Article | Received 30 September 2025; Revised 17 December 2025; Accepted 23 December 2025; Published 12 January 2026
https://doi.org/10.55092/aias20260001

Beyond single-vector threats: perceptual and decisional joint attacks in multi-agent deep reinforcement learning

Weiqi Guo, Guanjun Liu* and Ziyuan Zhou

The Department of Computer Science, Tongji University, Shanghai 201804, China

* Correspondence author; E-mail: liuguanjun@tongji.edu.cn.

Highlights:

- We propose a general framework for joint perceptual-decisional adversarial attacks in MADRL.
- Extensive comparative experiments show joint attacks have a synergistic and more damaging effect.
- We demonstrate that existing single-vector defenses are insufficient against this joint threat.

Abstract: Multi-agent deep reinforcement learning (MADRL) has shown potential in solving complex cooperation and competition tasks among agents. Algorithms based on the Actor-Critic architecture have demonstrated excellent performance in continuous spaces. However, the robustness of such algorithms against adversarial attacks has not been thoroughly investigated. Existing studies primarily focus on either Perceptual attacks or Decisional attacks in isolation, without considering the impact of combining these two approaches. In this paper, we propose the Perceptual-Decisional Joint Attack (PDJA) framework that induces a strong synergistic disruption in MADRL systems. The framework executes a sequential, two-stage attack: (1) First, it perturbs the agent's perception by utilizing the actor's gradients. (2) Then, based on the agent's reaction to this perturbed state, it attacks the agent's decision by using the critic's Q-value to apply a final perturbation directly to the output action. We evaluate our attack framework in the Multi-Agent Particle Environment (MPE) and the Multi-Agent MuJoCo (MAMuJoCo), demonstrating the synergistic effect of disrupting agent perception and decision-making, and the limitations of current defense strategies in handling this joint attacks.

Keywords: multi-agent deep reinforcement learning; adversarial attack; adversarial robustness; perception-decision joint attack

1. Introduction

MADRL has been successfully applied to challenges such as virtual games [1,2] and physical tasks [3–6]. MADRL methods often employ the actor-critic network with the centralized training and decentralized execution (CTDE) framework, such as MADDPG [7], COMA [8], MAPPO [9], and FACMAC [10]. This paradigm is widely adopted for its effectiveness in handling high-dimensional continuous action

Copyright©2026 by the authors. Published by ELSP. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium provided the original work is properly cited.

spaces. However, MADRL algorithms are vulnerable to adversarial attacks because they are built on deep neural networks. Even small, unnoticeable changes to inputs can cause serious failures in cooperation and decision-making [11,12].

Existing studies on adversarial attacks in MADRL primarily focus on either the perceptual or the decisional dimension. In perceptual attacks, the adversary directly perturbs an agent's input state or observation, misleading the policy network into perceiving a distorted state [11,13,14]. In decisional attacks, the adversary's goal is to disrupt an agent's behavior by directly manipulating its output action [15,16] or by introducing adversarial agents into the environment [17,18]. While these single-vector studies provide valuable insights, they overlook a potential synergistic effects: vulnerabilities in perception and decision-making are not independent. Specifically, when an agent acts on a misinterpreted state, its decision space may shift toward a local optimum; a subsequent attack on this decision space can then exploit this vulnerability, causing the agent to commit actions significantly worse than those induced by a state-only attack. To our knowledge, no study has investigated attacks that target both an agent's perception and decisions simultaneously. This leaves a critical gap in our understanding of MADRL vulnerabilities.

To address this gap, we propose the PDJA framework. PDJA is a two-stage and gradient-based mechanism designed to uncover perception and decision-making's coupled vulnerabilities. In the first stage, it uses actor network to generate a perceptual perturbation. In the second stage, based on the perturbed perception, it leverages the critic network to craft a subsequent decisional perturbation. By designing a sequential mechanism to connect these two stages, PDJA ensures that the decisional perturbation exploits the weakness exposed by the perceptual perturbation. We evaluate PDJA in the MPE and MAMuJoCo. Our results show that PDJA creates a strong synergistic disruption, significantly outperforming single-vector attacks. Furthermore, we demonstrate that existing defenses focusing solely on state robustness (e.g., PAAD [19], ATLA [20]) or action robustness (e.g., M3DDPG [21]) are insufficient against this joint attack. Our contributions are summarized as follows:

- We propose PDJA, a generalized framework for constructing joint attacks that target both agent perception (via the actor network) and decision-making (via the critic network).
- We introduce a formal quantitative metric, the Synergy Ratio (ρ), to mathematically define and measure the synergistic effect of perceptual and decisional joint attacks.
- We conduct extensive experiments using two MADRL algorithms (FACMAC, MADDPG) and three defense baselines (PAAD, ATLA, M3DDPG). The results demonstrate that PDJA significantly degrades team rewards and effectively bypasses single-vector defenses. Visualization analysis further reveals that PDJA's effectiveness stems from its ability to exploit the critic's gradient sensitivity and induce large action deviations.

To facilitate a clear understanding of the technical terms, algorithms and environmental scenarios discussed in this study, a comprehensive list of abbreviations is provided in Table 1. The remainder of this paper is organized as follows: Section 2 reviews related work on adversarial attacks and defenses in MADRL. Section 3 introduces the mathematical preliminaries. Section 4 formally defines and details our proposed PDJA framework, and introduces the synergy metric. Section 5 presents the experimental setup and analyses the results. Finally, Section 6 concludes the paper.

Table 1. List of abbreviations.

Abbreviation	Description
MADRL	Multi-Agent Deep Reinforcement Learning
CTDE	Centralized Training with Decentralized Execution
MDP	Markov Decision Process
MG	Markov Game
Dec-POMDP	Decentralized Partially Observable Markov Decision Process
PDJA	Perceptual-Decisional Joint Attack
FGSM	Fast Gradient Sign Method
PGD	Projected Gradient Descent
SGLD	Stochastic Gradient Langevin Dynamics
RN	Random Noise
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
FACMAC	Factored Multi-Agent Centralized Policy Gradients
PAAD	Policy Adversarial Actor-Director
ATLA	Alternating Training with Learned Adversary
M3DDPG	Minimax Multi-Agent Deep Deterministic Policy Gradient
3a	MPE Scenario with 3 agents and 1 prey
6a	MPE Scenario with 6 agents and 2 preys
2-Humanoid	MAMuJoCo Scenario with 2-Agent Humanoid
4-Ant	MAMuJoCo Scenario with 4-Agent Ant
3a-FACMAC	E.g., Environment is 3a, trained with the FACMAC algorithm
KDE	Kernel Density Estimation

2. Related work

2.1. Adversarial attacks in RL

The vulnerability of Deep Learning models was exposed by gradient-based methods like FGSM [22], PGD [23], SGLD [24] and JSMA [25], which utilize white-box gradients to craft perturbations. Huang *et al.* [12] and Tu *et al.* [26] extended these techniques to the RL domain, proving they can effectively degrade learned policies. Our work intentionally builds upon foundational white-box methods (FGSM, PGD, SGLD) and simple black-box baselines (Random Noise). We adopt these fundamental methods rather than state-of-the-art white-box or black-box attackers for three critical reasons:

(1) Avoid Confounding Factors: Simple white-box methods can ensure that any performance degradation is attributable to the joint nature of the attack and the synergy between stages, rather than the sophistication of the gradient optimizer itself.

(2) Worst-Case Guarantee: White-box attacks mathematically approximate the upper bound of vulnerability. If a defense mechanism can withstand gradients, it provides a strong theoretical guarantee against weaker, heuristic, or black-box attacks.

(3) Necessary Tool for Defense: Robust learning paradigms, such as Adversarial Training (e.g., ATLA, PAAD), fundamentally require an efficient white-box optimizer in their inner loop. Without this rigorous foundation, one cannot train truly robust agents.

In Multi-Agent Deep Reinforcement Learning, due to agent inter-dependencies, the mechanisms and

impacts of attacks become substantially more intricate. Most existing studies focus on either perceptual attacks or decisional attacks. Perceptual attacks focus on perturbing the observation to mislead the actor network's state interpretation [11,13,14,26]. In decisional attacks, the adversary's goal is to disrupt an agent's decision-making process, forcing it to execute suboptimal actions , thereby indirectly affecting the next state through environmental interaction. It can be achieved in two ways: either by manipulating an agent's output action [15,16], or by adding adversarial agents into the environment [17,18]. However, these methodologies maximize damage in either the state space or the action space independently, overlooking the latent coupled vulnerability in the Actor-Critic pipeline. Our work identifies this gap and positions PDJA as a systematic framework to bridge these dimensions, demonstrating that a coordinated perceptual-decisional attack yields a synergistic effect. To our knowledge, MARLSafe [27], which is the only related work involving multidimensional perturbations, fails to explore their interactions.

Significant effort within these single-vector studies focus on sub-problems such as whom to attack or when to attack. For instance, Zhou *et al.* [13] proposed mechanisms to select critical victim agents, and Hu and Zhang [15] designed sparse attacks to perturb actions only at specific timesteps. In contrast, PDJA employs a random victim selection strategy and executes attacks at every timestep. This design choice is deliberate, for we can ensure that the observed performance degradation is attributable to the synergy between state and action perturbations and evaluate the system's worst-case robustness under sustained pressure.

2.2. Defense mechanisms in multi-agent RL

Defense strategies primarily facilitate the generation of perturbations in adversarial training to uncover potential model risks. Broadly, these defenses focus on three main dimensions: state robustness [19,20,28,29], action robustness [15,16,21,30], and reward robustness [31]. In this work, we concentrate on state and action defenses to align with our perceptual-decisional joint threat model. Specifically, we select PAAD [19] and ATLA [20] as baselines for state robustness, and adopt M3DDPG [21] for action robustness. PAAD uses a "Director-Actor" mechanism to guide agents to learn to defend against worst-case perturbations in the observation space. ATLA utilizes an alternating training scheme, where an adversary is trained to generate adversarial perturbations. M3DDPG introduces a minimax optimization objective, training agents under the assumption that their counterparts will take adversarial actions. We choose these methods because they directly fortify the state and action channels targeted by PDJA. If PDJA can bypass these specialized defenses, it can prove that defending single vectors in isolation is insufficient against a joint attack.

3. Preliminaries

In this section, we establish the necessary theoretical preliminaries for multi-agent deep reinforcement learning and adversarial attacks. The mathematical symbols and notations used for formulating the problem and the proposed joint attack framework are summarized in Table 2.

Table 2. List of notations.

Symbol	Description
N	Number of agents
S, s	Global state space and a specific state
A_i, a_i	Action space and a specific action for agent i
$\mathbf{a}$	Joint action vector $[a_1, a_2, \ldots, a_N]$
Ω_i, o_i	Observation space and local observation for agent i
V, m	Set of victim agents and the number of victims
R, r	Reward function and a specific reward value
γ	Discount factor
$\mu_i(\cdot;\theta_i)$	Deterministic policy (Actor) of agent i with parameters θ_i
Q_i, Q_{tot}	Individual and centralized total action-value functions (Critic)
ϕ	Parameters of the critic and mixing networks
D	Experience replay buffer
$o_{i,\text{adv}}, a_{i,\text{adv}}$	Adversarially perturbed observation and action
$a_{i,\text{mid}}$	Intermediate action generated from perturbed observation
δ_s, δ_a	Perturbation vectors for state and action spaces
$\varepsilon_s, \varepsilon_a$	Maximum perturbation budgets for state and action
α	Step size for iterative gradient-based attacks
$L_{\text{state}}, L_{\text{action}}$	Loss functions for perceptual and decisional attacks
$\Pi_{x,\varepsilon}$	Projection operator onto the ε-ball around x
$\Delta(\cdot)$	Attack Damage (reduction in reward)
ρ	Synergy Ratio quantifying the effect of joint attacks

3.1. Reinforcement learning formalisms

A single-agent reinforcement learning problem is typically modeled as a Markov Decision Process (MDP), which is defined as $\langle S, A, P, R, \gamma \rangle$, where S is the state space, A is the action space, $P : S \times A \to S$ is the state transition probability function, $R : S \times A \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. Multi-agent systems can be modeled based on the agents' observability. Our work considers two primary formalisms: Markov Game (MG) [32] and Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [33].

An MG is a direct extension of the MDP. It is defined by the tuple $\langle S, \{A_i\}_{i=1}^N, P, \{R_i\}_{i=1}^N, \gamma \rangle$, where N is the number of agents, S is the global state space, A_i is the action space for agent i, $P : S \times A \to S$ is the state transition function for the joint action $A = \times_{i=1}^N A_i$, and $R_i : S \times A \to \mathbb{R}$ is the reward function for agent i. All agents in an MG have full observability of the global state $s \in S$.

A Dec-POMDP provides a more realistic model where agents operate with limited observation. It is defined by the tuple $\langle S, \{A_i\}_{i=1}^N, P, \{R_i\}_{i=1}^N, \{\Omega_i\}_{i=1}^N, O, \gamma \rangle$, where Ω_i is the observation space for agent i and $O : S \times A \to \Omega$ is the observation function for the joint observation $\Omega = \times_{i=1}^N \Omega_i$. Instead of the global state S, each agent receives a local observation $o_i \in \Omega_i$. Agents in a Dec-POMDP must make decisions based on this partial information.

3.2. Multi-agent actor-critic

The Actor-Critic architecture is a classic method in MADRL. We focus on algorithms that utilize CTDE, where a centralized critic is used during training to guide the decentralized actors during execution.

MADDPG is a policy-based algorithm for MADRL in continuous spaces. It assigns each agent i a centralized action-value function $Q_i(s, a_1, a_2, \ldots, a_N; \phi_i)$, where s represents global state information (e.g., $s = (o_1, o_2, \ldots, o_N)$). The critic network updates by minimizing the following loss function:

$$\begin{aligned} L(\phi_i) &= \mathbb{E}_{s,\mathbf{a},r,s'}\left[(y - Q_i(s, a_1, \ldots, a_N; \phi_i))^2\right], \\ y &= r_i + \gamma Q_i'(s', a_1', \ldots, a_N'; \phi_i')|_{a_k' = \mu_k'(o_k')} \end{aligned} \tag{1}$$

where s' is the next global state information, Q_i' is the target critic network of agent i, and $a_k' = \mu_k'(o_k')$ is the action selected by the agent k's target actor network μ_k'.

FACMAC integrates the actor-critic architecture with the QMIX value decomposition technique. Based on MADDPG, it introduces a centralized total critic Q_{tot} to combine information from all agents:

$$Q_{\text{tot}}(s, \mathbf{a}; \phi) = g_{\text{mix}}(s, Q_1(o_1, a_1; \phi_1), \ldots, Q_N(o_N, a_N; \phi_N); \phi_{\text{mix}}) \tag{2}$$

where $Q_i(o_i, a_i; \phi_i)$ is the agent i's local Q-value, g_{mix} is a non-linear network mixing all agents' local Q-values, and $\mathbf{a} = \{a_1, \ldots, a_N\}$ is the joint action. $\phi = \{\phi_1, \ldots, \phi_N, \phi_{\text{mix}}\}$ denotes the parameters of the entire critic network, which is updated by minimizing the loss:

$$\begin{aligned} L(\phi) &= \mathbb{E}_{(s,\mathbf{a},r,s')\sim D}\left[(y_{\text{tot}} - Q_{\text{tot}}(s, \mathbf{a}; \phi))^2\right], \\ y_{\text{tot}} &= r + \gamma Q_{\text{tot}}'(s', \mathbf{a}'; \phi') \end{aligned} \tag{3}$$

The actor network of each agent is updated through deterministic policy gradients:

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{s,\mathbf{a}\sim D}[\nabla_{\theta_i} \mu_i(o_i) \nabla_{a_i} Q_{\text{tot}}(s, \mathbf{a}; \phi)|_{a_k = \mu_k(o_k)}] \tag{4}$$

This structure allows each agent to consider the impact of its own action on the whole team's performance.

3.3. Foundational perturbation generation algorithms

Since we focus on studying the synergistic effects of the proposed joint attacks, we construct the attack modules using several basic methods. Let x be the clean input vector (representing either observation o or action a), x_{adv} be the perturbed input vector, L be the loss function, and ε be the maximum perturbation magnitude (i.e., the L_∞ norm constraint).

Random Noise (RN) is a non-gradient and black-box method. The perturbation δ is sampled from a uniform distribution $U[-\varepsilon, \varepsilon]$, and x_{adv} is calculated as:

$$x_{\text{adv}} = x + U[-\varepsilon, \varepsilon] \tag{5}$$

Fast Gradient Sign Method (FGSM) is an efficient, single-step and white-box method. It first calculates the gradient of the input vector through the loss function $L(x)$, then takes the sign of the gradient and multiplies it by ε, and adds the resulting perturbation to the input vector. It is calculated as:

$$x_{\text{adv}} = x + \varepsilon \cdot \text{sign}(\nabla_x L(x)) \tag{6}$$

Projected Gradient Descent (PGD) is an iterative extension of FGSM. For a total of K iterations, the gradient sign is multiplied by a small step size α (such as $\alpha = \varepsilon/K$), which is added to the current perturbed input vector x_k to obtain x_{k+1}. Then project x_{k+1} into the L_∞ norm space of x by the clip operation $\Pi_{x,\varepsilon}$.

$$\begin{aligned} x_0 &= x + U[-\varepsilon, \varepsilon] \\ x_{k+1} &= \Pi_{x,\varepsilon}\left(x_k + \alpha \cdot \text{sign}(\nabla_{x_k} L(x_k))\right) \end{aligned} \tag{7}$$

Stochastic Gradient Langevin Dynamics (SGLD) is a method that introduces stochasticity into PGD. At each step, Gaussian noise $N(0, \sigma^2)$ is added to the update, which helps to find more diverse and effective adversarial examples. The update rule is:

$$x_{k+1} = \Pi_{x,\varepsilon}\left(x_k + \alpha \cdot \text{sign}(\nabla_{x_k} L(x_k)) + N(0, \sigma^2)\right) \tag{8}$$

4. Methodology

In this section, we propose the PDJA framework. We first define the threat model. Then we propose the PDJA framework, detailing how different attack algorithms (white-box and black-box) are integrated. Finally, we introduce a quantitative metric to measure the synergistic effect of PDJA.

4.1. Threat model

The PDJA framework supports both White-box attacks (utilizing gradients from Actor and Critic networks) and Black-box attacks (which only access input observations and output actions). To clearly define the adversary, we establish a threat model encompassing four key attributes: Knowledge, Capabilities, Frequency, and Victim Selection. The specific assumptions and constraints for each attribute are summarized in Table 3. In our experiments, we employ foundational gradient-based methods (FGSM, PGD, SGLD) to represent white-box capabilities and Random Noise (RN) as a representative black-box baseline, as defined in Section 3.3.

Table 3. Definition of the threat model in PDJA, categorizing the adversary's Knowledge (White-box *vs.* Black-box) and specifying general constraints on Capabilities, Frequency, and Victim Selection strategy.

Attribute	Description
Knowledge	White-box: Full access to all internal model components, including Actor (θ) and Critic (ϕ) parameters, gradients (∇_θ, ∇_a), and specific architectural modules (e.g., Mixing networks in FACMAC). Black-box: Access restricted strictly to the agents' observations (o_t) and output actions (a_t) at each timestep; no access to network parameters, gradients, or intermediate computations.
Capabilities	Inject perturbations to local observations (o_i) and output actions (a_i), constrained by L_∞-norm budgets ε_s and ε_a, respectively.
Frequency	Attacks are executed at every timestep t during the evaluation episode.
Victim Selection	At each timestep, a subset of m agents ($V \subset \{1, \ldots, N\}$) is randomly selected as victims.

While the white-box assumption imposes stringent requirements, it represents a threat that remains plausible in real-world scenarios. Consider a cooperative drone system, where each drone's state includes its position and velocity, and its action is a vector controlling rotor thrust. The swarm's goal is to maintain formation while tracking a target. A white-box attack could occur if a third-party vendor with access to the control software leaks the deployed model parameters. An attacker could then launch a joint attack:

first, they inject a minor error into a drone's GPS data (perceptual attack), making it believe it has drifted. While this alone might only cause a minor self-correction, our joint attack follows up with a calculated perturbation to the drone's motor commands (decisional attack). The real-world consequence is that the drone overshoots violently, risking a mid-air collision. Therefore, analyzing white-box scenario is not merely a theoretical exercise but a necessary step in auditing the security of high-stakes.

4.2. The PDJA framework

PDJA is a two-stage, sequential adversarial mechanism designed to uncover the coupled vulnerabilities in MADRL systems. In the first stage, the attacker perturbs the observation received by the agentt; subsequently, in the second stage, it manipulates the agent's actions generated based on this compromised inputs. The first stage targets the agent's actor network while the second stage focuses on the critic network.

Stage 1: Perceptual Perturbation

The goal of the first stage is to mislead the agent's policy network μ_i. For a selected victim agent i, we introduce a perturbation δ_s to its local observation o_i. The perturbed observation $o_{i,\text{adv}}$ is generated as:

$$o_{i,\text{adv}} = o_i + \delta_s, \qquad \|\delta_s\|_\infty \leq \varepsilon_s \tag{9}$$

The generation of δ_s depends on the attack type. For gradient-based methods (FGSM, PGD, SGLD), the attacker aims to maximize the deviation of the agent's output action. We define the perceptual loss function L_{state} as the negative squared Euclidean distance between the action under clean observation and perturbed observation:

$$L_{\text{state}}(o_i) = -\|\mu_i(o_i + \delta_s; \theta_i) - \mu_i(o_i; \theta_i)\|_2^2 \tag{10}$$

The perturbation δ_s is updated by ascending the gradient $\nabla_{o_i} L_{\text{state}}$. Taking PGD as a representative white-box method, the iterative update rule is formally defined as:

$$\delta_s \leftarrow \Pi_{\varepsilon_s}(\delta_s + \alpha \cdot \text{sign}(\nabla_{o_i} L_{\text{state}})) \tag{11}$$

For Black-box method RN, the attacker simply samples the perturbation from a uniform distribution: $\delta_s \sim U[-\varepsilon_s, \varepsilon_s]$. This stage results in an intermediate perturbed action $a_{i,\text{mid}} = \mu_i(o_{i,\text{adv}}; \theta_i)$, which carries the error propagated from the perceptual distortion.

Stage 2: Decisional Perturbation

The attacker observes the intermediate action $a_{i,\text{mid}}$ and the current global state information. The goal is to apply a further perturbation δ_a directly to the action space to minimize the team's expected return. The final malicious action $a_{i,\text{adv}}$ is:

$$a_{i,\text{adv}} = a_{i,\text{mid}} + \delta_a, \qquad \|\delta_a\|_\infty \leq \varepsilon_a \tag{12}$$

The generation of δ_a also depends on the attack type. For gradient-based methods, the attacker utilizes the Critic network Q to guide the attack and seeks to minimize the Q-value. The decisional loss function L_{action} is formulated based on the specific Critic architecture:

$$L_{\text{action}} = \begin{cases} -Q_i(s, a_1, \ldots, a_{i,\text{mid}}, \ldots, a_N; \phi_i), & \text{for MADDPG (Individual Critic)} \\ -Q_{\text{tot}}(s, a_1, \ldots, a_{i,\text{mid}}, \ldots, a_N; \phi), & \text{for FACMAC (Mixing Critic)} \end{cases} \tag{13}$$

For other MADRL algorithms, the definition of L_{action} is similar, provided that the gradients of the Q-function output can be backpropagated to the input action. Similar to the state phase, for PGD-based attacks, the action perturbation is updated via:

$$\delta_a \leftarrow \Pi_{\varepsilon_a}(\delta_a + \alpha \cdot \text{sign}(\nabla_{a_{i,\text{mid}}} L_{\text{action}})) \tag{14}$$

For Black-box method RN, similar to Stage 1, δ_a is sampled from $U[-\varepsilon_a, \varepsilon_a]$.

The overall workflow of the framework is detailed in Algorithm 1 and visualised in Figure 1. At each timestep, the attacker selects a subset of m agents from the N agents as victims, denoted by the set V. The observations of these victims $\{o_i\}_{i\in V}$ are first processed by the Perception Perturbator, generating perturbed observations $\{o_{i,\text{adv}}\}_{i\in V}$. Subsequently, the victim agents compute intermediate actions $\{a_{i,\text{mid}}\}_{i\in V}$ based on $\{o_{i,\text{adv}}\}_{i\in V}$. Then the Decision Perturbator yields the final malicious actions $\{a_{i,\text{adv}}\}_{i\in V}$. The final joint action submitted to the environment consists of the perturbed actions $\{a_{i,\text{adv}}\}_{i\in V}$ for victims and unaltered original actions $\{a_j\}_{j\notin V}$ for non-victim agents.

Algorithm 1 The Perceptual-Decisional Joint Attack (PDJA) Framework

Input: Number of agents N; Number of victims m; Joint observation $\mathbf{o}$; Global state s; Actor networks $\{\mu_i\}$; Critic network Q; Budgets $\varepsilon_s, \varepsilon_a$.
Output: Final joint action $\mathbf{a}_{\text{adv}}$.

Randomly select a set V of m victim indices from $\{1, \ldots, N\}$.
Initialize perturbed joint observation $\mathbf{o}_{\text{adv}} \leftarrow \mathbf{o}$.
// Stage 1: Perceptual Perturbation
for each victim index $i \in V$ **do**
    Generate state perturbation δ_s via white-box optimization (e.g., Equation (11)) or black-box sampling.
    $o_{i,\text{adv}} \leftarrow \Pi_{\varepsilon_s}(o_i + \delta_s)$. // Apply perturbation and clip
end for
Compute intermediate joint action $\mathbf{a}_{\text{mid}}$ where $a_{i,\text{mid}} = \mu_i(o_{i,\text{adv}})$ for all i.
Initialize final joint action $\mathbf{a}_{\text{adv}} \leftarrow \mathbf{a}_{\text{mid}}$.
// Stage 2: Decisional Perturbation
for each victim index $i \in V$ **do**
    Generate action perturbation δ_a via white-box optimization (e.g., Equation (14)) or black-box sampling.
    $a_{i,\text{adv}} \leftarrow \Pi_{\varepsilon_a}(a_{i,\text{mid}} + \delta_a)$. // Apply perturbation and clip
end for
return Final joint action $\mathbf{a}_{\text{adv}}$.

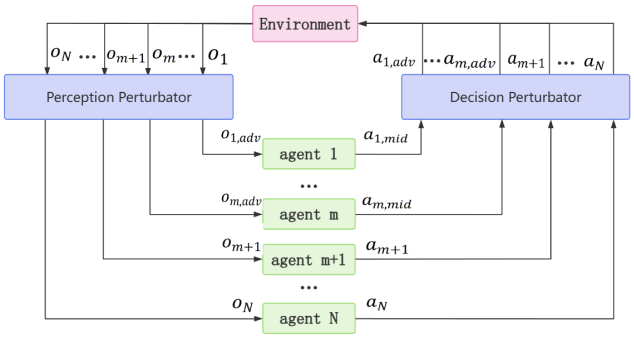

Figure 1. Two-stage workflow of the PDJA Framework. For the victim set V, clean observations $\{o_i\}_{i\in V}$ are perturbed into $\{o_{i,\text{adv}}\}_{i\in V}$ (Stage 1), leading to intermediate actions $\{a_{i,\text{mid}}\}_{i\in V}$ which are subsequently perturbed into final malicious actions $\{a_{i,\text{adv}}\}_{i\in V}$ (Stage 2).

4.3. Quantitative metric: synergy ratio

To formally quantify the synergistic effectiveness of joint attacks, we introduce the Synergy Ratio (ρ). This metric assesses whether the combined attack yields a performance degradation that exceeds the capability of its strongest individual component.

Let R_{clean} be the mean cumulative reward in a clean environment. Let R_{state} and R_{action} denote the rewards under perceptual-only and decisional-only attacks, respectively, and R_{joint} be the reward under the joint attack. We define the Attack Damage (Δ) as the reduction in reward: $\Delta(\cdot) = R_{\text{clean}} - R_{(\cdot)}$. The Synergy Ratio is defined as:

$$\rho = \frac{\Delta(\text{Joint})}{\max(\Delta(\text{State}), \Delta(\text{Action}))} = \frac{R_{\text{clean}} - R_{\text{joint}}}{\max(R_{\text{clean}} - R_{\text{state}}, R_{\text{clean}} - R_{\text{action}})} \tag{15}$$

This ratio allows us to categorize the interaction between perceptual and decisional attacks into three distinct modes, providing a clear mathematical standard for interpreting the experimental results. The interpretation of different ρ values is summarized in Table 4.

Table 4. Interpretation of the Synergy Ratio (ρ).

Value Range	Interaction Type	Interpretation
$\rho > 1$	Positive Synergy	The joint attack causes more damage than the most effective single-vector attack, indicating a mutually reinforcing mechanism.
$\rho \approx 1$	Redundancy	The joint attack is dominated by one vector, offering no significant additional benefit over the single strongest attack.
$\rho < 1$	Interference	The perturbations in state and action spaces hinder each other, reducing overall effectiveness.

5. Experiments

5.1. Experimental setup

5.1.1. Environments

In test the effectiveness and generalization of PDJA, we select four cooperative tasks from two widely used multi-agent environment: MPE [7] and MAMuJoCo [10]. MPE is a multi-agent environment that supports communication, cooperation, competition, predation and other scenarios. We selected two tasks from the predator-prey scenario: three agents and one prey (3a scenario) , and six agents and two preys (6a scenario). MAMujoco is an extended multi-agent environment based on Mujoco, supporting body dynamics simulations and articulation controls. We selected two tasks (2-Agent Humanoid and 4-Agent Ant) from this framework. To facilitate subsequent presentation of experimental results, we abbreviate these four scenarios as 3a, 6a, 2-Humanoid, and 4-Ant, respectively. We choose them to ensure our evaluation covers a diverse range of agent counts, task complexities, and state-action space structures. Key details are summarized in Table 5.

Table 5. Details of the experimental environments. "Obs. Dim." and "Action Dim." refer to the dimensions and value ranges for a single agent.

Environment	Scenario	# Agents	Obs. Dim. (Range)	Action Dim. (Range)
MPE	3a	3	16 ([−1, 1])	2 ([−1, 1])
MPE	6a	6	30 ([−1, 1])	2 ([−1, 1])
MAMuJoCo	2-Humanoid	2	18 ([−1, 1])	9 ([−0.4, 0.4])
MAMuJoCo	4-Ant	4	7 ([−1, 1])	2 ([−0.4, 0.4])

5.1.2. Benchmark vanilla models

Since our PDJA framework relies on both the actor and critic networks, we select target models that exemplify the Actor-Critic architecture across different MADRL paradigms. We conduct experiments using MADDPG (policy-based paradigm) and FACMAC (value-decomposition paradigm), trained across all four scenarios (3a, 6a, 2-Humanoid, 4-Ant).

In MADDPG, the actor and critic learning rates are set to 0.01 and 0.05, respectively. The actor network is a Recurrent Neural Network with a Gated Recurrent Unit cell and a hidden dimension of 64, processing local observations. The centralized critic is a feed-forward MLP (two hidden layers of size 64) that takes the global state and joint actions as input. To prevent overfitting, we apply a gradient norm clip of 0.5 and a weight decay of 1×10^{-4} to the critic optimizer. In FACMAC, the learning rates are configured as 0.05 for actors and 0.01 for the critic and mixer networks. The actor architecture mirrors that of MADDPG. FACMAC employs individual critics for each agent (3-layer MLP, 64 hidden units), which output local Q-values. These are aggregated into a global Q_{tot} via a QMixer network with an embedding dimension of 64 and a state-conditioned hypernetwork, enabling efficient factorization of the joint value function. Both MADDPG and FACMAC utilize the Adam optimizer and maintain a replay buffer capacity of 5,000. Exploration is facilitated by adding Gaussian noise with a standard deviation of 0.1 to the actions. The discount factor γ is set to 0.85, and soft target updates are applied with $\tau = 0.001$.

5.1.3. Benchmark defended models

We also evaluate PDJA against models trained with various defense mechanisms. Specifically, we select PAAD [19] and ATLA [20] as baselines for state robustness, and M3DDPG [21] for action robustness. For PAAD and ATLA, we implement both methods strictly following the methodology and hyperparameters detailed in [14]. For M3DDPG, We adapted it to our centralized critic framework via a key modification: during the critic update process, a single adversarial perturbation δ_a is applied to the joint target action a' with a budget $\varepsilon_{\text{adv}} = 0.001$, instead of perturbing each agent's target action individually.

We restrict our defense experiments to the MPE 3a and 6a environments. Preliminary experiments indicated that PAAD and ATLA failed to converge in the complex MAMuJoCo tasks, resulting in negative rewards. This failure is likely due to suboptimal hyperparameter configurations inherent to these algorithms, rather than errors in our implementation. To ensure a fair comparison based on defended models, we exclude the 2-Humanoid and 4-Ant scenarios from this subsection.

Benchmark attack baselines: To the best of our knowledge, PDJA is the first joint perceptual-decisional attack framework for MADRL. There are no existing state-action joint attack methods available for direct

external comparison. Consequently, our evaluation strategy focuses on validating the synergy of PDJA itself. To this end, we design two classes of baselines:

Ablation Baselines: These baselines utilize the exact same optimization techniques (e.g., PGD) as PDJA but are restricted to a single attack vector. This ablation ensures that the performance difference is attributable to the synergy of the attack rather than the optimizer.

- State-Only Attack (Ablation of Stage 2): The attacker targets the agent's observation using the actor network but leaves the output action unperturbed ($\varepsilon_a = 0$).
- Action-Only Attack (Ablation of Stage 1): The attacker accepts the clean observation as input ($\varepsilon_s = 0$) but manipulates the output action using the critic network to minimize the Q-value.

Internal Variations: To investigate the impact of different optimization strategies within our framework, we instantiate the PDJA module with various foundational algorithms described in Section 3.3: RN, FGSM, PGD, and SGLD. This allows us to compare the effectiveness of white-box gradient-based joint attacks against black-box random joint attacks.

5.1.4. Attack hyperparameter settings

Optimization Strategies: We instantiate the attack modules using RN, FGSM, PGD, and SGLD. For iterative methods PGD and SGLD, we set the number of iterations $K = 20$ and the step size $\alpha = \varepsilon/K$. For the core effectiveness analysis (Section 5.2, we adopt PGD as the standard white-box optimizer for both stages due to its stability and strength. For the broad analysis of attack configurations (Section 5.3), we test all 16 combinations (e.g., FGSM for state and PGD for action).

Perturbation Budgets: For the fixed-budget attack experiments (Section 5.2), in the 2-Humanoid and 4-Ant scenarios of MAMuJoCo, the state budget ε_s is set to 0.5 while the action budget ε_a is set to 0.08. In the 3a and 6a scenarios of MPE, ε_s is set to 0.02 and ε_a is set to 0.2. For the variable-budget attack experiments (Section 5.4), we set ε_a to range from 5% to 30% of the action space's range, and ε_s from 0.5% to 3% of the state space's range.

Victim Selection: In the core effectiveness analysis (Section 5.2), the number of victim agents is fixed: we attack $m = 3$ agents in 3a and 6a, and $m = 2$ agents in 2-Humanoid and 4-Ant. In the impact analysis of attack configurations (Section 5.3), we vary the number of victims m from 1 to 3 to investigate the scalability of the attack.

5.1.5. Evaluation protocol

To ensure statistical reliability, we strictly follow a rigorous evaluation protocol. We perform $N_{\text{test}} = 20$ independent testing runs for each model-attack setting. In each run, the agents interact with the environment for 100 episodes, and the mean cumulative reward is recorded. We report the mean and standard deviation across these N_{test} runs. A lower mean reward indicates a more effective attack. Furthermore, we conduct Welch's t-test to verify statistical significance ($p < 0.05$) when comparing PDJA against baselines. We also compute the Synergy Ratio (ρ) defined in Section 4.3. A higher ρ (> 1) indicates a stronger synergistic effect, validating that the joint attack mechanism successfully amplifies the damage beyond simple summation.

5.2. *Evidence of synergy in joint attacks*

In this part, we present the core evaluation of PDJA against both vanilla and defended models. The detailed results are summarized in Table 6 (Vanilla Models) and Table 7 (Defended Models). We report the mean cumulative reward over $N_{\text{test}} = 20$ runs, along with the Synergy Ratio (ρ) to quantify the joint effect. Statistical significance is verified via Welch's t-test ($p < 0.05$) and marked with asterisks ($*$).

Synergistic Vulnerability: A key finding from Table 6 is that PDJA consistently achieves the lowest reward in 6 out of 8 settings, with Synergy Ratios (ρ) exceeding 1.0 in most cases. For instance, in the 6a environment, PDJA achieves a remarkable $\rho = 1.75$ against MADDPG, reducing the reward to 30.92. This degradation is statistically significant ($p < 0.05$) compared to both State-Only (34.96) and Action-Only (32.20) attacks. These results confirm our central hypothesis: combining perceptual and decisional perturbations creates a synergistic destructive effect that is more damaging than attacking either vector in isolation.

Impact of Model Training Quality: The results in Table 6 also reveal the complexity of interaction between attack stages. In the 4-Ant environment, for MADDPG, the Action-Only attack is devastating (reward drops to -197.97), whereas adding state perturbation in PDJA actually mitigates this damage (reward recovers to 60.57). We found that its baseline reward (816.55) is lower than the reward achieved by taking no action (approximately 1000), indicating that the policy was not effectively trained. In such cases where the actor and critic networks are inaccurate, their gradients are unreliable, directly explaining the negative synergy. We hypothesize that the initial state perturbation, guided by the unreliable gradients of a poorly trained actor, shifted the agent into a policy region where the critic's gradient for the subsequent action attack became counterproductive. This finding highlights a crucial insight: the effectiveness of gradient-based joint attacks is contingent on whether the target model is well-trained. A well-trained model provides reliable gradients, allowing joint attacks to be more effective. Conversely, for a poorly trained model, a joint attack may be less effective. However, a poorly trained model already diminishes the practical necessity of launching a effective attack against it in the first place.

Bypassing Single-Vector Defenses: The core purpose of any defense, whether state-based (PAAD, ATLA) or action-based (M3DDPG), is to ensure that perturbed inputs ultimately lead to high-value actions. State defenses aim to map perturbed observations back to high-value actions, while action defenses aim to maintain a flat, high-value Q-landscape around the clean action. However, these defenses are optimized to handle single-vector perturbations and thus have a limited robust operational range. The results in Table 7 demonstrates that PDJA can effectively pushes the agents beyond their robust boundaries. For state-defended models like MADDPG+PAAD in the 3a environment, the State-Only attack is largely nullified (reward of 138.16 *vs.* clean 135.77). However, PDJA forces its final action into a low-value region that the state defense was not trained to handle, causing the reward to collapse to 101.52 with a synergy ratio of $\rho = 1.33$. More critically, even against action-robust M3DDPG, PDJA remains highly effective in the 3a environment ($\rho = 1.15$). PDJA first uses the state perturbation to shift the agent's policy, creating a larger and more vulnerable attack surface for the subsequent action attack. It then generates a final action that is far more deviant than the action defense is capable of handling, thus overwhelming the defense. This confirms that PDJA's ability to break through defensive boundaries is significantly stronger than that of single-vector attacks.

In summary, the statistical evidence and high Synergy Ratios strongly support the superiority of PDJA. It not only amplifies damage through synergy in vanilla models but also explores the robust boundary of defenses to trigger system-wide failure. Our results also highlight the critical need for joint defense mechanisms in building secure multi-agent systems.

Table 6. Comparison of mean cumulative rewards ($\pm$ Std) under different attacks on vanilla models over $N_{\text{test}} = 20$ independent testing runs. The symbol $*$ indicates that the reward difference between the specific baseline method (No Attack, State-Only, or Action-Only) and PDJA is statistically significant ($p < 0.05$, Welch's t-test). ρ denotes the Synergy Ratio. Bold values indicate the attack method that achieves the lowest reward (most effective attack).

Environment	Algorithm	No Attack	State-Only	Action-Only	PDJA	ρ
3a	FACMAC	$190.54 \pm 9.90^*$	$185.74 \pm 6.82^*$	$132.41 \pm 6.40^*$	$\mathbf{125.44 \pm 9.16}$	1.12
	MADDPG	$131.58 \pm 7.71^*$	$137.47 \pm 7.71^*$	107.56 ± 6.64	$\mathbf{104.32 \pm 5.78}$	1.14
6a	FACMAC	$308.70 \pm 10.34^*$	$305.53 \pm 11.37^*$	$265.43 \pm 8.20^*$	$\mathbf{260.02 \pm 7.46}$	1.12
	MADDPG	$33.91 \pm 2.81^*$	$34.96 \pm 3.06^*$	32.20 ± 2.72	$\mathbf{30.92 \pm 3.58}$	1.75
4-Ant	FACMAC	$1321.63 \pm 85.05^*$	$1253.42 \pm 62.37^*$	$\mathbf{497.86 \pm 27.92}^*$	642.77 ± 24.66	0.82
	MADDPG	$816.55 \pm 36.99^*$	$814.84 \pm 22.30^*$	$\mathbf{-197.97 \pm 14.06}^*$	60.57 ± 10.96	0.75
2-Humanoid	FACMAC	$580.03 \pm 8.19^*$	$429.97 \pm 6.02^*$	$556.14 \pm 7.13^*$	$\mathbf{425.48 \pm 7.33}$	1.03
	MADDPG	$517.95 \pm 14.16^*$	$309.77 \pm 8.78^*$	$421.83 \pm 10.47^*$	$\mathbf{239.70 \pm 10.35}$	1.34

Table 7. Comparison of mean cumulative rewards ($\pm$ Std) under different attacks on defended models over $N_{\text{test}} = 20$ independent testing runs. Other notations and significance markers follow the same convention as in Table 6.

Environment	Defended Model	No Attack	State-Only	Action-Only	PDJA	ρ
3a	MADDPG + ATLA	$123.96 \pm 7.23^*$	$118.16 \pm 8.59^*$	$91.99 \pm 7.20^*$	$\mathbf{86.18 \pm 4.28}$	1.18
	MADDPG + PAAD	$135.77 \pm 7.43^*$	$138.16 \pm 8.12^*$	$110.10 \pm 5.68^*$	$\mathbf{101.52 \pm 5.98}$	1.33
	MADDPG + M3DDPG	$134.89 \pm 6.29^*$	$134.38 \pm 7.38^*$	$107.25 \pm 5.91^*$	$\mathbf{103.16 \pm 5.78}$	1.15
	FACMAC + ATLA	$148.52 \pm 6.07^*$	$138.17 \pm 8.54^*$	$114.84 \pm 6.93^*$	$\mathbf{108.37 \pm 5.68}$	1.19
	FACMAC + PAAD	$160.34 \pm 8.45^*$	$151.01 \pm 8.11^*$	$122.35 \pm 5.63^*$	$\mathbf{116.36 \pm 5.05}$	1.16
6a	MADDPG + ATLA	48.88 ± 3.50	$51.05 \pm 3.50^*$	47.23 ± 3.88	$\mathbf{47.13 \pm 3.47}$	1.06
	MADDPG + PAAD	$46.50 \pm 4.27^*$	44.24 ± 4.08	44.13 ± 3.92	$\mathbf{43.10 \pm 3.42}$	1.43
	MADDPG + M3DDPG	38.09 ± 3.98	37.96 ± 3.39	$\mathbf{36.97 \pm 2.62}$	37.29 ± 3.27	0.72
	FACMAC + ATLA	$311.87 \pm 11.44^*$	$308.30 \pm 7.88^*$	274.50 ± 9.10	$\mathbf{269.48 \pm 8.04}$	1.13
	FACMAC + PAAD	$304.35 \pm 14.47^*$	$308.27 \pm 8.29^*$	$280.67 \pm 10.72^*$	$\mathbf{273.67 \pm 8.86}$	1.30

5.3. Impact of attack configurations

To provide a more granular analysis of joint attacks, we investigate how the choice of optimization strategies (RN,FGSM,PGD,SGLD) and the number of victim agents (m) influence the overall effectiveness. We use the well-trained 3a-FACMAC model for this experiment. The detailed results are presented in Table 8, with a visual summary in Figure 2.

Scalability with Number of Victims: Figure 2 clearly illustrates that for nearly all attack configurations, the reward degradation scales with the number of attacked agents (m). The green solid lines (PDJA) consistently trend downwards as m increases from 1 to 3, demonstrating that PDJA effectively leverages a larger attack surface. Notably, the slope of degradation for joint attacks is often steeper than that of single-vector attacks (blue and orange lines), suggesting that the synergistic effect becomes more pronounced as more agents are compromised.

Context-Dependency of Optimal Attack Algorithms: While PDJA demonstrates general effectiveness, Table 8 reveals that the optimal combination of foundational algorithms is highly context-dependent. A key observation is that theoretically simpler or less computationally intensive methods can sometimes outperform more complex ones. For instance, when attacking three agents ($m = 3$), the combination of SGLD (State) and PGD (Action) yields the lowest reward of 118.50. This aligns with the expectation that iterative, gradient-based methods are powerful. However, when attacking only two agents ($m = 2$), the combination of RN (State) and FGSM (Action) achieves a competitive low reward of 141.76. This suggests that under certain conditions, a simpler joint attack can create a higher disruptive pattern.

This experiment provides a crucial insight: there is no universally optimal joint attack configuration. The most effective combination depends on the environment, the target model and the number of compromised agents. It highlights the complexity of multi-agent vulnerabilities and the need for adaptive defense strategies that can anticipate a wide range of attack patterns.

Table 8. Comparison of mean cumulative rewards ($\pm$ Std) on the 3a-FACMAC vanilla model under different joint attack configurations. Each result is averaged over $N_{\text{test}} = 20$ independent testing runs. Bold values highlight the most effective attack configuration (lowest reward).

		Decisional Attack				
Perceptual Attack	***m***	**No Attack**	**RN**	**FGSM**	**PGD**	**SGLD**
No Attack	1	190.54 ± 9.90	187.59 ± 5.91	179.80 ± 8.05	171.31 ± 10.01	174.12 ± 7.45
	2		182.28 ± 8.78	157.60 ± 6.61	156.92 ± 7.30	155.67 ± 8.65
	3		181.43 ± 7.09	131.91 ± 6.65	132.41 ± 6.40	147.28 ± 6.95
RN	1	191.30 ± 10.68	192.58 ± 7.93	173.88 ± 10.93	165.36 ± 8.93	181.23 ± 9.02
	2	190.60 ± 8.68	178.93 ± 8.09	$\mathbf{141.76 \pm 6.97}$	153.33 ± 8.72	157.63 ± 7.16
	3	192.14 ± 6.79	178.59 ± 8.24	128.75 ± 5.56	136.25 ± 8.77	140.84 ± 6.34
FGSM	1	185.88 ± 8.31	176.15 ± 7.38	$\mathbf{161.64 \pm 7.71}$	163.53 ± 8.68	168.55 ± 6.91
	2	178.57 ± 7.71	178.80 ± 6.48	148.33 ± 6.92	146.88 ± 6.49	155.40 ± 5.96
	3	170.18 ± 7.35	172.86 ± 9.04	128.31 ± 6.19	126.45 ± 7.48	142.20 ± 6.47
PGD	1	190.21 ± 8.87	181.05 ± 9.17	169.12 ± 6.04	164.56 ± 6.94	174.22 ± 9.31
	2	182.36 ± 6.97	186.00 ± 9.24	147.10 ± 6.76	151.62 ± 4.77	158.21 ± 8.21
	3	185.74 ± 6.82	173.53 ± 8.87	128.70 ± 4.51	125.44 ± 9.16	138.99 ± 8.23
SGLD	1	180.75 ± 7.00	189.45 ± 5.93	168.48 ± 7.65	166.35 ± 7.00	171.17 ± 7.65
	2	178.93 ± 8.15	187.40 ± 8.10	147.96 ± 9.18	142.05 ± 7.36	152.38 ± 7.76
	3	172.85 ± 7.81	167.46 ± 8.11	128.34 ± 6.98	$\mathbf{118.50 \pm 6.29}$	144.46 ± 5.81

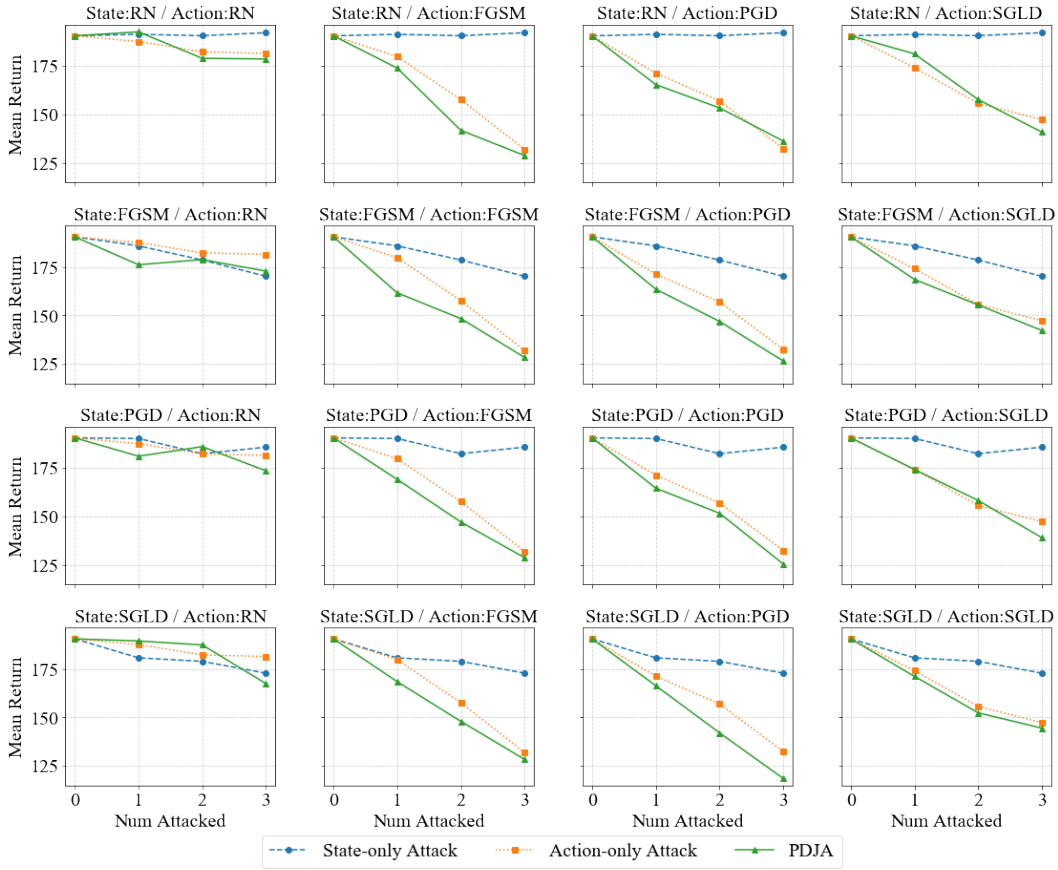

Figure 2. Visual comparison of attack effectiveness on the 3a-FACMAC. Each subplots corresponds to a joint attack combination (Perceptual, Decisional). Green solid line represents joint attack. Blue dashed line represents perceptual-only attack. Orange dotted line represents decisional-only attack. The number of victim agents increases from 0 to 3.

5.4. Quantifying synergy with varying attack budgets

While Section 5.2confirmed the existence of synergistic effects, this section aims to quantify how perceptual budgets (ε_s) and decisional budgets (ε_a) jointly affect agent performance. We selected the MPE 3a and 6a environments for this analysis, both utilizing the FACMAC algorithm. We employed the combination of SGLD(State) and PGD(Action) for 3a and PGD(State) and SGLS(Action) for 6a, as these were identified as highly effective in our prior experiments. We generated 36 distinct budget combinations by varying ε_s across $\{0, 0.005, 0.01, 0.015, 0.02, 0.03\}$ and ε_a across $\{0, 0.05, 0.1, 0.15, 0.2, 0.3\}$. The detailed results are provided in Table 9 and rendered as a 3D surface plot in Figure 3.

As detailed in Table 9, in the majority of the 25 joint attack combinations, the mean cumulative reward was lower than that of the corresponding state-only or action-only attacks. The most potent attacks consistently occur at high values of both ε_s and ε_a. In Figure 3, the blue dashed lines (state-only attacks, where $\varepsilon_a = 0$) and red dotted lines (action-only attacks, where $\varepsilon_s = 0$) show a clear downward trend in rewards as their respective budgets increase. The gray mesh, representing the joint attack space, slopes downward not just along the axes but most steeply where both ε_s and ε_a are at their maximum. This trend is consistent in both the 3a and 6a environments. This budget-varying experiment provides clear evidence for our hypothesis that the vulnerabilities in an agent's perception and decision-making processes are fundamentally coupled.

Table 9. Mean cumulative reward (± Std) under varying state (ε_s) and action (ε_a) attack budgets. Each result is averaged Bold values highlight the budget setting that results in the lowest reward.

		ε_a					
Environment	ε_s	**0**	**0.05**	**0.1**	**0.15**	**0.2**	**0.3**
3a-FACMAC	0	190.54 ± 9.90	180.87 ± 8.65	168.29 ± 8.42	157.29 ± 5.41	133.06 ± 6.13	93.71 ± 8.14
	0.005	189.75 ± 10.28	191.16 ± 6.93	172.49 ± 7.05	150.25 ± 7.89	136.46 ± 7.31	103.41 ± 6.08
	0.010	180.75 ± 8.34	181.91 ± 8.65	160.49 ± 8.78	147.94 ± 5.83	125.35 ± 6.89	103.05 ± 6.85
	0.015	182.23 ± 6.41	170.99 ± 8.41	150.94 ± 7.83	146.81 ± 7.60	124.53 ± 6.80	99.92 ± 6.71
	0.020	180.65 ± 6.29	171.44 ± 6.37	151.56 ± 8.47	148.62 ± 6.05	127.07 ± 8.13	101.41 ± 4.59
	0.030	163.56 ± 8.63	159.51 ± 7.65	145.32 ± 6.40	126.97 ± 4.95	116.57 ± 5.20	**86.03 ± 5.18**
6a-FACMAC	0	308.70 ± 10.34	302.73 ± 11.02	297.57 ± 8.08	283.17 ± 8.11	277.04 ± 9.19	249.64 ± 5.12
	0.005	314.04 ± 9.04	303.62 ± 9.54	298.94 ± 8.38	275.92 ± 8.90	264.00 ± 5.91	246.22 ± 8.96
	0.010	306.52 ± 9.57	306.62 ± 11.30	289.88 ± 8.00	279.89 ± 9.28	275.75 ± 6.05	244.65 ± 8.91
	0.015	313.35 ± 9.39	307.17 ± 10.26	289.80 ± 10.02	278.32 ± 7.43	259.10 ± 6.52	248.23 ± 8.73
	0.020	307.90 ± 8.90	294.98 ± 7.08	284.53 ± 9.61	283.56 ± 9.62	268.73 ± 7.40	**233.65 ± 7.19**
	0.030	299.82 ± 8.78	289.11 ± 9.12	274.96 ± 7.50	269.17 ± 10.01	264.44 ± 9.85	234.86 ± 8.92

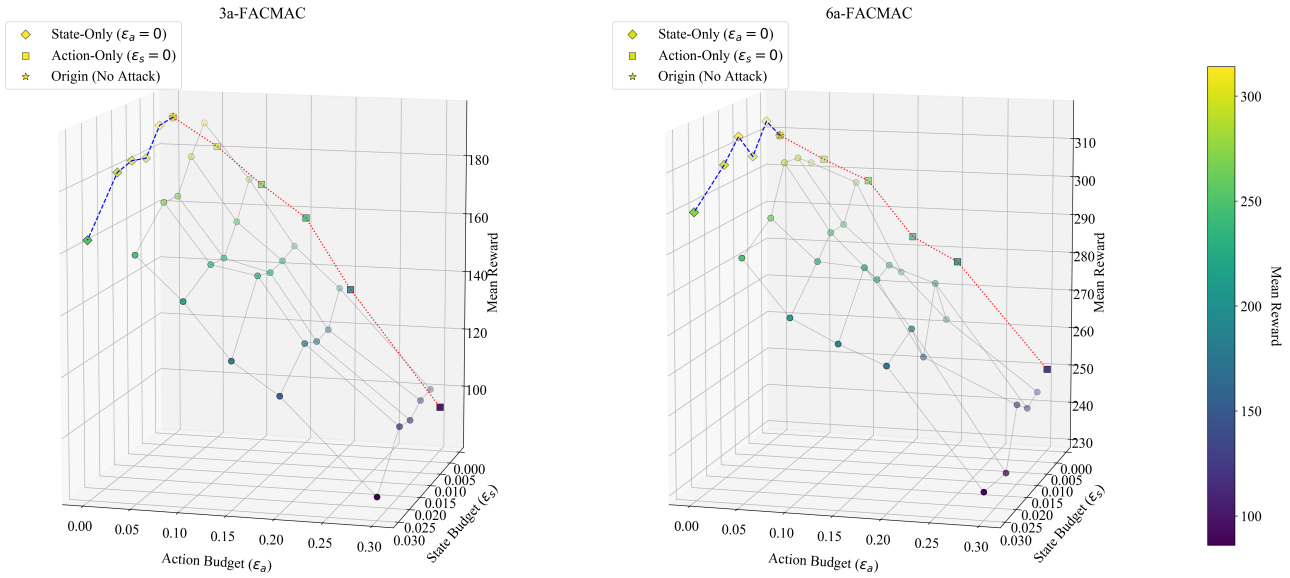

Figure 3. Mean cumulative reward under different ε_s and ε_a settings for the 3a-FACMAC model (left) and the 6a-FACMAC model (right). The blue dashed line represents State-Only attacks ($\varepsilon_a = 0$), while the red dotted line represents Action-Only attacks ($\varepsilon_s = 0$). The gray mesh illustrates the joint attack space.

5.5. Mechanism analysis via Q-value vsualization

To further investigate the mechanism behind PDJA's synergistic effect, we analyze how different attacks impact the critic's value estimation. We conducted experiments on four vanilla models (3a-FACMAC, 6a-FACMAC, 3a-MADDPG, and 6a-MADDPG). For each model, we evaluated four attack configurations (No Attack, State-Only, Action-Only, and PDJA), all instantiated with PGD. We ran 500 test episodes for each setting, with each episode lasting about 25 timesteps. In MADDPG, we utilize the output from the single centralized critic Q, while in FACMAC, the factored joint action-value Q_{tot} is used. We randomly

collected 10,000 such Q-value samples per configuration to calculate the Kernel Density Estimation (KDE) distributions and their corresponding means, as visualized in Figure 4.

The results in Figure 4 reveal two distinct patterns. For 3a-FACMAC and 3a-MADDPG, the Q-value distribution under PDJA is shifted to the left, with its mean being the lowest. This aligns with the reward degradation results in Table 6, indicating that PDJA effectively guides agents into regions that the critic correctly identifies as having low long-term value. However, an interesting phenomenon occurs in 6a-FACMAC and 6a-MADDPG. Despite PDJA being the most effective attack in terms of reward in Table 6, its mean Q-value is not the lowest. This seemingly counter-intuitive result highlights that gradient-based attacks are driven by the gradient of the value function (∇Q), not its absolute magnitude. In environments like 6a, the Q-value landscape is extremely flat and concentrated within a narrow range (40-45 for 6a-FACMAC and 15-18 for 6a-MADDPG). Even a minuscule difference in Q-value (e.g., 0.01) can correspond to a steep gradient. In environments with a flat or narrow Q-value landscape, a small Q-value difference can still produce a large gradient. PDJA may exploit such gradients to find a "deceptive cliff": a state-action pair whose immediate Q-value is not low, but which leads to a catastrophic cascade of future states that the single-step critic fails to foresee. This confirms that PDJA's strength lies in leveraging the critic's gradient sensitivity, even when the absolute Q-values themselves do not appear pessimistic.

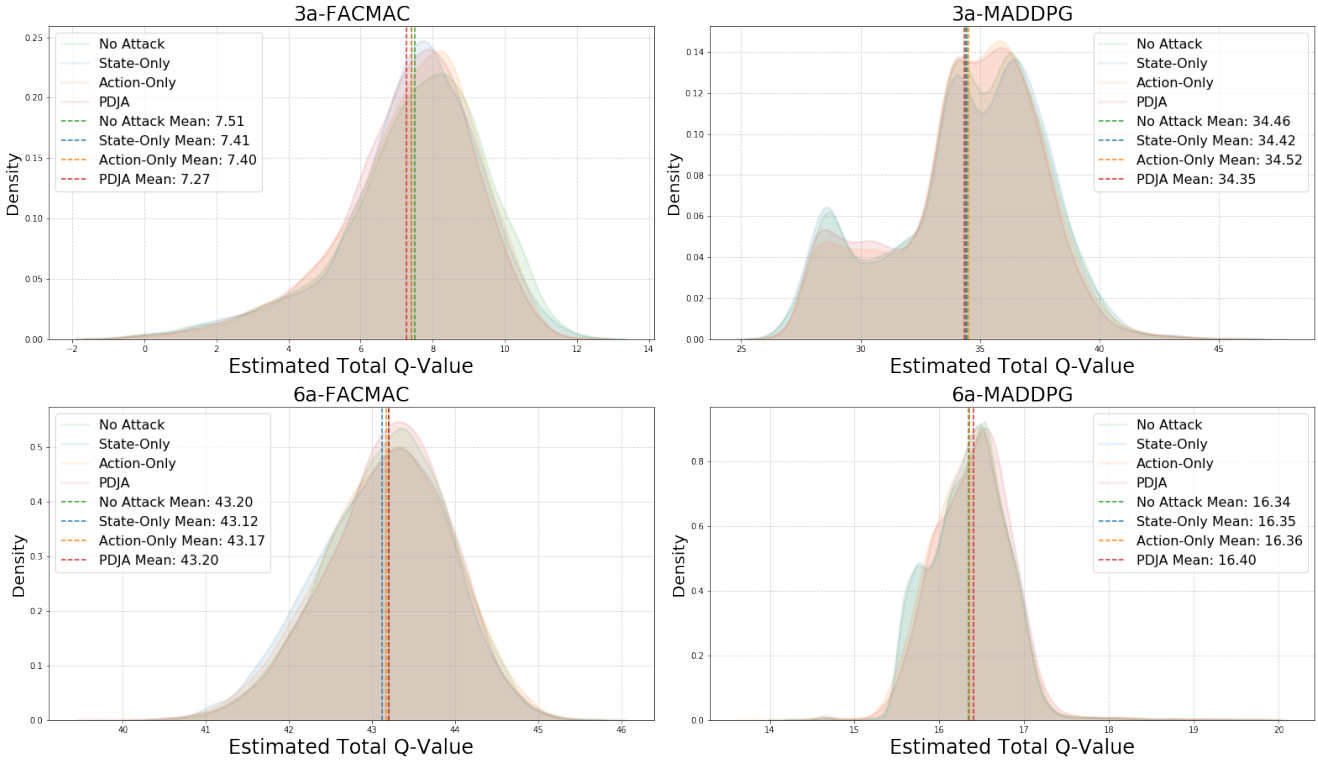

Figure 4. Distribution of total Q-values under different attack configurations. Each subplot corresponds to a specific model: **(a)** 3a-FACMAC; **(b)** 3a-MADDPG; **(c)** 6a-FACMAC; **(d)** 6a-MADDPG. The distributions for No Attack, State-Only, Action-Only, and PDJA are represented by green, blue, orange and red curves. The dashed vertical lines of the corresponding color indicate the mean of each distribution.

5.6. *Mechanism analysis via action deviation*

The results from Table 6 show a strong link between attack effectiveness and action deviation. To further analyze the attack mechanism, we now visualize the distribution of these deviations. We use the same four vanilla models and attack configurations as in the Q-value analysis. For each victim agent i in each timestep, we compute the L2 norm distance between its final executed action $a_{i,\text{adv}}$ and its original clean action a_i: $\|a_{i,\text{adv}} - a_i\|_2$. We collected 10,000 such distance samples for each configuration to plot the frequency distributions and their means, as shown in Figure 5.

The results in Figure 5 provide compelling evidence for the superiority of PDJA. Across all four models, a clear and consistent ordering of mean action deviation is observed: State-Only < Action-Only < PDJA. For instance, in 3a-MADDPG, the mean deviation for PDJA (0.38) is significantly larger than that for Action-Only (0.27) and State-Only (0.21). This is directly correlated with the degradation of the rewards in Table 6, which confirms that a higher deviation of action is a strong indicator of a more effective attack. This visualization also illuminates the mechanism behind the synergistic effect: While a State-Only attack can occasionally cause large action deviations, its distribution is heavily skewed towards zero. PDJA first effectively perturbs the state, unlocking a more vulnerable action landscape. Then it allows the subsequent action attack to generate a final action with a much larger deviation.

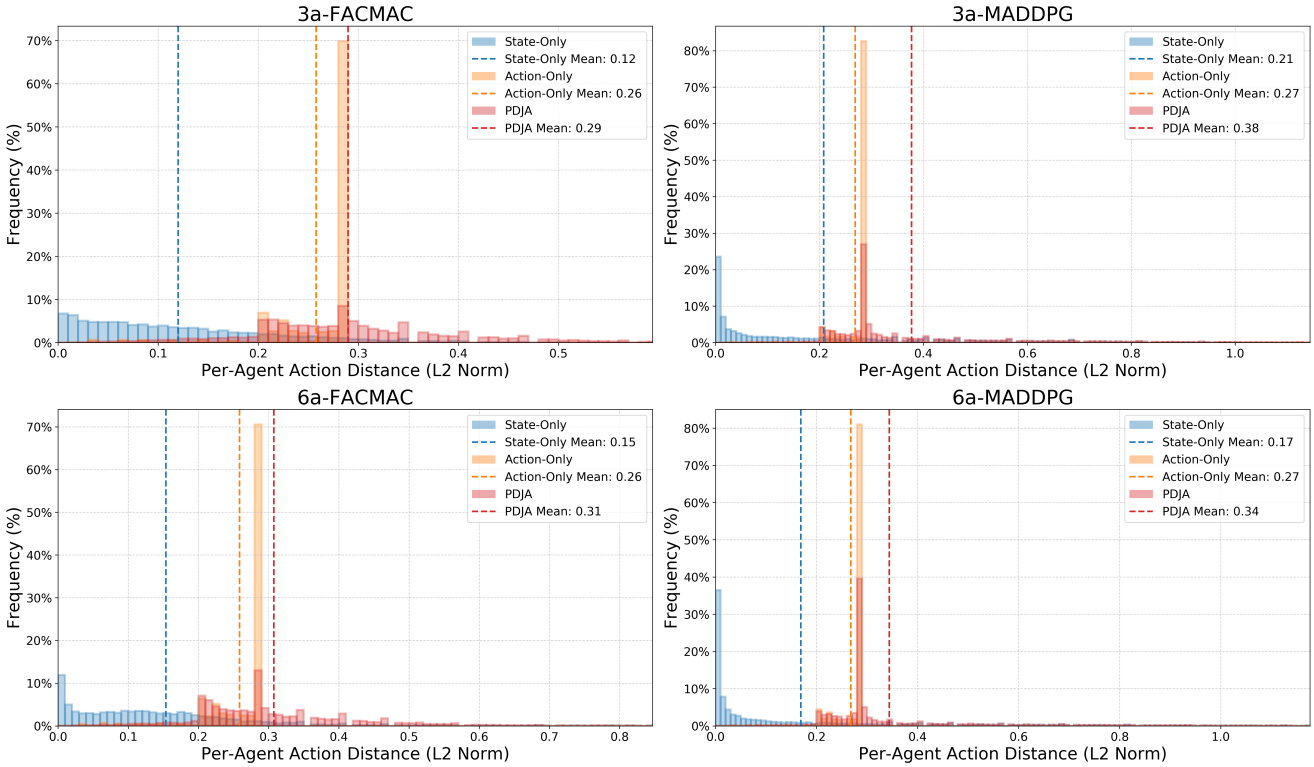

Figure 5. Distribution of per-agent action distance (L2 Norm) under different attack configurations. The histograms for State-Only, Action-Only, and PDJA are represented by blue, orange, and red, respectively. The dashed vertical lines of the corresponding color indicate the mean deviation for each attack type.

6. Conclusion

In this paper, we address a research gap in the MADRL system by proposing the PDJA framework. We demonstrated that simultaneously perturbing an agent's perception and decision-making using gradients from both the actor and critic networks can induce a powerful synergistic effect. Through the experiments on algorithms (FACMAC, MADDPG) and environments (MPE, MAMuJoCo), we quantitatively validated this synergy using our proposed Synergy Ratio (ρ) and showed that PDJA outperforms single-vector attacks. Furthermore, existing defenses like PAAD, ATLA and M3DDPG, which focus solely on either the state or action space in isolation, can be systematically bypassed by our joint attack.

However, PDJA operates under a white-box assumption, requiring full access to the target model's parameters and gradients. While this is essential for establishing theoretical upper bounds on vulnerability and developing robust defenses, its practical deployment by external attackers is constrained to scenarios involving explicit model access, such as insider threats or leaked model weights. Furthermore, our analysis revealed that the efficacy of PDJA is contingent on the availability of high-quality gradients from a well-trained model. PDJA can exhibit negative synergy against poorly converged policies, where the unreliable gradients from a suboptimal critic may guide the attack in a counterproductive direction.

The vulnerabilities exposed by PDJA also highlight potential societal risks if MADRL systems are deployed in autonomous vehicle coordination or drone swarms. A sophisticated attacker could leverage similar joint attack techniques to trigger system-wide failures with minimal, hard-to-detect perturbations. Therefore, identifying these risks proactively is a prerequisite for deploying safe and trustworthy multi-agent systems. Our work serves not as a guide for malicious actors, but as a stress test to expose blind spots in current defenses.

Future work should focus on two primary directions. The first is the development of black-box joint attacks, which would more closely mimic real-world external threats. Second, our findings strongly motivate research into joint defense mechanisms capable of protecting the entire perception-to-decision pipeline, rather than just isolated components. Ultimately, this work serves as a foundational step toward building more secure and reliable multi-agent systems.

Data availability statement

The data supporting the findings of this work were generated through training and evaluation processes. The source code is available in the GitHub repository: https://github.com/wwweeeiiiqqqiii/PDJA.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 92582104.

Authors' contribution

Conceptualization, Ziyuan Zhou and Guanjun Liu; methodology, Weiqi Guo, Ziyuan Zhou and Guanjun Liu; software, Weiqi Guo; validation, Ziyuan Zhou and Guanjun Liu; formal analysis, Weiqi Guo; investigation, Weiqi Guo; writing—original draft preparation, Weiqi Guo; writing—review and editing, Ziyuan Zhou and

Guanjun Liu; visualization, Weiqi Guo; supervision, Guanjun Liu and Ziyuan Zhou; project administration, Guanjun Liu. All authors have read and agreed to the published version of the manuscript.

Conflicts of interests

The authors declare no conflict of interest.

References

[1] Vinyals O, Babuschkin I, Chung J, Mathieu M, Jaderberg M, *et al.* Alphastar: mastering the real-time strategy game starcraft ii. 2019. Available: https://deepmind.google/blog/alphastar-mastering-the-real-time-strategy-game-starcraft-ii/ (accessed on 30 December 2025).

[2] Berner C, Brockman G, Chan B, Cheung V, Debiak P, *et al.* Dota 2 with large scale deep reinforcement learning. *arXiv* 2019, arXiv:1912.06680.

[3] Chen Y, Liu M, Everett M, How JP. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, Singapore, May 27–June 3, 2017, pp. 285–292.

[4] Lin K, Zhao R, Xu Z, Zhou J. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, London, UK, August 19–23, 2018, pp. 1774–1783.

[5] Zhang W, Liu H, Xiong H, Xu T, Wang F, *et al.* RLCharge: imitative multi-agent spatiotemporal reinforcement learning for electric vehicle charging station recommendation. *IEEE Trans. Knowl. Data Eng.* 2022, 35(6):6290–6304.

[6] Chen S, Liu G, Zhou Z, Zhang K, Wang J. Robust multi-agent reinforcement learning method based on adversarial domain randomization for real-world dual-uav cooperation. *IEEE Trans. Intell. Veh.* 2023, 9(1):1615–1627.

[7] Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, *et al.* Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, Long Beach, USA, December 4–9, 2017, p. 30.

[8] Foerster J, Farquhar G, Afouras T, Nardelli N, Whiteson S. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, USA, February 2–7, 2018.

[9] Yu C, Velu A, Vinitsky E, Gao J, Wang Y, *et al.* The surprising effectiveness of ppo in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, New Orleans, USA, November 28–December 9, 2022, pp. 24611–24624.

[10] Peng B, Rashid T, Schroeder de Witt C, Kamienny PA, Torr P, *et al.* Facmac: factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems*, Online, December 6–14, 2021, pp. 12208–12221.

[11] Lin J, Dzeparoska K, Zhang SQ, Leon-Garcia A, Papernot N. On the robustness of cooperative multi-agent reinforcement learning. In *2020 IEEE Security and Privacy Workshops (SPW)*, San Francisco, USA, May 18–21, 2020, pp. 62–68.

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

[13] Zhou Z, Liu G. Robustness testing for multi-agent reinforcement learning: state perturbations on critical agents. In *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI)*, Kraków, Poland, September 30–October 4, 2023, pp. 3131–3139.

[14] Zhou Z, Liu G, Guo W, Zhou M. Adversarial attacks on multiagent deep reinforcement learning models in continuous action space. *IEEE Trans. Syst. Man Cybern.: Syst.* 2024, 54(12):7633–7646.

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

[16] Sun C, Kim DK, How JP. ROMAX: certifiably robust deep multiagent reinforcement learning via convex relaxation. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, Philadelphia, USA, May 23–27, 2022, pp. 5503–5510.

[17] Gleave A, Dennis M, Wild C, Kant N, Levine S, *et al.* Adversarial policies: attacking deep reinforcement learning. *arXiv* 2019, arXiv:1905.10615.

[18] Li S, Guo J, Xiu J, Zheng Y, Feng P, *et al.* Attacking cooperative multi-agent reinforcement learning by adversarial minority influence. *Neural Networks* 2025, 191:107747.

[19] Sun Y, Zheng R, Liang Y, Huang F. Who is the strongest enemy? Towards optimal and efficient evasion attacks in deep RL. *arXiv* 2021, arXiv:2106.05087.

[20] Zhang H, Chen H, Boning D, Hsieh CJ. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv* 2021, arXiv:2101.08452.

[21] Li S, Wu Y, Cui X, Dong H, Fang F, *et al.* Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, USA, January 27–February 1, 2019, pp. 4213–4220.

[22] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. *arXiv* 2014, arXiv:1412.6572.

[23] Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards deep learning models resistant to adversarial attacks. *arXiv* 2017, arXiv:1706.06083.

[24] Welling M, Teh YW. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, USA, June 28–July 2, 2011, pp. 681–688.

[25] Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, *et al.* The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, Saarbrücken, Germany, March 21–24, 2016, pp. 372–387.

[26] Tu J, Wang T, Wang J, Manivasagam S, Ren M, *et al.* Adversarial attacks on multi-agent communication. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Montreal, Canada, October 11–17, 2021, pp. 7768–7777.

[27] Guo J, Chen Y, Hao Y, Yin Z, Yu Y, *et al.* 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)*, New Orleans, USA, June 18–24, 2022, pp. 115–122.

[28] Zhou Z, Liu G, Zhou M. A robust mean-field actor-critic reinforcement learning against adversarial perturbations on agent states. *IEEE Trans. Neural Networks Learn. Syst.* 2023, 35(10):14370–14381.

[29] Han S, Su S, He S, Han S, Yang H, *et al.* What is the solution for state-adversarial multi-agent reinforcement learning? *arXiv* 2022, arXiv:2212.02705.

[30] Tessler C, Efroni Y, Mannor S. Action robust reinforcement learning and applications in continuous control. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, USA, June 10–15, 2019, pp. 6215–6224.

[31] Phan T, Belzner L, Gabor T, Sedlmeier A, Ritz F, *et al.* Resilient multi-agent reinforcement learning with adversarial value decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Online, February 2–9, 2021, pp. 11308–11316.

[32] Shapley LS. Stochastic games. *Proc. Natl. Acad. Sci. U.S.A.* 1953, 39(10):1095–1100.

[33] Bernstein DS, Givan R, Immerman N, Zilberstein S. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* 2002, 27(4):819–840.