search performed in step two of the attack.

OW: To use RL to train an adversarial policy, we first turn to the formulation of c-MARL. Following the notations in Section II-B, the goal is to maximize the sum of discounted team rewards over time t:

$$\underset{\theta^1,\theta^2,\ldots\theta^N}{\operatorname{argmax}} \sum_{t=0}^{\infty} \gamma^t R_t(s_t, \boldsymbol{a_t}, s_{t+1}) \tag{1}$$

where θ represents the parameters for each agent's policy, γ is the reward discount factor, s_t is the state, and s_{t+1} is determined by a transition function T in the environment. The goal of centralized training is to maximize the total reward by tuning the parameters of all the agents' policy.

In an adversarial environment, we can consider the policies of all the non-victim agents as fixed, i.e. deterministic, hence from the victim's perspective, they are considered as part of the environment. Thus, we can formulate our attack as a single agent RL problem, with the goal of minimizing R:

$$\underset{\theta'^v}{\operatorname{argmin}} \sum_{t=0}^{\infty} \gamma^t R_t(s_t, a_t^v, s_{t+1}) \tag{2}$$

where a_t^v is the victim agent's action at time step t, and θ'^v parameterizes the victim agent's policy. With Equation 2, we can train an adversarial policy that selects actions for the victim agent that will minimize R. In our implementation, we use DQN to train such an adversarial policy.

OWR: Despite OW's ability to learn an effective adversarial policy, this may not be sufficient for the adversary to succeed. Some of the target actions returned by the adversarial policy in this first step may be difficult to achieve given a limited perturbation budget when crafting an adversarial example for the victim agent in the second step. Thus, we propose OWR, which regularizes the objective optimized to train the adversarial policy. The new penalty reflects how difficult it would be to find an adversarial example for the predicted target.

Indeed, we observe that the adversarial example search sometimes fails to achieve a target action when the difference between the target action Q-value and the original action Q-value becomes too large. Intuitively, some actions are harder to achieve than others through adversarial examples because they are further away in the agent's decision space. To improve the attack success rate, we observe that the victim does not need to perform the most optimal action at each time step, instead it should focus on selecting the most optimal actions for the steps that have high negative impact on the team reward.

To incorporate this idea, we modify the DQN training used in OW to include a regularization term. The loss function of the attacker's Q network with regularization is defined as:

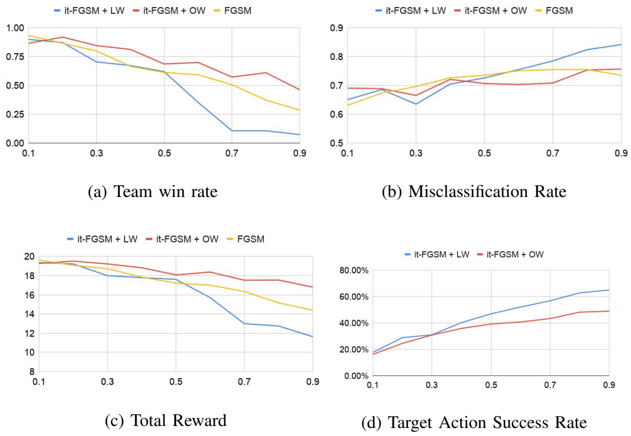
$$L = \sum_{b=1}^{B} \sum_{t=1}^{T} [(y_{target}^b - Q_{target}(o_t^{v,b}, a_t^{v,b}))^2 + \lambda {d_{diff}}^2] \tag{3}$$

where $y_{target}^b = r_t^b + \gamma max_{a_{t+1}} Q_{target}(o_{t+1}^{v,b}, a_{t+1}^{v,b})$, r_t^b is the reward for time step t, d_{diff} is the Q-value difference between the target action and the original best action, λ is the weight of the d_{diff} regularization, and b represents the batch index.

B. *Adversarial example generation to lure victim agents*

Next, the attack perturbs observations to have the victim select the action output by the adversarial policy from Section III-A. One could hope to apply existing gradient-based adversarial example algorithms. However, the attack needs to be targeted. Indeed, we find in Figure 3a that naively applying

Authorized licensed use limited to: Mohammed VI Polytechnic University. Downloaded on January 07,2025 at 14:44:42 UTC from IEEE Xplore. Restrictions apply.



(a) Team win rate

(b) Misclassification Rate

(c) Total Reward

(d) Target Action Success Rate

Fig. 3: Performance of FGSM-based methods as a function of the perturbation budget ϵ.

the FGSM [13] results in total team rewards (28.7%) that remain higher than what could be achieved by a random action baseline shown in Table I (3.4%). The attack also needs to retain sufficient flexibility to handle constraints placed on low-dimensional inputs observed from the agent's environment. This would require finding a differentiable loss function that hardcodes some of these constraints for the CW method [14] to be applicable.

To more effectively craft adversarial examples for reducing R, we extended existing approaches to create two new methods: *Iterative target-based FGSM method (it-FGSM)* and *Dynamic budget JSMA attack (d-JSMA)*.

it-FGSM: We adapt the i-FGSM [15] algorithm to a targeted method by taking gradient on the target Q-value instead of the loss, that is changing $\nabla_X L(X)$ to $\nabla_X Q_{target}(X)$.

Figure 3 shows the performance of it-FGSM compared to FGSM. it-FGSM methods are more effective in reducing the total team reward and team win rate. it-FGSM + LW achieves the best performance by reducing the team reward and team winning rate to 11.63 and 7.33%, respectively. However, we observe the performance of it-FGSM + OW is sub-optimal due to the lower target action success rate. Therefore, we conclude that the FGSM-based method is still not sufficiently effective for our problem setting.

d-JSMA: JSMA [16] is by default a targeted method, so it is suitable for our setting. It constructs a saliency map and perturbs the features with the highest saliency score (i.e. features that have large positive gradients for the target class and large negative sum of gradients for the non-target class). In our attempt to apply JSMA to our problem, we encountered two challenges, hence proposed d-JSMA to overcome them:

1. **Two features bidirectional perturbation**: In our setting, when perturbing only a single feature, it is difficult to find features that have a positive gradient to our target action, but negative sum for the rest. Figure 4 shows an example gradient of action Q-values, where the gradients of different actions with respect to a single feature are often similar. Hence, perturbing a feature that increases the target action's Q-value inadvertently causes an increase to the rest of the actions (i.e. all Q-values change in the same direction). This is understandable in an RL setting, as changing a certain feature (e.g. victim's health has the highest gradient value in Figure 4) may impact the final team reward regardless of the action.

We address this issue by perturbing two features at the same time, by modifying Algorithm 3 proposed in [16]. Formally, in d-JSMA, the attacker wants to perturb two input features simultaneously to increase the output Q-value of the target action Q_t and decrease the output Q-values of the non-target actions, until the $\operatorname{argmax}(Q)$ selects the target action. We can accomplish this by perturbing the input features using the following saliency map $\boldsymbol{S}(\boldsymbol{X}, t)$ and perturbation direction $d_{perturbation}(\boldsymbol{X}, t)$

$$\boldsymbol{S}(\boldsymbol{X},t)[i,j] = \begin{cases} 0 \text{ if } \left(\frac{\partial \boldsymbol{Q}_t(\boldsymbol{X})}{\partial \boldsymbol{X_i}} + \frac{\partial \boldsymbol{Q}_t(\boldsymbol{X})}{\partial \boldsymbol{X_j}}\right)\left[\sum\limits_{k \neq t}\left(\frac{\partial \boldsymbol{Q}_k(\boldsymbol{X})}{\partial \boldsymbol{X_i}} + \frac{\partial \boldsymbol{Q}_k(\boldsymbol{X})}{\partial \boldsymbol{X_j}}\right)\right] > 0 \\ -\left(\frac{\partial \boldsymbol{Q}_t(\boldsymbol{X})}{\partial \boldsymbol{X_i}} + \frac{\partial \boldsymbol{Q}_t(\boldsymbol{X})}{\partial \boldsymbol{X_j}}\right)\left[\sum\limits_{k \neq t}\left(\frac{\partial \boldsymbol{Q}_k(\boldsymbol{X})}{\partial \boldsymbol{X_i}} + \frac{\partial \boldsymbol{Q}_k(\boldsymbol{X})}{\partial \boldsymbol{X_j}}\right)\right] \text{ otherwise} \end{cases} \tag{4}$$

$$d_{perturbation}(\boldsymbol{X}, t)[i,j] = Sign\left(\frac{\partial \boldsymbol{Q}_t(\boldsymbol{X})}{\partial \boldsymbol{X_i}} + \frac{\partial \boldsymbol{Q}_t(\boldsymbol{X})}{\partial \boldsymbol{X_j}}\right) \tag{5}$$

where $\boldsymbol{X}$ is the victim agent's observation, i, j are the indices of the input features. The condition in the first line of Equation 4 filters out feature pairs whose perturbation leads to Q-value changes in the same direction for both target and non-target actions. If a pair of features with indices i, j passes this filter, then its perturbation direction is determined by $d_{perturbation}$. To find an adversarial example, our method iteratively finds a pair of input features that has the highest saliency score and following:

$$\begin{aligned} &\boldsymbol{X}_{adv,0} = \boldsymbol{X} \\ &\boldsymbol{X}_{adv,t_{iter}}[i] = \boldsymbol{X}_{adv,t_{iter}-1}[i] + d_{perturbation} \times \theta \\ &\boldsymbol{X}_{adv,t_{iter}}[j] = \boldsymbol{X}_{adv,t_{iter}-1}[j] + d_{perturbation} \times \theta \end{aligned} \tag{6}$$

where the θ is the step size of the perturbation, and t_{iter} is the number of iterations. For each time step in a c-MARL game, this method is invoked iteratively until a successful adversarial example is found or the t_{iter} reaches max iteration.

The method finds a saliency score for each pair of feature gradients and creates a bigger saliency map. By considering two feature gradients at the same time, we obtain more accurate estimations for the output Q-value contour of the model, which further helps us find better options to explore the output space. This method differs from the original JSMA implementation in that it considers both the positive target action Q-value gradient, as well as the negative ones (in which case the $d_{perturbation}$ is negative). We refer to this method as JSMA, using the notation d-JSMA to denote dynamic θ, and JSMA for a fixed θ. To further improve the efficiency, more than two feature points can be considered at the same time,

Authorized licensed use limited to: Mohammed VI Polytechnic University. Downloaded on January 07,2025 at 14:44:42 UTC from IEEE Xplore. Restrictions apply.

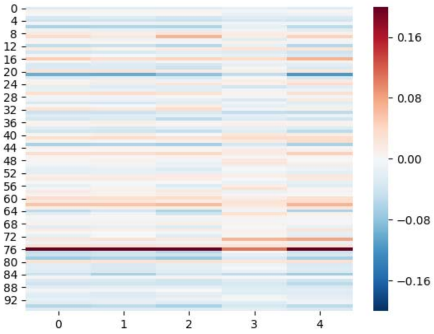


Fig. 4: Heat map of the gradient of action Q-values with respect to the input features. The x-axis denotes the available actions a_i, in this case $0 \leq i \leq 4$; y-axis denotes the observation features, o_i, $0 \leq i \leq 96$

however, the calculation complexity is $O(n^d)$ (d is the number of features).

2. **Dynamic $\boldsymbol{\theta}$**: JSMA has a hyperparameter θ which controls the perturbation size for each attack iteration. In our problem setup, we found that a fixed θ does not perform well, and a large θ does not always outperform a small θ. One hypothesis for this behavior is due to the fast-changing gradient in the output contour surface. To achieve better attack performance, we propose using dynamic θ in the d-JSMA method. For each time step in the c-MARL game, we start with a low θ value for our attack, and if the attack is not successful, we increase the size of θ and retry the attack until we reach the maximum θ value.

IV. Experiments

A. Experimental Setup:

We evaluate our attacks on the StarCraft Multi-Agent Challenge (SMAC) [7]. StarCraft II is a real-time strategy game used widely as a benchmark for RL research. We used the ”2s3z” SMAC map (2 ”Stalkers”, 3 ”Zealots” per team) for our scenarios. Our ally team is a c-MARL system whose goal is to defeat the opponents. For our attack, one of the ally Stalkers is the victim, and the rest of the allies use their original policy, trained in a centralized manner to achieve high team win rate. More details on the features and actions can be found in [7].

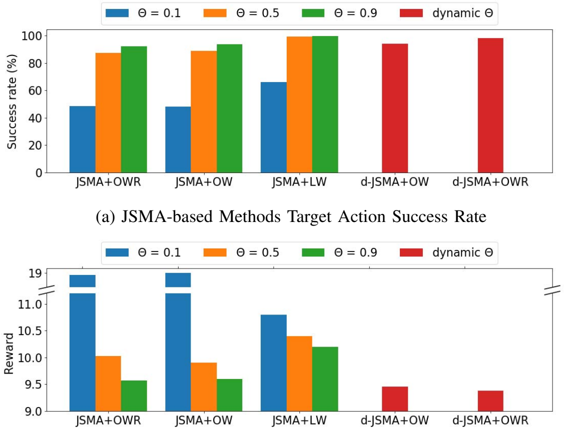
B. Results and Discussion:

1) Learning adversarial policy: To learn an adversarial policy for the victim agent, we applied the methods from III-A. We trained an adversarial policy for the victim to select sub-optimal actions that minimize the total reward. To evaluate the performance of the policy, we directly control the action of the victim based on the output of our adversarial policy. We ran each attack method for 500 games and presented the results in table I. From the results, OW and OWR have the highest negative impact on team reward and win rate, with 100% loss rate, making these two methods the most efficient.

2) d-JSMA methods: We evaluate d-JSMA with fixed attack step size θ (0.1 - 0.9), and a dynamic one. We run 100 games for each configuration and report their average results. For the target action, we combine d-JSMA with LW, OW and OWR.

TABLE I: Learning adversarial policies to select the victim’s target action.

	Average Reward	Team Win Rate
Baseline	20.00	99.80%
Random	11.10	3.40%
QMIX Worst	10.58	1.41%
Local Worst	10.52	1.00%
Optimized Worst (OW)	9.39	**0%**
Optimized Worst with Reg (OWR)	**9.35**	**0%**



(a) JSMA-based Methods Target Action Success Rate

(b) JSMA-based Methods Total Reward

Fig. 5: JSMA-based Methods Evaluation under Different θ

Figure 5 shows the total reward, and target action success rate for d-JSMA and JSMA with different θ values. Table II shows the team win rate for both methods. The results show that as we increase the input perturbation attack budget, the total team reward goes down as expected. At a low budget, JSMA+LW has relatively better attack performance. If a higher budget is available, then d-JSMA+OWR is able to achieve the lowest reward. Moreover, d-JSMA+OWR achieves better performance than JSMA+LW, even if the target action success rate is slightly lower, hence showing that the underlying OWR action selection method is more efficient. In this experiment, all the features are assumed to be continuous. Considering some of the features in the SMAC benchmark are discrete in nature, we have also conducted experiments by only perturbing the continuous features, and we did not notice a major performance difference.

3) Overall performance & budget comparison of all methods: We evaluate the overall performance of the two-step attack methods that combine the learning adversarial policy methods and the adversarial example generation methods. Table III shows the best attack performance for each method and the total team reward reduction achieved. Based on the results, d-JSMA+OWR achieves the best attack performance. It achieves 10.62 reward reduction, which is close to the one achieved by directly controlling the victim with OWR (10.65). Moreover, d-JSMA+OWR outperforms d-JSMA+OW, so we conclude that adding regularization to OW as described in III, does provide a benefit when combining the two attack steps.

In table III, we also show the evaluation for the perturbation

Authorized licensed use limited to: Mohammed VI Polytechnic University. Downloaded on January 07,2025 at 14:44:42 UTC from IEEE Xplore. Restrictions apply.

TABLE II: Team Win Rate of JSMA-based Methods.

	θ	LW	OW	OWR
JSMA	0.1	2.0%	83.0%	81.0%
	0.5	0.0%	1.0%	1.0%
	0.9	0.0%	0.0%	0.0%
d-JSMA	dynamic	0.00%	0.00%	0.00%

TABLE III: Average perturbation required (in terms of L1 norm) v.s. total reward reduction for JSMA and d-JSMA methods.

	Total Reward Reduction	Average L1 Norm Perturbation
d-JSMA+OWR	**10.62**	8.33
d-JSMA+OW	10.54	9.01
JSMA+LW ($\theta = 0.9$)	9.78	**6.34**
JSMA+OW ($\theta = 0.9$)	10.42	9.62
JSMA+OWR ($\theta = 0.9$)	10.43	9.80
it-FGSM+LW	8.36	20.37

budget. d-JSMA+OWR requires a lower budget than most other JSMA methods, while achieving the best total reward reduction. Figure 6 shows the perturbation distribution, where the d-JSMA+OWR method has the highest peak densities close to 0, showing that our regularization method effectively reduces the required perturbation. d-JSMA+OWR also has a lower density than d-JSMA+OW after 5 L1 norm, confirming regularization's effectiveness. Finally, d-JSMA-based methods generally have higher density in the lower perturbation range compared to JSMA methods, thus can achieve good performance with a relatively low budget.

C. Attack Strategy in StarCraft benchmark environment

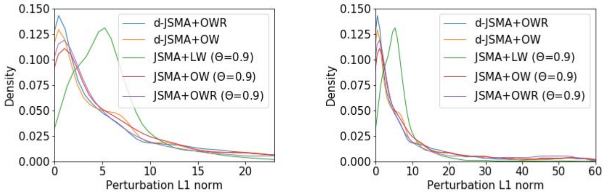
In this section, we examine the learned attack strategy and behaviors. Using LW or QMIX worst action selection policy (Figure 7a) results in the victim walking away from the opponent's range of sight—staying hidden, even after its team is defeated. The episode ends with the victim alive.

1) Agent behaviour under OW and OWR adversarial policies: At the beginning of an episode, the victim moves away from its team immediately, hiding until its team is almost defeated (as in 7a). Then, it moves towards the opponents to be defeated, as in 7b. Once within their sight range, most often it backs away, resulting in it being defeated. We have even observed the victim luring some of its team at times.

2) Two-step attack - d-JSMA+OWR: By only perturbing the victim's observation, our two-step attack can lure the victim to behave similarly as being directly controlled by the OWR policy. We can see the victim occasionally select sub-optimal actions, but the attack method is able to quickly recover.

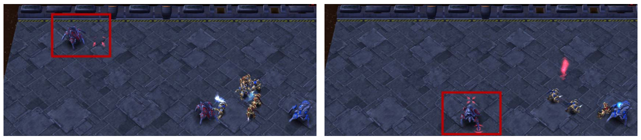
V. Discussion on Defense

To improve robustness of c-MARL systems, adversarial manipulations of both team agents and the environment need to be taken into account during training. Being able to isolate the impact of malicious agents is crucial in achieving robustness. One possible defense mechanism is to have each agent estimate action values or reward function for other agents, and use the estimate to potentially identify malicious agents. Techniques from Inverse Reinforcement Learning and model-based RL can be applied here. Another possible defense method involves formulating all agents as potential adversaries during the MARL centralized training, so that an agent can react better to adversarial actions during execution.



(a) Perturbation distribution from 0 to 20 (b) Perturbation distribution from 0 to 70

Fig. 6: Distribution of required perturbations for different attack methods.



(a) QMIX worst, local worst adversarial policy.

(b) OW, OWR adversarial policy and combined two-step attack (d-JSMA + OWR)

Fig. 7: Screenshots from replayed episodes. Victim agent shown in rectangle.

VI. Related Work

Prior work has demonstrated that RL is vulnerable to input perturbation attacks, where the attacker modifies the agent's observation with the goal of degrading its performance; for example, using the FGSM [13] to attack three RL networks [5]. Other attacks reduce the number of adversarial examples needed to decrease the agent's reward [17] or trigger misbehavior of the agent after a delay [18]. The learning process of DQN itself can be attacked [19], by constructing a replica model of the victim and transferring adversarial examples from the FGSM and JSMA [16] techniques to craft adversarial examples. None of this work studies c-MARL, and the effects of cooperation.

The closest work to ours is perhaps Gleave et al. [6]. They focus on attacks in the competitive multi-agent setting whereas we instead consider cooperative teams of agents. Furthermore, they assume full control over one of the agents (i.e., they were able to directly control the agent's actions), whereas we only perturb the agent's environment.

VII. Conclusions

We presented the first robustness analysis of cooperative teams of RL agents. Using the StarCraft II multi-agent environment as a benchmark, we proposed a two-step attack that combines 1) learning an adversarial policy and 2) gradient-based adversarial example generation. We demonstrated the possibility of controlling the total team reward by attacking a single agent. We hope our attack method can serve as a tool to assist evaluating and validating the robustness of c-MARL.

Authorized licensed use limited to: Mohammed VI Polytechnic University. Downloaded on January 07,2025 at 14:44:42 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] M. Wiering, “Multi-agent reinforcement learning for traffic light control,” in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML’2000)*, 2000, pp. 1151–1158.

[2] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “Safe, multi-agent, reinforcement learning for autonomous driving,” *arXiv preprint arXiv:1610.03295*, 2016.

[3] C. de Vrieze, S. Barratt, D. Tsai, and A. Sahai, “Cooperative multi-agent reinforcement learning for low-level wireless communication,” *arXiv preprint arXiv:1801.04541*, 2018.

[4] S. Q. Zhang, Q. Zhang, and J. Lin, “Efficient communication in multi-agent reinforcement learning via variance based control,” in *Advances in Neural Information Processing Systems*, 2019, pp. 3230–3239.

[5] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, “Adversarial attacks on neural network policies,” *arXiv preprint arXiv:1702.02284*, 2017.

[6] A. Gleave, M. Dennis, N. Kant, C. Wild, S. Levine, and S. Russell, “Adversarial policies: Attacking deep reinforcement learning,” *arXiv preprint arXiv:1905.10615*, 2019.

[7] M. Samvelyan, T. Rashid, C. Schroeder de Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, “The starcraft multi-agent challenge,” in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 2186–2188.

[8] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against machine learning,” in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. ACM, 2017, pp. 506–519.

[9] A. Athalye, N. Carlini, and D. Wagner, “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” *arXiv preprint arXiv:1802.00420*, 2018.

[10] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, “Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning,” *arXiv preprint arXiv:1803.11485*, 2018.

[11] F. A. Oliehoek, C. Amato *et al.*, *A concise introduction to decentralized POMDPs*. Springer, 2016, vol. 1.

[12] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, “Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models,” in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 15–26.

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.

[14] N. Carlini and D. Wagner, “Towards evaluating the robustness of neural networks,” in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[15] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial examples in the physical world,” *arXiv preprint arXiv:1607.02533*, 2016.

[16] 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.

[17] 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 preprint arXiv:1703.06748*, 2017.

[18] Y. Zhao, I. Shumailov, H. Cui, X. Gao, R. Mullins, and R. Anderson, “Blackbox attacks on reinforcement learning agents using approximated temporal information,” *arXiv preprint arXiv:1909.02918*, 2019.

[19] V. Behzadan and A. Munir, “Vulnerability of deep reinforcement learning to policy induction attacks,” in *International Conference on Machine Learning and Data Mining in Pattern Recognition*. Springer, 2017, pp. 262–275.

ACKNOWLEDGMENTS

This research was supported in part by CIFAR.

Authorized licensed use limited to: Mohammed VI Polytechnic University. Downloaded on January 07,2025 at 14:44:42 UTC from IEEE Xplore. Restrictions apply.