SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2024, Vol. 67, Iss. 5, 152104:1–152104:15
https://doi.org/10.1007/s11432-021-3688-y

Understanding adversarial attacks on observations in deep reinforcement learning

You QIAOBEN[1], Chengyang YING[1], Xinning ZHOU[1], Hang SU[1,2], Jun ZHU[1,2*] & Bo ZHANG[1]

[1]*Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology, Tsinghua-Bosch Joint Center for Machine Learning, Institute for Artificial Intelligence, Tsinghua University, Beijing* 100084*, China;*
[2]*Peng Cheng Laboratory, Shenzhen* 518055*, China*

Received 13 December 2021/Revised 14 September 2022/Accepted 17 November 2022/Published online 26 April 2024

Abstract Deep reinforcement learning models are vulnerable to adversarial attacks that can decrease the cumulative expected reward of a victim by manipulating its observations. Despite the efficiency of previous optimization-based methods for generating adversarial noise in supervised learning, such methods might not achieve the lowest cumulative reward since they do not generally explore the environmental dynamics. Herein, a framework is provided to better understand the existing methods by reformulating the problem of adversarial attacks on reinforcement learning in the function space. The reformulation approach adopted herein generates an optimal adversary in the function space of targeted attacks, repelling them via a generic two-stage framework. In the first stage, a deceptive policy is trained by hacking the environment and discovering a set of trajectories routing to the lowest reward or the worst-case performance. Next, the adversary misleads the victim to imitate the deceptive policy by perturbing the observations. Compared to existing approaches, it is theoretically shown that our adversary is strong under an appropriate noise level. Extensive experiments demonstrate the superiority of the proposed method in terms of efficiency and effectiveness, achieving state-of-the-art performance in both Atari and MuJoCo environments.

Keywords deep learning, reinforcement learning, adversarial robustness, adversarial attack

1 Introduction

The increasing sophistication and ubiquity of reinforcement learning (RL) have resulted in an impressive performance in the Atari games [1–3] and numerous other tasks [4,5]. However, the performance remains vulnerable when an adversary accesses inputs to an RL system and implements malicious attacks to deceive a deep policy [6–9]. Thus, a deep RL (DRL) agent may take suboptimal (or even harmful) actions to degrade the performance of a trained RL agent. As the RL-based frameworks become increasingly widely deployed in real-world scenarios, it becomes an indispensable prerequisite to understanding adversarial attacks against the DRL policies, especially for safety-related or life-critical applications, such as industrial robots and self-driving vehicles. Though adversarial robustness has been extensively studied in supervised learning [10–14], the adversarial vulnerability of the DRL has been less investigated [6,9,15]. Since the seminal work [16], only a few studies have addressed this problem. In the DRL, the adversary can choose different parts to attack, i.e., observation, transition, and reward. In this work, we mainly focus on adversarial attacks on observations, i.e., the perturbation only depends on the current state and does not change the state-action transition. Particularly, most of the previous optimization-based methods conducted adversarial attacks following a supervised approach that aims to change the behavioral characteristics of an agent directly [6]. However, as illustrated in Figure 1, for the same observation, different attacks will mislead the agent to take different actions. Therefore, it may not minimize the accumulated reward to achieve the optimal attack on the RL since the adversary does not alter the environmental dynamics. Recent learning-based methods [9,17] have tried generating an optimal adversary

* Corresponding author (email: dcszj@mail.tsinghua.edu.cn)

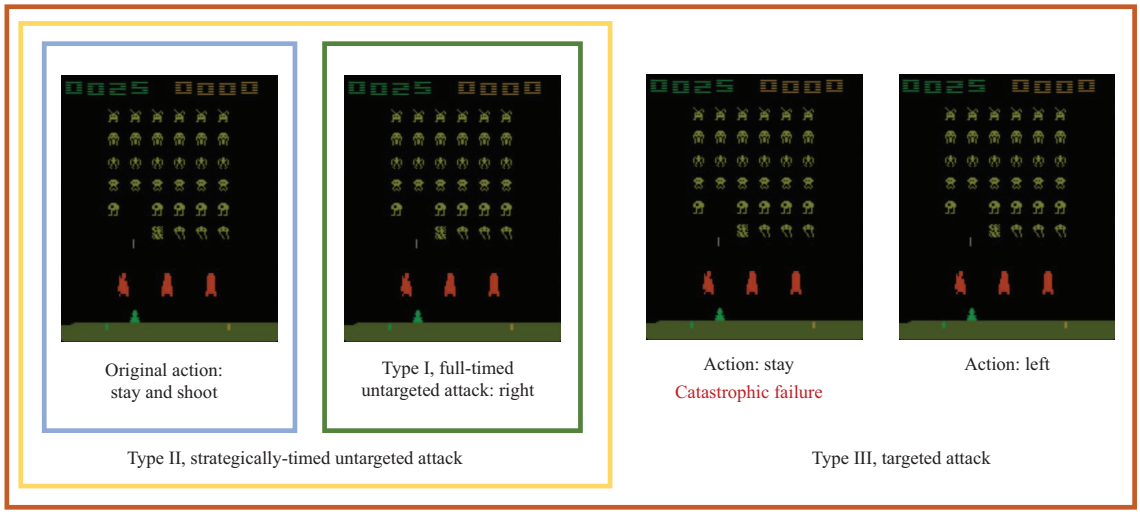

Figure 1 Actions taken by the victim policy under an adversarial attack against Atari. The optimal action for the agent in the current state is to "stay and shoot" — and get the bonus; while the worst action is to "stay" which can lead to the death of the agent. Most of the early methods can be categorized into the full-timed or strategically-timed untargeted attacks, which may not achieve the worst case action since they usually do not infer the MDP's dynamics. The more recent studies explore the targeted attacks which lead to the catastrophic consequences. In this paper, we provide a comprehensive understanding for the present methods with a generic framework in the function space and identify the limitations of these methods.

by learning a global mapping from the original states to the adversarial states, which provides a possible solution to obtain the worst-case agent reward. However, these methods may suffer from inefficiency due to the high-dimensional state representations (e.g., the state dimension is > 6000 in Atari). Therefore, it is imperative to investigate whether we can effectively and efficiently generate adversarial noise by leveraging the adversarial attacks to assess the robustness of the RL. Herein, we study the optimization-based methods due to their efficient implementation and potential to explore the dynamics of the Markov decision process (MDP). To better understand the pros and cons of these current methods, we first categorize them by considering the manner of noise generation and their attack frequencies. The first approach is the full-timed untargeted attack that misleads the agent to a suboptimal action [6, 8], and the second approach is a more covert and low-frequency strategically-timed attack, which only attacks the agent at a small subset of time steps [18]. Note that both methods try to mislead the agent to a suboptimal action rather than the worst-case action, which is referred to as an untargeted attack herein. The third approach lures the agent to a specified malicious policy using a targeted attack [8, 19, 20] which allows the adversary to choose when and how to attack by adjusting the target action. Then, we present a reformulation of these attacks in the function space, where each type corresponds to a particular function space. Obviously, targeted attacks are much stronger than untargeted attacks, i.e., the space corresponding to targeted attacks exceeds that for untargeted attacks. After close examinations, we further find that all the previous targeted attacks follow a pessimistic assumption on the adversary, i.e., the performance of the attacked policy is close to the victim policy, which may hinder the performance. To address this issue, we introduce a more reasonable optimistic assumption where the attacked policy performance should be close to the worst-case agent, which allows the adversary to explore a much larger space and generate a more powerful adversary. Following the comprehensive analysis of the present methods, we propose a novel optimization-based method that aims to minimize the expected accumulated reward for the victim policy under an alternative optimistic assumption. To the best of our knowledge, our method is the first optimization-based method to consider the dynamics of the MDP by approximately estimating the attacked policy performance. Specifically, we reformulate the task as a two-stage optimization problem by introducing an intermediate deceptive policy to explore the environment, which serves as a targeted policy to achieve the worst-case agent. Instead of maximizing the Q-values of the least preferred action [20], our method generates adversarial perturbations by minimizing the KL-divergence between the deceptive and victim policies, which can therefore satisfy our optimistic assumption. Furthermore, we provide a theoretical analysis and show that the results from our algorithm are in a tighter upper bound for the performance of the attacked agent than other existing adversaries under an appropriate noise level. Extensive experiments on both the MuJoCo and Atari environments show that our attacks

achieve state-of-the-art performance with the lowest rewards in the vast majority of the environments, compared with strong optimization-based and learning-based baselines. Furthermore, our algorithm is the first optimization-based method to achieve nearly the lowest reward in several Atari environments by exploring the MDP dynamics, and is much more efficient than the learning-based methods.

2 Preliminaries

We first briefly review the recent adversarial attack algorithms for DRLs and present the framework of the state-adversarial MDP (SA-MDP).

2.1 Adversarial attacks in DRL

In DRL, the adversary can choose different parts to attack, like observations and environment dynamics. Modifying observations and dynamics have structurally different impacts on the agent and some current work [21] hopes to find the connection between them. Moreover, since the adversary often does not have the authority to modify the original state in a simulator, in this paper, we mainly focus on the setting that perturbs the observations. In general, the present methods can be categorized into optimization-based and learning-based attacks. At each state, the adversary in optimization-based attacks generates a perturbation with the optimization-based adversarial attack in supervised learning. Based on the manner of noise generation and attack frequency, we can further classify the optimization-based adversary into three categories. The first type [6] applies an untargeted attack algorithm to mislead the agent to choose a sub-optimal action at each time step, while the second type provides heuristic adversaries that attack the agent on a subset of time steps, using a solver of an untargeted attack [7,18]. The third type of adversaries misleads the agent to some heuristic target actions by leveraging the Q-value functions [8, 19, 20]. In a learning-based attack, the adversary learns a mapping and directly generates a perturbation with this mapping. A recent example is [9], which provides a function approximator to learn the perturbation under the framework [8] with superior performance over optimization-based methods in MuJoCo environments. However, the previous learning-based methods can be inefficient in high-dimensional environments.

2.2 State-adversarial Markov decision process

Zhang et al. [8, 9] presented a unified framework of an SA-MDP, which is a modified MDP for the perturbation on state observations. Formally, let $\mathcal{S}$ be the state set and $\mathcal{F}(\mathcal{S})$ be the set of all distributions on $\mathcal{S}$. SA-MDP has an attacker set G, and any attacker $g : \mathcal{S} \to \mathcal{F}(\mathcal{S})$ can perturb the state observation which is configured as s to $\bar{s} \sim g(\cdot|s)$, where $g(\cdot|s)$ is the distribution of the perturbed state. In particular, SA-MDP can be represented as a 6-tuple as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{B}, P_a, \mathcal{R}, \gamma)$, where $\mathcal{A}$ is the action set, and $\mathcal{B}(s)$ is the disturbed state set (usually small) around state s; P_a is a transition function, $\mathcal{R}$ is a reward function, and γ is a discount factor. An agent acts following a policy π. In SA-MDP, the agent takes the action as

$$\pi_g(a|s) \equiv \sum_{\bar{s} \in S} g(\bar{s}|s)\pi(a|\bar{s}), \tag{1}$$

where π is the victim policy to be attacked and $g(\bar{s}|s)$ is the mapping for the original state to the adversarial state. The adversary aims to minimize the expected total reward of π by applying the perturbations as

$$g^* = \arg\min_{g \in G} \mathbb{E}_{a_t \sim \pi_g(\cdot|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]. \tag{2}$$

For notation simplicity, we omit the initial state distribution, and denote $a \sim \pi_g$ instead of $a_t \sim \pi_g(\cdot|s_t)$ in this paper.

3 Understanding adversarial attacks in function spaces

We first provide a reformulation of the SA-MDP in the function space, and then show that the existing three types of optimization-based adversarial algorithm can be characterized as different subspaces; finally, we provide a comprehensive analysis of the function space which motivates us to find a better adversary.

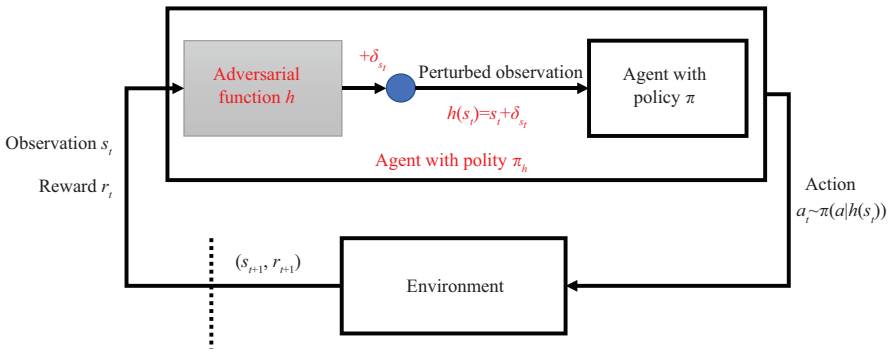

Figure 2 Adversarial attacks on the victim's observation. Given an agent with policy π, the attacker applies an adversarial function h to the observation. The victim's observation state is perturbed to $h(s_t) = s_t + \delta_{s_t}$, yielding a sub-optimal action. The victim agent consequentially behaves as $a_t \sim \pi(a|h(s_t))$ which can be recognized as an agent with adversarial policy $a_t \sim \pi_h(a|s_t)$.

Table 1 The typical existing optimization-based attacks in DRL are categorized into three types based on algorithm for noise generation and the attack frequency, which are corresponding to three function spaces

Function space	Attack manner	Attack frequency	Methods
$\mathcal{H}_1$	Untargeted attack	Full-timed	Huang et al. [6]; Zhang et al. [8]
$\mathcal{H}_2$	Untargeted attack	Strategically-timed	Kos and Song [7]; Lin et al. [18]; Sun et al. [22]; Yang et al. [23]; Inkawhich et al. [24]
$\mathcal{H}_3$	Targeted attack	Full-timed	Pattanaik et al. [19]; Lin et al. [20]; Zhang et al. [8]

3.1 Reformulating SA-MDP in function spaces

In order to provide an in-depth understanding of the current methods, we reformulate SA-MDP in the function space of $\mathcal{H}$ by classifying the existing optimization-based attack algorithms into different types. As illustrated in Figure 2, given a pre-trained policy π as the victim policy, the adversary aims to minimize the expected total reward of π by applying the perturbations δ_{s_t} to state s_t as $h(s_t) = s_t + \delta_{s_t}$. Let $\mathcal{H}$ be the space of the adversary's function as

$$\mathcal{H} = \{h \big| \|h(s) - s\|_p \leqslant \epsilon, \ \forall s \in \mathcal{S}\}, \tag{3}$$

where the constant ϵ is the level of the l_p-norm adversarial noise that measures the ability of an adversary. We can reformulate problem (2) by solving the optimal function in the function space as $h^* \in \mathcal{H}$. Suppose a function as $\pi_h : \mathcal{S} \to \mathcal{F}(\mathcal{A})$ such that

$$\pi_h(a|s) \equiv \pi(a|h(s)), \tag{4}$$

which means an agent with policy $\pi_h(a|s)$ behaves similarly to the victim agent with policy π when the observed state is perturbed to $h(s)$. With our setting of attacker h and function space $\mathcal{H}$, we can derive the optimal function set $\mathcal{H}^*$ to generate the adversarial perturbation by rewriting problem (2):

$$\mathcal{H}^* = \left\{ h^* \ \middle| h^* = \arg\min_{h \in H} \left[R(\pi_h) \triangleq \mathbb{E}_{a \sim \pi_h} \sum_{t=0}^{\infty} \gamma^t r_t \right] \right\}, \tag{5}$$

where $R(\pi_h)$ is the expected total reward.

3.2 Categorizing adversaries in function spaces

In this part, we classify the existing optimization-based methods with three subspaces in the function space $\mathcal{H}$ as in Table 1 [6–8, 18–20, 22–24]. The first two types of subspaces refer to the untargeted adversaries that generate adversarial noise by only considering the current states. In the third subspace, we consider an adversary that is able to generate adversarial noise by specifying the targeted action or policy at each time step.

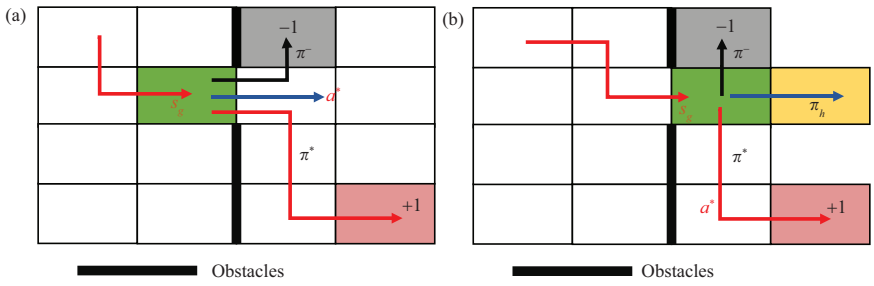

Figure 3 An example in Grid World with two obstacles for (a) $\mathcal{H}_1$ and (b) $\mathcal{H}_2$.

3.2.1 *The function space of full-timed untargeted attack of* $\mathcal{H}_1$

We start by introducing the adversary set $\mathcal{H}_1$, which misleads the agent to take sub-optimal actions at each state s by using untargeted adversaries as follows.

Definition 1. Assume Φ_u is the set of the untargeted attack algorithms, for $\forall \phi \in \Phi_u$; $\mathcal{H}_1$ is the function space to represent the adversary $h \in \mathcal{H}$ which generates the perturbed state for each $\phi \in \Phi_u$ as

$$\mathcal{H}_1 = \{h \in \mathcal{H} | h(s) \equiv \phi_\pi^\epsilon(s),\ \exists \phi \in \Phi_u,\ \forall s \in \mathcal{S}\},$$

where ϕ_π^ϵ is an instantiated mapping from the original state space $\mathcal{S}$ to the adversarial state space $\mathcal{S}$ parameterized by the noise level ϵ and the victim policy π, i.e., $\|\phi_\pi^\epsilon(s) - s\|_p \leqslant \epsilon$ holds for any state s.

As an untargeted attack algorithm, $\phi \in \Phi_u$ maximizes the divergence between the victim policy π and the attacked policy π_h at state s. For instance, as a representative work in $\mathcal{H}_1$ [6], the distance is evaluated by the cross-entropy loss between the attacked policy and the policy that takes the action a with the maximum probability of the victim policy. Ref. [8] further improve the methods by maximizing the KL divergence between the victim policy and the attacked policy with projected gradient descent (PGD) [13] or convex relaxations [25] of neural networks for each state. However, in some environments, any adversary in $\mathcal{H}_1$ may not be in $\mathcal{H}^*$, i.e., one cannot achieve the worst case of expected accumulated reward. We present Proposition 1 to show the weakness of $\mathcal{H}_1$.

Proposition 1 (Weakness of $\mathcal{H}_1$)**.** There exists an MDP such that $\forall \pi \in \Pi^*$, the optimal function h^* is not included in the space of $\mathcal{H}_1$, i.e., $\mathcal{H}^* \cap \mathcal{H}_1 = \emptyset$, where Π^* is the optimal policy set that can maximize the accumulated reward for an agent.

Proposition 1 is apparent and we provide a constructed example in Figure 3(a) for illustration. The agent gains reward -1 in the grey state, gains reward $+1$ in the red state, and 0 in other states. The grey state and the red state are the terminal states. The agent can only move right or down in the left part. Then the optimal policy (red arrow) $\pi^* \in \Pi^*$ must reach the green state s_g and move right with a^*. By Definition 1, the full-timed untargeted attacks aim to change the original actions as

$$\forall h_1 \in \mathcal{H}_1, \pi_{h_1}(a^*|s_g) < 1,$$

which prevents the victim from taking the minimum reward (reach the grey state). However, every worst-case policy with the flipped reward (black arrow) also reach the green state and move right, which means

$$\forall h^* \in \mathcal{H}^*, \pi_{h^*}(a^*|s_g) = 1.$$

Then $V^{\pi_{h_1}}(s_g) > V^{\pi_{h^*}}(s_g)$. From this example, we can illustrate that $\mathcal{H}^* \cap \mathcal{H}_1 = \emptyset$. From Proposition 1, we can see that it is not always necessary to generate noise at each time step to minimize the expected reward of the attacked policy. The second type of strategically-timed untargeted attacks can be more effective in this aspect, though they also suffer some limitations, as detailed below.

3.2.2 *The function space of strategically-timed untargeted attack of* $\mathcal{H}_2$

We characterize the second type of strategically-timed untargeted attack which can choose either to perturb the observation or to maintain the origin observation. Formally, we characterize this type as the second function space $\mathcal{H}_2$ as follows.

Definition 2. $\mathcal{H}_2$ is the function space to represent the adversary $h \in \mathcal{H}$ which only attacks the agent at a small subset of time steps strategically as

$$\mathcal{H}_2 = \big\{h \in \mathcal{H} | h(s) \equiv \mathbb{I}_a(s)\phi_\pi^\epsilon(s) + (1 - \mathbb{I}_a(s))s,\ \exists \phi \in \Phi_u,\ \forall s \in \mathcal{S}\big\},$$

where $\phi_\pi^\epsilon(s)$ is an instantiated mapping of ϕ parameterized by ϵ and π as in Definition 1, and $\mathbb{I}_a(s)$ indicates whether the adversary attack at state s or not, i.e., $\mathbb{I}_a(s) = 1$ when the adversary attacks at state s, otherwise $\mathbb{I}_a(s) = 0$.

There exist a few attempts in this direction [7, 18, 22, 23], which aim at reducing the frequency of attacks rather than enhancing the attacker's capabilities. These algorithms heuristically control whether to attack based on the current state s. Although the attacked policy in $\mathcal{H}_2$ has one more option at each state, i.e., it can act as the victim policy, this adversary may still not be in $\mathcal{H}^*$. We also present Proposition 2 to show the weakness of $\mathcal{H}_2$, which is similar to Proposition 1.

Proposition 2 (Weakness of $\mathcal{H}_2$)**.** There exists an MDP such that $\forall \pi \in \Pi^*$, the optimal function h^* is not included in the space of $\mathcal{H}_2$, i.e., $\mathcal{H}^* \cap \mathcal{H}_2 = \emptyset$, where Π^* is the optimal policy set that can maximize the accumulated reward for an agent.

Proof. Suppose there exist $|\Phi_u|$ attack algorithms. We provide a constructed example in Figure 3(b) for $|\Phi_u| = 1$, without losing generality. Since strategically-timed attacked policy (in Definition 2) is not related to the worst state in the environment, there exists an MDP in Proposition 2. According to Definition 6, there are three possible strategically-timed attacked policy at green state s_g: {right, down}, {up, down}, or {left, down}. In Figure 3(b), we suppose that the strategically-timed attacked policy moves right or down in green state s_g. The agent gains reward $+1$ in the red state and 0 in other states. The agent gains reward -1 in the grey state. The grey state and the red state are the terminal states. Then the optimal policy (red arrow) $\pi^* \in \Pi^*$ must reach the green state s_g and move right with a^*, i.e., $V^{\pi_{h_2}}(s_g) \geqslant 0$. However, every worst-case policy with the adversarial reward (black arrow) also reaches the grey state, i.e., $V^{\pi_{h_2}}(s_g) > V^{\pi_{h^*}}(s_g) = -1$. From this example, we can illustrate that $\mathcal{H}^* \cap \mathcal{H}_2 = \emptyset$. When the strategically-timed attacked policy at s_g is {top, down} or {left, down}, we can give another example by simply setting the yellow state as the worst case with reward -1.

3.2.3 *The function space of targeted attack of $\mathcal{H}_3$*

Compared to the untargeted attack algorithms, the targeted attack algorithms allows flexibility in specifying the target action or the target policy, which is categorized as the third function space $\mathcal{H}_3$.

Definition 3. Assume Φ_t is the set of all targeted attack algorithms, for $\forall \phi \in \Phi_t$, $\mathcal{H}_3$ is the function space to represent the adversary which generates the perturbed state $\phi_{\pi,\pi'}^\epsilon(s)$ at each state s as

$$\mathcal{H}_3 = \{h \in \mathcal{H} | h(s) \equiv \phi_{\pi,\pi'}^\epsilon(s),\ \exists \pi' \in \Pi_{\text{adv}},\ \forall s \in \mathcal{S}\},$$

where $\phi_{\pi,\pi'}^\epsilon$ is an instantiated mapping of ϕ parameterized by the noise level ϵ, the victim policy π and the target policy π' as in Definition 1, and Π_{adv} is a policy set that is accessible to the adversary.

As a targeted attack, the algorithm ϕ aims at finding an adversarial example with the adversarial noise level ϵ to minimize the distance between $\pi'(\cdot|s)$ and $\pi_h(\cdot|s)$. π' belongs to Π_{adv}, and Π_{adv} is accessible to the adversary without the adversarial optimizer. For example, $\exists \pi^* \in \Pi^*$, the optimal policy π^* satisfies: $\pi^* \in \Pi_{\text{adv}}$ when the environment is accessible to the adversary. Theoretically, it is also worth noting that the adversary can access a worst policy $\pi^- \in \Pi^-$ when the adversary can explore the environment with the flipped reward.

3.3 Discussion on the function spaces

In this subsection, we make a comprehensive analysis by finding the limitations of the present methods and providing insights on where and how to find optimal adversary h^* as follows.

3.3.1 *The function space $\mathcal{H}_3$ is generally larger than the function space $\mathcal{H}_1$ and $\mathcal{H}_2$*

We first investigate the advantages of the targeted adversaries. If $\pi_{\mathcal{H}_1}$, $\pi_{\mathcal{H}_2}$, and $\pi_{\mathcal{H}_3}$ are the policies set in the corresponding function space, we find the advantage of $\mathcal{H}_3$ as follows.

Theorem 3 (The advantage of $\mathcal{H}_3$)**.** With an appropriate noise level ϵ and policy set Π_{adv}, the policy set satisfy $\pi_{\mathcal{H}_1} \subseteq \pi_{\mathcal{H}_2} \subseteq \pi_{\mathcal{H}_3}$.

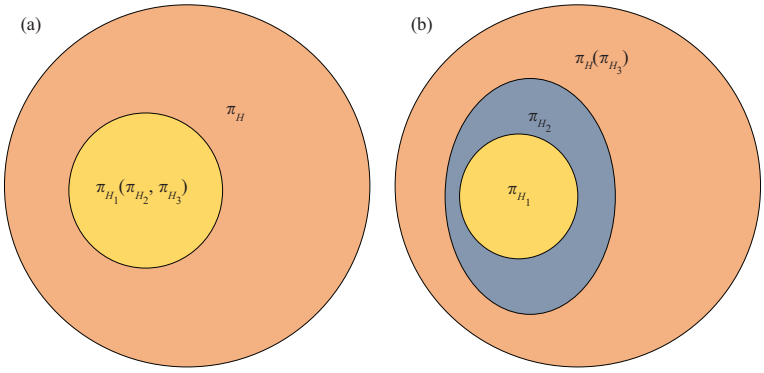

Figure 4 Relationship among the policy sets $\pi_{\mathcal{H}}$, $\pi_{\mathcal{H}_1}$, $\pi_{\mathcal{H}_2}$, and $\pi_{\mathcal{H}_3}$ with different noise levels ϵ. (a) $\epsilon = 0$, $\forall \Pi_{\text{adv}} \subseteq \Pi$; (b) $\epsilon \to \infty$, $\Pi_{\text{adv}} = \Pi$.

Proof. Figure 4 shows the relationship among these policy sets. Obviously, as $\mathcal{H}_1 \subseteq \mathcal{H}_2$, we always have $\pi_{\mathcal{H}_1} \subseteq \pi_{\mathcal{H}_2}$. Without the limitation on noise level ϵ and policy set Π_{adv}, $\pi_{\mathcal{H}_3}$ is equal to $\Pi_{\text{adv}} = \Pi = \pi_{\mathcal{H}}$. Therefore, we can get $\pi_{\mathcal{H}_2} \subseteq \Pi = \pi_{\mathcal{H}_3}$.

Therefore, we solve the where-to-find problem by finding the optimal solution h^* in $\mathcal{H}_3$, and next we will show how to solve it.

3.3.2 *Assumptions on the adversary in the function space $\mathcal{H}_3$*

After careful examinations on the present targeted attacks in the function space of $\mathcal{H}_3$, we find that the existing methods [8, 19, 20] implicitly follow a "pessimistic" assumption as follows.

Assumption 1. For the adversary $h \in \mathcal{H}_3$ and the victim policy π, the pessimistic assumption states:

$$Q^{\pi_h}(s, a) \approx Q^{\pi}(s, a), \tag{6}$$

where $Q^{\pi_h}(s, a)$ is the the Q-value function of the attacked policy π_h.

We call this assumption as the pessimistic assumption because it assumes that the performance of the attacked policy is close to the victim policy. In other words, we pessimistically assume the effect of the adversary. The objective function of existing approaches [19] is exactly the same as the elementary function in (5) by substituting Q^{π_h} with Q^{π}; and the objective function in [8] can be obtained by substituted Q^{π_h} into the elementary function in (5) with a robust Q-values function Q^{RS}.

However, the pessimistic assumption may be inappropriate since practically, the performance of the adversarial attack [8, 19] is close to the worst policy π^- rather than the victim policy π. Therefore, we present another reasonable "optimistic" assumption as follows.

Assumption 2. For the adversary $h \in \mathcal{H}_3$ and π^- is the worst policy, the optimistic assumption states

$$Q^{\pi_h}(s, a) \approx Q^{\pi^-}(s, a), \tag{7}$$

where Q^{π_h} and Q^{π^-} are Q-value functions for the attacked policy π_h and the worst policy π^-, respectively.

We call this assumption as the optimistic assumption because we assume that the performance of the attacked policy is close to the performance of the worst policy, which means we are optimistic to the effect of the adversary. We follow this optimistic assumption and further design a two-stage optimization method to solve this problem in the following, which is the first attempt following this assumption to the best of our knowledge.

4 Methodology

Following previous discussions, we now present an adversarial attack to obtain the adversarial policy in function space $\mathcal{H}_3$ under the optimistic assumption and the pseudo code is outlined in Algorithm 1.

Algorithm 1 Two-stage attack

Require: Victim policy π.

1: **Stage 1**: learn the deceptive policy π^- using RL algorithm (like proximal policy optimization (PPO)) to solve

$$\pi^- \in \arg\min_{\pi\in\Pi} R(\pi). \tag{8}$$

2: **Stage 2**: when observe current state s, calculate the adversarial noise via the deceptive policy

$$\hat{h}^*(s) = \arg\min_{h\in\mathcal{H}_3} D_{\mathrm{KL}}(\pi_h(\cdot|s)\|\pi^-(\cdot|s)), \tag{9}$$

and apply it to current observation.

4.1 Two-stage optimization

The original problem (5) is intractable to solve since the attacker is required to infer the environmental dynamics and the exploration mechanism in DRL, which inevitably leads to a shift in the distribution of the state. To address this issue, we first introduce a deceptive policy to explore the worst case in the environment that can minimize the accumulated reward. Moreover, we denote Π_d as the set of deceptive policies, which is formally defined as follows.

Definition 4 (The deceptive policy set of Π_d)**.** Π_d is a set of deceptive policies which can minimize the accumulated reward on an MDP as

$$\Pi_d = \left\{\pi|\pi = \arg\min_{\pi\ \in\Pi} R(\pi)\right\}.$$

Theoretically, it is noted that if the adversary can interact with the environment then the adversary can learn a deceptive policy, i.e., $\Pi_d \cap \Pi_{\mathrm{adv}} \neq \emptyset$. Moreover, the deceptive policy can also be specified if the adversary has some expert knowledge that can help the adversary to minimize the victim's reward. Following the optimistic assumption, we use Q-values $Q^{\pi^-}(s,a)$ of another deceptive policy $\pi^- \in \Pi_d \cap \Pi_{\mathrm{adv}}$ to estimate $Q^{\pi_h}(s,a)$. As policy $\pi^- \in \Pi_d$ such that $\pi^- = \arg\min_{\pi\in\Pi} \mathbb{E}_{a\sim\pi}\left[Q^{\pi}(s,a)\right]$, an optimal solution of $\pi_h = \arg\min_{\pi\in\Pi} \mathbb{E}_{a\sim\pi}[Q^{\pi^-}(s,a)]$ is $\pi_h = \pi^-$. Therefore, we reformulate problem (5) as minimizing the distance between π_h and π^- on a trajectory determined by policy π^-:

$$h^* = \arg\min_{h\in\mathcal{H}_3} \mathbb{E}_{s\sim d^{\pi^-}}\left[D_{\mathrm{TV}}(\pi_h(\cdot|s)\|\pi^-(\cdot|s))\right], \tag{10}$$

where $\pi^- \in \Pi_d \cap \Pi_{\mathrm{adv}}$ is the targeted policy to explore the worst case in the MDPs. Here, $D_{\mathrm{TV}}(\cdot\|\cdot)$ is the total variation (TV) distance between two policy distributions. Specifically, d^π is the distribution for future state under the policy as $d^\pi(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s|\pi)$. Since the deceptive policy set Π_d is a subset of Π, we can reduce the search space by only considering the policies trained with the flipped reward for the victim, yielding a more efficient optimization. Considering that the TV distance does not admit a closed-form expression for the statistical mixture distributions (e.g., GMM models) [26], it often is necessary to conduct Monte Carlo approximations or a numerical integration. Nevertheless, these operations do not guarantee the deterministic lower and upper bounds. To address this issue, we propose to use KL-divergence which can be used to upper bound the TV distance [27] and is widely used in the constraint for trust regions in RL algorithms [28]. In particular, we introduce a new objective with KL-divergence instead of problem (10) as

$$h^* = \arg\min_{h\in\mathcal{H}_3} \mathbb{E}_{s\sim d^{\pi^-}}\left[D_{\mathrm{KL}}(\pi_h(\cdot|s)\|\pi^-(\cdot|s))\right]. \tag{11}$$

The problem is also intractable because the state-distribution is generated by π_h which is in a complex function space. Therefore, we reformulate it as: (1) A two-stage optimization problem. By inferring the deceptive policy as an intermediate step:

$$\hat{h}^*(s) = \arg\min_{h\in\mathcal{H}_3} D_{\mathrm{KL}}(\pi_h(\cdot|s)\|\pi^-(\cdot|s)),\ \forall s\in\mathcal{S}, \tag{12}$$

where $\pi^- \in \Pi_d \cap \Pi_{\mathrm{adv}}$ is the intermediate deceptive policy. Afterwards, we directly solve problem (12) in function space $\mathcal{H}_3$ by defining the target policy as π^-. (2) Solving the two-stage optimization problem.

Note that when the policy π^- is given, problem (12) can be seen as a typical targeted attack in supervised learning. The adversary independently adds perturbation on each state s and treats $\pi^-(\cdot|s)$ as a target function. This is different from previous work [6], which is an untargeted attack. In this paper, we propose to use the PGD [13], which is a popular method for targeted attacks. The adversary iteratively updates the observation as

$$s_{k+1} = s_k - \eta,$$

where k is the number of iterations, $s_0 = s$ is the original observation, and η is the perturbation. The perturbation is based on the type of norm. For the commonly used l_2-norm, the perturbation can be computed by the PGD on the negative loss of (12) as

$$\eta = \frac{\epsilon' \sqrt{|s|} \nabla_s (D_{\mathrm{KL}}(\pi(\cdot|s_k) \| \pi^-(s)))}{||\nabla_s (D_{\mathrm{KL}}(\pi(\cdot|s_k) \| \pi^-(\cdot|s)))||_2},$$

where ϵ' is the step size to control the distance between the resultant and the original observations.

4.2 Theoretical analysis

We now present a theoretical analysis in a practical setting that provides a performance bound for our algorithm. First, we define the α-deceptive policy set $\Pi^-(\alpha)$ to express a class of policies, after which we use this set to measure the adversary's capability.

Definition 5 (The α-deceptive policy set $\Pi^-(\alpha)$)**.** An α-deceptive policy set $\Pi^-(\alpha)$ corresponds to the policies that are α-close to the performance of $\pi^- \in \Pi_d$ as

$$\Pi^-(\alpha) = \{\pi | R(\pi) \leqslant R(\pi^-) + \alpha, \pi \in \Pi\}.$$

With the definition of α-deceptive policy set $\Pi^-(\alpha)$, we further define the ability of an adversary with parameter $\hat{\alpha}$.

Definition 6 (The $\hat{\alpha}$-adversary)**.** Let Π_{adv} be a policy set that is accessible to the adversary, an adversary is an $\hat{\alpha}$-adversary if

$$\hat{\alpha} = \inf\{\alpha | \Pi_{\mathrm{adv}} \cap \Pi^-(\alpha) \neq \emptyset\}.$$

It is noted that the adversary is a 0-adversary when it can minimize the reward (or reach the worst case state) by exploring the environments or obtaining the environmental dynamics from an expert. With the definition of the adversary's capability $\hat{\alpha}$, we further provide an upper bound of the performance after an attack by an $\hat{\alpha}$-adversary.

Lemma 1 (Upper bound of the $\hat{\alpha}$-adversary's performance)**.** Let the adversary be an $\hat{\alpha}$-adversary, the performance of the perturbed policy π_h satisfies

$$R(\pi_h) \leqslant \hat{\alpha} + \frac{C\beta_1}{1-\gamma} + \frac{2\gamma C\sqrt{\beta_0/2}}{(1-\gamma)^2} + R(\pi^-),$$

where $\beta_0 = \max_{s \in S} \|D_{\mathrm{KL}}(\pi_h(\cdot|s) \| \pi^-(\cdot|s))\|$, $C = \max_s |\mathbb{E}_{a \sim \pi_h}[A^{\pi^-}(s,a)]|$ and $\beta_1 = \max_{s,a} \|\frac{\pi_h(a|s)}{\pi^-(a|s)} - 1\|$, A is the advantage function.

The complete derivation is provided in Appendix A. Lemma 1 implies that the performance of the adversarial attack decreases as the adversary's capability $\hat{\alpha}$ and the distance between policy π_h and π^- decrease. We further show that under an appropriate noise level, our attacked method can be stronger than the existing sub-optimal adversary, as summarized in Theorem 4.

Theorem 4 ($\hat{\alpha}$-adversary is stronger than other adversary under some conditions)**.** Let e be an arbitrary adversarial attack algorithm, and we set $\alpha_e = R(\pi_e) - R(\pi^-)$ and $\beta_1 = \max_{s,a} \|\frac{\pi_h(a|s)}{\pi^-(a|s)} - 1\|$. If β_1 satisfies

$$\beta_1 < \frac{-\sqrt{2}\gamma C + \sqrt{2\gamma^2 C^2 + 4(\alpha_e - \hat{\alpha})(1-\gamma)^3}}{2(1-\gamma)C},$$

then the performance of the victim policy after perturbed by our algorithm satisfies $R(\pi_h) < R(\pi_e)$. Here C follows the definition in Lemma 1.

The proof of Theorem 4 is provided in Appendix A. Based on this analysis, we reduced β_1 by targeted attacks in our experiments and obtained promising results in both Atari and MuJoCo environments.

5 Experiments

In this section, we conduct extensive experiments to evaluate and show the effectiveness of our method.

5.1 Experimental setup

Implementation details for MuJoCo. First, we evaluate the effectiveness of the adversarial attacks on four OpenAI Gym MuJoCo continuous environments, namely Ant, Hopper, Walker2d, and HalfCheetah, using the implementation and pretrained models in [9].

Implementation details for Atari. Further, we conduct experiments to evaluate the adversarial attacks on four Atari games: Pong, BeamRider, Qbert, and SpaceInvaders. The victim policies are mainly trained by three classic RL algorithms, namely deep Q-network (DQN) [1, 29], advantage actor-critic (A2C) [30], and PPO [31]. We use DQN and A2C implementations in [32] and PPO implementation in [33]. All these policies achieve competitive performance in the Atari environments.

Details of deceptive policies. We train five deceptive policies to evaluate the two-stage adversarial attacks. For PPO victims, we train the deceptive policy by PPO, while for A2C and DQN victims, we train the deceptive policies by A2C. To reduce the variance of the two-stage attack performance, we evaluate the two-stage attacks by choosing the policy with the median reward in the five deceptive policies. All deceptive policies are trained with the same hyper-parameters as their victims.

Additional details. In MuJoCo environments, we use the released victim policies in [9]. In Atari environments, the victim policy is trained with GeForce GTX 1080Ti 11G and Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40 GHz. We train 10 M steps for DQN, which needs fewer than three days each. We also train 40 M steps for A2C, which needs no more than one day each. Furthermore, we train 10 M steps for PPO, which needs no more than one day each. The deceptive policy is trained for 1 M steps each in Atari environments and 1000 steps each in MuJoCo environments, which are significantly fewer than the number of steps required for the victim policy. It takes less than a minute to generate the adversarial observation in all Atari environments by the two-stage attack (fast gradient sign method, FGSM) and less than two minutes to generate the adversarial observation in Pong, SpaceInvaders, and Qbert, as well as less than 10 min in BeamRider by the two-stage attack (PGD). In MuJoCo environments, it takes less than a minute for the two-stage attack (PGD) for each trajectory. Note that we use the KL-divergence between the attacked and deceptive policies plus the entropy of the attacked policy to generate a smoother attacked policy.

Evaluation on MuJoCo environments. For comparison sake, we evaluate our methods against SOTA representative works following [9], including random attacks, the stochastic maximal action difference (MAD) attack [8], and snooping attack [31] for untargeted attacks, along with the critic attack [19] and robust Sarsa (RS) attack [8] for targeted attacks. As deterministic MAD does not apply to the continuous action space environment, we follow previous work [8] and just evaluate stochastic MAD. We also compare our method with a SOTA learning-based attack, the optimal attack [9]. For fair comparisons, we choose a similar setup in [9]. Particularly, we consider the perturbation in l_∞-norm and run the agents without attacks, as well as under attacks for 50 episodes, and report the mean and standard deviation of the episode reward.

Evaluation on Atari environments. As learning-based methods are inefficient in the Atari environment, we evaluate the vulnerability of the victim policy with random attacks and three optimization-based attacks, including the deterministic MAD attack [6] and stochastic MAD attack [8] for untargeted attacks, as well as the critic attack targeted attacks [19]. For fair comparisons, we consider a similar setup as the previous work. Particularly, we use perturbation of l_2-norm as in [6, 9, 19], and run the agents without attacks, as well as under attacks for five episodes during testing.

5.2 Experimental results

In this experiment, we study the vulnerability of the victim policy to the proposed two-stage attack. We generate adversarial perturbations with the deceptive policy by two common adversarial optimizers: the FGSM [12] and the PGD [13]. Specifically, we generate adversarial perturbations with the deceptive policy by PGD with ten iterations.

Results in MuJoCo environments. Table 2 presents the results on the attacking PPO agents in MuJoCo environments. In most tasks, the two-stage approach markedly outperforms all other baselines. First, as a targeted attack, the two-stage attack achieves a lower reward than two untargeted attacks,

Table 2 The average reward of the victim policy (PPO) under adversarial attack on MuJoCo[a)]

Adversary	Ant	Hopper	Walker2d	HalfCheetah
Noise level of ϵ	0.15	0.07	0.15	0.15
No noise	5861.10 ± 609.63	3290.41 ± 397.13	4491.23 ± 674.98	7102.41 ± 121.03
Random	5528.39 ± 609.63	2850.41 ± 797.75	4275.17 ± 871.11	5336.31 ± 1574.15
Critic	5071.47 ± 970.93	2090.47 ± 1025.50	4131.07 ± 639.36	5584.03 ± 879.24
Stochastic MAD	1648.23 ± 820.80	1868.66 ± 686.15	901.84 ± 475.67	1798.58 ± 1014.83
RS	412.87 ± 247.87	2808.47± 871.20	1299.01 ± 307.40	338.81 ± 526.71
Snooping	3978.19 ± 658.76	1832.97 ± 562.45	1265.88 ± 782.60	1967.98 ± 1077.90
"Optimal" attack	−493.22 ± 40.49	637.30 ± 3.32	879.85 ± 36.14	**−657.60 ± 288.10**
Our two-stage	**−841.01 ± 494.88**	**112.49 ± 148.64**	**−29.93 ± 7.35**	−314.65 ± 54.75

a) We bold the best attack reward (the lowest attacked policy's reward) over all attacks.

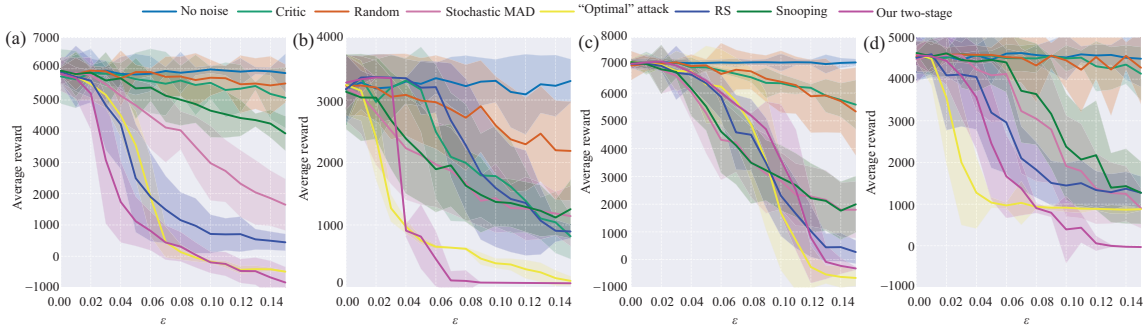

Figure 5 Attacking PPO agents in MuJoCo environments under different ϵ noise level. Each color in this figure represents the results of 50 episodes. The shadowed area is the variance of the return of episodes. In the remaining part, we omit the description of the shadowed area as they all represent the variance of the return of episodes. (a) Ant; (b) Hopper; (c) HalfCheetah; (d) Walker2d.

Table 3 The average reward of the victim policy (DQN) under adversarial attack on Atari[a)]

Adversary	Optimizer	Qbert	Pong	SpaceInvaders	BeamRider
Noise level of ϵ	–	0.007	0.004	0.012	0.014
No noise	–	9402.1 ± 1719.2	20.5 ± 0.6	509.0 ± 135.3	2251.5 ± 198.6
Random	–	7380.0 ± 542.8	19.9 ± 1.2	216.8 ± 61.9	976.2 ± 440.3
Critic	FGSM	192.5 ± 110.1	−19.0 ± 2.8	101.0 ± 46.3	134.2 ± 81.5
	PGD(10)	157.5 ± 60.5	−19.7 ± 1.6	96.0 ± 86.9	149.6 ± 76.0
Deterministic	FGSM	275.0 ± 169.4	−20.6 ± 0.3	79.5 ± 33.7	189.2 ± 59.8
MAD	PGD(10)	210.0 ± 206.3	**−21.0 ± 0.0**	171.8 ± 23.1	211.2 ± 33.5
Stochastic	FGSM	246.3 ± 82.3	−18.3 ± 3.3	111.3 ± 59.9	451.0 ± 118.0
MAD	PGD(10)	237.5 ± 110.1	−20.9 ± 0.1	189.5 ± 33.4	525.8 ± 83.2
Our two-stage	FGSM	193.8 ± 85.1	**−21.0 ± 0.0**	47.0 ± 26.5	308.0 ± 121.1
	PGD(10)	**13.8 ± 13.9**	**−21.0 ± 0.0**	**0.0 ± 0.0**	**6.6 ± 7.3**

a) We bold the best attack reward (the lowest attacked policy's reward) over all attacks.

the stochastic MAD and snooping attacks. It indicates that targeted attacks are much stronger than untargeted attacks in optimization-based algorithms. Besides, compared to targeted attacks such as the critic and RS attacks following the pessimistic assumption, our two-stage attack under the optimistic assumption can significantly reduce the reward of the attacked policy. It indicates that we can find a more effective adversary with the optimistic assumption. Additionally, our optimization-based two-stage attack outperforms the learning-based optimal attack in most environments, indicating that we can design an effective attack by leveraging an efficient optimizer. Figure 5 shows that the lowest performance of the attacked victim policy achieves a lower reward than other attacked policies. Additionally, our algorithm misleads the agent to the opposite direction (reward < 0) in Walker2d, indicating that our attack is stronger than all previous attacks.

Results in Atari environments. Tables 3–5 present the results on the attacking DQN, A2C, and PPO agents, indicating that our two-stage attack significantly outperforms the alternative methods in the Atari environments. Particularly, for the Atari environments SpaceInvaders and BeamRider, as a targeted attack, our two-stage attack achieves significantly lower reward than two untargeted attacks of

Table 4 The average reward of the victim policy (A2C) under adversarial attack on Atari[a)]

Adversary	Optimizer	Qbert	Pong	SpaceInvaders	BeamRider
Noise level of ϵ	–	0.006	0.008	0.014	0.014
No noise	–	13564.0 ± 1716.7	21.0 ± 0.0	634.2 ± 148.3	15222.1 ± 2797.3
Random	–	14394.0 ± 1085.6	21.0 ± 0.0	459.2 ± 129.0	14938.8 ± 1813.5
Critic	FGSM	168.0 ± 108.7	**−21.0 ± 0.0**	150.0 ± 79.2	207.7 ± 76.2
	PGD(10)	437.0 ± 103.6	**−21.0 ± 0.0**	148.4 ± 97.8	248.2 ± 48.3
Deterministic	FGSM	193.0 ± 132.3	**−21.0 ± 0.0**	109.4 ± 25.0	279.8 ± 25.7
MAD	PGD(10)	412.0 ± 168.9	−21.0 ± 0.1	140.4 ± 76.3	416.3 ± 102.2
Stochastic	FGSM	156.0 ± 75.6	**−21.0 ± 0.0**	183.0 ±98.6	529.3 ± 75.5
MAD	PGD(10)	341.0 ± 107.4	−20.8 ± 0.3	132.8 ± 70.8	586.4 ± 62.4
Our two-stage	FGSM	**47.0 ± 39.7**	**−21.0 ± 0.0**	80.4 ± 60.6	455.4 ± 130.3
	PGD(10)	301.0 ± 88.4	**−21.0 ± 0.0**	**26.6 ± 34.4**	**188.3 ± 127.0**

a) We bold the best attack reward (the lowest attacked policy's reward) over all attacks.

Table 5 The average reward of the victim policy (PPO) under adversarial attack on Atari[a)]

Adversary	Optimizer	Qbert	Pong	SpaceInvaders	BeamRider
Noise level of ϵ	–	0.002	0.001	0.014	0.008
No noise	–	16999.0 ± 2008.7	20.6 ± 0.3	952.2 ± 229.4	1873.4 ± 771.1
Random	–	16275.0 ± 1002.5	20.7 ± 0.4	768.6 ± 186.3	1788.2 ± 368.7
Stochastic	FGSM	331.0 ± 122.3	−18.5 ± 0.6	209.4 ± 135.5	526.0 ± 43.8
MAD	PGD(10)	340.0 ± 130.8	−19.0 ± 1.0	172.4 ± 31.6	508.0 ± 74.0
Our two-stage	FGSM	**127.0 ± 90.3**	**−21.0 ± 0.0**	173.0 ± 34.9	**364.5 ± 153.7**
	PGD(10)	271.0 ± 67.0	**−21.0 ± 0.0**	**166.0 ± 38.4**	371.4 ± 162.6

a) In each line we bold the best attack reward (the lowest attacked policy's reward) over all attacks.

the stochastic MAD and deterministic MAD attacks. These results show that it is better to use the targeted attack algorithm to generate the perturbation. Also, despite the fact that the critic attack and our two-stage attack are all targeted attacks, our two-stage attack under the optimistic assumption significantly outperforms the critic attack with the same optimizer on the attacking DRL agents in most environments. It indicates that we can adopt the optimistic assumption rather than the pessimistic assumption. Generally, adversaries with PGD are stronger than those with FGSM, indicating that a more powerful optimizer leads to a more powerful adversary. Figures 6–8 show that the performance of the attacked victim policy trained by DQN, A2C, and PPO is lower than other attacked policies under most of the noise level. This trend indicates that our approach highly applies to different noise levels. Besides, the lowest performance of the attacked policy under the two-stage attack is lower than all other adversaries in all settings. This result indicates the adversary follows the optimistic assumption and is stronger than other adversaries. Additionally, the deceptive policy is trained for 1 M steps each in the Atari environments and 1000 steps each in MuJoCo environments, which is significantly fewer than the number of steps required for the victim policy (e.g., 40 M steps for the A2C and 10 M steps for the PPO in Atari). Our optimization-based attacks generate the adversarial noise efficiently in Atari, which is a high-dimensional space, indicating the efficiency of our methods. However, the learning-based attacks are required to learn a high-dimensional mapping in the MDP over the state and action space described (e.g., the dimension is > 6000 in Atari), which is intractable to generally solve.

Nearly lowest reward in Atari environments. Our attack achieves nearly optimal performance on the attacking DQN agents in Table 3, which demonstrates the rationality of our optimistic assumption on the effectiveness of our adversary. Figures 9–11 show the rewards of the attacked policy under a two-stage attack and the best performance under other attacks with different noise levels. Besides, we show the lowest reward for each task. Evidently, our adversary achieves nearly the lowest reward when attacking DQN and A2C agents. Besides, it achieves competitive performance when attacking the PPO agents. These results also demonstrate the rationality of our optimistic assumption.

6 Conclusion

We reformulate the SA-MDP in the function space and provide a framework to categorize and understand the existing optimization-based adversarial attacks on RL. We show that the adversary should be solved in

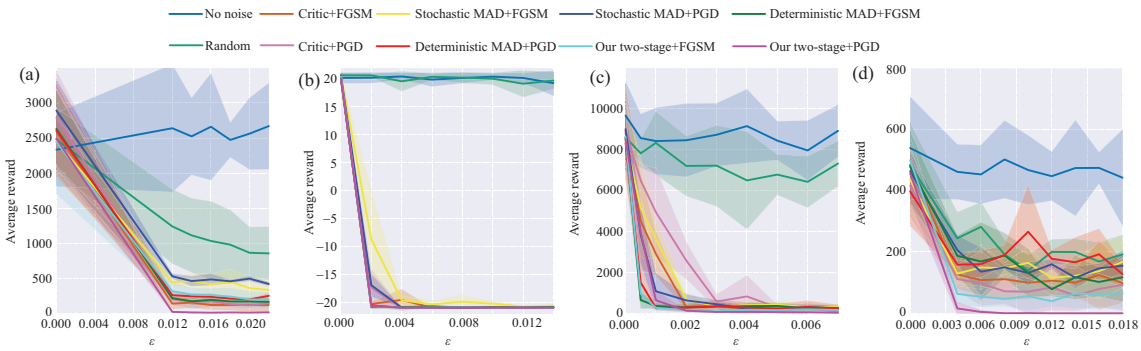

Figure 6 Attacking DQN agents in Atari environments under different ϵ noise level. Each color in this figure represents the results of 50 episodes. (a) BeamRider; (b) Pong; (c) Qbert; (d) SpaceInvaders.

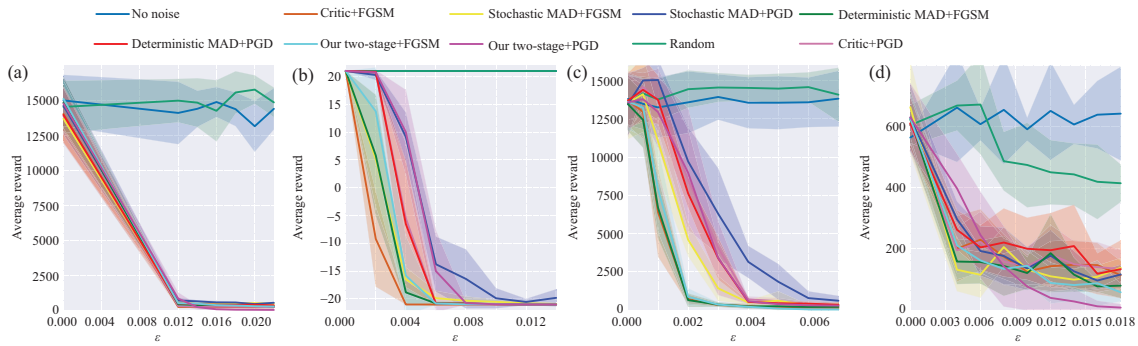

Figure 7 Attacking A2C agents in Atari environments under different ϵ noise level. Each color in this figure represents the results of 50 episodes. (a) BeamRider; (b) Pong; (c) Qbert; (d) SpaceInvaders.

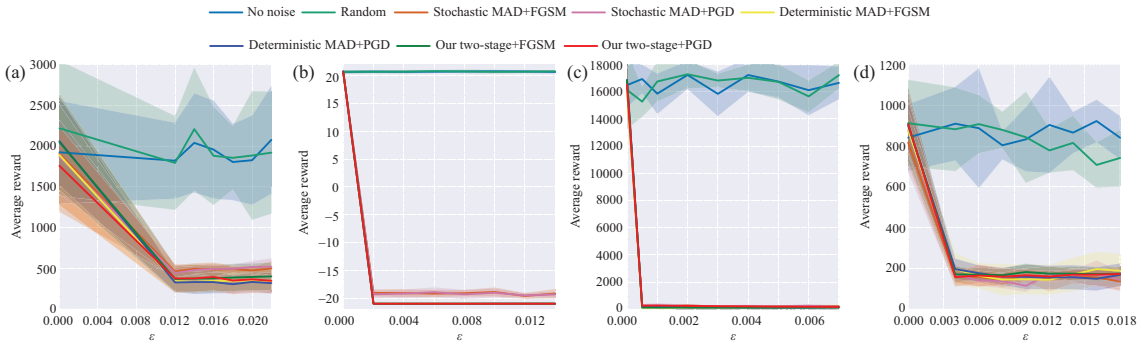

Figure 8 Attacking PPO agents in Atari environments under different ϵ noise level. Each color in this figure represents the results of 50 episodes. (a) BeamRider; (b) Pong; (c) Qbert; (d) SpaceInvaders.

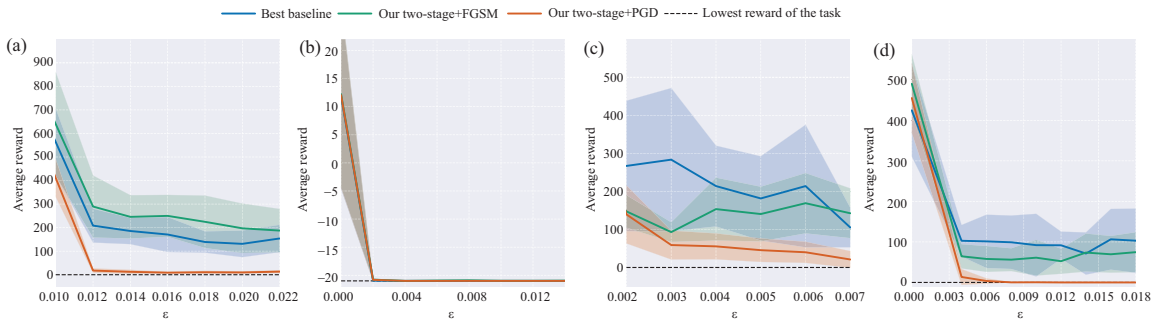

Figure 9 Attacking DQN agents in Atari environments under different ϵ noise level. The attacked policy under our two-stage attack achieves the nearly lowest reward. Each color in this figure represents the results of 50 episodes. (a) BeamRider; (b) Pong; (c) Qbert; (d) SpaceInvaders.

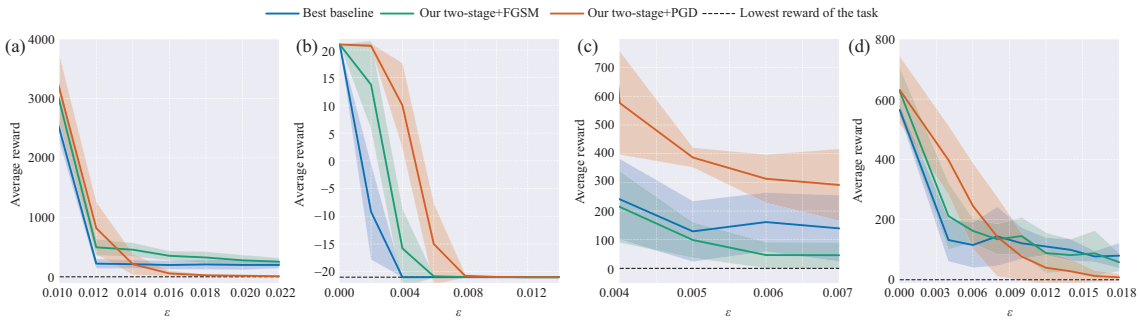

Figure 10 Attacking A2C agents in Atari environments under different ϵ noise level. The attacked policy under our two-stage attack achieves the nearly lowest reward. Each color in this figure represents the results of 50 episodes. (a) BeamRider; (b) Pong; (c) Qbert; (d) SpaceInvaders.

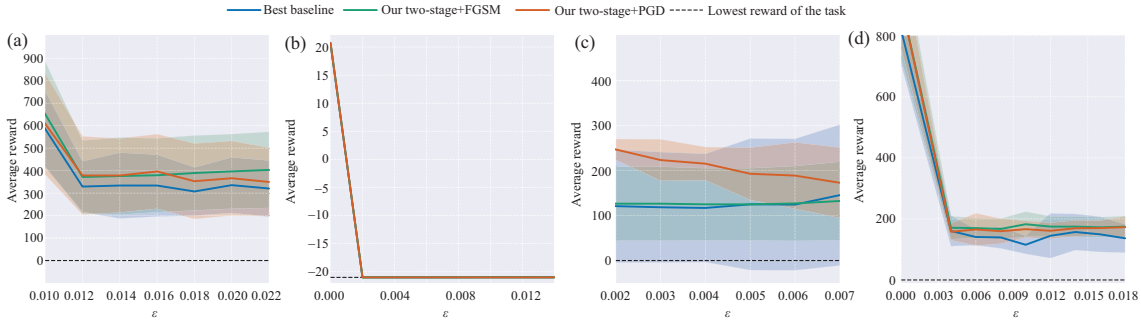

Figure 11 Attacking PPO agents in Atari environments under different ϵ noise level. Comparing to the other baselines, the reward of the attacked policy under our two-stage attack is closer to the lowest reward in each task. Each color in this figure represents the results of 50 episodes. (a) BeamRider; (b) Pong; (c) Qbert; (d) SpaceInvaders.

the function space of targeted attacks following the optimistic assumption. Based on our understanding, we propose a two-stage method that can effectively and efficiently generate adversarial noise on the RL observation. Extensive experiments in both Atari and MuJoCo environments show the superiority of our method, which provides a possible way to assess the adversarial robustness of RL.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant Nos. 2020AAA0104304, 2017YFA0700904), National Natural Science Foundation of China (Grant Nos. 61620106010, 62061136001, 61621136008, 62076147, U19B2034, U1811461, U19A2081), Beijing NSF Project (Grant No. JQ19016), Beijing Academy of Artificial Intelligence (BAAI), Tsinghua-Huawei Joint Research Program, Tsinghua Institute for Guo Qiang, Tsinghua-OPPO Joint Research Center for Future Terminal Technology and Tsinghua-China Mobile Communications Group Co., Ltd. Joint Institute.

Supporting information Appendix A. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Mnih V, Kavukcuoglu K, Silver D, et al. Playing Atari with deep reinforcement learning. 2013. ArXiv:1312.5602
2 Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518: 529–533
3 Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning. In: Proceedings of International Conference on Machine Learning (ICML), 2016. 1928–1937
4 Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529: 484–489
5 Schrittwieser J, Antonoglou I, Hubert T, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. Nature, 2020, 588: 604–609
6 Huang S, Papernot N, Goodfellow I, et al. Adversarial attacks on neural network policies. 2017. ArXiv:1702.02284
7 Kos J, Song D. Delving into adversarial attacks on deep policies. 2017. ArXiv:1705.06452
8 Zhang H, Chen H G, Xiao C W, et al. Robust deep reinforcement learning against adversarial perturbations on state observations. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2020. 21024–21037
9 Zhang H, Chen H G, Boning D S, et al. Robust reinforcement learning on state observations with learned optimal adversary. In: Proceedings of International Conference on Learning Representations (ICLR), 2021
10 Biggio B, Corona I, Maiorca D, et al. Evasion attacks against machine learning at test time. In: Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD), 2013. 387–402
11 Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks. In: Proceedings of International Conference onLearning Representations (ICLR), 2014
12 Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. 2014. ArXiv:1412.6572

13 Madry A, Makelov A, Schmidt L, et al. Towards deep learning models resistant to adversarial attacks. In: Proceedings of International Conference on Learning Representations (ICML), 2018
14 Dong Y P, Liao F Z, Pang T Y, et al. Boosting adversarial attacks with momentum. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 9185–9193
15 Xiao C W, Pan X L, He W, et al. Characterizing attacks on deep reinforcement learning. 2019. ArXiv:1907.09470
16 Mandlekar A, Zhu Y K, Garg A, et al. Adversarially robust policy learning: active construction of physically-plausible perturbations. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robotsand Systems (IROS), 2017. 3932–3939
17 Russo A, Proutiere A. Optimal attacks on reinforcement learning policies. 2019. ArXiv:1907.13548
18 Lin Y C, Hong Z W, Liao Y H, et al. Tactics of adversarial attack on deep reinforcement learning agents. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), 2017. 3756–3762
19 Pattanaik A, Tang Z, Liu S, et al. Robust deep reinforcement learning with adversarial attacks. In: Proceedings of the 17th International Conference on Autonomous Agents and Multi Agent Systems (AAMAS), 2018. 2040–2042
20 Lin J Y, Dzeparoska K, Zhang S Q, et al. On the robustness of cooperative multi agent reinforcement learning. In: Proceedings of IEEE Security and Privacy Workshops (SPW), 2020. 62–68
21 Ying C, Zhou X, Su H, et al. Towards safe reinforcement learning via constraining conditional value-at-risk. 2022. ArXiv:2206.04436
22 Sun J W, Zhang T W, Xie X, et al. Stealthy and efficient adversarial attacks against deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2020
23 Yang C H H, Qi J, Chen P Y, et al. Enhanced adversarial strategically-timed attacks against deep reinforcement learning. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020. 3407–3411
24 Inkawhich M, Chen Y, Li H. Snooping attacks on deep reinforcement learning. In: Proceedings of the 19th International Conference on Autonomous Agents and Multi Agent Systems (AAMAS), 2020. 557–565
25 Zhang H, Weng T W, Chen P Y, et al. Efficient neural network robustness certification with general activation functions. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2018. 4939–4948
26 Nielsen F, Sun K. Guaranteed deterministic bounds on the total variation distance between univariate mixtures. In: Proceedings of the 28th International Workshop on Machine Learning for Signal Processing (MLSP), 2018. 1–6
27 Achiam J, Held D, Tamar A, et al. Constrained policy optimization. In: Proceedings of International Conference on Machine Learning (ICML), 2017. 22–31
28 Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization. 2017. ArXiv:1707.06347
29 Horgan D, Quan J, Budden D, et al. Distributed prioritized experience replay. In: Proceedings of International Conference on Learning Representations (ICLR), 2018
30 Konda V R, Tsitsiklis J N. Actor-critic algorithms. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2000. 1008–1014
31 Schulman J, Levine S, Abbeel P, et al. Trust region policy optimization. In: Proceedings of International Conference on Machine Learning (ICML), 2015. 1889–1897
32 Oikarinen T, Weng T W, Daniel L. Robust deep reinforcement learning through adversarial loss. 2020. ArXiv:2008.01976
33 Kostrikov I. PyTorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail