# Adversarial Attack on Black-Box Multi-Agent by Adaptive Perturbation

**Jianming Chen**[1,2,3,4], **Yawen Wang**[1,2,3,4*], **Junjie Wang**[1,2,3,4*], **Xiaofei Xie**[5], **Yuanzhe Hu**[1,2,3,4], **Qing Wang**[1,2,3,4], **Fanjiang Xu**[1,2,3,4*]

[1] Institute of Software Chinese Academy of Sciences, Beijing, China
[2] Science & Technology on Integrated Information System Laboratory, Beijing, China
[3] State Key Laboratory of Complex System Modeling and Simulation Technology, Beijing, China
[4] University of Chinese Academy of Sciences, Beijing, China
[5] Singapore Management University, Singapore
{jianming2023, yawen2018, junjie}@iscas.ac.cn, xfxie@smu.edu.sg, {yuanzhe, wq, fanjiang}@iscas.ac.cn

## Abstract

Evaluating security and reliability for multi-agent systems (MAS) is urgent as they become increasingly prevalent in various applications. As an evaluation technique, existing adversarial attack frameworks face certain limitations, e.g., impracticality due to the requirement of white-box information or high control authority, and a lack of stealthiness or effectiveness as they often target all agents or specific fixed agents. To address these issues, we propose AdapAM, a novel framework for adversarial attacks on black-box MAS. AdapAM incorporates two key components: (1) *Adaptive Selection Policy* simultaneously selects the victim and determines the anticipated malicious action (the action would lead to the worst impact on MAS), balancing effectiveness and stealthiness. (2) *Proxy-based Perturbation to Induce Malicious Action* utilizes generative adversarial imitation learning to approximate the target MAS, allowing AdapAM to generate perturbed observations using white-box information and thus induce victims to execute malicious action in black-box settings. We evaluate AdapAM across eight multi-agent environments and compare it with four state-of-the-art and commonly-used baselines. Results demonstrate that AdapAM achieves the best attack performance in different perturbation rates. Besides, AdapAM-generated perturbations are the least noisy and hardest to detect, emphasizing the stealthiness.

## Introduction

Recent years have witnessed sensational advances in reinforcement learning (RL) across many prominent sequential decision-making problems (Mhammedi, Foster, and Rakhlin 2024; Ma et al. 2024). As these problems have grown in complexity, the field has transitioned from using primarily single-agent RL algorithms to multi-agent RL (MARL) algorithms, which are playing increasingly significant roles in various domains, e.g., unmanned aerial vehicles (Liu et al. 2023; Zhang et al. 2023), industrial robots (Chen et al. 2025b; Gu et al. 2023), and auto-driving (Petrillo et al. 2018; Yeh and Soo 2024). These multi-agent systems (MAS) rely on multiple agents collaborating to achieve complex tasks, where each agent operates based on decentralized decision-making and shared information. However, the increasing deployment of such MAS has also made them attractive targets for adversarial attacks, raising concerns about their security and reliability in critical environments (Zhang et al. 2024b).

Adversarial attacks are techniques used to assess the security and reliability of artificial intelligence (AI) systems by deliberately introducing inputs that can mislead the system into making incorrect decisions (Shen et al. 2024; Zhang et al. 2024a). For example, some imperceptible perturbations are added to the inputs of the attacked model to make that model produce the wrong output (Hong et al. 2024; Fang et al. 2024). The adversarial attacks against machine learning (ML) systems have been extensively studied (Chan and Cheng 2025; Zhang et al. 2025), including targeting single-agent systems (Huang et al. 2017; Zhang et al. 2020, 2021). However, in MAS, the increasing interactions and dependencies between agents, as well as across time steps, introduce significant challenges that require further investigation to develop an effective adversarial attack framework (Chen et al. 2025a).

The current research on adversarial attacks against MAS usually relies on unrealistic assumptions about the problem setting. Many studies assume that attackers have high-level access to the victim MAS, such as requiring internal network outputs or other white-box information (Zhou et al. 2024; Lin et al. 2020), or even the action control authority of target agents (Gleave et al. 2020; Li et al. 2024). However, such assumptions are impractical in practical applications. In reality, scenarios are more likely to involve the strict black-box setting, where attackers cannot directly access internal models or parameters of the system. Instead, they can only implement threats by perturbing the observations of the target agents (Kraus et al. 2020). This strict black-box setting is not only more practical but also presents greater challenges and is therefore more worthwhile to study.

Furthermore, existing adversarial attack methods for MAS often struggle with a trade-off between effectiveness and stealthiness. For example, some approaches (Guo et al. 2022; Han et al. 2024) add perturbations to all agents, which leads to poor stealthiness (in terms of the number of perturbed agents). Some methods only consider specific mem-

*Corresponding authors.

bers within the MAS as fixed adversaries (Li et al. 2024), or fail to provide effective agent selection strategies (Zhou et al. 2024), making it difficult to achieve efficient attacks in the diverse state transitions of the multi-agent Markov Decision Process (MDP) (Wen et al. 2022). Under constraints like limited attack budgets and stealthiness requirements, existing methods exhibit a lack of effectiveness and stealthiness. Therefore, there is an urgent need for an attack method that can effectively and stealthily operate in strict black-box settings, thereby comprehensively assessing the robustness and safety of the MAS.

Drawing from the aforementioned issues, we propose **AdapAM**, a novel learning-based framework for ***Adap***tive adversarial ***A***ttacks on the black-box ***M***AS. Our attack operates in a strict black-box setting, where only the observations and corresponding actions of the victim MAS can be accessed. Besides, only manipulating and perturbing the observations of agents is allowed, rather than actions. AdapAM adaptively selects the most important agent and determines the anticipated malicious action (achieved by specifically perturbing its observations), maximizing the effectiveness of the attack. We define malicious action as the action we aim to induce the adversary to execute by perturbing its observation, which causes the worst impact on the victim MAS. It avoids targeting multiple agents, thereby achieving both effectiveness and stealthiness in the attack. The primary focus of this approach is to adaptively select the most important agent as the adversary and determine the corresponding malicious actions. We first design an adaptive selection policy, which learns to select the adversary and determine malicious action at each time step, based on the environment state. The training of this policy is modeled as an RL process that optimizes the objective based on reducing the reward of the target agent. Secondly, since determining the observations that can lead to malicious actions requires knowledge of the internal model (i.e., white-box setting), we conduct the perturbation generation through proxy agents while operating in a strict black-box setting. The proxy agents are trained in a generative adversarial imitation learning way with a mapping from observation to action that approximates the victim MAS. Therefore, the white-box information of proxy agents can be utilized to better produce perturbed observations.

We evaluate AdapAM in eight popular multi-agent environments and compare it with four state-of-the-art and commonly-used baselines. The results indicate that, compared with the baselines, AdapAM achieves almost optimal attack performance against both normal MAS and robust MAS across all environments and various perturbation rate settings, including both reward and win rate metrics. Additionally, we evaluate the stealthiness of our AdapAM in two aspects: first, by calculating the distance between the observations before and after adding perturbations; second, by assessing the detection rate using existing attack detection methods to determine whether the victim MAS is under attack. Experimental results show that the perturbations added by AdapAM are the smallest and the attacks it causes are the most difficult to detect, demonstrating that AdapAM has better stealthiness.

The main contributions of this work are as follows.

- A novel black-box adversarial attack framework for MAS, learning an adaptive policy to select adversary and anticipated malicious action at each time step.
- The perturbations generated via proxy agents, where the proxy agents are the approximation of the victim MAS to provide white-box information and address the challenge of generating perturbations in a black-box setting.
- Experimental evaluation of attack performance, stealthiness of AdapAM in eight multi-agent tasks, which outperforms four state-of-the-art and commonly-used baselines and demonstrates the effects of adversary and malicious action selection.

## Related Work

### Adversarial Attack

Adversarial attack (Shen et al. 2024) is a type of attack targeting ML models, particularly deep neural network (DNN). By introducing perturbations to the input data, attackers aim to mislead the model into making incorrect predictions or classifications. These perturbations are generated using either white-box (Papernot et al. 2016; Moosavi-Dezfooli, Fawzi, and Frossard 2016) or black-box (Guo et al. 2019; Andriushchenko et al. 2020) based methods and are usually undetectable to humans, but can have a significant impact on the outputs of models. These findings have attracted a lot of attention since they suggest that ML models may lack robustness in some specific situations.

Regarding the adversarial attacks against agent systems, the studies (Huang et al. 2017) and (Zhang et al. 2020) utilize gradient-based adversary attacks to generate adversarial perturbations of the state. However, they only mislead an agent to do a wrong action and may not lead to the minimal expected reward. In (Zhang et al. 2021), the adversary is modeled as a Markov decision process (MDP), and RL is employed to address it, which can work well in a low-dimensional state space. To extend this method to a high-dimensional one, the work (Sun et al. 2022) proposes a two-step attack framework for advising the worst-case action and generating perturbations.

Despite these advancements, the majority of existing work primarily concentrates on single-agent systems, often neglecting the critical aspect of selecting which agents should be designated as adversaries in multi-agent reinforcement learning (MARL) models. This consideration is vital, as the effectiveness of adversarial attacks can vary significantly depending on the specific agents targeted.

### Adversarial Attack on MAS

(Lin et al. 2020) attacks a default agent in the MAS by generating perturbed observations in a JSMA-based algorithm (Papernot et al. 2016), which is a white-box approach. (Guo et al. 2022) and (Han et al. 2024) consider all agents in the MAS as victims of the attack. The attack has superior performance due to the large number of agents affected, but such attacks are extremely easy to detect. (Zhou et al. 2024) proposes an optimization algorithm optimized with a joint action-value function to select key agents as victims for the attack. However, the value function is unavailable in strict

black-box situations. In addition, the current environmental state is not considered in the selection. Considering only attacking a fixed victim or not selecting a victim based on the state of the environment leads to ineffective attacks.

Attacking the actions of the agent in MAS is another scenario. (Gleave et al. 2020) shows that in a two-agent competitive environment, the actions of one agent can be manipulated to fool the other. (Li et al. 2024) controls a fixed agent in a cooperative MAS, learning the action that is most harmful to the other agents. However, it is a strong assumption to have action manipulation permissions for the target agent, which is a rare case in practice.

## Threat Model and Problem Statement

### Threat Model

In this paper, we propose an adaptive adversarial attack targeting black-box MAS.

**Definition 1.** A MAS consists of a set of agents, denoted as $M = \{m_1, m_2, \ldots, m_n\}$, where $|M| = n \geq 2$. Each agent $m_i$ has the independent local observation space $O_i$ and the action space $A_i$. MAS typically maintains internal collaboration mechanisms to collectively accomplish a task (e.g., to counter another MAS or to achieve some goals).

**Attacker's Objective:** Previous work (Chen et al. 2025a; Zhou et al. 2024) has shown that attacking only a small number of the most critical agents can significantly degrade the performance of the victim MAS. Therefore, if it is possible to select an adversary at each time step adaptively, it can avoid perturbing a large number of agents to ensure stealthiness. On the other hand, the selected adversary needs to have a sufficient influence on the victim MAS to ensure the effectiveness of the attack. Based on this, the attacker's objective is to adaptively select an agent as the target adversary at each time step, using perturbations to induce it to execute specific malicious actions to degrade the performance of the MAS.

**Attacker's Capability:** The attacker possesses control over the environment, which is a common situation. Specifically, during the training phase, the attacker can fully control the actions of the adversary. For instance, in real-world multi-agent autonomous driving, a driver can seize control to override the outputs of the autonomous agent. In the simulation environment, we can achieve this by tampering with the action signals received by the environment. In the deployment phase of the attack, the attacker can only manipulate the state observation provided to the adversary agent, aiming to mislead its policy through perturbations. It is a strict setting, which can be achieved by adding a patch as a perturbation to the agent camera (Wei et al. 2023).

We assume that the victim MAS policy is fixed, i.e., the parameters in the deployment phase are frozen (Gu et al. 2023; Zhang et al. 2023). Finally, the attack we study adheres to a strict black-box setting, where control of actions for the selected adversary is achieved through environmental perturbations, without access to the algorithm used for training or constructing agents and their network architecture.

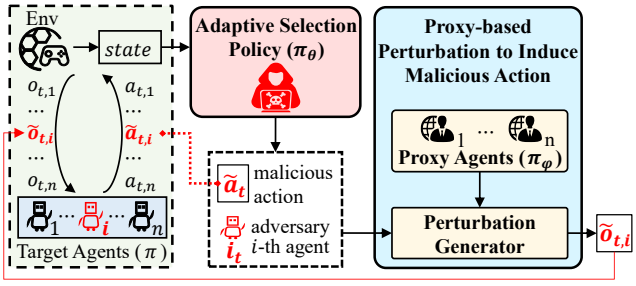


Figure 1: The overview of our proposed AdapAM.

### Problem Statement

Formally, the multi-agent Markov Decision Process (MDP) (Lu et al. 2021) is defined as follows:

$$G = (n, \mathcal{S}, \{\mathcal{A}_i\}, \{\mathcal{O}_i\}, \pi, T, R), \tag{1}$$

where $n$ represents the number of agents, indicating the total count of decision-makers involved. $\mathcal{S}$ denotes the global state space, which encompasses all possible states $s$ that describe the current configuration of all agents and their environment. $\mathcal{A}_i$ denotes the set of actions that the $i$-th agent can choose from, while the joint action space of MAS can be represented as the Cartesian product of all individual action spaces. The observation space $\mathcal{O}_i$ refers to the set of information that the $i$-th agent can perceive, which is typically a subset of the global state, reflecting the partial information available to the agents. We consider a problem setting where a joint policy $\pi = \{\pi_1, ..., \pi_n\}$ has been well trained. At each time-step $t$, the $i$-th agent obtains its local observation $o_{t,i}$ derived from the global environment state $s_t$. The policy $\pi_i(a_{t,i}|o_{t,i})$ of the $i$-th agent determines the action $a_{t,i}$ solely based on its local observation $o_{t,i}$. The joint action $a_t = \{a_{t,1}, ..., a_{t,n}\}$ transitions the system to the next state $s_{t+1}$ according to the state transition function $T(s_{t+1}|s_t, a_t)$. Thereby, a global reward $r_t$ is obtained according to function $R(s_t, a_t, s_{t+1})$.

We further decompose the problem into two subproblems, as follows:

**(1) Adaptive Selection Policy**

The attacker's policy is defined as the following MDP:

$$G^\alpha = (\mathcal{S}, \mathcal{A}^\alpha, T, R^\alpha). \tag{2}$$

Retain the same state space $\mathcal{S}$ and state transition function $T$ as in Equation 1. However, the attacker's policy needs to adaptively select the agent to target as an adversary and the corresponding malicious action based on the state. $\mathcal{A}^\alpha$ denotes the action space of the policy, which outputs the attack action $\tilde{a}_i$. $\tilde{a}_i$ represents the i-th agent (serves as adversary) and its corresponding malicious action $\tilde{a}$ (sampled from $\mathcal{A}_i$). In addition, the attacker's reward $R^\alpha$ aims to degrade the performance of the victim MAS.

**(2) Perturbation to Induce Malicious Action**

When we obtain $\tilde{a}_i$, we do not directly control the adversary's actions to execute the attack. Instead, we generate perturbed observation for the adversary to induce it to output the malicious actions we anticipate. Therefore, we need

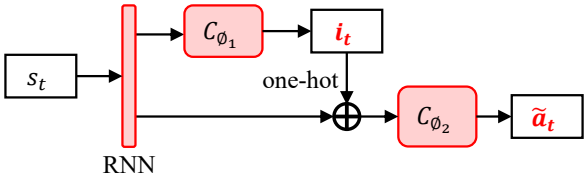


Figure 2: The architecture of the Adaptive Selection Policy.

a perturbed observation $\tilde{o}_i$, as follows:

$$\begin{aligned} \tilde{o}_i &= o_i + \delta, \\ \tilde{o}_i &= \arg\max_{||\delta|| \leq \epsilon} \mathbb{1}(\pi_i(o_i + \delta) = \tilde{a}_i), \end{aligned} \tag{3}$$

where $\delta$ is the adversarial perturbation to be added, $\epsilon$ represents the scale of the perturbation, and $\mathbb{1}(\cdot)$ is the indicator function (which is true when $\pi_i$ outputs the malicious action $\tilde{a}_i$ based on the perturbed observation $o_i + \delta$). We need to find a perturbation that satisfies both Equation 3.

## Approach

We propose AdapAM, a framework to adaptively select the adversary and determine malicious action on the target MAS. The adversary represents the most critical agent in the victim MAS, while the malicious action means the action induced by perturbing the adversary's observation to cause the worst impact on the target MAS. AdapAM utilizes white-box information from proxy agents to generate a perturbed observation, which is then injected into the observation of adversary. As shown in Figure 1, AdapAM primarily consists of two modules: **Adaptive Selection Policy** and **Proxy-based Perturbation to Induce Malicious Action**.

The target MAS comprises $n$ agents interacting with the environment. As previously discussed, each agent plays a distinct role, whose actions have varying degrees of impact. To maximize the effectiveness of an attack under a limited attack budget, it is necessary to select the most important agent as the adversary and the most malicious action based on the environmental state. Therefore, we design an adaptive selection policy to select the adversary agent $i_t$ and a specific malicious action $\tilde{a}_t$ desired to be performed by $i_t$.

To mislead $i_t$ into executing the action $\tilde{a}_t$ in a black-box setting, the proxy agents are required to be trained in advance first. The policies of proxy agents approximate those of target agents, ensuring transferability between the two. Utilizing the gradient information provided by the proxy model, we can employ the white-box method to generate a perturbed observation $\tilde{o}_{t,i}$ corresponding to the desired malicious action $\tilde{a}_t$. The $\tilde{o}_{t,i}$ is then injected into the observation of the adversary agent, effectively inducing it to execute $\tilde{a}_t$.

### Adaptive Selection Policy

We design a learning-based adaptive selection policy, which aims to adaptively select the adversary agent and determine its anticipated malicious action according to the environment state, thus significantly degrading the performance of the entire MAS with a limited attack budget.

**Algorithm 1:** Adaptive Selection Policy Learning

**Input:** Initial network parameters $\phi$, $\theta_1$, $\theta_2$, $\bar{\theta}_1$, $\bar{\theta}_2$
**Output:** Updated parameters $\phi$, $\theta_1$, $\theta_2$
1 **Initialization:** episode buffer $\mathcal{D} \leftarrow \emptyset$, weight parameter $\mu = 0.005$
2 **foreach** *iteration* **do**
3  **foreach** *environment step* **do**
4  Sample $(i_t, \tilde{a}_t) \sim \pi_\phi(i_t, \tilde{a}_t | s_t)$
5  Generate perturbation $\tilde{o}_{t,i}$, which induces adversary agent $i_t$ to take action $\tilde{a}_t$
6  Inject perturbation $o_{t,i} \leftarrow \tilde{o}_{t,i}$
7  Sample $a_t \sim \pi(a_t | o_t)$
8  Sample next state $s_{t+1} \sim T(s_{t+1} | s_t, a_t)$
9  Get attack reward $r_t^a = -R(s_t, a_t, s_{t+1})$
10  Save data $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, i_t, \tilde{a}_t, r_t^a, s_{t+1})\}$
11  **end**
12  **foreach** *gradient step* **do**
13  **for** $k \in \{1, 2\}$ **do**
14  Update critic: $\theta_k \leftarrow \theta_k - \hat{\nabla}_{\theta_k} J_Q(\theta_k)$
15  **end**
16  Update policy: $\phi \leftarrow \phi - \hat{\nabla}_\phi J_\pi(\phi)$
17  **for** $k \in \{1, 2\}$ **do**
18  Update target critic: $\bar{\theta}_i \leftarrow \mu\theta_k + (1 - \mu)\bar{\theta}_i$
19  **end**
20  **end**
21 **end**

The adaptive selection policy $\pi_\phi(i_t, \tilde{a}_t | s_t)$, parameterized by $\phi$, learns to determine which adversary agent $i_t$ to attack and the malicious action $\tilde{a}_t$ it intends the adversary to take, at the time-step $t$ and given the current environmental state $s_t$. If we directly learn $i_t$ and $\tilde{a}_t$ simultaneously, the search space is large, making convergence difficult. Therefore, technically, $\pi_\phi(i_t, \tilde{a}_t | s_t)$ actually includes two classifiers: $C_{\phi_1}(i_t | s_t)$ for selecting $i_t$ and $C_{\phi_2}(\tilde{a}_t | s_t, i_t)$ for selecting $\tilde{a}_t$. After $C_{\phi_1}(i_t | s_t)$ output $i_t$, we concatenate $s_t$ with the one-hot vector of $i_t$, and then input them together into the $C_{\phi_2}(\tilde{a}_t | s_t, i_t)$ for selecting $\tilde{a}_t$. As shown in Figure 2, it is a hierarchical design to help reduce the search space

Besides, we design the learning process of $\pi_\phi(i_t, \tilde{a}_t | s_t)$ based on SAC (Christodoulou 2019), as shown in Algorithm 1. The process alternates between collecting experience buffers $\mathcal{D}$ from the environment with the current policy and updating the network parameters using the stochastic gradients from batches sampled from $\mathcal{D}$.

Lines 2 to 11 of the Algorithm 1 describe the procedure of collecting buffers. At each time-step, $i_t$ and $\tilde{a}_t$ are sampled from $\pi_\phi(i_t, \tilde{a}_t | s_t)$. Then AdapAM generates the corresponding perturbed observation $\tilde{o}_{t,i}$ based on $i_t$ and $\tilde{a}_t$, using the module of *Proxy-based Perturbation to Induce Malicious Action*, which will be presented later. $\tilde{o}_{t,i}$ will then be injected into the observation of the target agent $i_t$, and the target MAS decides the actions based on the injected observations. After calculating the next state according to the state transition function and corresponding at-

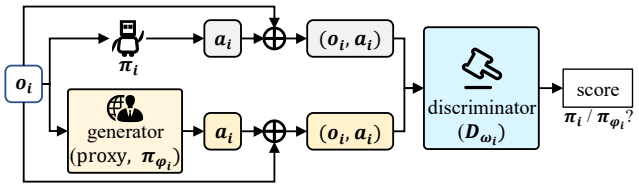


Figure 3: The setting of the training proxy agent.

tack reward $r_t^a$, the episode data are stored into $\mathcal{D}$, where $r_t^a = -R(s_t, a_t, s_{t+1})$.

Lines 12 to 20 of the Algorithm 1 describe the procedure of updating network parameters. The Q-function (i.e., critic) parameters can be trained to minimize the following soft Bellman residual (Haarnoja et al. 2018):

$$\begin{aligned} y &= r_t^a + \gamma(\bar{\theta}(s_{t+1}, i_{t+1}, \tilde{a}_{t+1}) \\ &- \alpha \log \pi_\phi(i_{t+1}, \tilde{a}_{t+1}|s_{t+1})), \\ J_Q(\theta) &= \mathbb{E}_{(s_t, i_t, \tilde{a}_t)\sim\mathcal{D}}[\frac{1}{2}(Q_\theta(s_t, i_t, \tilde{a}_t) - y)^2], \end{aligned} \tag{4}$$

where $\alpha$ is a weighting parameter to balance exploration and learning. The policy parameters can be updated by minimizing the following expected KL-divergence:

$$\begin{aligned} J_\pi(\phi) &= \mathbb{E}_{s_t\sim\mathcal{D}}[\mathbb{E}_{(i_t, \tilde{a}_t)\sim\pi_\phi}[\alpha \log(\pi_\phi(i_t, \tilde{a}_t|s_t)) \\ &- Q_\theta(s_t, i_t, \tilde{a}_t)]]. \end{aligned} \tag{5}$$

In line 14 and line 16 of Algorithm 1, the $\hat{\nabla}_{\theta_k} J_Q(\theta_k)$ and $\hat{\nabla}_\phi J_\pi(\phi)$ represent the gradients of $J_Q(\theta)$ and $J_\pi(\phi)$, which are used to update the critic and policy networks, respectively. Besides, in line 14 and line 18, we use 2 critic networks and 2 target critic networks to calculate Q-values following the setting in SAC (Haarnoja et al. 2018), which have been shown to stabilize and boost training.

## Proxy-based Perturbation to Induce Malicious Action

To efficiently generate perturbed observation that induces adversaries to perform specific malicious actions in the black-box MAS, we train proxy agents to provide white-box information for perturbation generation.

**Training the Proxy Agents** The policy of the $i$-th target agent $\pi_i(a_i|o_i)$, represents the mapping $o_i \rightarrow a_i$, from observation to action. If internal information of the mapping can be obtained (such as the intermediate layers' outputs and network gradients), it would be easy to generate perturbations that induce specific actions. However, this information cannot be directly accessed in a black-box setting To this end, a proxy agent is needed, which learns a policy (denoted as $\pi_{\psi_i}$) to approximate this mapping, to provide a pipeline for generating perturbed observation from malicious action:

$$\pi_{\psi_i}(a_i|o_i) \approx \pi_i(a_i|o_i). \tag{6}$$

We utilize Multi-Agent Generative Adversarial Imitation Learning (MAGAIL) (Song et al. 2018) to train proxy agents $\pi_{\psi_i}$. The proxy agents imitate the behavior of target agents by learning their policy through adversarial imitation and serve as surrogates for generating perturbations.

As shown in Figure 3, for each $i$-th agent, we have a discriminator (denoted as $D_{\omega_i}$) and a generator (denoted as $\pi_{\psi_i}$), i.e., proxy policy. The $D_{\omega_i}$ maps observation-action pairs to the score, which is optimized to discriminate the action sampled by the $i$-th target agent (with policy $\pi_i$) from the action sampled by proxy policy $\pi_{\psi_i}$.

**Perturbation Generator** After the proxy agents are trained, the internal information can be used to generate the perturbed observation corresponding to the malicious action. Given the malicious action $\tilde{a}_t$ at time-step $t$, the goal of the perturbation generator is to find $\tilde{o}_{t,i}$ such that $argmax\pi_i(\tilde{o}_{t,i}) = \tilde{a}_t$.

By utilizing the proxy agents, we leverage the white-boxed C&W attack technique (Carlini and Wagner 2017) to generate the perturbed observation. This method can misguide the target agent with a 100% success rate, which can increase the success rate of the attack. The $\tilde{o}_{t,i}$ can be found by minimizing the following function:

$$||o_{t,i} - \tilde{o}_{t,i}|| + f(\tilde{o}_{t,i}). \tag{7}$$

The first term in Equation 7 represents the distance between the perturbed observation and the original observation, i.e., the magnitude of the perturbation. We use Euclidean distance (Bardes, Ponce, and LeCun 2022) to ensure that small changes in each pixel point are calculated. The second term represents the objective function of ensuring that $\pi_i(\tilde{o}_{t,i}) = \tilde{a}_t$, i.e., it measures whether the input $\tilde{o}_{t,i}$ leads the policy $\pi_i$ to output the target action $\tilde{a}_t$. Suggested by (Carlini and Wagner 2017), we define $f(\tilde{o}_{t,i})$ as:

$$f(\tilde{o}_{t,i}) = \max(\max\{Z(a|\tilde{o}_{t,i}, a \neq \tilde{a}_t)\} - Z(\tilde{a}_t|\tilde{o}_{t,i}), 0), \tag{8}$$

where $Z(a|\tilde{o}_{t,i})$ denotes the output of the second-to-last network layer (i.e., the logits output), with $a$ as the target action.

# Experiment

## Experimental Setup

Our experiments are conducted on three popular multi-agent benchmarks with different characteristics, selecting two to three environments from each benchmark as follows.

**StarCraft Multi-Agent Challenge (SMAC).** SMAC (Samvelyan et al. 2019) simulates battle scenarios in which a team of controlled agents must destroy the built-in enemy team. We consider two environments in SMAC, which vary in the number and types of units controlled by agents.

**Google Research Football (GF).** GF (Kurach et al. 2020) provides the scenarios of controlling a team to play football against the built-in team. We choose two environments in GF, which vary in the number of players and the tactics.

**Multi-Agent Particle Environments (MPE).** MPE (Lowe et al. 2017) consists of navigation tasks, where agents need to control particles to reach the target landmarks. We study two of these tasks, which mainly differ in whether communication is required between agents.

We implement four state-of-the-art and popular baseline approaches for adversarial attacks in each multi-agent environment.

| Env | Metric | Normal Target MAS | | | | | | Robust Target MAS | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Origin | AdapAM (ours) | MASafe | AMCA | AMI | Lin | Origin | AdapAM (ours) | MASafe | AMCA | AMI | Lin |
| SMAC-1c3s5z | Reward | 20 | **9.02** | 9.36 | 10.34 | 12.27 | 13.68 | 19.06 | **9.12** | 10.79 | 11.02 | 12.59 | 14.80 |
| | Win Rate | 100% | **0%** | **0%** | **0%** | 17% | 25% | 91% | **0%** | **0%** | **0%** | 19% | 29% |
| SMAC-8m | Reward | 20 | **9.57** | 11.08 | 12.36 | 13.30 | 14.56 | 19.28 | **9.80** | 11.63 | 12.05 | 13.16 | 15.08 |
| | Win Rate | 100% | **0%** | 6% | 17% | 25% | 32% | 94% | **0%** | 10% | 14% | 22% | 37% |
| SMAC-bane_vs_bane | Reward | 20 | 13.21 | **12.25** | 14.63 | 15.52 | 16.02 | 18.79 | 13.73 | **13.05** | 14.82 | 15.22 | 15.70 |
| | Win Rate | 100% | 22% | **16%** | 30% | 36% | 43% | 90% | 24% | **21%** | 31% | 33% | 38% |
| SMAC-27m_vs_30m | Reward | 19.24 | 13.87 | **13.19** | 14.94 | 15.68 | 16.78 | 18.55 | 14.33 | **13.36** | 15.21 | 15.59 | 16.24 |
| | Win Rate | 94% | 28% | **23%** | 31% | 39% | 47% | 84% | 29% | **25%** | 33% | 37% | 45% |
| GF-counterattack | Reward | 5.21 | **0.56** | 0.59 | 0.69 | 0.95 | 1.08 | 4.89 | **0.49** | 0.78 | 0.76 | 1.08 | 1.27 |
| | Win Rate | 100% | **0%** | **0%** | 2% | 5% | 6% | 91% | **0%** | 3% | 3% | 6% | 13% |
| GF-3_vs_1 | Reward | 5.43 | **0.99** | 1.07 | 1.21 | 1.37 | 1.67 | 5.06 | **0.87** | 1.19 | 1.24 | 1.36 | 1.75 |
| | Win Rate | 100% | **0%** | **0%** | 12% | 16% | 21% | 98% | **0%** | 10% | 12% | 16% | 23% |
| MPE-spread | Reward | -546.99 | **-1059.49** | -1056.90 | -1018.49 | -1006.24 | -968.30 | -596.13 | **-1024.21** | -1006.11 | -971.57 | -972.15 | -953.28 |
| MPE-reference | Reward | -9.72 | **-38.80** | -37.71 | -36.61 | -36.82 | -35.08 | -11.06 | **-36.49** | -34.95 | -34.42 | -34.26 | -33.74 |

Table 1: Attack performance of different environments and MAS, measured by the reward or win rate (the lower, the better).

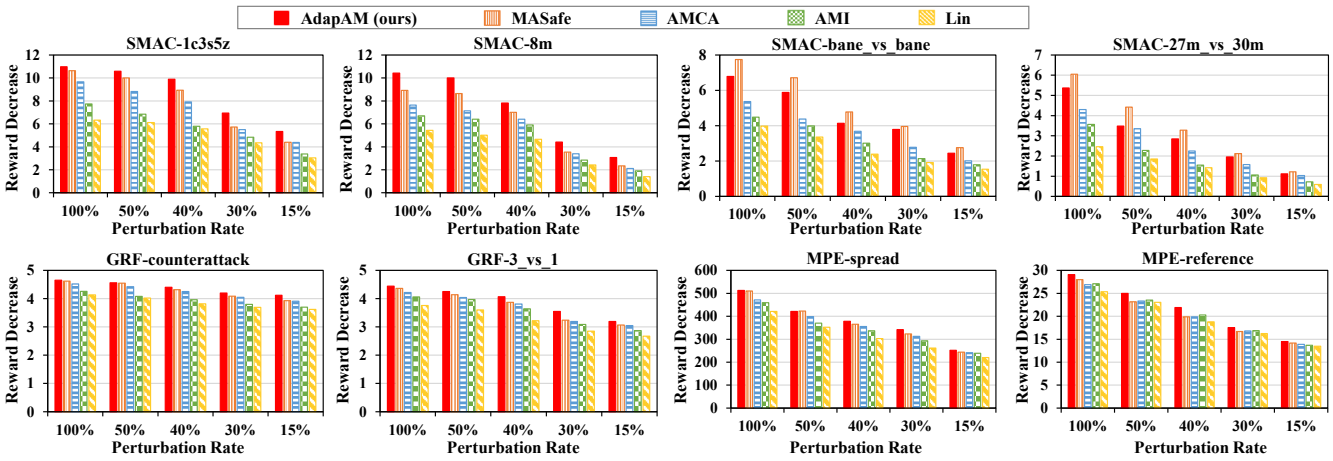


Figure 4: The decrease in reward after being attacked at different perturbation rates, on the *Normal Target MAS*.

**MASafe** (Guo et al. 2022): applying the random perturbations to the observations of all agents.

**AMCA** (Zhou et al. 2024): identifying important agents with a differential evolutionary algorithm and generating state perturbations after learning malicious actions.

**AMI** (Li et al. 2024): directly controlling the default agent, learning attack actions based on mutual information. It needs access to control the target agent.

**Lin** (Lin et al. 2020): generating observation perturbations corresponding to malicious actions against the default and fixed agent.

## Attack Performance

We evaluate the attack performance of our method in terms of its ability to degrade the overall functionality of a multi-agent system (MAS). Specifically, the effectiveness of the attack is measured by the reduction in the *Reward* and *Win Rate* of the target system across multiple benchmark environments. In addition, our attacks target two types of MAS: *Normal Target MAS*, which represents common MAS trained using basic MARL methods (using QMIX (Rashid et al. 2018)), and *Robust Target MAS*, which represents the MAS with enhanced resistance to attacks, trained using a robustness-enhancing method called ROMANCE (Yuan et al. 2023). By introducing attackers, ROMANCE enables the policy to encounter diverse adversarial attacks as an auxiliary during adversarial training, so that it is trained to be highly robust under various perturbations.

Table 1 compares the experimental results of the two target MAS in terms of *Reward* and *Win Rate*, in case of the *origin* (non-attacked) performance and the performance under different attack methods. Notably, win or loss is not defined in the MPE-spread and MPE-reference environments, so attack performance is measured only by reward.

Figure 4 compares the attack performance in the case of different perturbation rates on normal target MAS, where the attack performance is represented by the decrease in *reward* and *win rate*, respectively. The higher the decrease, the better the attack performance. The perturbation rate represents the percentage of time-steps to perform the attack in each episode. The results in Table 1 are for cases where the perturbation rate is 100%.

**Results.** In all environments, our proposed AdapAM significantly reduces both the reward and win rate. In the environments where the number of agents is less than 10, AdapAM effectively degrades the MAS's ability to achieve its objec-

| Env | AdapAM | MASafe | AMCA | Lin |
|---|---|---|---|---|
| SMAC-1c3s5z | **0.14** | 0.35 | 0.25 | 0.19 |
| SMAC-8m | **0.12** | 0.33 | 0.21 | 0.15 |
| SMAC-bane_vs_bane | **0.16** | 0.37 | 0.27 | 0.22 |
| SMAC-27m_vs_30m | **0.19** | 0.39 | 0.29 | 0.23 |
| GF-counterattack | **0.12** | 0.32 | 0.21 | 0.16 |
| GF-3_vs_1 | **0.10** | 0.31 | 0.20 | 0.15 |
| MPE-spread | **0.13** | 0.32 | 0.26 | 0.21 |
| MPE-reference | **0.14** | 0.40 | 0.22 | 0.19 |

Table 2: Stealthiness: the magnitude of perturbations (the lower, the better), on the normal target MAS.

| Env | AdapAM | MASafe | AMCA | AMI | Lin |
|---|---|---|---|---|---|
| SMAC-1c3s5z | **0.57** | 0.82 | 0.61 | 0.72 | 0.66 |
| SMAC-8m | **0.59** | 0.86 | 0.64 | 0.77 | 0.69 |
| SMAC-bane_vs_bane | **0.39** | 0.90 | 0.58 | 0.67 | 0.61 |
| SMAC-27m_vs_30m | **0.36** | 0.91 | 0.57 | 0.65 | 0.62 |
| GF-counterattack | **0.61** | 0.87 | 0.68 | 0.76 | 0.71 |
| GF-3_vs_1 | **0.67** | 0.91 | 0.70 | 0.79 | 0.74 |
| MPE-spread | **0.63** | 0.85 | 0.69 | 0.78 | 0.74 |
| MPE-reference | **0.71** | 0.92 | 0.76 | 0.81 | 0.79 |

Table 3: Stealthiness: the F1 score when facing attack detection (the lower, the better), on the normal target MAS.

tives (i.e., the Win Rate drops to $0\%$) and outperforms all baselines. Only when attacking in the *SMAC-bane_vs_bane* and *SMAC-27m_vs_30m* environments, AdapAM does not achieve optimal performance, being slightly weaker than MASafe. While MASafe perturbs all agents, we only choose one agent to attack. Even with a significant difference in the number of perturbed agents, AdapAM does not fall far behind MASafe, which further demonstrates the importance of selecting the target agent and malicious actions.

Besides, when attacking the Robust Target MAS, AdapAM achieves a more significant attack effect in all cases. It shows the capability of AdapAM to maintain superior attack performance against robust MAS. Similarly, as shown in Figure 4, AdapAM almost always outperforms baseline methods under different perturbation rates, except in environments (i.e., *SMAC-bane_vs_bane* and *SMAC-27m_vs_30m*) with a large number of agents where it is outperformed by MASafe.

Distinguished from other methods that attack default victim agents or select victim agents in ways that are not effective enough, the superior performance of AdapAM is due to its ability to learn victim-agent and malicious action in different environments and states. It is worth mentioning that although MASafe demonstrates outstanding attack performance, this is because it is the only method that applies perturbations to all agents. This approach leads to poor stealthiness, which is analyzed later.

### Stealthiness Evaluation

We evaluate the stealthiness from two aspects. First, we measured the magnitude of perturbations added by different methods, calculating the distance between the perturbed observation $\tilde{o}$ and the original observation $o$ using the L-$\infty$ norm (Warde-Farley and Goodfellow 2016) as shown below:

$$\|\tilde{o} - o\|_\infty = \max((\tilde{o}^1 - o^1), \ldots, (\tilde{o}^m - o^m)), \quad (9)$$

where $\tilde{o}^x$ or $o^x$ represents the $x$-th element in the observation vector. The smaller the L-$\infty$ distance, the more imperceptible the added perturbation is, i.e., better stealthiness. Since AMI attack directly manipulates the action of agent rather than adding perturbations, we did not calculate its perturbation distance. Its stealthiness is evaluated solely based on the abnormal action detection, described below. Table 2 represents the results of the magnitude of perturbations.

In addition to the perturbation magnitude, we use the method proposed in (Kazari, Shereen, and Dán 2023) to detect the abnormal actions of victim agents. This method predicts the action distribution of all agents based on the state and detects abnormal actions by calculating the normality scores. The F1 score of the detection results is used to represent stealthiness. The lower the F1 score, the harder it is to detect, and the more stealthy it is. Table 3 lists the results.

**Results.** As shown in Table 2, the perturbations added by AdapAM are the smallest in magnitude, outperforming the baselines. Moreover, the results in Table 3 demonstrate that AdapAM is the most difficult to detect, outperforming the baselines. Although MASafe exhibits an attack performance nearly comparable to AdapAM, and even outperforms AdapAM in one case, the results in Tables 2 and 3 indicate that the stealthiness of MASafe is the worst. The reason is that the high attack performance of MASafe stems from the fact that the perturbations are applied to all agents, in a way that does not take stealthiness into account at all.

On the contrary, AdapAM guarantees stealthiness by adaptively choosing only one victim agent through the adaptive selection policy. Besides, AdapAM leverages proxy agents, enabling the use of white-box-based adversarial example generation methods to easily craft undetectable perturbations, thereby ensuring its stealthiness.

## Conclusion

This paper proposes AdapAM for effective and stealthy black-box adversarial attacks on MAS. AdapAM includes two key components: (1) an adaptive selection policy that adaptively selects victim agents and determines malicious actions to consider simultaneously effectiveness and stealthiness; (2) a perturbation generation module using proxy agents trained to approximate the target MAS, effectively generating observation perturbations injected to victims for executing malicious actions. Experimental results across eight environments demonstrate the superior attack performance and better stealthiness of AdapAM, compared to four baselines. Our future work will focus on extending AdapAM to larger and more diverse scenarios and designing robust defense MAS based on the insights from AdapAM.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China Grant No.62232016 and No.62506355, Basic Research Program of ISCAS Grant No. ISCAS-JCZD-202304 and No.ISCAS-JCZD-202405, Strategic Priority Research Program of Chinese Academy of Sciences Grant No. XDB0900000, Major Program of ISCAS Grant No. ISCAS-ZD-202302, the National Research Foundation, Singapore, and the Cyber Security Agency under its National Cybersecurity R&D Programme (NCRP25-P04-TAICeN). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Cyber Security Agency of Singapore. The authors would like to thank the anonymous reviewers for their valuable comments.

## References

Andriushchenko, M.; Croce, F.; Flammarion, N.; and Hein, M. 2020. Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search. In *Computer Vision - ECCV 2020 - 16th European Conference*, volume 12368, 484–501. Springer.

Bardes, A.; Ponce, J.; and LeCun, Y. 2022. VICRegL: Self-Supervised Learning of Local Visual Features. In *Advances in Neural Information Processing Systems, NeurIPS*, volume 35, 8799–8810.

Carlini, N.; and Wagner, D. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, 39–57.

Chan, K. H.; and Cheng, B. H. C. 2025. Evoattack: suppressive adversarial attacks against object detection models using evolutionary search. *Autom. Softw. Eng.*, 32(1): 3.

Chen, J.; Wang, Y.; Wang, J.; Xie, X.; jun Hu; Wang, Q.; and Xu, F. 2025a. Understanding Individual Agent Importance in Multi-Agent System via Counterfactual Reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(15): 15785–15794.

Chen, J.; Wang, Y.; Wang, J.; Xie, X.; Wang, D.; Wang, Q.; and Xu, F. 2025b. Demo2Test: Transfer Testing of Agent in Competitive Environment with Failure Demonstrations. *ACM Trans. Softw. Eng. Methodol.*, 34(2).

Christodoulou, P. 2019. Soft Actor-Critic for Discrete Action Settings. arXiv:1910.07207.

Fang, Z.; Wang, T.; Zhao, L.; Zhang, S.; Li, B.; Ge, Y.; Li, Q.; Shen, C.; and Wang, Q. 2024. Zero-query adversarial attack on black-box automatic speech recognition systems. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 630–644.

Gleave, A.; Dennis, M.; Wild, C.; Kant, N.; Levine, S.; and Russell, S. 2020. Adversarial Policies: Attacking Deep Reinforcement Learning. In *International Conference on Learning Representations*.

Gu, S.; Grudzien Kuba, J.; Chen, Y.; Du, Y.; Yang, L.; Knoll, A.; and Yang, Y. 2023. Safe multi-agent reinforcement learning for multi-robot control. *Artificial Intelligence*, 319: 103905.

Guo, C.; Gardner, J. R.; You, Y.; Wilson, A. G.; and Weinberger, K. Q. 2019. Simple Black-box Adversarial Attacks. In *Proceedings of the 36th International Conference on Machine Learning, ICML*, volume 97, 2484–2493.

Guo, J.; Chen, Y.; Hao, Y.; Yin, Z.; Yu, Y.; and Li, S. 2022. Towards Comprehensive Testing on the Robustness of Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 115–122.

Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; and Levine, S. 2018. Soft Actor-Critic Algorithms and Applications. *arXiv*, abs/1812.05905.

Han, S.; Su, S.; He, S.; Han, S.; Yang, H.; Zou, S.; and Miao, F. 2024. What is the Solution for State-Adversarial Multi-Agent Reinforcement Learning? *Transactions on Machine Learning Research*.

Hong, H.; Zhang, X.; Wang, B.; Ba, Z.; and Hong, Y. 2024. Certifiable Black-Box Attacks with Randomized Adversarial Examples: Breaking Defenses with Provable Confidence. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 600–614.

Huang, S. H.; Papernot, N.; Goodfellow, I. J.; Duan, Y.; and Abbeel, P. 2017. Adversarial Attacks on Neural Network Policies. In *5th International Conference on Learning Representations, ICLR*.

Kazari, K.; Shereen, E.; and Dán, G. 2023. Decentralized Anomaly Detection in Cooperative Multi-Agent Reinforcement Learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023*, 162–170.

Kraus, S.; Azaria, A.; Fiosina, J.; Greve, M.; Hazon, N.; Kolbe, L.; Lembcke, T.-B.; Muller, J. P.; Schleibaum, S.; and Vollrath, M. 2020. AI for Explaining Decisions in Multi-Agent Environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 13534–13538.

Kurach, K.; Raichuk, A.; Stańczyk, P.; Zajac, M.; Bachem, O.; Espeholt, L.; Riquelme, C.; Vincent, D.; Michalski, M.; Bousquet, O.; and Gelly, S. 2020. Google Research Football: A Novel Reinforcement Learning Environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4501–4510.

Li, S.; Guo, J.; Xiu, J.; Zheng, Y.; Feng, P.; Yu, X.; Liu, A.; Yang, Y.; An, B.; Wu, W.; and Liu, X. 2024. Attacking Cooperative Multi-Agent Reinforcement Learning by Adversarial Minority Influence. arXiv:2302.03322.

Lin, J.; Dzeparoska, K.; Zhang, S. Q.; Leon-Garcia, A.; and Papernot, N. 2020. On the Robustness of Cooperative Multi-Agent Reinforcement Learning. In *2020 IEEE Security and Privacy Workshops (SPW)*, 62–68.

Liu, D.; Dou, L.; Zhang, R.; Zhang, X.; and Zong, Q. 2023. Multi-Agent Reinforcement Learning-Based Coordinated Dynamic Task Allocation for Heterogenous UAVs. *IEEE Transactions on Vehicular Technology*, 72: 4372–4383.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-Agent Actor-Critic for Mixed

Cooperative-Competitive Environments. *Neural Information Processing Systems, NeurIPS*.

Lu, S.; Zhang, K.; Chen, T.; Başar, T.; and Horesh, L. 2021. Decentralized Policy Gradient Descent Ascent for Safe Multi-Agent Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10): 8767–8775.

Ma, C.; Li, A.; Du, Y.; Dong, H.; and Yang, Y. 2024. Efficient and scalable reinforcement learning for large-scale network control. *Nature Machine Intelligence*, 6(9): 1006–1020.

Mhammedi, Z.; Foster, D. J.; and Rakhlin, A. 2024. The Power of Resets in Online Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 37, 12334–12407.

Moosavi-Dezfooli, S.; Fawzi, A.; and Frossard, P. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2574–2582.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016. The Limitations of Deep Learning in Adversarial Settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, 372–387.

Petrillo, A.; Salvi, A.; Santini, S.; and Valente, A. S. 2018. Adaptive multi-agents synchronization for collaborative driving of autonomous vehicles with multiple communication delays. *Transportation Research Part C: Emerging Technologies*, 86: 372–392.

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4295–4304.

Samvelyan, M.; Rashid, T.; Schroeder de Witt, C.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2186–2188.

Shen, M.; Li, C.; Li, Q.; Lu, H.; Zhu, L.; and Xu, K. 2024. Transferability of White-box Perturbations: Query-Efficient Adversarial Attacks against Commercial DNN Services. In *33rd USENIX Security Symposium (USENIX Security 24)*, 2991–3008. Philadelphia, PA.

Song, J.; Ren, H.; Sadigh, D.; and Ermon, S. 2018. Multi-Agent Generative Adversarial Imitation Learning. In *Advances in Neural Information Processing Systems, NeurIPS*, volume 31. Curran Associates, Inc.

Sun, Y.; Zheng, R.; Liang, Y.; and Huang, F. 2022. Who Is the Strongest Enemy? Towards Optimal and Efficient Evasion Attacks in Deep RL. In *International Conference on Learning Representations*.

Warde-Farley, D.; and Goodfellow, I. 2016. Adversarial Perturbations of Deep Neural Networks. In *Perturbations, Optimization, and Statistics*. The MIT Press.

Wei, X.; Guo, Y.; Yu, J.; and Zhang, B. 2023. Simultaneously Optimizing Perturbations and Positions for Black-Box Adversarial Patch Attacks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7): 9041–9054.

Wen, M.; Kuba, J.; Lin, R.; Zhang, W.; Wen, Y.; Wang, J.; and Yang, Y. 2022. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. In *Advances in Neural Information Processing Systems, NeurIPS*, volume 35, 16509–16521.

Yeh, J.-C.; and Soo, V.-W. 2024. Toward Socially Friendly Autonomous Driving Using Multi-agent Deep Reinforcement Learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, 2573–2575.

Yuan, L.; Zhang, Z.; Xue, K.; Yin, H.; Chen, F.; Guan, C.; Li, L.; Qian, C.; and Yu, Y. 2023. Robust Multi-Agent Coordination via Evolutionary Generation of Auxiliary Adversarial Attackers. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI*, 11753–11762.

Zhang, G.; Ma, X.; Zhang, H.; Xiang, Z.; Ji, X.; Yang, Y.; Cheng, X.; and Hu, P. 2024a. LaserAdv: Laser Adversarial Attacks on Speech Recognition Systems. In *33rd USENIX Security Symposium (USENIX Security 24)*, 3945–3961. Philadelphia, PA.

Zhang, H.; Chen, H.; Boning, D. S.; and Hsieh, C.-J. 2021. Robust Reinforcement Learning on State Observations with Learned Optimal Adversary. In *International Conference on Learning Representations*.

Zhang, H.; Chen, H.; Xiao, C.; Li, B.; Liu, M.; Boning, D.; and Hsieh, C.-J. 2020. Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. In *Advances in Neural Information Processing Systems, NeurIPS*, volume 33, 21024–21037.

Zhang, J.; Hong, Y.; Cheng, D.; Zhang, L.; and Zhao, Q. 2025. Defending adversarial attacks in Graph Neural Networks via tensor enhancement. *Pattern Recognit.*, 158: 110954.

Zhang, L.; Li, L.; Wei, W.; Song, H.; Yang, Y.; and Liang, J. 2024b. Scalable Constrained Policy Optimization for Safe Multi-agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 37, 138698–138730.

Zhang, R.; Zong, Q.; Zhang, X.; Dou, L.; and Tian, B. 2023. Game of Drones: Multi-UAV Pursuit-Evasion Game With Online Motion Planning by Deep Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10): 7900–7909.

Zhou, Z.; Liu, G.; Guo, W.; and Zhou, M. 2024. Adversarial Attacks on Multiagent Deep Reinforcement Learning Models in Continuous Action Space. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(12): 7633–7646.