A Comprehensive Evaluation Framework for Multi-Agent Reinforcement Learning

Zonglei Jing*
SKLCCSE, Beihang University
Beijing, China
raykr@buaa.edu.cn

Xiaojun Chang*
Mingxuan Zhu*
ace_chang.cc@foxmail.com
clr_zmx@126.com
China Academy of Aerospace Science and Innovation
Beijing, China

Simin Li
SKLCCSE, Beihang University
Beijing, China
lisiminsimon@buaa.edu.cn

Aishan Liu
SKLCCSE, Beihang University
Beijing, China
liuaishan@buaa.edu.cn

Xiaoqian Li
Taishan University
Taian, China
xiaoqianli0324@126.com

Xianglong Liu†
SKLCCSE, Beihang University
Beijing, China
xlliu@buaa.edu.cn

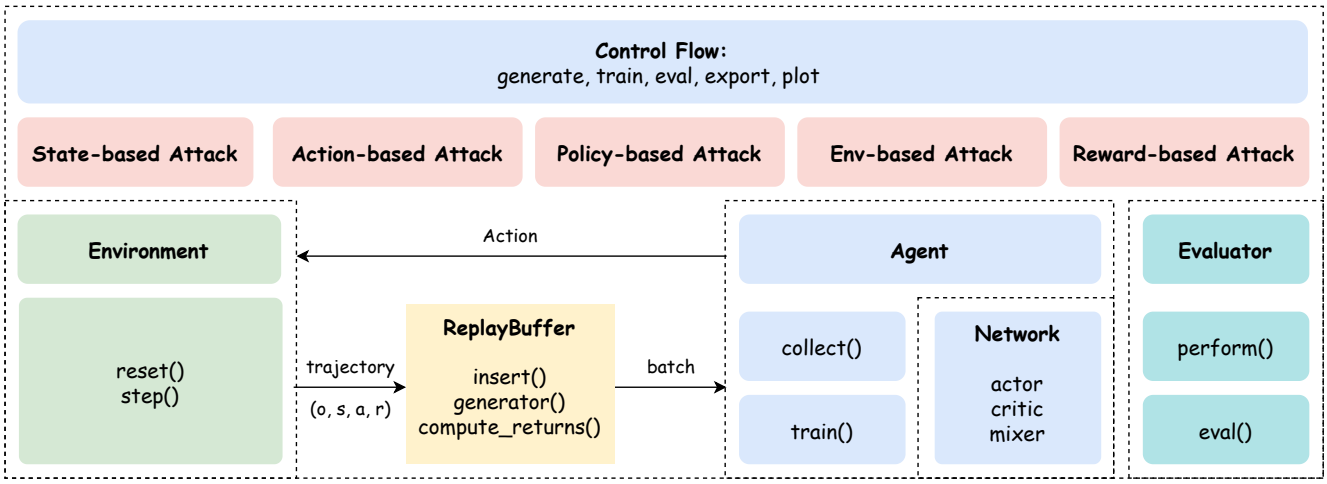

Figure 1: Overview of the architecture of the proposed platform.

ABSTRACT

In recent years, multi-agent reinforcement learning (MARL) has been increasingly applied to swarm intelligence, offering effective solutions for complex decision-making in multi-agent systems. However, MARL models remain vulnerable to imperceptible perturbations, which can lead to erroneous decisions and potentially cause system disruptions or failures. While existing research on adversarial attacks predominantly focuses on traditional supervised learning and single-agent reinforcement learning, the extension to multi-agent scenarios has been largely overlooked. Furthermore, a comprehensive platform for evaluating MARL robustness is lacking. To address this gap, we propose an integrated framework for robust evaluation of MARL. The platform integrates over six classical MARL algorithms, spanning both on-policy and off-policy architectures. It supports more than ten interactive environments across a diverse range of application domains, including gaming, sports, robotics, autonomous vehicles, drones, traffic signal control, and power systems. We have implemented over eight adversarial attack methods—targeting policies, states/observations, actions, rewards, and environments—covering all stages of the MDP closed-loop. Additionally, the platform includes more than five robustness evaluation metrics, addressing both self-model and inter-model perspectives. It supports automated batch experiments and allows for easy customization of agents and environments, enabling seamless integration for rapid MARL robustness research. Our experiments further validate the platform's efficacy in evaluating and improving MARL robustness.

CCS CONCEPTS

• **Software and its engineering** → *Software safety*; • **Computer systems organization** → **Reliability**; • **General and reference** → **Metrics**; **Evaluation**.

*Both authors contributed equally to this research.
†Corresponding author.

KEYWORDS

multi-agent reinforcement learning, robustness evaluation, platform

ACM Reference Format:

Zonglei Jing, Xiaojun Chang, Mingxuan Zhu, Simin Li, Aishan Liu, Xiaoqian Li, and Xianglong Liu. 2024. A Comprehensive Evaluation Framework for Multi-Agent Reinforcement Learning. In *Proceedings of (The Sixth International Conference on Distributed Artificial Intelligence).* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

In recent years, multi-agent reinforcement learning (MARL) has gained widespread application in swarm intelligence, providing effective solutions for complex decision-making in multi-agent systems [13, 38, 39]. However, MARL models face significant robustness challenges, as they are susceptible to imperceptible perturbations that can lead to erroneous agent decisions, potentially resulting in system-wide disruptions or failures. Furthermore, these models are exposed to adversarial attacks at various stages of the Markov decision process (MDP) [26, 36, 42]. Despite the critical importance of addressing robustness in MARL, existing research on adversarial attacks has largely focused on single-agent environments, overlooking the distinct challenges posed by multi-agent systems.

When evaluating MARL algorithms, it is crucial to account for their distinct characteristics, such as non-unique perturbation targets, indirect attack objectives, and varying attacker perspectives. These complexities make it challenging to directly adapt well-established adversarial attack frameworks from other domains, such as AISafety [8] and RobustART [33]. The unique nature of multi-agent interactions requires tailored approaches, as traditional adversarial systems may not fully address the intricacies of MARL environments.

Recently, the field of MARL has seen the development of several high-quality benchmark frameworks. The PyMARL project [30] introduced the SMAC [30] benchmark suite, offering a modular framework for evaluating various MARL algorithms. Building on this, the EPyMARL project [27] extended its scope by incorporating additional algorithms and environments. Yu et al. [37] subsequently provided a comprehensive evaluation of MAPPO across four benchmark environments, highlighting its performance in multi-agent settings. Researchers also developed the AdaptAUG framework [40], an adaptive system that applies data augmentations to enhance the sample efficiency and performance of MARL algorithms. Additionally, Hu et al. introduced MARLlib [11], a framework built on RLlib [21], supporting a wide range of environments and algorithms for efficient MARL benchmarking. However, these frameworks primarily focus on performance evaluation and do not address robustness assessment. Developing a comprehensive robustness evaluation platform for multi-agent reinforcement learning is therefore essential, providing crucial technical support for both research and industry applications.

In this work, we introduce a comprehensive model robustness evaluation framework for multi-agent reinforcement learning, featuring a rigorous and cohesive set of evaluation metrics. The framework integrates over six classical MARL algorithms, encompassing both on-policy and off-policy architectures, and supports more than ten interactive environments across diverse application domains such as gaming, sports, robotics, autonomous vehicles, drones, traffic signal control, and power systems. We have implemented over eight adversarial attack methods, including state/observation-based, policy-based, action-based, reward-based, and environment-based attacks, covering all phases of the MDP closed-loop. Additionally, the framework includes more than five robustness evaluation metrics, addressing both self-model and inter-model perspectives. It also supports automated batch experiments and allows users to define custom agents and environments, providing an efficient and user-friendly platform for advancing MARL robustness research.

To thoroughly demonstrate the effectiveness of our evaluation framework, we conducted a series of experiments across multiple environments, employing various adversarial attack strategies. All evaluation experiments were performed on our proposed adversarial robustness evaluation platform, which we anticipate will assist researchers in gaining a deeper understanding of adversarial factors and contribute to the enhancement of model robustness. Our contributions are as follows:

- We unify on-policy and off-policy MARL algorithms into an broad actor-critic architecture, provide various adversarial attack methods for different stages, support customized agent and environment, and provide a flexible configuration automated batch experiment workflow.
- Based on our framework, we provide an open-sourced platform, which containing 6+ MARL algorithms, 10+ MARL environments, 8+ adversarial attack methods, also could fully evaluate model robustness through a total of 5+ self-model-oriented and inter-model-oriented robustness metrics.
- We conduct extensive experiments to verify the effectiveness of the proposed platform, and the results demonstrate that the platform can effectively evaluate the robustness of MARL algorithms. Meanwhile, we provide suggestions on the selection of proper metrics with examples.

2 RELATED WORK

2.1 Problem Formulation

Multi-agent reinforcement learning can be modeled as a decentralized partially observable Markov decision process (Dec-POMDP), which can be defined by a set of key elements $\langle N, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\{1,\ldots,N\}}, \{\pi^i\}_{i\in\{1,\ldots,N\}}, P, \{R^i\}_{i\in\{1,\ldots,N\}}, \gamma\rangle$ [36].

- N: the number of agents, where $N \geqslant 2$ is referred as multi-agent cases, and $N = 1$ degenerates to a single-agent MDP.
- $\mathcal{S}$: the set of environment states shared by all agents. In multi-agent scenarios, the environmental state takes into account the state information of all agents, usually represented as $\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_N$.
- $\mathcal{A}^i$: the set of actions of agent i. We denote $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_N$.
- π^i: the set of policy of agent i.
- $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$: for each time step t, given agents' joint actions $a \in \mathcal{A}$, the transition probability from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ in the next time step.

- $R^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$: the reward function that returns a scalar value to the agent i for a transition from (s, a) to s'.
- $\gamma \in [0, 1]$ is the discount factor that represents the value of time.

The goal of multi-agent reinforcement learning is to find a joint policy $\pi^* = \{\pi_1^*, \pi_2^*, \ldots, \pi_N^*\}$ that maximizes the cumulative discounted return G over time.

2.2 Adversarial Attack in Reinforcement Learning

Adversarial attacks involve small, often imperceptible perturbations to the inputs of a neural network, which lead the model to produce incorrect outputs. These attacks are generally categorized into two types: white-box attacks, where the attacker has full knowledge of the model's architecture and parameters, and black-box attacks, where the attacker only interacts with the model through its inputs and outputs, without direct access to its internal structure.

The study of adversarial attacks has naturally extended to deep reinforcement learning. Huang et al. [12] were the first to attack strategies derived from deep reinforcement learning by applying the FGSM algorithm [6], which is widely used in machine learning. They generated adversarial perturbations and directly added them to the agent's observations, successfully disrupting the performance of deep learning agents. Li et al. [17] introduced Adversarial Minority Influence (AMI), a practical black-box attack that exploits the complexity of multi-agent interactions in centralized MARL (c-MARL) to deceive agents into collectively harmful outcomes. This marked the first successful attack on both real-world robot swarms and simulated environments. To counter such vulnerabilities, they also proposed the BARDec-POMDP framework, which enhances c-MARL robustness against Byzantine failures by enabling agents to learn adaptive policies. This framework helps agents collaborate with identified allies and improves resilience and micromanagement in diverse scenarios [18].

2.3 Robustness Evaluation in Reinforcement Learning

In reinforcement learning, robustness is primarily defined as the ability of an algorithm to sustain stable and reliable performance despite environmental uncertainties and adversarial attacks. The objective of robust reinforcement learning is to enhance the algorithm's performance in worst-case scenarios, which may be either deterministic or statistical in nature. Environmental uncertainty typically stems from discrepancies between training and testing conditions, such as the divergence between simulated and real-world environments, the non-stationarity of the test domain, variations in the distribution of training and testing tasks, or the partial observability of environmental transitions and rewards. This focus on robustness is crucial for deploying reinforcement learning systems in practical, dynamic settings where conditions can vary unpredictably.

To accurately evaluate robustness, appropriate platforms and evaluation metrics are required. For example, the MuJoCo environment is commonly used to test robustness in continuous environments [34]. In addition, there are platforms for observation-based attacks, such as Atari games [24], and control task environments with safety zones [7, 15]. Using adversarial attack methods for robust evaluation is a common practice, such as FGSM [6], PGD [23], C&W [2], and other adversarial attack algorithms. In addition, there are some robustness evaluation methods based on adversarial training [23]. These methods have been widely used in the field of computer vision, but their application in the field of reinforcement learning is still limited. Guo et al. [9] proposed MARLSafe, a novel robustness evaluation framework for cooperative multi-agent reinforcement learning (c-MARL) algorithms, which comprehensively evaluates three aspects: state, action, and reward robustness. Li et al. [19] proposed Mutual Information Regularization as Robust Regularization (MIR3), a framework that uses off-policy evaluation to implicitly optimize robustness against worst-case adversaries in multi-agent reinforcement learning, achieving better robustness and training efficiency without requiring explicit threat scenarios.

The research on robustness in multi-agent reinforcement learning remains incomplete, presenting challenges in effectively and conveniently evaluating the resilience of MARL systems. A unified MARL robustness evaluation platform that integrates diverse algorithms, environments, and performance metrics is crucial for advancing the field.

3 ARCHITECTURE DESIGN

Building on the Heterogeneous-Agent Reinforcement Learning (HARL) library [41], a PyTorch-based framework that accommodates various MARL algorithms, our platform incorporates expanded support for heterogeneous agents. We have integrated additional MARL algorithms, several real-world simulated environments, as well as full types of adversarial attack methods, and extended robustness evaluation capability as shown in Table 1. Furthermore, we have developed a comprehensive set of metrics for evaluating the robustness of MARL algorithms, enabling thorough assessments of their resilience. To facilitate efficient evaluation,

Table 1: Comparison of frameworks for multi-agent reinforcement learning

Framework	Multi-Agent	Robustness	Customizable	Real-world Simulated Environments	Heterogeneous-Agent	Full Types Attacks
HARL [41]	✓			✓	✓	
MARLlib [11]	✓		✓	✓		
PyMARL [30]	✓					
EPyMARL [27]	✓					
MARLSafe [9]	✓	✓				
Ours*	✓	✓	✓	✓	✓	✓

Table 2: Comparison of features in multi-agent reinforcement learning algorithms

Algorithm	Architecture	Policy Type	Action Space	Replay Buffer	Task mode	Learning Type
MAPPO [37]	Actor-Critic	on-policy	Continuous/Discrete	N	Cooperative + Collaborative + Competitive + Mixed	CC
HAPPO [41]	Actor-Critic	on-policy	Continuous/Discrete	N	Cooperative + Collaborative + Competitive + Mixed	CC
MADDPG [22]	Policy-Q	off-policy	Continuous	Y	Cooperative + Collaborative + Competitive + Mixed	CC
QMIX [29]	Q-Mixer	off-policy	Discrete	Y	cooperative	VD
VDN [31]	Q-Mixer	off-policy	Discrete	Y	cooperative	VD
IQL [32]	Q	off-policy	Discrete	Y	Cooperative + Collaborative + Competitive + Mixed	IL

we have also implemented robustness evaluation functionality and designed pipeline tools for workflow management. The architecture of our platform is shown in Figure 1, which mainly consists of the RL core module, the adversarial attack module, the robustness tuning module, and the automated workflow pipeline.

Specifically, the RL core module includes components such as *Agent*, *Environment*, *Replay Buffer*, *Runner*, and *Evaluator*. The *Agent* is responsible for decision-making and interaction, the *Environment* is responsible for simulation and interaction, the *Replay Buffer* is responsible for storing and sampling experiences, the *Runner* is responsible for model training, and the *Evaluator* is responsible for model evaluation. These components together support a complete MARL closed-loop workflow, on which the other modules are built.

In addition to the built-in MARL algorithms and environments, we also provide custom Agent and Environment functions to facilitate users to access the environment and evaluation algorithms more quickly and conduct customized experiments according to their own needs.

The adversarial attack module implements various adversarial attack methods based on state, action, reward, and environment, supporting the attack and robustness evaluation of MARL algorithms.

The automated workflow pipeline realizes the automated execution of processes such as model training, adversarial attacks, robustness evaluation, data export, visualization, etc., making it convenient for users to quickly conduct MARL robustness research.

3.1 MARL Algorithms

Our platform integrates multiple MARL algorithms, each characterized by distinct architectures, learning types, and action spaces, as detailed in Table 2. Some algorithms operate on a Q-learning architecture, such as QMIX, VDN, and IQL, while others are based on a broad actor-critic architecture like MAPPO. Additionally, some algorithms, such as MADDPG, blend Policy Gradient and Q-learning architectures. Each algorithm is implemented natively, posing considerable engineering challenges when integrating a diverse array of architectures, policies, and coding practices into a unified adversarial attack evaluation framework.

To address these challenges, we have chosen the broad actor-critic structure as the foundational architecture for the platform's algorithms, thereby standardizing various approaches under this umbrella. For instance, we have adapted QMIX into an Actor-Mixer structure, which supports the Q-learning architecture through the computation and updating of Q-values. This standardization simplifies the integration process and enhances the platform's ability to systematically evaluate adversarial robustness across different MARL algorithms.

3.2 Environments

The platform supports a variety of MARL environments, as shown in Table 3. In the design, we considered various types of environments, such as cooperative, collaborative, competitive, and mixed, as well as fully observable, partially observable, discrete action space, continuous action space, and other environment characteristics. In addition to supporting classic environments in the field of multi-agent reinforcement learning, such as SMAC, MPE, MAMuJoCo, etc., it also supports some environments related to real-world applications, such as MetaDrive involving autonomous driving, Bi-DexHands involving robot collaboration, Quadrotor Swarms involving drone collaboration, as well as real-life network system control, voltage control, and other environments. This provides researchers with a rich experimental environment, which helps to more comprehensively evaluate the robustness of MARL algorithms and provides references for practical applications.

Table 3: The taxonomy of platform supported environments

Name	Learning Mode	Observability	Action Space	Interact Mode
SMAC [30]	Cooperative + Collaborative	Partial	Discrete	Simultaneous
SMACv2 [5]	Cooperative + Collaborative	Partial	Discrete	Simultaneous
MPE [25]	Cooperative + Collaborative + Competitive + Mixed	Full	Discrete + Continuous	Simultaneous / Asynchronous
MAMuJoCo [28]	Cooperative + Collaborative	Partial	Continuous	Simultaneous
Google Research Football [14]	Cooperative + Collaborative	Full	Discrete	Simultaneous
MetaDrive [16]	Collaborative	Partial	Continuous	Simultaneous
Bi-DexHands [3]	Cooperative	Partial	Continuous	Simultaneous
Quadrotor Swarms [1]	Collaborative	Partial	Continuous	Simultaneous
Network System Control [4]	Cooperative	Partial	Discrete	Simultaneous
Voltage Control [35]	Cooperative	Partial	Continuous	Simultaneous

3.3 Evaluation Metrics

In reinforcement learning, the interaction between the agent and the environment is continuous, making the average episode reward a common metric for evaluating model performance. However, unlike accuracy in image classification, which is a dimensionless percentage, the reward is a numerical value. Additionally, the initial reward value of a reinforcement learning model during training is not zero; it can be either positive or negative, and these values may differ across various environments. Consequently, while the global reward can indicate performance changes before and after an attack, it is not suitable for comparing performance differences across various environments [20]. To address this issue, we proposed and implemented several metrics from the perspectives of self-model-oriented and inter-model-oriented robustness.

3.3.1 *Self-model-oriented robustness metric.*

Definition 3.1. We abandon the use of reward difference to directly measure robustness. Instead, we use the **performance variation amount** (R) of the model before and after training as the evaluation basis, which formula is:

$$\mathrm{R}_i = r_i - r_i^0, \tag{1}$$

where i refers to MARL models with different implementation details or optimization techniques under the same algorithm, r_i^0 refers to the average episode reward value that can be obtained by evaluating the model once before normal training, and r_i refers to the average episode reward value that can be obtained by evaluating the model once after normal training. Therefore, for a MARL model that is trained normally with reasonable parameters, its R value range is $(0, +\infty)$.

Definition 3.2. Based on definition 3.1, we define the model's **Self-Robustness Change Rate** (SRR) as follows:

$$\mathrm{SRR}_i = \frac{r_i^A - r_i}{R_i}, \tag{2}$$

where r_i^A refers to the reward value that the model can obtain after being evaluated once after being adversarial attacked. Since the SRR is a relative metric, its value range is $(-\infty, +\infty)$, where its value indicates the magnitude of performance change, and the sign indicates the direction of performance change after being adversarial. A positive number indicates improved performance, while a negative number indicates a decline in performance.

3.3.2 *Inter-model-oriented robustness metric.*

The SRR indicator evaluates the model itself and cannot assess the differences between different models. To address this, we propose three inter-model-oriented robustness metrics.

Definition 3.3. Based on definition 3.2, we can obtain the **Relative Self-Robustness Change Rate** (rSRR) relative to the baseline by calculating the SRR error between the target model and the baseline model, with the following formula:

$$\mathrm{rSRR}_t = \mathrm{SRR}_t - \mathrm{SRR}_{baseline}, \tag{3}$$

where t represents the target model which which is a homogeneous model that uses the same algorithm as the baseline model but with different implementation details. When $\mathrm{rSRR} > 0$, it means that the target model has better robustness than the baseline model, otherwise worse.

Definition 3.4. **Tricked Performance Change Rate** (TPR) refers to the performance change of the fine-tuned model compared to the baseline model without adversarial attacks. The formula is as follows:

$$\mathrm{TPR}_t = \frac{r_t - r_{baseline}}{R_{baseline}}, \tag{4}$$

where $\mathrm{TPR} > 0$ indicates that the fine-tuned model performs better than the baseline model, conversely worse.

Definition 3.5. **Tricked Robustness Change Rate** (TRR) refers to the robustness change of the fine-tuned model compared to the baseline model under adversarial attacked. The formula is as follows:

$$\begin{aligned}\mathrm{TRR}_t &= \frac{(r_t^A - r_{baseline}) - (r_{baseline}^A - r_{baseline})}{R_{baseline}} \\ &= \frac{r_t^A - r_{baseline}^A}{R_{baseline}},\end{aligned} \tag{5}$$

where $\mathrm{TRR} > 0$ indicates that the fine-tuned model is more robust than the baseline model, conversely worse. It is important to note that the difference between TRR and rSRR lies in their evaluation methods. TRR assesses the change in performance before and after training the baseline model, whereas rSRR evaluates the model directly by calculating each SRR and obtaining the result through subtraction.

3.3.3 *Comprehensive robustness metric.*

By integrating the three metrics mentioned above, a comprehensive metric is used to characterize the overall performance and robustness of a certain one fine-tuned model, which is calculated using the weighted average of TPR, TRR, and rSRR values, collectively known as the **Comprehensive Robustness Change Rate** (CR). The formula for this calculation is as follows:

$$\begin{aligned}\mathrm{CR}_t &= \alpha \mathrm{TPR}_t + \beta \mathrm{TRR}_t + \gamma \mathrm{rSRR}_t \\ &\text{s.t. } \alpha + \beta + \gamma = 1, 0 \le \alpha, \beta, \gamma \le 1.\end{aligned} \tag{6}$$

This indicator combines metrics from various models to assess both performance and robustness. It can be tailored for practical applications using custom weighting coefficients to balance these two aspects. For instance, with weights set at $\alpha = 0.5$, $\beta = 0.5$, and $\gamma = 0$, it integrates the model's original performance before an attack with its robustness performance afterward. This is achieved through a weighted average of the model's TPR and TRR metrics, providing a comprehensive evaluation of the model's overall robustness.

3.4 Adversarial Attack Methods

The Markov decision process involves multiple stages and closed-loop features, adding complexity and security risks. For instance, adversaries can attack agents at various stages. To thoroughly assess the robustness of MARL algorithms, we implement a range of adversarial attack methods, detailed in Table 4. These methods are categorized into state-based, action-based, reward-based, policy-based, and environment-based types, allowing attackers to select the most suitable approach for their evaluation needs.

Table 4: Comparison of features in multi-agent reinforcement learning adversarial attack algorithms

Method Name	Attack Type	Effective Phase	Adversary's Knowledge	Multi-step Execution Required
Random Noise [20]	State-based	Evaluation phase	White-Box/Black-Box	N
Iterative Perturbation [20]	State-based	Evaluation phase	White-Box	N
Adaptive Action Induction [20]	State-based	Evaluation phase	White-Box	Y
Random Action	Action-based	Evaluation phase	Black-Box	N
Flipping Reward Signals [10]	Reward-based	Training phase	White-Box	Y
Random Policy	Policy-based	Evaluation phase	White-Box	N
Traitor [20]	Policy-based	Evaluation phase	White-Box	Y
Environmental Factor Search	Environment-based	Evaluation phase	Black-Box	N

In addition to the adversarial attack methods outlined in [20] (e.g., Random Noise, Iterative Perturbation, Adaptive Action, and Traitor), we have enhanced the platform by incorporating new methods. These include action-based attacks (e.g., Random Action), policy-based attacks (e.g., Random Policy), environment-based attacks (e.g., Environmental Factor Search), and reward-based attacks (e.g., Flipping Reward Signals). This expansion allows for a broader range of adversarial attack scenarios.

3.4.1 Random Noise Attack. We introduce random noise following a Gaussian distribution with a mean of 0 and a variance of σ to the partial observations generated by the environment. This noisy observation is then fed into the agent's policy, enabling adversarial attacks on the agent.

3.4.2 Iterative Perturbation Attack. We implement a gradient-based iterative perturbation algorithm (e.g., PGD) to perturb the agent's state multiple times in order to achieve adversarial attacks on the agent. The model used to compute the gradient is the victim model, which is the evaluated MARL model. From the perspective of the PGD algorithm, it performs an untargeted iterative attack, aimed at misleading the agent's policy away from the observation distribution that can output the optimal action.

3.4.3 Adaptive Action Induction Attack. Due to the nature of reinforcement learning, greedy attacks at each time step may not lead to a substantial decrease in total return during long-term planning. To address the limitations of the iterative perturbation attack method, we developed an improved version called the adaptive action induction attack. This approach involves training a reward target that opposes that of the victim. By utilizing the gradient information from the adversarial agent's policy model, we iteratively perturb the victim's partial observations to execute a long-range attack on the victim's policy.

3.4.4 Random Action Attack. Perturbing the global state or local observations obtained during the data sampling phase can indirectly affect the actions output by the agent's policy. Directly altering the output actions does not require knowledge of the agent's policy model, allowing for a fully black-box attack. The Random Action attack algorithm is an action-based black-box attack that randomly replaces the actions produced by a specific agent. This method assesses whether the overall system can maintain robustness when a local agent makes suboptimal or incorrect decisions.

3.4.5 Flipping Reward Signals Attack. In reinforcement learning, reward signals are essential for agent training. If these signals are flipped, agents may learn incorrect policies, which can negatively impact their performance. The Flipping Reward Signals attack algorithm specifically targets an agent by altering its reward signals. This method assesses the agent's robustness when exposed to incorrect reward signals during training.

3.4.6 Random Policy Attack. In reinforcement learning, an agent's policy is the core of its behavior. By randomly replacing the policies of some agents, the Random Policy attack method assesses the robustness of the entire system when the policies of local agents change.

3.4.7 Traitor Attack. Traitor attacks are policy-based attacks that involve training a traitor agent to act as an adversary. The traitor agent is trained to maximize the victim agent's inverse reward, thereby misleading its policy and enabling the minority to influence the majority. This method evaluates the robustness of the victim agent when faced with a traitor agent.

3.4.8 Environmental Factor Search Attack. Environmental factors significantly influence agent learning and decision-making, as agents interact with their surroundings. The Environmental Factor Search attack method evaluates the robustness of agents by combining, searching, or perturbing these factors. This approach is vital for understanding agent generalization.

These attack algorithms target various stages of reinforcement learning and have distinct implementation logics. Consequently, different categories of attack algorithms cannot be directly applied to the same code location; instead, they must be implemented separately. To resolve this, we abstract and encapsulate the implementation logic of these algorithms, unifying them under a single adversarial attack process based on their categories. By inheriting abstract classes, we can implement the logic for each adversarial attack algorithm, accommodating various categories.

For instance, state-based attack algorithms, such as random noise attacks and iterative perturbation attacks, are grouped under a state-based attack process. Similarly, policy-based attack algorithms, including random policy attacks and traitor attacks, fall under a policy-based attack process. For each attack process, we implement both on-policy and off-policy algorithm logic according to different learning strategies. This approach allows us to use a unified adversarial attack call interface to support multiple adversarial attack algorithms on a single model.

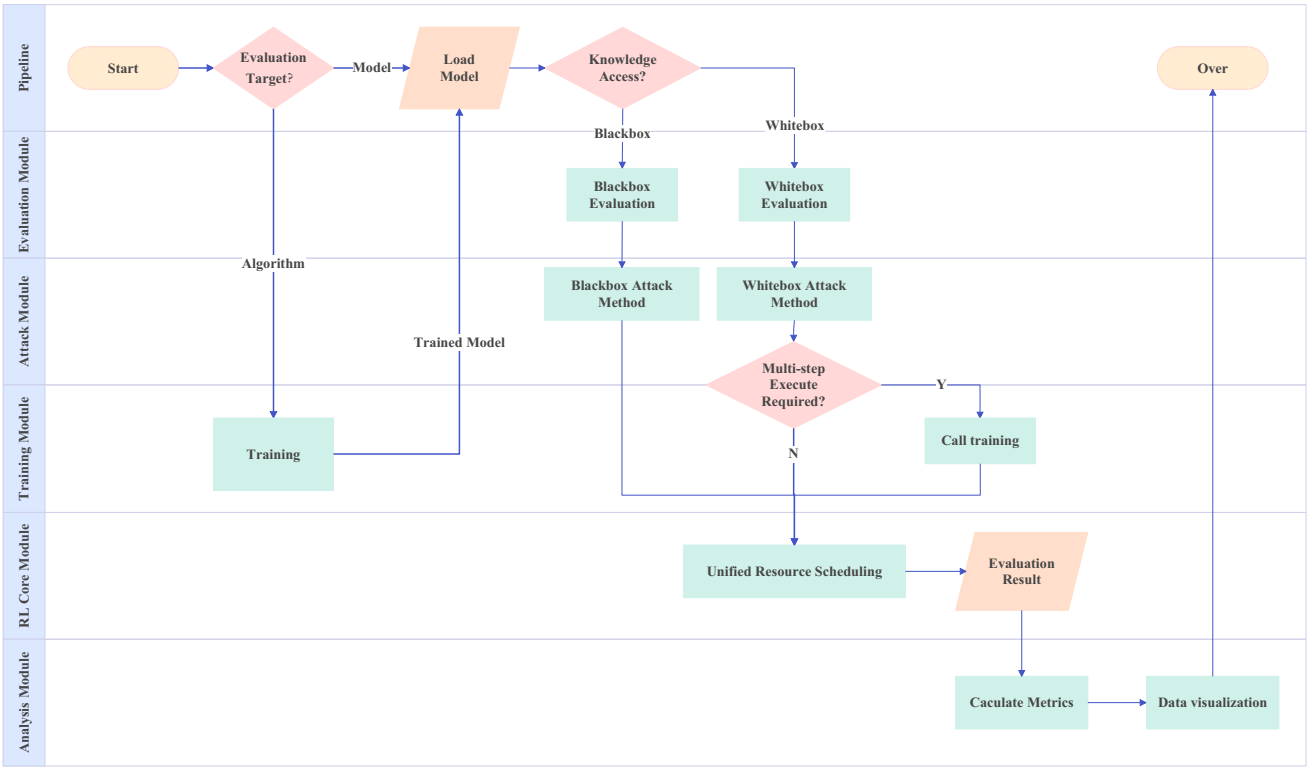

Figure 2: The evaluation process of the platform, including model training, adversarial attack, robustness evaluation, and visualization analysis.

3.5 Evaluation Architecture

The platform's core business process is evaluating the robustness of multi-agent reinforcement learning models. This involves the interaction and integration of various functional modules within the platform.

As illustrated in Figure 2, the evaluation process begins by identifying whether the focus is on pre-trained models or algorithms. If the target is an algorithm, the platform's training module is activated for pre-training. Once the pre-trained model is ready, the model to be tested is loaded, and the evaluation module determines if it is a white-box or black-box model.

For a white-box model (e.g., one with accessible source code and policy checkpoints), it must comply with the platform's interface requirements. The evaluation module then invokes white-box adversarial attack methods from the attack module to assess the model.

In contrast, for a black-box model (e.g., one that accepts input observations and query actions via an HTTP API), the evaluation module retrieves data from the model's query interface and employs black-box adversarial attack methods from the attack module for evaluation.

In white-box adversarial attack methods, the evaluation module assesses whether multi-step execution is needed based on the attack principles. If required, it activates the training module to develop the adversarial policy. Once the evaluation configuration and specific processes are complete, the RL core module will conduct unified scheduling to execute the entire evaluation process and obtain the results.

The analysis module is then activated to export and merge the evaluation data, calculate comprehensive metrics, visualize the results, and generate the final analysis, completing the model evaluation process.

3.6 Customized Agent and Environment

In addition to supporting multiple algorithms and environments, our platform offers extensibility for user-defined agents and environments.

The *Agent* class abstracts the agent's policy, providing a standardized interface for receiving observations and outputting actions. Users can seamlessly integrate their custom agent policies by inheriting the *Agent* class and implementing the required interface. This feature allows users to train and evaluate any custom agent policy within the platform.

The *EnvExample* class abstracts environment interactions, and we have defined its interaction interface following the OpenAI Gym's interface specifications. In addition to the core methods (e.g., step and reset), we require the provision of information such as the number of agents, action space, observation space, and global state space. Similar to the *Agent* class, users can seamlessly integrate their custom environments by inheriting the *EnvExample* class and implementing the necessary interface. This enables users to apply the platform's MARL algorithms to custom environments.

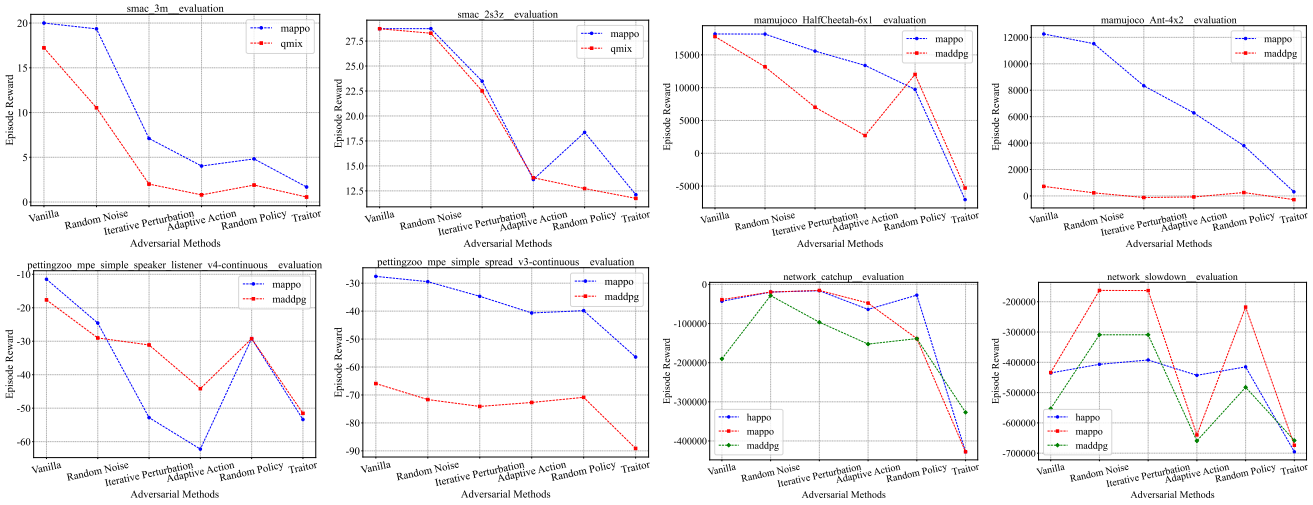

Figure 3: The visualization results of the mamujoco_HalfCheetah-6x1_mappo experimental group.

Through this approach, we aim to provide greater flexibility, allowing users to tailor the platform to meet a wider range of practical needs.

4 EXPERIMENT

To evaluate the platform's effectiveness and its robust evaluation capabilities, we conducted experiments to assess how various adversarial attack methods affect the robustness of multi-agent reinforcement learning models. We selected four MARL environments with both discrete and continuous action spaces, totaling 8 different scenarios: SMAC (i.e., 3m, 2s3z), MAMujoco (i.e., HalfCheetah-6x1, Ant-4x2), MPE (i.e., simple_spread, simple_speaker_listener) and Network (i.e., catchup, slowdown). We choose MARL algorithms that are suitable for each environment, including MAPPO, MADDPG, QMIX, and HAPPO.

We started by training the initial multi-agent reinforcement learning models. Each model was then evaluated against up to five different adversarial attacks (e.g., Random Noise, Iterative Perturbation, Adaptive Action Induction, Random Policy, Traitor). We then statistically analyzed the impact of different adversarial attack methods on the robustness of different algorithms, as illustrated in Figure 3. The points in the figure are the episode reward values under each discrete adversarial attack method. The dotted line connects all the values of the same algorithm in order to intuitively represent its trend and verify the strength of the adversarial attack method.

From the perspective of adversarial attack strength, we can analyze the experimental results through the trend of the line graph and draw the following experimental conclusions: 1) The adversarial attack method we implemented is effective and successfully impacts the model's performance. 2) Multi-step execution necessitated adversarial attack methods (e.g., Adaptive Action Induction, Traitor) have worst-case attack capabilities generally.

From the perspective of algorithms, we can compare the robustness of different algorithms in the same scenario and then expand to the general comparison of different algorithms in a wide range of scenarios. Then we arrive at the following conclusions: 1) In most environments and scenarios, MAPPO generally outperforms MADDPG in terms of performance and robustness. 2) Algorithms with weaker vanilla performance tend to be more robust against specific adversarial perturbations. For instance, MADDPG demonstrates greater resilience than MAPPO when subjected to state-based iterative perturbations methods (e.g., Iterative Perturbation, Adaptive Action Induction) in the mpe_simple_speaker_listener scenario. 3) The trends of different algorithms in the same scenario show a consistent pattern, indicating that the impact of the adversarial algorithms is stable.

5 CONCLUSION

In this study, we have developed a comprehensive evaluation framework for multi-agent reinforcement learning that supports over six algorithms, ten interactive environments, eight adversarial attack methods, and five robustness evaluation metrics. Our proposed metrics are designed to provide insights from two essential perspectives: self-model-oriented and inter-model-oriented evaluations. Furthermore, our platform offers an out-of-the-box solution for assessing model robustness. To validate the effectiveness of our framework, we conducted a series of experiments across various environments, utilizing different adversarial attack strategies facilitated by our platform. The results demonstrate that the robustness of MARL models can be effectively evaluated using these adversarial methods, highlighting potential vulnerabilities and the effectiveness of different defense strategies. Based on our findings, we present discussions and recommendations aimed at enhancing model robustness. Our framework is designed to provide a rigorous and thorough evaluation, enabling a deeper understanding of the resilience characteristics of MARL systems. We hope this contribution will aid other researchers in exploring the factors influencing robustness in MARL and in developing strategies to enhance the resilience of these systems.

REFERENCES

[1] Sumeet Batra, Zhehui Huang, Aleksei Petrenko, Tushar Kumar, Artem Molchanov, and Gaurav S. Sukhatme. 2022. Decentralized Control of Quadrotor Swarms with End-to-end Deep Reinforcement Learning. In *Proceedings of the 5th Conference on Robot Learning*. PMLR, 576–586.

[2] Nicholas Carlini and David Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. 39–57. https://doi.org/10.1109/SP.2017.49

[3] Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. 2022. Towards Human-Level Bimanual Dexterous Manipulation with Reinforcement Learning. *Advances in Neural Information Processing Systems* 35 (Dec. 2022), 5150–5163.

[4] Tianshu Chu, Sandeep Chinchali, and Sachin Katti. 2020. Multi-Agent Reinforcement Learning for Networked System Control. https://doi.org/10.48550/arXiv.2004.01339 arXiv:2004.01339 [cs, stat]

[5] Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N. Foerster, and Shimon Whiteson. 2022. SMACv2: An Improved Benchmark for Cooperative Multi-Agent Reinforcement Learning. arXiv:2212.07489 [cs]

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. https://doi.org/10.48550/arXiv.1412.6572 arXiv:1412.6572 [cs, stat]

[7] Sven Gronauer. 2023. *Bullet-Safety-Gym: A Framework for Constrained Reinforcement Learning*. Technical Report. [object Object]. https://doi.org/10.14459/2022MD1639974

[8] Jun Guo, Wei Bao, Jiakai Wang, Yuqing Ma, Xinghai Gao, Gang Xiao, Aishan Liu, Jian Dong, Xianglong Liu, and Wenjun Wu. 2023. A Comprehensive Evaluation Framework for Deep Model Robustness. *Pattern Recognition* 137 (May 2023), 109308. https://doi.org/10.1016/j.patcog.2023.109308

[9] Jun Guo, Yonghong Chen, Yihang Hao, Zixin Yin, Yin Yu, and Simin Li. 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*. 115–122.

[10] Yi Han, Benjamin I. P. Rubinstein, Tamas Abraham, Tansu Alpcan, Olivier De Vel, Sarah Erfani, David Hubczenko, Christopher Leckie, and Paul Montague. 2018. Reinforcement Learning for Autonomous Defence in Software-Defined Networking. https://doi.org/10.48550/arXiv.1808.05770 arXiv:1808.05770 [cs, stat]

[11] Siyi Hu, Yifan Zhong, Minquan Gao, Weixun Wang, Hao Dong, Zhihui Li, Xiaodan Liang, Xiaojun Chang, and Yaodong Yang. 2022. MARLlib: Extending RLlib for Multi-agent Reinforcement Learning. arXiv:2210.13708 [cs]

[12] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. 2017. Adversarial Attacks on Neural Network Policies. https://doi.org/10.48550/arXiv.1702.02284 arXiv:1702.02284

[13] Zonglei Jing, Xiaoqian Li, Peijun Ju, and Huanshui Zhang. 2024. Optimal control and filtering for hierarchical decision problems with h_ınfty constraint based on stackelberg strategy. *IEEE Trans. Automat. Control* 69, 9 (Sept. 2024), 6238–6245. https://doi.org/10.1109/TAC.2024.3374487

[14] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. 2020. Google Research Football: A Novel Reinforcement Learning Environment. https://doi.org/10.48550/arXiv.1907.11180 arXiv:1907.11180 [cs, stat]

[15] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. 2017. AI Safety Gridworlds. arXiv:1711.09883 [cs]

[16] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. 2022. MetaDrive: Composing Diverse Driving Scenarios for Generalizable Reinforcement Learning. arXiv:2109.12674 [cs]

[17] Simin Li, Jun Guo, Jingqiao Xiu, Pu Feng, Xin Yu, Aishan Liu, Wenjun Wu, and Xianglong Liu. 2023. Attacking cooperative multi-agent reinforcement learning by adversarial minority influence. *arXiv preprint arXiv:2302.03322* (2023).

[18] Simin Li, Jun Guo, Jingqiao Xiu, Ruixiao Xu, Xin Yu, Jiakai Wang, Aishan Liu, Yaodong Yang, and Xianglong Liu. 2023. Byzantine robust cooperative multi-agent reinforcement learning as a bayesian game. *arXiv preprint arXiv:2305.12872* (2023).

[19] Simin Li, Ruixiao Xu, Jun Guo, Pu Feng, Jiakai Wang, Aishan Liu, Yaodong Yang, Xianglong Liu, and Weifeng Lv. 2023. MIR2: Towards Provably Robust Multi-Agent Reinforcement Learning by Mutual Information Regularization. *arXiv preprint arXiv:2310.09833* (2023).

[20] Xiaoqian Li, Peijun Ju, and Zonglei Jing. 2024. Can Tricks Affect Robustness of Multi-Agent Reinforcement Learning?. In *2024 43rd Chinese Control Conference (CCC)*. Kunming, China. in press.

[21] Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. arXiv:1712.09381 [cs]

[22] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2020. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. arXiv:1706.02275

[23] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2019. Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv:1706.06083 [cs, stat]* (Sept. 2019). https://doi.org/10.48550/ARXIV.1706.06083 arXiv:1706.06083 [cs, stat]

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.1312.5602 arXiv:1312.5602

[25] Igor Mordatch and Pieter Abbeel. 2018. Emergence of Grounded Compositional Language in Multi-Agent Populations. arXiv:1703.04908 [cs]

[26] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht. 2019. Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning. https://doi.org/10.48550/arXiv.1906.04737 arXiv:1906.04737 [cs, stat]

[27] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2020. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. https://doi.org/10.48550/arXiv.2006.07869

[28] Bei Peng, Tabish Rashid, Christian A. Schroeder De Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. FACMAC: Factored Multi-Agent Centralised Policy Gradients. https://doi.org/10.48550/arXiv.2003.06709 arXiv:2003.06709 [cs, stat]

[29] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. https://doi.org/10.48550/arXiv.1803.11485 arXiv:1803.11485 [cs, stat]

[30] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. arXiv:1902.04043 [cs, stat]

[31] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2017. Value-Decomposition Networks For Cooperative Multi-Agent Learning. arXiv:1706.05296 [cs]

[32] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[33] Shiyu Tang, Ruihao Gong, Yan Wang, Aishan Liu, Jiakai Wang, Xinyun Chen, Fengwei Yu, Xianglong Liu, Dawn Song, Alan Yuille, Philip H. S. Torr, and Dacheng Tao. 2022. RobustART: Benchmarking Robustness on Architecture Design and Training Techniques. arXiv:2109.05211 [cs]

[34] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A Physics Engine for Model-Based Control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

[35] Jianhong Wang, Wangkun Xu, Yunjie Gu, Wenbin Song, and Tim C. Green. 2022. Multi-Agent Reinforcement Learning for Active Voltage Control on Power Distribution Networks. https://doi.org/10.48550/arXiv.2110.14300 arXiv:2110.14300 [cs]

[36] Yaodong Yang and Jun Wang. 2021. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. https://doi.org/10.48550/arXiv.2011.00583 arXiv:2011.00583

[37] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games. https://doi.org/10.48550/arXiv.2103.01955 arXiv:2103.01955 [cs]

[38] Xin Yu, Rongye Shi, Pu Feng, Yongkai Tian, Simin Li, Shuhao Liao, and Wenjun Wu. 2024. Leveraging Partial Symmetry for Multi-Agent Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17583–17590.

[39] Xin Yu, Rongye Shi, Pu Feng, Yongkai Tian, Jie Luo, and Wenjun Wu. 2023. ESP: Exploiting Symmetry Prior for Multi-Agent Reinforcement Learning. In *ECAI 2023*. IOS Press, 2946–2953.

[40] Xin Yu, Yongkai Tian, Li Wang, Pu Feng, Wenjun Wu, and Rongye Shi. 2024. AdaptAUG: Adaptive Data Augmentation Framework for Multi-Agent Reinforcement Learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 10814–10820.

[41] Yifan Zhong, Jakub Grudzien Kuba, Siyi Hu, Jiaming Ji, and Yaodong Yang. 2023. Heterogeneous-Agent Reinforcement Learning. https://doi.org/10.48550/arXiv.2304.09870 arXiv:2304.09870 [cs]

[42] Yihe Zhou, Shunyu Liu, Yunpeng Qing, Kaixuan Chen, Tongya Zheng, Yanhao Huang, Jie Song, and Mingli Song. 2023. Is Centralized Training with Decentralized Execution Framework Centralized Enough for MARL? arXiv:2305.17352 [cs]