# A Robust Mean-Field Actor-Critic Reinforcement Learning Against Adversarial Perturbations on Agent States

Ziyuan Zhou, Guanjun Liu, *Senior Member, IEEE*, and Mengchu Zhou, *Fellow, IEEE*

***Abstract*— Multiagent deep reinforcement learning (DRL) makes optimal decisions dependent on system states observed by agents, but any uncertainty on the observations may mislead agents to take wrong actions. The mean-field actor-critic (MFAC) reinforcement learning is well-known in the multiagent field since it can effectively handle a scalability problem. However, it is sensitive to state perturbations that can significantly degrade the team rewards. This work proposes a Robust MFAC (RoMFAC) reinforcement learning that has two innovations: 1) a new objective function of training actors, composed of a *policy gradient function* that is related to the expected cumulative discount reward on sampled clean states and an *action loss function* that represents the difference between actions taken on clean and adversarial states and 2) a repetitive regularization of the action loss, ensuring the trained actors to obtain excellent performance. Furthermore, this work proposes a game model named a state-adversarial stochastic game (SASG). Despite the Nash equilibrium of SASG may not exist, adversarial perturbations to states in the RoMFAC are proven to be defensible based on SASG. Experimental results show that RoMFAC is robust against adversarial perturbations while maintaining its competitive performance in environments without perturbations.**

***Index Terms*— Mean-field actor-critic (MFAC) reinforcement learning, multiagent systems, robustness, state-adversarial stochastic game (SASG).**

## I. Introduction

DEEP learning has achieved significant success in such fields as computer vision [1], [2], [3], [4], [5], [6] and intelligent fault diagnosis [7], [8], [9]. However, deep neural networks are generally trained and tested by using independent and identically distributed data, and thus possibly make incorrect predictions when there are perturbations on samples even though these perturbations may be insignificant [10], [11]. Deep learning has been combined with reinforcement learning to train the control policy of an agent [12], [13], [14], [15]. Some studies have shown that agents trained by deep reinforcement learning (DRL) are also vulnerable to adversarial attacks, e.g., agents are likely to perform undesirable actions when their state observations are perturbed [16]. In fact, agents can receive perturbed state observations due to sensor errors or malicious attacks, which can cause serious issues in many applications such as autonomous driving and unmanned aerial vehicles. Therefore, robust DRL is important in the single-/multiagent field.

Many tasks require multiple agents to work together rather than acting independently in a cooperative or competitive environment. Multiagent DRL (MADRL) is proposed to maximize team rewards. Many practical and effective MADRL approaches have been proposed, such as policy-based methods including multiagent deep deterministic policy gradient (MADDPG) [17], multiactor-attention-critic reinforcement learning [18], game abstraction mechanism based on two-stage attention network [19] and value-based methods including value decomposition networks (VDNs) [20] and duplex dueling multiagent $Q$-learning [21]. But they usually face a big challenge: poor scalability, which significantly limits their applications in the real world. mean-field actor-critic (MFAC) [22] applies the mean-field theory to MADRL and thus successfully improves the scalability. However, our experiments illustrate that it is sensitive and not robust to state perturbations.

The works in [23] and [24] are very interesting since they explore the robustness of MADRL. The former considers the action perturbations rather than the state perturbations. Though the latter considers the state perturbations, it does not provide any defense method. This article is to explore the state perturbations of MAFC as well as the defense to them.

Compared with single-agent situations, multiagent situations face the following challenges: 1) the total number of perturbed agents is unknown and 2) perturbations on some agents can influence others. Facing these challenges, this work aims to propose a robust MFAC (RoMFAC) and makes the following novel contributions.

1) It proposes a novel objective function for training actors, which consists of a *policy gradient function* related to the expected cumulative discount reward on sampled clean states and an *action loss function* representing the difference between actions taken on clean and adversarial states. It designs a repetitive regularization method for the action loss, thereby ensuring that the trained actors obtain great performance not only in clean states but also in adversarial ones.

Manuscript received 13 November 2022; revised 28 April 2023; accepted 17 May 2023. Date of publication 5 June 2023; date of current version 8 October 2024. This work was supported in part by the Shanghai Science and Technology Committee under Grant 22511105500 and in part by the National Nature Science Foundation of China under Grant 62172299 and Grant 62032019. *(Corresponding author: Guanjun Liu.)*

Ziyuan Zhou and Guanjun Liu are with the Department of Computer Science, Tongji University, Shanghai 201804, China (e-mail: ziyuanzhou@tongji.edu.cn; liuguanjun@tongji.edu.cn).

Mengchu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2023.3278715, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2023.3278715

2) It defines a *state-adversarial stochastic game* (SASG) by extending the objective function of RoMFAC to a stochastic game (SG) and exploring the basic properties of SASG. This work proves that SASG does not necessarily have a Nash equilibrium under the joint optimal adversarial perturbation but it can still defend against perturbations.
3) We conduct experiments on two scenarios of the many-agent reinforcement learning system (MAgent) [25]. The results show that RoMFAC can improve the robustness under white-box attacks on states without degrading the performance of clean states.

The rest of this article is organized as follows. Section II presents related work of the adversarial attacks and defends on DRL. Section III recalls the SG and MFAC. We introduce our RoMFAC framework in Section IV and define an SASG in Section V. Then we illustrate the advantages of RoMFAC on robustness in comparison with some state-of-the-art methods in Section VI. Finally, we conclude this work and propose future work in Section VII.

## II. Related Work

### A. Adversarial Attacks on Single-Agent DRL

In classification tasks, the methods for generating and defending against adversarial examples have been extensively studied. Studies of adversarial attacks and defenses on single-agent DRL have recently emerged. The adversarial attacks on single-agent DRL algorithms can be divided into four categories [16] according to the following four factors: the state space with adversarial perturbations, the reward function with adversarial perturbations, the action space with adversarial perturbations and the model space with adversarial perturbations. Huang et al. [26] employ fast gradient sign method (FGSM) [10] to generate adversarial examples of input states, showing that adversarial attacks are also effective in the DRL policy network. To make the attack on an agent more stealthy and efficient, Sun et al. [27] introduce two adversarial attack techniques: the critical point attack and the antagonist attack.

### B. Robust Training for Single-Agent DRL

Defense methods against attacks in the single-agent field are usually classified into six categories [16]: adversarial training, defensive distillation, robust learning, game theoretic approach, benchmarking/watermarking, and adversarial detection. Zhang et al. [28] propose a novel Markov decision process (SA-MDP) that considers state-adversarial perturbations, and provides a theoretical foundation for robust single-agent reinforcement learning. They develop the principle of policy regularization that can possibly be applied to many DRL algorithms. Based on SA-MDP, an alternate training framework with learned adversaries is proposed in [29]. Oikarinen et al. [30] propose the robust ADversarIAl loss (RADIAL-RL) method, which can improve the robustness of DRL under the $\ell_p$ norm boundary against attacks with lower computational complexity. This article focuses on robust learning and expands the theoretical results and policy regularization in SA-MDP to multiagent DRL. We only consider the case that the states received by an agent have perturbations, but these perturbations do not change the Markov property of its environment. This case is the same as the previous research in [28], [29], and [30]. Furthermore, for the case of state perturbations, we define a new game (i.e., SASG) and discuss its basic properties based on the conclusions of SA-MDP.

### C. Attacks and Defenses for Multiagent DRL

Motivated by single-agent DRL [12], multiagent reinforcement learning has changed from the tabular method to deep learning recently. However, there are few studies on adversarial attacks and robust training in multiagent DRL [31]. We summarize them into adversarial attacks and backdoor ones. For the former, Lin et al. [24] propose the method of generating adversarial examples for MADRL, but do not provide a robust defense method. Li et al. [23] propose the minimax multiagent deep deterministic policy gradient (M3DDPG) which is an extension of MADDPG and makes policies of agents keep a strong generalization ability even if the opponent's policies change. They also present a robust optimization method that effectively solves the problem of the high complexity of min-max calculation in a continuous action space. However, they lack the defense against state perturbations which is exactly the purpose of our work.

For backdoor attacks, Chen et al. [32] introduce a pioneering backdoor attack framework, denoted as MARNet, that caters to the cooperative multiagent reinforcement learning (c-MARL) settings. The efficacy of this framework is assessed on two c-MARL algorithms VDN [20] and monotonic value function factorization (QMIX) [33]. Wang et al. [34] conduct research on the backdoor attack in DRL-based autonomous vehicle controllers. Their results illustrate a trigger based on traffic physics principles in order to prevent such attacks. Additionally, they explore a trigger of backdoor attacks in MARL systems and propose a technique named BACKDOORL to detect and prevent such attacks. In another relevant study, Guo et al. [35] propose a backdoor detection method in MARL systems by utilizing policy cleanse to detect and mitigate Trojan agents and their trigger actions. Furthermore, they design a machine unlearning-based approach to effectively mitigate the identified backdoors.

### D. Generalization of Multiagent DRL

There exists a certain correlation between robustness and generalization in machine learning because they are both related to the nonindependent identical distribution of the training samples. Some methods that improve generalization can also be used to enhance models' robustness, while approaches to addressing robustness can boost the generalization performance.

There have been many studies on the generalization of deep learning. Zhang et al. [36] design a novel solution for temporal activity detection in videos that can detect activities not seen in training by utilizing an activity graph transformer and label embeddings. Yan et al. [37] propose a differentiable generative adversarial network search method for zero-shot

learning, and their aim is to search for an optimal architecture for both generator and discriminator through adversarial training. Robust DRL methods can also be used to improve the generalization of algorithms [38]. Yan et al. [39] demonstrate a robust method for training a real robot to grasp a tiny sphere by decomposing the end-to-end system into a vision module and a closed-loop controller module. This method achieves an effective domain transfer with a 90% success rate in real-world scenarios. For the generalization of MADRL, Candela et al. [40] devise a simulation platform that allows autonomous driving and utilizes a multiagent proximal policy optimization [41] in conjunction with domain randomization for the transfer of policies from simulation to practical applications. In our earlier research [42], we establish a simulation platform for multi-UAV transport and utilize the domain randomization method to aid in the transfer from simulation to reality. Moreover, we formulate a nonstationary version of Markov games and demonstrate the effectiveness of recurrent neural networks in dealing with nonstationary Markov games. The exploration of generalization methods can potentially provide novel insights for studying the robustness of MADRL, to be explored as future research.

## III. Preliminary

### A. Stochastic Game

SG [43] is a game with multiple agents (or players) and states, defined as a tuple $\langle \mathcal{S}, \mathcal{A}^1, \ldots, \mathcal{A}^N, r^1, \ldots, r^N, p, \gamma \rangle$ where $\mathcal{S}$ is the state space, $N$ is the number of agents, $\mathcal{A}^j$ is the action space of agent $j$, $r^j$ is the immediate reward of agent $j$, $\gamma \in [0, 1]$ is the discount factor, and $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \times \mathcal{S} \to [0, 1]$ denotes the probability distributions of the next states under the current state and joint action. The reward function $R^j(s, \boldsymbol{a}, s') : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \times \mathcal{S} \to \mathbb{R}$ represents the reward obtained by agent $j$ in state $s'$ after taking the joint action $\boldsymbol{a} \triangleq [a^1, \ldots, a^N]$ in state $s$.

For an $n$-player SG, there is at least one Nash equilibrium [44], which can be described as the joint policy $\boldsymbol{\pi}_* \triangleq (\pi_*^1, \ldots, \pi_*^N)$ such that $\forall s \in \mathcal{S}$

$$\begin{aligned} V_{\boldsymbol{\pi}_*}^j(s) &\triangleq V^j\left(s, \pi_*^1, \ldots, \pi_*^j, \ldots, \pi_*^N\right) \\ &\geq V^j\left(s, \pi_*^1, \ldots, \pi^j, \ldots, \pi_*^N\right) \end{aligned} \tag{1}$$

where $\boldsymbol{\pi}_*(\cdot|s) = \prod_{j=1}^N \pi_*^j(\cdot|s)$ is the probability distribution of the joint action $\boldsymbol{a}$ at state $s$ under the Nash equilibrium and $\pi^j$ is an arbitrary valid policy of agent $j$. $V_{\boldsymbol{\pi}_*}^j(s)$ is the value function of agent $j$ under state $s$ and the Nash equilibrium at time $t$, and it is calculated through the expected cumulative discount reward of agent $j$

$$V_{\boldsymbol{\pi}_*}^j(s) = \mathbb{E}_{\boldsymbol{\pi}_*, p}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}^j | s_t = s\right) \tag{2}$$

where $r_t^j$ denotes the reward of agent $j$ at the time $t$. The action-value $Q_{\boldsymbol{\pi}_*}^j(s, \boldsymbol{a})$ is defined as the expected cumulative discount reward of agent $j$ at a state $s$ and a joint action $\boldsymbol{a}$ of all agents under the Nash equilibrium

$$\begin{aligned} Q_{\boldsymbol{\pi}_*}^j(s, \boldsymbol{a}) &= \mathbb{E}_{\boldsymbol{\pi}_*, p}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}^j | s_t = s, \boldsymbol{a}_t = \boldsymbol{a}\right) \\ &= \sum_{s' \in \mathcal{S}} p\left(s'|s, \boldsymbol{a}\right)\left(R^j\left(s, \boldsymbol{a}, s'\right) + \gamma V_{\boldsymbol{\pi}_*}^j\left(s'\right)\right). \end{aligned} \tag{3}$$

According to (2) and (3), the value function can also be formulated as follows:

$$V_{\boldsymbol{\pi}_*}^j(s) = \mathbb{E}_{\boldsymbol{a} \sim \boldsymbol{\pi}_*}\left(Q_{\boldsymbol{\pi}_*}^j(s, \boldsymbol{a})\right). \tag{4}$$

### B. MFAC Reinforcement Learning

MFAC [22] uses the mean-field theory to transform the interaction of multiple agents into the interaction between two agents, which makes large-scale MADRL become possible. In MFAC, $Q^j(s, \boldsymbol{a})$ is decomposed into

$$Q^j(s, \boldsymbol{a}) = \frac{1}{|\mathcal{N}(j)|} \sum_{k \in \mathcal{N}(j)} Q^j(s, a^j, a^k) \tag{5}$$

through local interactions, where $\mathcal{N}(j)$ is the set of neighbors of agent $j$ and $Q^j(s, a^j, a^k)$ is the Q function of agent $j$ when only the actions of agents $k$ and $j$ are considered. It has been proven in [22]

$$Q^j(s, \boldsymbol{a}) \approx Q^j\left(s, a^j, \bar{a}^j\right) \tag{6}$$

where the mean action $\bar{a}^j$ of all neighbors of agent $j$ can be represented as an empirical distribution of the actions taken by these neighbors and obtained by calculating the average of $a^k$, while $a^k$ is sampled according to policy $\pi^k$ which is calculated by a neural network via the previous average action $\bar{a}_-^k$ of neighbors of agent $j$ (note that in this article, we consider all agents except agent $j$ as its neighbors)

$$\bar{a}^j = \frac{1}{|\mathcal{N}(j)|} \sum_{k, a^k \sim \pi^k\left(\cdot|s, \bar{a}_-^k\right)} a^k. \tag{7}$$

This approximation is the core of mean-field theory and ensures that all neighbors' influence on agent $j$ can be approximated as the central agent's influence on agent $j$. Detailed proof can be found in [22]. Note that each $a^k$ is a one-hot coding. $a^k$ with $D$ possible operations can be represented as $a^k = [a_1^k, \ldots, a_D^k]$. For example, if an agent in the environment has three valid actions and agent $j$ has two neighbors performing actions $[0, 1, 0]$ and $[1, 0, 0]$, respectively. Obviously, $\bar{a}^j$ is $[0.5, 0.5, 0]$. Then the policy $\pi^j$ is changed according to the current $s$ and $\bar{a}^j$.

The mean field $Q$-function can be updated according to the following recursive form:

$$Q_{\phi^j}\left(s, a^j, \bar{a}^j\right) = (1 - \alpha) Q_{\phi^j}\left(s, a^j, \bar{a}^j\right) + \alpha\left(r^j + \gamma V^j\left(s'\right)\right) \tag{8}$$

where $s'$ is the state at the next time step, $\alpha$ is the learning rate and $\phi^j$ is the parameters of the critic of agent $j$. The mean-field value function is calculated by the following equation:

$$V^j\left(s'\right) = \sum_{a^j} \pi_{\theta^j}\left(a^j|s', \bar{a}^j\right) \mathbb{E}_{\boldsymbol{a}^{-j} \sim \boldsymbol{\pi}_{\theta^{-j}}}\left(Q_{\phi^j}\left(s', a^j, \bar{a}^j\right)\right) \tag{9}$$

where $\theta^j$ is the parameters of the actor of agent $j$, $\boldsymbol{a}^{-j}$ represents the joint action of all agents except agent $j$ and $\boldsymbol{\pi}_{\theta^{-j}}$ represents the joint policy of all agents except agent $j$.

MFAC is an on-policy actor-critic method where the critic is trained by minimizing the loss function

$$\mathcal{L}_1(\phi^j) = (y^j - Q_{\phi^j}(s, a^j, \bar{a}^j))^2 \tag{10}$$

and the actor $\pi_{\theta^j}$ is updated by sampling policy gradients

$$\nabla_{\theta^j}\mathcal{J}(\theta^j) \triangleq \nabla_{\theta^j}V_{\pi_{\theta^j}}(s) \approx \nabla_{\theta^j}(\log\pi_{\theta^j}(s))\,Q_{\phi_-^j}(s', a_-^j, \bar{a}_-^j)|_{a_-^j=\pi_{\theta_-^j}(s)} \tag{11}$$

where $y^j = r^j + \gamma V_{\phi_-^j}(s')$ is the target mean field value calculated with the weights $\phi_-^j$, and $\phi_-^j$ and $\theta_-^j$ are parameters of target networks of agent $j$. Konda and Tsitsiklis [45] provides a derivation of (11). During the training process, $\phi^j$ and $\theta^j$ are alternately updated until convergence is achieved. Since the observed states are perturbed, this article proposes a new objective function to train an actor so that it can defend against state perturbations.

## IV. RoMFAC Framework

In this section, we introduce a novel framework for improving the robustness of MFAC against state perturbations. Our framework RoMFAC mainly contains the following innovative components: an action loss function and its repetitive regularization. Important concepts are described as follows.

1) A *clean state* of an agent is a state generated by the environment interacting with the agent.
2) A *perturbation* means a noise within a certain range that is usually imperceptible to humans.
3) Adversarial attack: In this work, we consider the situation that for an agent, its states rather than its actions are perturbed. An *adversarial attack* is to add perturbations to clean states in order to mislead agents' actions. Certainly, there are some papers [23], [46], [47] considering the case of directly adding perturbations to the actions of agents.
4) An *adversarial state* is a state generated by the adversarial attack.

### A. Action Loss Function

We propose a novel strategy for updating our policy network in order to learn a robust policy. In the worst case, the states of every agent are all attacked and thus we should optimize the expected cumulative discount reward corresponding to adversarial states. According to (9), the learning objective of our RoMFAC is to maximize the expected cumulative discount reward in the worst case, i.e.,

$$\max_{\theta^j} V^j(s) = \max_{\theta^j}\min_{\hat{s}^j}\sum_{a^j}\pi_{\theta^j}(a^j|\hat{s}^j, \bar{a}^j)\mathbb{E}_{\boldsymbol{a}^{-j}\sim\boldsymbol{\pi}_{\theta^{-j}}}\left(Q_{\phi^j}(\hat{s}^j, a^j, \bar{a}^j)\right) \tag{12}$$

where $\hat{s}^j$ is the adversarial state of agent $j$ calculated by (15). The inner-loop goal is to minimize the expected cumulative discount reward of agent $j$, and the outer-loop goal is to maximize the worst case. An action that an agent performs in a given state is determined only by the actor network. It has nothing to do with the critic network during the testing process. As a result, this article only considers the robustness of actor networks. The minimized part of (12) can be solved by maximizing the loss between actions taken on clean and adversarial states, and this loss is called *action loss* in this article, i.e.,

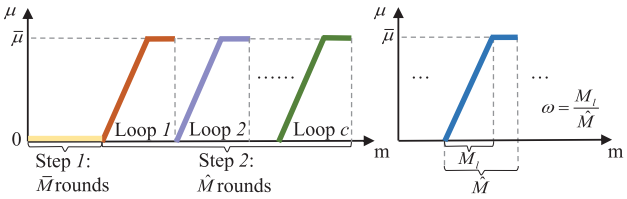

Fig. 1. Repetitive change of $\mu$. The different colors are used to distinguish different loops.

$$\mathcal{L}(\theta^j, \hat{s}^j) = \max_{\hat{s}^j\in\mathcal{B}^j}\left\{L(\theta^j, \hat{s}^j, z^j)|\hat{s}^j = s + \delta^j\right\} \tag{13}$$

where $s$ is the clean state, $\delta^j$ is the adversarial perturbation of agent $j$ and generated according to state $s$, and $\mathcal{B}^j$ is the set of adversarial states of agent $j$. We can label the action with the highest probability $z^j = \arg\max_{a^j}\pi_{\theta^j}(a^j|s, \bar{a}^j)$ because the actor of agent $j$ outputs the probability distribution of actions. $L$ is the cross-entropy loss function of actions taken on clean and adversarial states, i.e.,

$$L(\theta^j, \hat{s}^j, z^j) = -\sum_{a^j}[z^j = a^j]\log(\pi_{\theta^j}(a^j|\hat{s}^j, \bar{a}^j)) \tag{14}$$

where $[z^j = a^j]$ is the Iverson bracket whose value is 1 if the statement $z^j = a^j$ is true, and 0 otherwise. Note that, $a^j$ and $z^j$ are generated by the same policy.

The projected gradient descent (PGD) [11] is currently recognized as one of the methods that can generate stronger adversarial perturbations. Since we utilize a simply projected convex set in this article, the complexity of each PGD is low [48]. Therefore, we use PGD to generate the adversarial perturbation $\delta^j$ for the policy network of agent $j$ from the aspect of both the computing time and the defense performance. The PGD uses $U$ steps of the calculations of gradient ascent with clip to produce the adversarial states. In the $(u+1)$th step we have

$$\hat{s}_{u+1}^j = \text{clip}\left(\hat{s}_u^j + \beta\text{sgn}\left(\nabla_{\hat{s}_u^j}L(\theta^j, \hat{s}_u^j, z^j)\right)\right)|_{\hat{s}_0^j=s} \tag{15}$$

where $\beta$ is the step-size, $\hat{s}_u^j$ represents the adversarial state in the $u$th step and $u < U$. Notice, $\hat{s}_u^j$ is initialized by $s$. If the effective value range of $s$ is $[x, y]$, then

$$\text{clip}(s) = \begin{cases} s, & s \in [x, y] \\ x, & s < x \\ y, & s > y. \end{cases} \tag{16}$$

The outer-loop learning objective is to optimize the worst case, i.e., minimize the difference of actions taken on adversarial and clean states. The actor $\pi_{\theta^j}$ is trained with gradient descent by minimizing

$$-\mathcal{J}(\theta^j) + \mu\mathcal{L}(\theta^j) \tag{17}$$

where $\mu$ is a weight factor governing the trade-off between the two parts.

### B. Repetitive Regularization of the Action Loss

For the weight factor $\mu$ of the action loss, if it is too large, there may be a vanishing and exploding gradient so that the training is unstable. On the other hand, if it is too small, the action loss will not work. Therefore, regularizing

**Algorithm 1** RoMFAC

Initialize $Q_{\phi^j}$, $Q_{\phi_-^j}$, $\pi_{\theta^j}$, $\pi_{\theta_-^j}$ and $\bar{a}^j$, $\forall j \in \{1, \ldots, N\}$
**for** $m = 1, 2, \ldots, \bar{M} + c\hat{M}$ **do**
  For each agent $j$, sample action $a^j = \pi_{\theta^j}(s)$ and compute the new mean action $\bar{\boldsymbol{a}} = [\bar{a}^1, \ldots, \bar{a}^N]$ according to (7);
  Take the joint action $\boldsymbol{a} = [a^1, \ldots, a^N]$ and observe the next state $s'$ and the reward $\boldsymbol{R} = [R^1, \ldots, R^N]$;
  Put $\langle s, \boldsymbol{a}, s', \boldsymbol{R}, \bar{\boldsymbol{a}} \rangle$ in the replay buffer $\mathcal{D}$;
  **for** $j = 1$ to $N$ **do**
    Sample a batch of $K$ examples $\langle s_i, \boldsymbol{a}_i, \boldsymbol{R}_i, s'_i, \bar{\boldsymbol{a}}_i \rangle|_{i=1,\ldots,K}$ from $\mathcal{D}$;
    Set $y^j = R^j + \gamma V_{\phi_-^j}(s')$;
    Update the critic based on (10);
    Compute $\mu$ based on (18)
    **if** $\mu > 0$ **then**
      Generate the adversarial state by using PGD attack based on (15);
      Compute the action loss $\mathcal{L}^i(\theta^j, \hat{s}_i^j)$;
    **end if**
    Update the actor based on (17);
  **end for**
  Update the target network parameters for each agent $j$ using learning rates $\alpha_\theta$ and $\alpha_\phi$:

$$\theta_-^j \leftarrow \alpha_\theta \theta^j + (1 - \alpha_\theta)\theta_-^j$$
$$\phi_-^j \leftarrow \alpha_\phi \phi^j + (1 - \alpha_\phi)\phi_-^j$$

**end for**

the loss related to adversarial perturbations is often used in many robust single-agent reinforcement learning research [28], [29]. They usually use the grid search method to produce a fixed value for the weight factor $\mu$, and the perturbation bound $\epsilon$ gradually increases to a given value in the whole training process. RoMFAC is an on-policy learning method in which the optimized objective function depends on both clean states and adversarial ones and the calculation of adversarial states depends on clean ones. In addition, clean states are collected by agents according to the current policy. Therefore, if the prior approach is used to update the neural network parameters, some bad actions can be generated so that agents collect poor data and then affect the next update.

To solve this problem, we propose a repetitive regularization method for our action loss in order to dynamically balance the performance of agents on clean and adversarial states. If $\mu = 0$, only the clean state is considered during the training process. $\mu$ gradually increases, meaning that the weight of action loss gradually increases, and during the whole process, the coefficient of the clean state is always 1. After the agent gradually adapts to the adversarial state (i.e., $\mu$ increases linearly by one loop), it starts learning from the clean state again. By repeatedly adapting to the adversarial state, the agent can obtain good ability in the adversarial state while maintaining performance in the clean state.

*Step 1:* We train a network until it is stable with $\mu = 0$;
*Step 2:* We continually train it through $c$ loops. In every loop, $\mu$ increases linearly from 0 to a given upper bound $\overline{\mu}$.

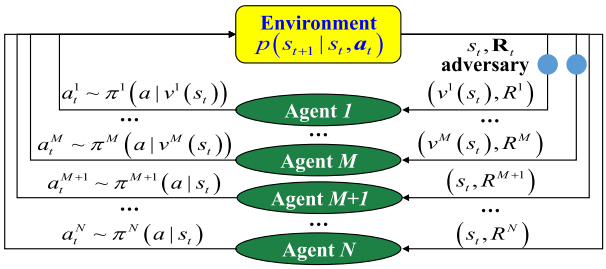


Fig. 2. Illustration of an SASG.

During the whole training process, the perturbation bound $\epsilon$ is a fixed value. In addition, $c$ and $\overline{\mu}$ are two hyper-parameters. Fig. 1 shows the idea of our repetitive regularization method. In fact, $\mu$ can be calculated by the following formula for the $m$th round:

$$\mu(m) = \frac{\min\{\max\{m - \bar{M}, 0\} \mod \hat{M}, \omega\hat{M}\}}{\omega\hat{M}}\overline{\mu} \quad (18)$$

where $\bar{M}$ is the number of training rounds in step 1 and $\hat{M}$ is the number of training rounds in every loop. This repetitive change can make agents explore more new positive behaviors while simultaneously increasing the robustness against adversarial states. Our RoMFAC is presented in Algorithm 1.

### C. Time Complexity of RoMFAC

We analyze the overall time complexity of Algorithm 1.

*Theorem 1:* If the time complexity of sampling action $a^j$ is $D_1$, computing the mean action $\bar{a}$ is $D_2$, taking the joint action $\boldsymbol{a}$ interacting with the environment is $D_3$, and summation is $D_4$, respectively, the executing time complexity of RoMFAC is

$$g_1 = O\big((\bar{M} + c\hat{M})(D_1 + D_2 + D_3)\big)$$

where $D_2 = O(N(N-1) * D_4)$.

If the time complexity of computing the gradient of (10) and (14) are $d_1$ and $d_2$, respectively, executing every step in PGD is $d_3$, and computing (11) is $d_4$, the training time complexity of MFAC is

$$g_2 = O\big(N(\bar{M} + c\hat{M})(d_1 + d_4)\big).$$

The time complexity of the repetitive regularization method is

$$g_3 = O\big(N(c\hat{M}(d_2 + Ud_3))\big).$$

The time complexity of RoMFAC is $O(g_1 + g_2 + g_3)$.

## V. State-Adversarial SG

In this section, we define a class of games: SASG to which the objective function in Section IV can be applied. SASG allows adversarial perturbations, and we prove that adversarial perturbations can be defended in theory.

*Definition 1 (SASG):* An SASG can be defined as a tuple $\langle \mathcal{S}, \mathcal{A}^1, \ldots, \mathcal{A}^N, \mathcal{B}^1, \ldots, \mathcal{B}^M, r^1, \ldots, r^N, p, \gamma \rangle$ where $\mathcal{B}^j$ is the set of adversarial states of agent $j$, and $M$ is the number of attacked agents such that $M \leq N$.

We define the adversarial perturbation $v^j(s)$ of agent $j$ as a deterministic function $v^j : \mathcal{S} \to \mathcal{B}^j$. Obviously, it is only dependent on the current state $s$ and remains constant over time. As shown in Fig. 2, $v^j(s)$ only perturbs the state of agent

$j$, while the state transition function of the environment is still based on clean states. Therefore, the environment discussed in this article is still based on the Markov property. The value and action-value functions of SASG are similar to those of SG

$$\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}}(s) = \mathbb{E}_{\boldsymbol{\pi}\circ\boldsymbol{v},p}\left(\sum_{k=0}^{\infty}\gamma^k r^j_{t+k+1}|s_t = s\right) \tag{19}$$

$$\widehat{Q}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}}(s,\boldsymbol{a}) = \mathbb{E}_{\boldsymbol{\pi}\circ\boldsymbol{v},p}\left(\sum_{k=0}^{\infty}\gamma^k r^j_{t+k+1}|s_t = s, \boldsymbol{a}_t = \boldsymbol{a}\right) \tag{20}$$

where $\boldsymbol{v} \triangleq (v^1, \ldots, v^M)$ denotes the joint adversarial perturbation and $\boldsymbol{\pi} \circ \boldsymbol{v}$ denotes the joint policy under the joint adversarial perturbation: $\boldsymbol{\pi} \circ \boldsymbol{v} \triangleq \boldsymbol{\pi}(\cdot|s, \boldsymbol{v}(s)) = \prod_{j=1}^{M}\pi^j(\cdot|v^j(s))\prod_{j'=M+1}^{N}\pi^{j'}(\cdot|s)$.

Given the joint policy $\boldsymbol{\pi} : \mathcal{S} \to \mathrm{PD}(\mathcal{A}^1 \times \cdots \times \mathcal{A}^N)$ and the joint adversarial perturbation $\boldsymbol{v} : \mathcal{S} \to \mathcal{B}^1 \times \cdots \times \mathcal{B}^M$, we have Bellman equations with fixed $\boldsymbol{\pi}$ and $\boldsymbol{v}$ as follows:

$$\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}}(s) = \sum_{\boldsymbol{a}\in\mathcal{A}^1\times\cdots\times\mathcal{A}^N}\boldsymbol{\pi}(\boldsymbol{a}|s, \boldsymbol{v}(s)) \times \sum_{s'\in\mathcal{S}}\left(p(s'|s,\boldsymbol{a})\left(R^j(s,\boldsymbol{a},s') + \gamma\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}}(s')\right)\right) \tag{21}$$

$$\widehat{Q}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}}(s) = \sum_{s'\in\mathcal{S}}p(s'|s,\boldsymbol{a})\left(R^j(s,\boldsymbol{a},s') + \gamma\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}}(s')\right) \tag{22}$$

where PD stands for the probability distribution. The joint policy $\boldsymbol{\pi}$ depends on both the clean state $s$ and the adversarial state $\boldsymbol{v}(s)$. On the other hand, $v^j(s)$ is generated based on the clean state and the actor network of attacked agents. The derivation of (21) and (22) can be found in Appendix A of our supplementary file. The goal of the joint optimal adversarial perturbation is to minimize the expected cumulative discount reward of every attacked agent, and thus the value function and the action-value function can, respectively, be written as follows:

$$\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}_*}(s) = \min_{v^j}\widehat{V}^j_{\boldsymbol{\pi}\circ\left(v^j,\boldsymbol{v}_*^{-j}\right)}(s) \tag{23}$$

$$\widehat{Q}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}_*}(s,\boldsymbol{a}) = \min_{v^j}\widehat{Q}^j_{\boldsymbol{\pi}\circ\left(v^j,\boldsymbol{v}_*^{-j}\right)}(s,\boldsymbol{a}) \tag{24}$$

where $\boldsymbol{v}_* \triangleq (v^1_*, \ldots, v^M_*)$ is the joint optimal adversarial perturbation, $v^j$ is an arbitrary valid adversarial perturbation and $\boldsymbol{v}_*^{-j} \triangleq (v^1_*, \ldots, v^{j-1}_*, v^{j+1}_*, \ldots, v^M_*)$.

In what follows we present some properties of SASG and their proofs can be seen in Appendixes B–E of our supplementary file.

*Theorem 2:* Every SASG has a joint optimal adversarial perturbation.

We can get the above conclusion through constructive proof, and Appendix B of our supplementary file shows the construction process and analysis. The following conclusion demonstrates the convergence of $\boldsymbol{v}_*$.

*Theorem 3 (Bellman Contraction of Agent $j$ for the Joint Optimal Adversarial Perturbation):* $\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}_*}$ lives in the finite dimensional Banach space $\mathbb{R}^{|\mathcal{S}|}$. For element $\widehat{V}^j \in \mathbb{R}^{|\mathcal{S}|}$, Bellman operator $\mathscr{L}^j$ with respect to the policy $\boldsymbol{\pi} \circ \boldsymbol{v}_*$ is defined as follows:

$$\left(\mathscr{L}^j\widehat{V}^j\right)(s) = \min_{v^j(s)\in\mathcal{B}^j}\sum_{\boldsymbol{a}\in\mathcal{A}^1\times\cdots\times\mathcal{A}^N}\left(\boldsymbol{\pi}\left(\boldsymbol{a}|s, v^j(s), \boldsymbol{v}_*^{-j}(s)\right)\right.$$

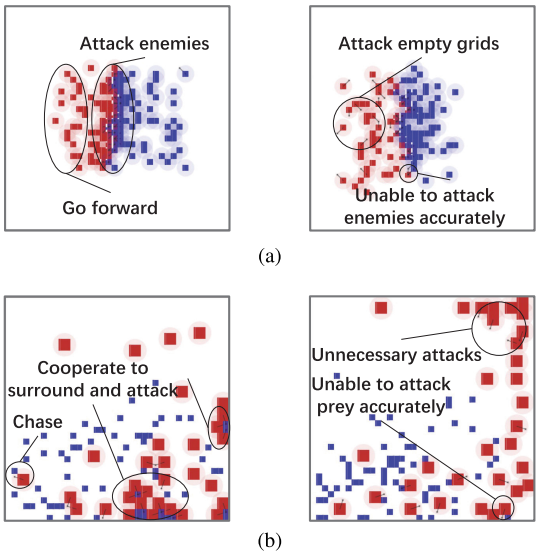


Fig. 3. Illustration of representative behaviors of MFAC agents in battle and pursuit scenarios. The left sides of (a) and (b) are behaviors of MFAC agents under clean states, and the right sides are behaviors under adversarial states.

$$\left.\sum_{s'\in\mathcal{S}}p(s'|s,\boldsymbol{a})\left(R^j(s,\boldsymbol{a},s') + \gamma\widehat{V}^j(s')\right)\right). \tag{25}$$

Then, the Bellman equation for the joint optimal adversarial perturbation $\boldsymbol{v}_*$ is $\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}_*} = \mathscr{L}^j\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}_*}$. Furthermore, $\mathscr{L}^j$ is a contraction that converges to $\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}_*}$.

Theorem 3 indicates that $\mathscr{L}^j$ converges to a unique fixed point, that is, the value function of the joint optimal adversarial perturbation is convergent. Similarly, the inner minimization of (12) is convergent. The next conclusion means that an SASG does not necessarily have the Nash equilibrium, which is the biggest difference from SG.

*Theorem 4:* Under the joint optimal adversarial perturbation, the Nash equilibrium of SASG may not always exist.

In our supplementary file, such a counterexample is constructed that has no Nash equilibrium. The joint policy $\boldsymbol{\pi}_*$ under the Nash equilibrium must have $\widehat{V}^j_{\boldsymbol{\pi}_*\circ\boldsymbol{v}_*}(s) \geq \widehat{V}^j_{(\pi^j,\boldsymbol{\pi}_*^{-j})\circ\boldsymbol{v}_*}(s)$ while the joint optimal adversarial perturbation $\boldsymbol{v}_*$ requires $\widehat{V}^j_{\boldsymbol{\pi}_*\circ\boldsymbol{v}_*}(s) \leq \widehat{V}^j_{\boldsymbol{\pi}_*\circ(v^j,\boldsymbol{v}_*^{-j})}(s)$. Therefore, for an SASG, we have to find a suboptimal joint policy to approach Nash equilibrium, and the following conclusion shows such a feasible defense against the joint optimal adversarial perturbation.

*Theorem 5:* Given the joint policy $\boldsymbol{\pi}$ and the joint optimal adversarial perturbation $\boldsymbol{v}_*$, we have

$$\max_{s\in\mathcal{S}}\left\{\widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}_*^{-j}}(s) - \widehat{V}^j_{\boldsymbol{\pi}\circ\boldsymbol{v}_*}(s)\right\} \leq \zeta\max_{s\in\mathcal{S}}\max_{\hat{s}^j\in\mathcal{B}^j}\mathrm{D_{TV}}\left(\boldsymbol{\pi}\left(\cdot|s,\hat{\boldsymbol{s}}^{-j}\right), \boldsymbol{\pi}\left(\cdot|s,\hat{s}^j,\hat{\boldsymbol{s}}^{-j}\right)\right) \tag{26}$$

where $\mathrm{D_{TV}}$ is the total variation, $\hat{\boldsymbol{s}}^{-j}$ is a group of adversarial states of all attacked agents except the agent $j$, i.e., $\hat{\boldsymbol{s}}^{-j} = \boldsymbol{v}_*^{-j}(s)$, $\hat{s}^j$ is an arbitrary valid adversarial state of agent $j$, and $\zeta \triangleq 2(1 + (\gamma/(1-\gamma)^2))\max_{s,\boldsymbol{a},s'}|R^j(s,\boldsymbol{a},s')|$ is a constant independent on $\boldsymbol{\pi}$.

As shown in (26), the joint policy between the clean state and perturbation one must be the same. Theorem 5 indicates that when there are adversarial states, the intervention of the

TABLE I
PERFORMANCE COMPARISONS AMONG THE PROPOSED METHOD AND BASELINES WITHOUT AN AGENT BEING ATTACKED. BOLD SCORES REPRESENT THE BEST PERFORMANCE

| SCENARIO | BATTLE | | | PURSUIT |
|---|---|---|---|---|
| METHODS | WINNING RATE | AVERAGE KILL | AVERAGE REWARD | AVERAGE REWARD |
| MFAC | 0.70±0.08 | 62.22±2.83 | 291.26±18.22 | 3674.04±498.83 |
| FGSM-M | 0.78±0.03 | 62.76±2.62 | 308.92±17.32 | 2240.49±522.99 |
| PGD-M | 0.76±0.06 | 62.55±2.64 | 306.16±17.15 | 2149.70±418.33 |
| RADIAL-RL | 0.52±0.04 | 60.13±4.53 | 281.91±23.12 | **5285.18±417.57** |
| RADIAL-M | 0.01±0.01 | 43.68±8.50 | 211.13±44.34 | 4954.42±412.89 |
| SA-MFAC | 0.36±0.13 | 56.74±5.78 | 274.84±32.07 | 1630.78±510.71 |
| ROMFAC | **0.90±0.11** | **62.83±2.77** | **312.88±19.73** | 3844.66±462.89 |

value function is small as long as the difference between action distributions is small. It mainly illustrates that when an adversary perturbs the state of an agent, the upper bound of its impact on the performance of the agent (i.e., the value function) depends on the difference between the output action distributions of the same policy function in different states. That is, the focus of defense is to minimize the difference between the distribution of actions output by the agent in the clean state and the adversarial state. This idea also supports our proposed RoMFAC in which the defense against state perturbations can thus be transferred into minimizing the cross-entropy loss between the output actions in the clean state and the adversarial state.

## VI. EXPERIMENTAL RESULTS AND ANALYSIS

We demonstrate the ability of RoMFAC to improve model robustness against adversarial perturbations.

### A. Environmental Settings

We use two scenarios of MAgent [25], the battle and pursuit scenarios, to illustrate the advantages of our method compared with other state-of-the-art ones. They can support hundreds of agents.

1) *Battle:* This is a cooperative and competitive scenario in which two groups of agents, A and B, interact. Each group of agents works together as a team to eliminate all opponent agents. There are 128 agents in total and 64 ones in each group. We use the default reward settings: 5 for killing an opponent agent, 0.2 for attacking an opponent agent, $-0.1$ for being attacked or killed, $-0.1$ for attacking an empty grid, and 0.005 per step.
2) *Pursuit:* This is a scenario of local cooperation. There are 32 predators and 64 prey. Similarly, we use the default reward settings: the predator receives $+1$ for attacking the prey, while the prey receives $-0.1$ for being attacked.

The state and action spaces of the two scenarios are described as follows.

1) *State Spaces:* The state space consists of two parts: high dimensional information of spacial local view and low dimensional nonspatial features. The spatial view consists of a group of rectangular channels, and nonspatial features include normalized position, last action, last reward, and ID embedding. They, respectively, correspond to $13 \times 13 \times 7$ and 34 continuous values in battle and to $10 \times 10 \times 5$ and 14 in pursuit.
2) *Action Spaces:* In battle and pursuit scenarios, there are 21 and 13 discrete actions, respectively.

### B. Experiment Settings

*1) Benchmark Methods:*

1) MFAC [22] does not apply any robust strategies.
2) FGSM-M is an adversarial training method that uses adversarial states generated by FGSM [10] to train a model.
3) PGD-M is another adversarial training method that uses adversarial states generated by PGD [11] to train a model.
4) RADIAL-RL [30] is a method using the interval bound propagation (IBP) [49] to construct the adversarial loss. The IBP technique calculates the upper and lower bounds for the effect of a certain range of perturbations on model performance without the need to know the perturbations.
5) RADIAL-M is a variant of MFAC by using IBP to construct the adversarial loss such as RADIAL-RL.
6) SA-MFAC, such as most robust training [28], uses a fixed weight factor $\mu$ and increased perturbation bound $\epsilon$ in the original training technique.

*2) Training:* In the battle scenario, we train seven models in self-play. In the pursuit scenario, both predators and prey use the same algorithm during the training process. We execute three loops for $\mu$ (i.e., $c = 3$) and $\epsilon = 0.075$. The other hyper-parameter settings of these algorithms are the same: during the whole training process, the temperature of the soft-max layer in actor is 0.1, the learning rate is $\alpha = 10^{-4}$, the weights of entropy and value in the total loss is 0.08 and 0.1, respectively.

*3) Testing:* For the convenience of comparison, we use advantageous actor-critic which is fully decentralized as the policies of opponent agents and prey but do not perturb their states. To evaluate the robustness of models trained by these algorithms, we create state-adversarial perturbations using a ten-step PGD with $\ell_\infty$ norm perturbation budget $\epsilon = 0.075$; and 1000 rounds cross-comparative and 100 rounds experiments with maximum time steps of 400 in the battle

TABLE II

PERFORMANCE COMPARISONS AMONG THE PROPOSED METHOD AND BASELINES IN THE BATTLE SCENARIO. BOLD SCORES REPRESENT THE BEST PERFORMANCE

| ATTACKED METHODS | | PGD | | | FGSM | | |
|---|---|---|---|---|---|---|---|
| METHODS | NOS | WINNING RATE | AVG KILL | AVG REWARD | WINNING RATE | AVG KILL | AVG REWARD |
| MFAC | 4 | 0.59±0.09 | 61.13±3.71 | 286.31±21.26 | 0.56±0.08 | 60.90±3.88 | 285.91±22.40 |
| FGSM-M | | 0.63±0.04 | 61.73±3.35 | 301.73±19.30 | 0.67±0.05 | 61.97±3.24 | 302.61±20.77 |
| PGD-M | | 0.67±0.02 | 61.85±3.31 | 302.21±19.35 | 0.64±0.04 | 61.60±3.48 | 300.72±20.20 |
| RADIAL-RL | | 0.41±0.11 | 58.36±5.62 | 271.66±27.16 | 0.36±0.13 | 58.43±5.24 | 272.14±26.53 |
| RADIAL-M | | 0.01±0.00 | 42.04±8.28 | 201.57±42.41 | 0.01±0.00 | 42.42±7.87 | 203.09±41.34 |
| SA-MFAC | | 0.35±0.13 | 56.51±5.64 | 276.61±30.62 | 0.39±0.11 | 56.80±5.68 | 278.12±31.00 |
| ROMFAC | | **0.89±0.10** | **62.63±3.22** | **310.62±21.33** | **0.88±0.12** | **62.44±3.41** | **309.60±22.41** |
| MFAC | 8 | 0.50±0.10 | 59.84±4.88 | 280.51±27.70 | 0.42±0.08 | 59.11±4.91 | 275.46±26.88 |
| FGSM-M | | 0.45±0.02 | 59.96±4.37 | 291.76±22.86 | 0.53±0.01 | 60.72±3.97 | 296.38±21.56 |
| PGD-M | | 0.56±0.04 | 60.74±3.97 | 294.76±20.88 | 0.57±0.00 | 60.82±4.05 | 296.34±21.47 |
| RADIAL-RL | | 0.27±0.19 | 55.73±6.80 | 256.25±33.47 | 0.23±0.19 | 55.21±6.64 | 254.06±32.40 |
| RADIAL-M | | 0.01±0.01 | 41.38±8.00 | 196.34±41.48 | 0.00±0.00 | 41.10±7.34 | 195.46±39.12 |
| SA-MFAC | | 0.37±0.15 | 56.35±5.90 | 272.40±32.31 | 0.37±0.16 | 55.97±6.02 | 273.52±32.84 |
| ROMFAC | | **0.85±0.10** | **62.27±3.83** | **308.98±23.54** | **0.85±0.12** | **62.21±3.70** | **308.12±23.12** |
| MFAC | 16 | 0.28±0.06 | 55.22±7.39 | 255.91±38.77 | 0.24±0.08 | 54.84±7.28 | 255.51±38.28 |
| FGSM-M | | 0.19±0.03 | 55.55±5.58 | 268.77±27.30 | 0.26±0.01 | 56.70±5.69 | 275.34±28.85 |
| PGD-M | | 0.22±0.02 | 55.86±5.40 | 269.25±25.39 | 0.28±0.03 | 56.69±5.59 | 275.42±27.59 |
| RADIAL-RL | | 0.17±0.19 | 50.55±8.72 | 228.14±43.95 | 0.18±0.19 | 50.15±8.67 | 225.88±43.14 |
| RADIAL-M | | 0.00±0.00 | 38.07±7.09 | 178.22±38.32 | 0.00±0.00 | 38.75±7.59 | 181.89±38.95 |
| SA-MFAC | | 0.35±0.11 | 56.56±5.72 | 273.62±30.74 | 0.33±0.08 | 56.13±5.75 | 274.53±31.33 |
| ROMFAC | | **0.87±0.15** | **62.37±3.43** | **308.51±22.45** | **0.87±0.12** | **62.35±3.47** | **308.37±22.41** |
| MFAC | 32 | 0.05±0.03 | 46.34±9.54 | 207.80±49.43 | 0.03±0.03 | 45.74±9.37 | 207.26±48.71 |
| FGSM-M | | 0.03±0.01 | 45.12±9.36 | 214.50±46.52 | 0.06±0.01 | 46.41±10.05 | 221.42±48.89 |
| PGD-M | | 0.07±0.01 | 45.09±10.33 | 214.24±50.53 | 0.13±0.02 | 47.17±10.35 | 224.90±49.21 |
| RADIAL-RL | | 0.02±0.04 | 40.43±9.99 | 173.35±52.67 | 0.01±0.02 | 39.92±9.45 | 172.50±47.73 |
| RADIAL-M | | 0.00±0.00 | 32.57±6.81 | 146.00±37.24 | 0.00±0.00 | 33.00±6.73 | 148.92±36.61 |
| SA-MFAC | | 0.27±0.12 | 55.75±5.57 | 270.07±29.69 | 0.24±0.13 | 55.08±5.86 | 269.31±31.70 |
| ROMFAC | | **0.84±0.11** | **62.22±3.55** | **306.46±22.40** | **0.85±0.13** | **62.25±3.62** | **307.02±22.82** |
| MFAC | 64 | 0.00±0.00 | 28.82±4.49 | 112.86±22.71 | 0.00±0.00 | 29.15±4.43 | 117.82±22.08 |
| FGSM-M | | 0.00±0.00 | 32.08±13.35 | 145.44±68.32 | 0.00±0.00 | 34.06±14.75 | 156.10±75.76 |
| PGD-M | | 0.00±0.00 | 32.74±15.20 | 148.66±78.38 | 0.00±0.00 | 33.74±15.62 | 154.37±80.84 |
| RADIAL-RL | | 0.00±0.00 | 23.75±5.29 | 84.32±27.45 | 0.00±0.00 | 23.92±4.84 | 88.50±24.02 |
| RADIAL-M | | 0.00±0.00 | 23.26±6.52 | 91.15±36.19 | 0.00±0.00 | 24.79±6.19 | 100.77±34.06 |
| SA-MFAC | | 0.24±0.14 | 54.98±5.78 | 265.23±31.04 | 0.25±0.14 | 54.86±5.84 | 267.21±31.63 |
| ROMFAC | | **0.83±0.12** | **61.74±4.35** | **302.30±27.32** | **0.84±0.16** | **61.97±4.16** | **304.56±25.27** |

TABLE III

PERFORMANCE COMPARISONS AMONG THE PROPOSED METHOD AND BASELINES IN THE PURSUIT SCENARIO. BOLD SCORES REPRESENT THE BEST PERFORMANCE. THERE ARE 32 PREDATORS IN THE PURSUIT SCENARIO SO THAT THE MAXIMUM NUMBER OF ATTACKED AGENTS IS 32

| ATTACK | METHODS | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|---|
| PGD | MFAC | 3618.11±434.19 | 3258.74±283.89 | 3012.28±377.84 | 2356.47±369.75 | 1088.57±373.91 |
| | FGSM-M | 2163.72±432.99 | 1995.18±387.70 | 1887.59±354.32 | 1627.69±476.25 | 929.65±413.18 |
| | PGD-M | 2059.20±383.81 | 2061.70±366.77 | 1871.49±391.06 | 1551.80±349.21 | 1135.15±359.40 |
| | RADIAL-RL | **5024.44±491.72** | **4962.54±401.90** | **4544.58±440.86** | **3949.76±523.12** | 2623.35±417.49 |
| | RADIAL-M | 4792.63±402.29 | 4717.31±413.55 | 4399.78±377.47 | 3878.02±435.38 | 2719.74±367.63 |
| | SA-MFAC | 1524.98±353.21 | 1489.56±576.48 | 1427.56±452.71 | 1484.44±496.08 | 1302.40±432.42 |
| | ROMFAC | 3857.26±498.60 | 3866.27±392.19 | 3815.51±408.88 | 3724.85±394.78 | **3714.23±485.07** |
| FGSM | MFAC | 3598.22±418.61 | 3383.29±410.60 | 2962.89±404.87 | 2409.75±462.30 | 1068.65±389.01 |
| | FGSM-M | 2155.51±403.68 | 2007.99±402.25 | 1989.81±383.69 | 1507.23±393.77 | 996.78±318.51 |
| | PGD-M | 2136.05±411.39 | 2065.67±422.87 | 2042.14±359.66 | 1658.83±376.30 | 1253.52±409.21 |
| | RADIAL-RL | **5071.39±579.16** | **4934.67±513.39** | **4658.96±499.40** | **3899.16±464.89** | 2554.81±483.15 |
| | RADIAL-M | 4795.82±420.06 | 4658.44±439.86 | 4438.49±386.81 | 3834.28±413.98 | 2697.24±357.38 |
| | SA-MFAC | 1593.44±559.52 | 1369.20±645.09 | 1508.01±542.68 | 1375.30±510.70 | 1252.98±435.63 |
| | ROMFAC | 3828.48±481.32 | 3872.49±470.55 | 3751.54±460.42 | 3808.64±398.25 | **3895.09±537.61** |

and pursuit scenario are executed, respectively. The maximum number of attacked agents in the battle and pursuit scenario is 64 and 32, respectively.

### C. Results and Discussions

*1) Experimental Results:* As demonstrated in Fig. 3, MFAC agents cannot collaborate normally when states are perturbed.

TABLE IV

COMPARISONS OF ROMFAC PERFORMANCE FOR VARIOUS $\epsilon$. THE NUMBER OF AGENTS ATTACKED IS REPRESENTED BY THE TABLE'S COLUMNS, WHILE THE VALUE OF $\epsilon$ IS REPRESENTED BY ITS ROWS. THE WIN RATE IS UTILIZED IN THE BATTLE SCENARIO, WHEREAS THE AVERAGE REWARD IS USED IN THE PURSUIT SCENARIO

| BATTLE | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|
| 0.025 | 0.94±0.07 | 0.91±0.01 | 0.89±0.09 | 0.88±0.12 | 0.86±0.12 |
| 0.125 | 0.86±0.14 | 0.83±0.11 | 0.83±0.13 | 0.77±0.12 | 0.63±0.16 |
| PURSUIT | 2 | 4 | 8 | 16 | 32 |
| 0.025 | 3981.78±439.32 | 3845.51±455.70 | 3827.86±379.50 | 3820.90±447.26 | 3776.92±459.34 |
| 0.125 | 3890.63±461.26 | 3907.06±512.27 | 3872.49±442.04 | 3718.72±458.73 | 3760.80±419.36 |

In the battle scenario, the previously learned policies of collaboration that a group of agents collaboratively go forward and attack another group are destroyed. They begin attacking empty grids but are unable to accurately attack opponent agents. In the pursuit scenario, the initially learned coordinated siege policy is destroyed, their movements are scattered, and they are unable to attack prey accurately.

The experimental results are shown in Tables I–III. Table I illustrates that FGSM-M, PGD-M, and RoMFAC improve the performance on clean states while RADIAL-RL, RADIAL-M, and SA-MFAC decrease in the battle scenario. In the pursuit scenario, RADIAL-RL, RADIAL-M, and RoMFAC enhance the performance in clean states while FGSM-M, PGD-M, and SA-MFAC exhibit a decline. RoMFAC indicates outstanding performance in both scenarios. As for the robustness of these methods, as shown in Table II, when robust training is not carried out, the cooperative policies are destroyed more seriously with more attacked agents. The performance of RoMFAC slightly decreases as the number of attacked agents grows, but it is still the best. As shown in Table III, it is seen that RADIAL-RL and RADIAL-M have higher rewards when the number of attacked agents is small. However, as the number of attacked agents increases, their rewards gradually decrease. In contrast, the reward of RoMFAC shows little fluctuation, demonstrating the effectiveness of our proposed method in enhancing model robustness. RADIAL-RL has the best performance in single-agent DRL, but the results in multiagent DRL are poor. We analyze the reason as follows: The input data of MAgent, i.e., the experimental environment used in this experiment, consists of two parts. When using RADIAL-RL in MAgent, the overlaps between output bounds of actions of the two parts of inputs must be calculated individually and then merged. At this time, some errors occur in the merging process and thus the results are very poor. Besides, RADIAL-M must consider the bound of the previous average action, and then merge this bound with the above two parts. This can make the merging error larger. Thus it is worse than RADIAL-RL.

The scalability of RoMFAC is the same as that MFAC. In the battle and pursuit scenarios, the number of agents is 64 and 32, respectively, which is the same as in the MFAC setting. Theoretically, the repetitive regularization method does not modify the mean-field theory and therefore does not affect the scalability. RoMFAC has lower sample efficiency than MFAC because RoMFAC considers the existence of adversarial states in the samples and thus requires more samples. MFAC only considers clean states and achieves good performance with 2000 training episodes on clean states. RoMFAC needs an additional 9000 training episodes to achieve similar performance on adversarial states. In deep learning, adversarial training often degrades the performance of a model on clean samples, while in reinforcement learning, adversarial training does generally not degrade the performance of an agent in the clean states, as shown in our experiments as well as the ones in [28] and [29]. Analyzing the reasons, we think that the emergence of perturbed states generally prompts the agent to explore the environment and thus to learn better behaviors.

Overall, we can conclude that as follows.

1) Constructing bounds on the effect of perturbations on model performance over a certain range using IBP may result in a loss of performance on the clean state because it needs to balance clean loss with adversarial one. However, the presence of the merging operation makes the boundary calculation less accurate, resulting in no performance improvement in the adversarial state.
2) SA-MFAC only considers the trade-off between clean loss and adversarial one. When the agent gradually adapts to the adversarial state, it has certain robustness, but it tends to reduce the performance in the clean state, which eventually makes the performance in the clean state and the adversarial state as poor as each other. Our repetitive regularization method well solves this problem.

In a word, the model trained by our RoMFAC has the highest robustness and does not degrade the performance of the model in a clean state.

*2) Generalization Analysis:* We analyse the generalization of RoMFAC from the following two aspects.

1) *Different Types of Perturbations:* We test the noise generated by FGSM [10] to demonstrate our algorithm's generalization on $\ell_\infty$ norm perturbations, and the experimental results are presented in Tables II and III. Overall, we can see that FGSM has a weaker attack effect than PGD. In the battle scenario, the networks trained by the baseline techniques can withstand FGSM attacks, but their effects are not as effective as our RoMFAC. The model trained by RADIAL-RL is also vulnerable to FGSM attacks in the pursuit scenario, while the model trained by our technique remains the strongest defense against FGSM attacks.
2) *Different $\epsilon$:* During the training process, we set $\epsilon = 0.075$. The experimental results are provided in Table IV for the situations of $\epsilon = 0.025$ and $\epsilon = 0.125$. We can

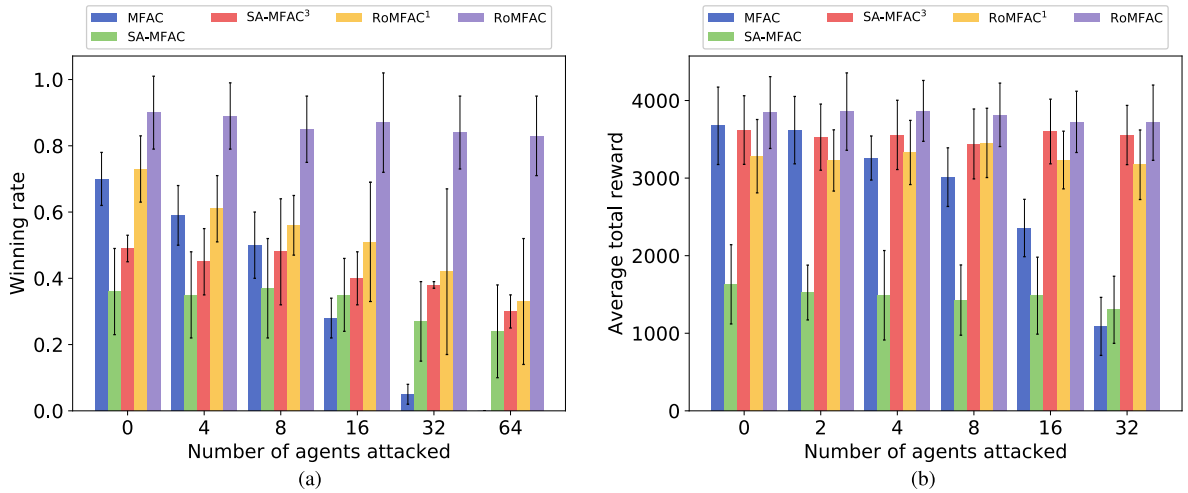


Fig. 4. Performance comparisons of the ablation study in two scenarios. (a) Battle. (b) Pursuit.

TABLE V

SOME HYPER-PARAMETERS AND NUMBERS OF TRAINING ROUNDS OF ROMFAC[1] AND ROMFAC. IN EVERY LOOP, $\mu$ INCREASES LINEARLY FROM 0 TO A GIVEN UPPER BOUND $\overline{\mu}$ AND THE VALUE OF $\epsilon$ REMAINS AT 0.075

| SCENARIO | METHODS | $\overline{\mu}$ | $\bar{M}$ | $\hat{M}$ | $\omega$ |
|---|---|---|---|---|---|
| BATTLE | ROMFAC[1] | 0.7 | 2000 | 3000 | 0.6 |
| | ROMFAC | 0.7 | 2000 | 3000 | 0.6 |
| PURSUIT | ROMFAC[1] | 0.7 | 2000 | 9000 | 0.2 |
| | ROMFAC | 0.7 | 2000 | 3000 | 0.6 |

TABLE VI

SOME HYPER-PARAMETERS AND NUMBERS OF TRAINING ROUNDS OF SA-MFAC AND SA-MFAC[3]. IN EVERY LOOP, $\epsilon$ INCREASES LINEARLY FROM 0 TO A GIVEN BOUND $\overline{\epsilon}$ AND THE VALUE OF $\mu$ REMAINS AT 0.7

| SCENARIO | METHODS | $\overline{\epsilon}$ | $\bar{M}$ | $\hat{M}$ | $\omega$ |
|---|---|---|---|---|---|
| BATTLE | SA-MFAC | 0.075 | 2000 | 3000 | 0.6 |
| | SA-MFAC[3] | 0.075 | 2000 | 3000 | 0.6 |
| PURSUIT | SA-MFAC | 0.075 | 2000 | 9000 | 0.2 |
| | SA-MFAC[3] | 0.075 | 2000 | 3000 | 0.6 |

observe that the model trained by RoMFAC has a good effect on $\epsilon$ less than 0.075. Even when the number of attacked agents is low, the performance of our method is better than the case that agents are not attacked. It has a protective impact for $\epsilon$ greater than 0.075, but the overall performance is less than that for $\epsilon = 0.075$.

### D. Ablation Study

1) *RoMFAC[1]:* We consider one variant of RoMFAC for ablation studies, namely the $\mu$ of RoMFAC just uses linear increase one time in the training process in order to show the effectiveness of our repetitive regularization.
2) *SA-MFAC[3]:* We apply our repetitive regularization technique to SA-MFAC, that is, $\mu$ remains constant, but $\epsilon$ changes repetitively.

To ensure that the slopes of $\mu$ in RoMFAC and RoMFAC[1] are identical, some related hyper-parameters and training rounds are shown in Table V. Similarly, Table VI shows the setting of hyper-parameters and training rounds in order to ensure that the slopes of $\epsilon$ in SA-MFAC and SA-MFAC[3] are the same.

The results are shown in Fig. 4. Comparing RoMFAC[1] and RoMFAC, we can see the significance of our repetitive regularization. The average total rewards obtained by SA-MFAC[3] are slightly better than SA-MFAC in the battle scenario, but they are obviously good in the pursuit scenario. Therefore, applying our repetitive regularization to SA-MFAC can also lead to a good result.

### E. Execution Time Analysis

RoMFAC uses the PGD method to generate adversarial samples. Due to the simple convex set, the computational complexity of each PGD iteration is low [48]. To investigate the effect of RoMFAC on execution time, we measure the training and testing time on a PC with an Intel[1] Core[2] i7-8700 CPU @ 3.20 GHz and 32.00G RAM. It is worth noting that RL is different from deep learning in which an agent needs to collect data by interacting with the environment. If we use GPU for experiments, data exchange between GPU and CPU increases the computation time. Therefore, we conduct experiments on CPU rather than GPU.

The training time is shown in Table VII. In the battle scenario, MFAC takes 2 h and 37 min to train 11 000 rounds, while RoMFAC takes 3 h and 29 min. The extra time spent by RoMFAC is the effect of repetitive regularization on the training process. Our method uses a ten-step PGD in computing the action loss function. Thus ten gradient ascents are required, while FGSM only needs to compute the gradient once. Therefore, adversarial training using FGSM takes less time than RoMFAC. The RADIAL method does not need to compute the adversarial loss by utilizing a gradient, and its

[1]Registered trademark.
[2]Trademarked.

TABLE VII
TRAINING TIME OF 11 000 ROUNDS

| Methods | BATTLE | PURSUIT |
|---|---|---|
| MFAC | 2 hours 37 minutes | 18 hours 30 minutes |
| FGSM-M | 3 hours 22 minutes | 25 hours 34 minutes |
| PGD-M | 7 hours 28 minutes | 44 hours 10 minutes |
| RADIAL-AC | 4 hours 22 minutes | 19 hours 3 minutes |
| RADIAL-M | 16 hours 44 minutes | 21 hours 30 minutes |
| SA-MFAC | 4 hours 47 minutes | 30 hours 37 minutes |
| RoMFAC | 3 hours 29 minutes | 30 hours 24 minutes |

TABLE VIII
ELAPSED TIME OF USING TEN-STEP PGD TO EXECUTE 1000 TIME STEPS

| NUMBERS OF ATTACKED AGENTS | 0 | 4 | 8 | 16 | 32 | 64 |
|---|---|---|---|---|---|---|
| ELAPSED TIME (S) | 4 | 16 | 18 | 20 | 23 | 32 |

running time is also less. In the battle scenario, because of the policy enhancement, it no longer takes 400-time steps for agents to end a round so the execution time does not fully reflect the time complexity. In the pursuit scenario, executing 11 000 rounds with 400-time steps per round gives a good reflection of the training time of each algorithm. Our approach is more efficient compared to adversarial training with PGD.

In addition, we evaluate the time when executing 1000 time steps under a ten-step PGD attack on different numbers of agents. The results are listed in Table VIII. Obviously, the time spent on repetitive regularization of RoMFAC and attack by PGD is acceptable compared to the training time of MFAC and the testing time without attack.

## VII. CONCLUSION

In this article, we present a robust training framework for the state-of-the-art reinforcement learning method MFAC. In our framework, the action loss function and its repetitive regularization play an important role in improving the robustness of the trained model. Moreover, we present SASG to establish a theoretical foundation in multiagent reinforcement learning with adversarial attacks and defenses. Our work is inspired by SA-MDP [28] which is a robust single-agent reinforcement learning against adversarial perturbations, while we extend their work to multiagent systems. To the best of our knowledge, there is little related work on adversary attacks and defenses of multiagent reinforcement learning [23], [24], [46], [47]. Lin et al. [24] first propose the method of adversarial states generation in MADRL which reduces the team reward by perturbing the state of only one agent, and does not provide any defense method. This article presents a robust defense technique that can deal with a number of attacked agents rather than only one agent. Our next work intends to extend our method to the other MADRL for continuous action spaces such as MADDPG [17] and take into account new types of perturbations such as Wasserstein adversarial examples [50]. Besides, we plan to consider how to construct the action loss without the adversarial states such as RADIAL-RL [30].

## REFERENCES


[1] M. Zhao, S. Zhong, X. Fu, B. Tang, and M. Pecht, "Deep residual shrinkage networks for fault diagnosis," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4681–4690, Jul. 2020.

[2] I. Ahmed, S. Din, G. Jeon, F. Piccialli, and G. Fortino, "Towards collaborative robotics in top view surveillance: A framework for multiple object tracking by detection using deep learning," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 7, pp. 1253–1270, Jul. 2021.

[3] X. Chang, P. Ren, P. Xu, Z. Li, X. Chen, and A. Hauptmann, "A comprehensive survey of scene graphs: Generation and application," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 1–26, Jan. 2023.

[4] S. Gao et al., "Fully complex-valued dendritic neuron model," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 2105–2118, Apr. 2023.

[5] P. Xiong, J. Liao, M. Zhou, A. Song, and P. X. Liu, "Deeply supervised subspace learning for cross-modal material perception of known and unknown objects," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 2259–2268, Feb. 2023.

[6] M. Zhao et al., "PCUNet: A context-aware deep network for coarse-to-fine point cloud completion," *IEEE Sensors J.*, vol. 22, no. 15, pp. 15098–15110, Aug. 2022.

[7] F. M. Shakiba, S. M. Azizi, and M. Zhou, "A transfer learning-based method to detect insulator faults of high-voltage transmission lines via aerial images: Distinguishing intact and broken insulator images," *IEEE Syst., Man, Cybern. Mag.*, vol. 8, no. 4, pp. 15–25, Oct. 2022.

[8] H. Li, G. Hu, J. Li, and M. Zhou, "Intelligent fault diagnosis for large-scale rotating machines using binarized deep neural networks and random forests," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 2, pp. 1109–1119, Apr. 2022.

[9] Y. Jiao, K. Yang, D. Song, and D. Tao, "TimeAutoAD: Autonomous anomaly detection with self-supervised contrastive loss for multivariate time series," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 3, pp. 1604–1619, May 2022.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[11] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, May 2018, pp. 1–23.

[12] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[13] L. Li, D. Li, T. Song, and X. Xu, "Actor–critic learning control with regularization and feature selection in policy gradient estimation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1217–1227, Mar. 2021.

[14] L. Li, D. Li, T. Song, and X. Xu, "Actor-critic learning control based on $\ell_2$-regularized temporal-difference prediction with gradient correction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 5899–5909, Apr. 2018.

[15] J. Gu, J. Wang, X. Guo, G. Liu, S. Qin, and Z. Bi, "A metaverse-based teaching building evacuation training system with deep reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 4, pp. 2209–2219, Apr. 2023.

[16] I. Ilahi et al., "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," *IEEE Trans. Artif. Intell.*, vol. 3, no. 2, pp. 90–109, Apr. 2022.

[17] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, Dec. 2017, pp. 6382–6393.

[18] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.* K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97, Jun. 2019, pp. 2961–2970.

[19] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *Proc. AAAI Conf. Artif. Intell.* New York, NY, USA: AAAI Press, vol. 34, Feb. 2020, pp. 7211–7218.

[20] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst.* Richland, SC, USA: International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 2085–2087.

[21] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent Q-learning," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, May 2021, pp. 1–27.

[22] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. 35th ICML*, J. Dy and A. Krause, Eds. vol. 80, Jul. 2018, pp. 5571–5580.

[23] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient," in *Proc. 33rd AAAI Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, vol. 33, Feb. 2019, pp. 4213–4220.

[24] J. Lin, K. Dzeparoska, S. Q. Zhang, A. Leon-Garcia, and N. Papernot, "On the robustness of cooperative multi-agent reinforcement learning," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2020, pp. 62–68.

[25] L. Zheng et al., "Magent: A many-agent reinforcement learning platform for artificial collective intelligence," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, Apr. 2018, pp. 1–2.

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

[27] J. Sun et al., "Stealthy and efficient adversarial attacks against deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, Feb. 2020, pp. 5883–5891.

[28] H. Zhang et al., "Robust deep reinforcement learning against adversarial perturbations on state observations," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, Dec. 2020, pp. 1–14.

[29] H. Zhang, H. Chen, D. S. Boning, and C. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, May 2021, pp. 1–16.

[30] T. P. Oikarinen, W. Zhang, A. Megretski, L. Daniel, and T. Weng, "Robust deep reinforcement learning through adversarial loss," in *Proc. 35th Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, Dec. 2021, pp. 26156–26167.

[31] Z. Zhou, G. Liu, and Y. Tang, "Multi-agent reinforcement learning: Methods, applications, visionary prospects, and challenges," 2023, *arXiv:2305.10091*.

[32] Y. Chen, Z. Zheng, and X. Gong, "MARNet: Backdoor attacks against cooperative multi-agent reinforcement learning," *IEEE Trans. Dependable Secure Comput.*, early access, Sep. 19, 2022, doi: 10.1109/TDSC.2022.3207429.

[33] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, pp. 1–51, Jan. 2020.

[34] Y. Wang, E. Sarkar, W. Li, M. Maniatakos, and S. E. Jabari, "Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4772–4787, 2021.

[35] J. Guo, A. Li, and C. Liu, "Backdoor detection and mitigation in competitive reinforcement learning," 2022, *arXiv:2202.03609*.

[36] L. Zhang et al., "TN-ZSTAD: Transferable network for zero-shot temporal activity detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3848–3861, Mar. 2023.

[37] C. Yan et al., "ZeroNAS: Differentiable generative adversarial networks search for zero-shot learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9733–9740, Dec. 2022.

[38] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2020, pp. 737–744.

[39] M. Yan, I. Frosio, S. Tyree, and J. Kautz, "Sim-to-real transfer of accurate grasping with eye-in-hand observations and continuous control," 2017, *arXiv:1712.03303*.

[40] E. Candela, L. Parada, L. Marques, T. Georgescu, Y. Demiris, and P. Angeloudis, "Transferring multi-agent reinforcement learning policies for autonomous driving using sim-to-real," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 8814–8820.

[41] C. Yu et al., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Advances in Neural Information Processing Systems*, vol. 35, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds. Red Hook, NY, USA: Curran Associates, 2022, pp. 24611–24624.

[42] H. Shi, G. Liu, K. Zhang, Z. Zhou, and J. Wang, "MARL Sim2real transfer: Merging physical reality with digital virtuality in metaverse," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 53, no. 4, pp. 2107–2117, Apr. 2023.

[43] L. S. Shapley, "Stochastic games," *Proc. Nat. Acad. Sci. USA*, vol. 39, no. 10, pp. 1095–1100, Oct. 1953.

[44] A. M. Fink, "Equilibrium in a stochastic $n$-person game," *Hiroshima Math. J.*, vol. 28, no. 1, p. 1, 1964.

[45] V. R. Konda and J. N. Tsitsiklis, "Actor-critic agorithms," in *Proc. 12th Int. Conf. Neural Inf. Process. Syst.* Cambridge, MA, USA: MIT Press, 1999, pp. 1008–1014.

[46] Y. Hu and Z. Zhang, "Sparse adversarial attack in multi-agent reinforcement learning," 2022, *arXiv:2205.09362*.

[47] C. Sun, D. Kim, and J. P. How, "ROMAX: Certifiably robust deep multiagent reinforcement learning via convex relaxation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 5503–5510.

[48] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, p. 334, Mar. 1997.

[49] S. Gowal et al., "On the effectiveness of interval bound propagation for training verifiably robust models," 2018, *arXiv:1810.12715*.

[50] E. Wong, F. Schmidt, and Z. Kolter, "Wasserstein adversarial examples via projected Sinkhorn iterations," in *Proc. 36th Int. Conf. Mach. Learn.*, K. Chaudhuri and R. Salakhutdinov, Eds. vol. 97, Jun. 2019, pp. 6808–6817.

**Ziyuan Zhou** received the B.S. degree from the China University of Mining and Technology, Xuzhou, China, in 2020. She is currently pursuing the Ph.D. degree in computer software and theory with the Department of Computer Science, Tongji University, Shanghai, China.

Her research interests include reinforcement learning, multi-agent system, and adversarial attacks.

**Guanjun Liu** (Senior Member, IEEE) received the Ph.D. degree in computer software and theory from Tongji University, Shanghai, China, in 2011.

He was a Post-Doctoral Research Fellow with the Singapore University of Technology and Design, Singapore, from 2011 to 2013, and the Humboldt University of Berlin, Berlin, Germany, from 2013 to 2014, supported by the Alexander von Humboldt Foundation. He is currently a Professor with the Department of Computer Science, Tongji University. He has authored or coauthore more than 140 papers and three books. His research interests include Petri net theory, model checking, machine learning, cyber-physical systems, and multi-UAV cooperation.

Dr. Liu served as a Guest Editor for some journals including the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, the *Mathematical Problems in Engineering*, and the *Journal of Software* (in Chinese), and also served as a Program Committee Member for many international conferences. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS.

**Mengchu Zhou** (Fellow, IEEE) received the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, where he is now a Distinguished Professor. He has more than 1100 publications including 14 books, more than 750 journal articles (more than 600 in IEEE TRANSACTIONS), 31 patents, and 32 book-chapters. His research interests include Petri nets, automation, the Internet of Things, and big data.

He is also a fellow of International Federation of Automatic Control (IFAC), American Association for the Advancement of Science (AAAS), Chinese Association of Automation (CAA), and National Academy of Inventors (NAI).