Who Is the Strongest Enemy? Towards Optimal and Efficient Evasion Attacks in Deep RL

Anonymous Author(s)
Affiliation
Address
email

Abstract

Evaluating the worst-case performance of a reinforcement learning (RL) agent under the strongest/optimal adversarial perturbations on state observations (within some constraints) is crucial for understanding the robustness of RL agents. However, finding the optimal adversary is challenging, in terms of both whether we can find the optimal attack and how efficiently we can find it. Existing works on adversarial RL either use heuristics-based methods that may not find the strongest adversary, or directly train an RL-based adversary by treating the agent as a part of the environment, which can find the optimal adversary but may become intractable in a large state space. In this paper, we propose a novel attacking algorithm which has an RL-based "director" searching for the optimal policy perturbation, and an "actor" crafting state perturbations following the directions from the director (i.e. the actor executes targeted attacks). Our proposed algorithm, PA-AD, is theoretically optimal against an RL agent and significantly improves the efficiency compared with prior RL-based works in environments with large or pixel state spaces. Empirical results show that our proposed PA-AD universally outperforms state-of-the-art attacking methods in a wide range of environments. Our method can be easily applied to any RL algorithms to evaluate and improve their robustness.

1 Introduction

Although Deep Reinforcement Learning (DRL) achieves incredible success in many applications, some recent works [7, 18, 30, 29] reveal that a well-trained RL agent may be vulnerable to *evasion attacks*, making it risky to deploy RL models in high-stakes applications. As in most related works, we consider a *state adversary* which adds imperceptible noise to the state observations of an agent such that the cumulative reward gained by the agent is reduced during test time.

In order to understand the vulnerability of an RL agent and to improve its certified robustness, it is important to evaluate the worst-case performance of the agent under any adversarial attacks with certain constraints. In other words, it is crucial to find the strongest/optimal adversary that can minimize the cumulative reward gained by the agent with fixed constraints.

However, finding the strongest adversary in RL is challenging. Many existing works directly apply evasion attack methods in supervised learning (SL) to RL, although RL is fundamentally different from SL due to the correlation in data samples and the uncertainty in environments. For example, Huang et al. [7], although pave the road for the first steps towards understanding evasion attacks in RL, treat every decision-making step similarly as a classification task, and perturb each state input so that the agent is

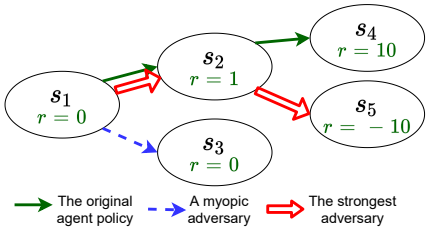

Figure 1: An Example that a myopic adversary is not the strongest.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

less likely to choose the action it would have selected without the perturbation. This type of attack is "myopic" since it does not plan for the future, and may not result in the strongest attack in many cases. Figure 1 shows an intuitive example, where a myopic adversary fails to let the agent obtain the lowest possible cumulative reward, since it does not "see" the "worst" state s_5 in the next step.

The computational efficiency arises as another challenge in practice, even if the strongest adversary can be identified in theory. A recent work [30] points out that learning the optimal state adversary is equivalent to learning an optimal policy in a new Markov Decision Process (MDP). A follow-up paper by Zhang et al. [29] shows that a learned optimal adversary significantly outperforms prior works in four MuJoCo games. However, the state space and the action space of the new MDP are both as large as the state space in the original environment, which is usually high-dimensional. For example, video games and autonomous driving systems use images as observations. In these environments, learning the state adversary directly as in Zhang et al. [29] becomes computationally intractable.

To overcome the two challenges above, **we propose a novel attack method called Policy Adversarial Actor Director (PA-AD)**. We design a "director" and an "actor" that collaboratively identify the optimal state perturbations. The director uses RL methods to learn the optimal perturbing directions in the policy space, and the actor perturbs states according to the director's directions (similar to a targeted attacker in SL). The director learns an MDP named *Policy Adversary MDP (PAMDP)*. Theoretical analysis shows that the optimal policy in PAMDP induces an optimal state adversary. Our PAMDP is more compact than the adversarial MDP defined by Zhang et al. [30, 29] and thus easier to be solved. With our proposed *director-actor collaborative mechanism*, PA-AD outperforms state-of-the-art evasion RL methods on various types of environments, including Atari games and MuJoCo tasks as shown in the experiments, verifying the optimality and efficiency of our method.

Summary of Contributions (1) We establish a theoretical understanding of the optimality of evasion attacks from the perspective of *policy perturbation*, allowing a more efficient implementation of optimal attacks. (2) Inspired by theoretical insights, we propose a novel attack method PA-AD which learns the optimal adversary efficiently. PA-AD is a general method that works on stochastic and deterministic victim policies, vectorized and pixel state spaces, as well as discrete and continuous action spaces. (3) Empirical study shows that PA-AD produces stronger attacks than previous attacking methods on various environments, and PA-AD can be combined with off-the-shelf robust RL training approaches to improve the robustness of RL agents.

2 Preliminaries and Notations

The Victim RL Agent In RL, an agent interacts with an environment modeled by a Markov Decision Process (MDP) denoted as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ is a state space with cardinality $|\mathcal{S}|$, $\mathcal{A}$ is an action space with cardinality $|\mathcal{A}|$, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function [1], $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. In this paper, we consider a setting where the state space is much larger than the action space, which arises in a wide variety of environments. For notation simplicity, our theoretical analysis focuses on a finite MDP, but our algorithm applies to continuous state spaces and continuous action spaces, as verified in experiments. The agent takes actions according to its *policy*, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. We suppose the victim uses a fixed policy π with a function approximator (e.g. a neural network) during test time. We denote the *space of all policies* as Π, which is a Cartesian product of $|\mathcal{S}|$ simplices. The *value* of a policy $\pi \in \Pi$ for state $s \in \mathcal{S}$ is defined as $V^\pi(s) = \mathbb{E}_{\pi,P}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s]$.

Evasion Attacker Evasion attacks are test-time attacks that aim to reduce the expected total reward gained by the agent/victim. As in most literature [7, 18, 30], we assume the attacker knows the victim policy π. However, the attacker does not know the environment dynamics, nor does it have the ability to change the environment directly. The attacker can observe the interactions between the victim agent and the environment, including states, actions and rewards. We focus on a typical *state adversary* [7, 18, 4, 30], which perturbs the state observations returned by the environment before the agent observes them. Note that the underlying states in the environment are not changed.

Formally, we model a state adversary by a function h which perturbs state $s \in \mathcal{S}$ into $\tilde{s} := h(s)$, so that the input to the agent's policy is $\tilde{s}$ instead of s. The amount of perturbation is usually small so that the attacks are hard to be perceived. In this paper, we consider the common ℓ_p-norm ball

[1] $\Delta(X)$ denotes the the space of probability distributions over X.

constraint: $\tilde{s}$ should be in $\mathcal{B}_\epsilon(s)$, where $\mathcal{B}_\epsilon(s)$ denotes an ℓ_p norm ball centered at s with radius ϵ (ϵ is a non-negative constant called the *budget* of the adversary). With the budget constraint, we define the *admissible state adversary* and the *admissible adversary set* as below.

Definition 1 (**Set of Admissible State Adversaries** H)**.** *A state adversary h is said to be admissible if $\forall s \in \mathcal{S}, h(s) \in \mathcal{B}_\epsilon(s)$. The set of all admissible state adversaries is denoted by H.*

Then the goal of the attacker is to find an adversary h^* in H that maximally reduces the cumulative reward of the agent. In this work, we propose a novel method to learn the optimal state adversary through the identification of an optimal policy perturbation defined and motivated in the next section.

3 Understanding the Optimal State Adversary via Policy Perturbations

We first motivate our idea of interpreting evasion attacks as perturbations of policies in Section 3.1 . Then in Section 3.2, we propose to find the optimal state adversary via the optimal policy perturbation.

3.1 Evasion Attacks Are Perturbations of Policies

Although existing literature usually considers state attacks and action attacks separately, we point out that evasion attacks, no matter applied to states or actions, are essentially equivalent to perturbing the agent's policy π into another policy $\pi_h \in \Pi$. For instance, as shown in Figure 2, if the adversary h alters state s into state $\tilde{s}$, then the victim selects an action $\tilde{a}$ based on distribution $\pi(\cdot|\tilde{s})$, the policy at the perturbed state $\tilde{s}$. This is equivalent to directly perturbing $\pi(\cdot|s)$ to $\pi_h(\cdot|s) := \pi(\cdot|\tilde{s})$. (See Appendix B for more detailed analysis including action adversaries.)

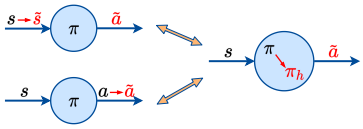

Figure 2: Equivalence between evasion attacks and policy perturbations.

In this paper, we aim to find the optimal state adversary through the identification of the "optimal policy perturbation", which has the following **merits**. **(1)** $\pi_h(\cdot|s)$ usually lies in a lower dimensional space than $h(s)$ for an arbitrary state $s \in \mathcal{S}$. For example, in Atari games, the action space is discrete and small (e.g. $|\mathcal{A}| = 18$), while a state is an image of dimension $C \times H \times W$, where C, H, W are the number of channels, the height and the width of the image, respectively. Then the state perturbation $h(s)$ has dimension $C \times H \times W$, much higher than the corresponding policy perturbation $\pi_h(\cdot|s)$ which has dimension $|\mathcal{A}|$. **(2)** It is easier to characterize the optimality of a policy perturbation than a state perturbation. How a state perturbation changes the value of the victim policy depends on the victim's policy and the environment dynamics, while the relationship between a policy perturbation and the value perturbation only depends on the environment. Our Theorem 4 in Section 3.2 and Theorem 11 in Appendix C both provide insights about how V^π changes as π changes continuously. **(3)** Policy perturbation captures the essence of evasion attacks, and unifies state and action attacks. Although this paper focuses on state-space adversaries, the learned "optimal policy perturbation" can also be used to conduct action-space attacks against the same victim.

3.2 Characterizing the Optimal Policy Adversary

As depicted in Figure 3, the policy perturbation serves as a bridge connecting the perturbations in the state space and the value space. In this section, we characterize how the policy changes as the state is perturbed, and how the value changes when the policy is perturbed. Then, we introduce the idea of finding the optimal state adversary by identifying the optimal "policy adversary".

We first define an Admissible Adversarial Policy Set (Adv-policy-set) $\mathcal{B}^H(\pi) \subset \Pi$ as the set of policies perturbed from π by all admissible state adversaries $h \in H$. In other words, when a state adversary perturbs states within an ℓ_p norm ball $\mathcal{B}_\epsilon(\cdot)$, the victim policy is perturbed within $\mathcal{B}^H(\pi)$.

Definition 2 (**Admissible Adversarial Policy Set (Adv-policy-set)** $\mathcal{B}^H(\pi)$)**.** *For an MDP $\mathcal{M}$, a fixed victim policy π, we define the admissible adversarial policy set (Adv-policy-set) w.r.t. π, denoted by $\mathcal{B}^H(\pi)$, as the set of policies that are perturbed from π by all admissible adversaries, i.e.,*

$$\mathcal{B}^H(\pi) := \{\pi_h \in \Pi : \exists h \in H \text{ s.t } \forall s, \pi_h(\cdot|s) = \pi(\cdot|h(s))\}. \tag{1}$$

Remarks (1) For a continuous function π (e.g., neural network), $\mathcal{B}^H(\pi)$ is connected and compact as in Figure 3(middle) (proven in Appendix C.2). (2) The value functions generated by all policies in the Adv-policy-set $\mathcal{B}^H(\pi)$ form a polytope (see Theorem 11 in Appendix C), as in Figure 3(right).

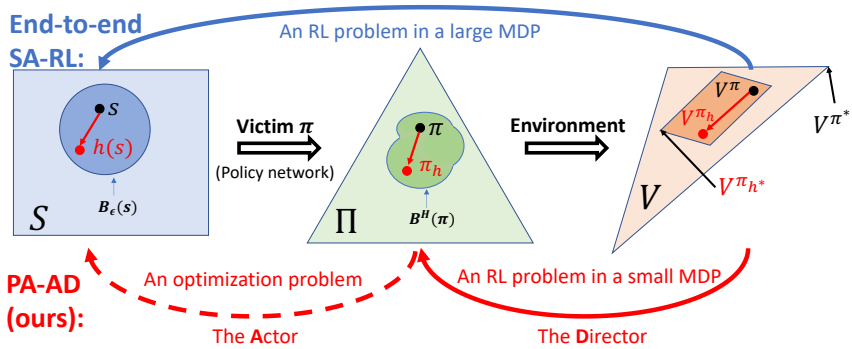

Figure 3: A state adversary h perturbs s into $h(s) \in \mathcal{B}_\epsilon(s)$ in the state space; hence, the victim's policy π is perturbed into π_h within the Adv-policy-set $\mathcal{B}^H(\pi)$; as a result, the expected total reward the victim can gain becomes V^{π_h} instead of V^π. Finding the optimal adversary h^* with the minimal $V^{\pi_{h^*}}$ is an RL problem. A prior work SA-RL [29] directly uses an RL agent to learn the best state adversary, which works for MDPs with small state spaces, but suffers from high complexity in larger MDPs. In this paper, we find the optimal state adversary h^* through identifying the optimal policy adversary π_{h^*}. By decoupling the attack problem into an RL part (finding the policy π_{h^*} with the lowest value) and a non-RL part (perturbing the state to let the victim execute π_{h^*}), our proposed attack method called PA-AD achieves both optimal and efficient attacks.

Given that the Adv-policy-set $\mathcal{B}^H(\pi)$ contains all the possible policies the victim may execute under admissible state perturbations, we can characterize the optimality of a state adversary through the lens of policy perturbations. Recall that the attacker's goal is to find a state adversary $h^* \in H$ that minimizes the victim's expected total reward. From the perspective of policy perturbation, the attacker's goal is to perturb the victim's policy to another policy $\pi_{h^*} \in \mathcal{B}^H(\pi)$ with the lowest value. Therefore, we can define the optimal state adversary and the optimal policy adversary as below.

Definition 3 (**Optimal State Adversary h^* and Optimal Policy Adversary π_{h^*}**). *For an MDP $\mathcal{M}$, a fixed policy π, and an admissible adversary set H,*
*(1) an **optimal state adversary** h^* satisfies $h^* \in \operatorname{argmin}_{h \in H} V^{\pi_h}(s), \forall s \in \mathcal{S}$, which leads to*
*(2) an **optimal policy adversary** π_{h^*} satisfies $\pi_{h^*} \in \operatorname{argmin}_{\pi_h \in \mathcal{B}^H(\pi)} V^{\pi_h}(s), \forall s \in \mathcal{S}$.*
Recall that π_h is the perturbed policy caused by adversary h, i.e., $\pi_h(\cdot|s) = \pi(\cdot|h(s)), \forall s \in \mathcal{S}$.

Definition 3 implies **an equivalent relationship between the optimal state adversary and the optimal policy adversary**: an optimal state adversary leads to an optimal policy adversary, and any state adversary that leads to an optimal policy adversary is optimal. Theorem 18 in Appendix E.1 shows that there always exists an optimal policy adversary for a fixed victim π, and learning the optimal policy adversary is an RL problem. (A similar result have been shown by Zhang et al. [30] for the optimal state adversary, while we focus on the policy perturbation.)

Due to the equivalence, one can first find an optimal policy adversary π_{h^*}, then search for the state adversary that can change the victim policy π into π_{h^*}. However, finding the optimal policy adversary in the Adv-policy-set $\mathcal{B}^H(\pi)$ is still challenging since $\mathcal{B}^H(\pi)$ is generated by all admissible state adversaries in H and is hard to compute. Then, how can we find the optimal policy adversary?

The optimal policy adversary can be found without fully characterizing the Adv-policy-set. Although it is impractical to directly compute the Adv-policy-set $\mathcal{B}^H(\pi)$, Theorem 4 shows that the "outermost boundary" of $\mathcal{B}^H(\pi)$ always contains an optimal policy adversary. Intuitively, a policy π' is in the outermost boundary of $\mathcal{B}^H(\pi)$ if and only if no policy in $\mathcal{B}^H(\pi)$ is farer away from π than π' in the direction $\pi' - \pi$. Therefore, if an adversary can perturb a policy along a direction, it should push the policy as far away as possible in this direction under the budget constraints. Then, the adversary is guaranteed to find an optimal policy adversary after trying all the perturbing directions. In contrast, such a guarantee does not exist for state adversaries, justifying the benefits of considering policy adversaries. Our proposed algorithm in Section 4 applies this idea to find the optimal attack: an RL-based director searches for the optimal perturbing direction, and an actor is responsible for pushing the policy to the outermost boundary of $\mathcal{B}^H(\pi)$ with a given direction.

Theorem 4. *For an MDP $\mathcal{M}$, a fixed policy π, and an admissible adversary set H, define the **outermost boundary** of the admissible adversarial policy set $\mathcal{B}^H(\pi)$ w.r.t π as*

$$\partial_\pi \mathcal{B}^H(\pi) := \{\pi' \in \mathcal{B}^H(\pi) : \forall s \in \mathcal{S}, \theta > 0, \nexists \hat{\pi} \in \mathcal{B}^H(\pi) \text{ s.t. } \hat{\pi}(\cdot|s) = \pi'(\cdot|s) + \theta(\pi'(\cdot|s) - \pi(\cdot|s))\}. \quad (2)$$

Then there exists a policy $\tilde{\pi} \in \partial_\pi \mathcal{B}^H(\pi)$, such that $\tilde{\pi}$ is the optimal policy adversary w.r.t. π.

We provide the proof of Theorem 4 in Appendix C.3, and demonstrate some examples of the outermost boundary in Figure 5 in Appendix C.5.

4 PA-AD: Optimal and Efficient Evasion Attack

In this section, we summarize some existing evasion RL methods with a focus on their optimality and efficiency. Then, we introduce our algorithm, *Policy Adversarial Actor Director (PA-AD)* that efficiently finds the optimal adversary for a fixed victim policy.

Efficient heuristic attacks are not optimal. Section 3 shows that finding the optimal adversary is an RL problem. Heuristic attacks [7, 18, 30] craft adversarial states at every step separately, without considering the correlations among steps. For instance, Zhang et al. [30] propose to maximize the difference between $\pi(\cdot|s)$ and $\pi(\cdot|\tilde{s})$, where $\tilde{s} \in \mathcal{B}_\epsilon(s)$ is the perturbed state. With this objective, the attacker can use gradient-based methods to find $\tilde{s}$ efficiently. However, due to the uncertainty in the environment, it is not guaranteed that $\pi(\cdot|\tilde{s})$ will lead the agent to gain a low cumulative reward. In Appendix E.4, we categorize existing heuristic attack methods into four types, and prove that there exist scenarios where these heuristic methods may not find the strongest adversary.

An optimal end-to-end RL attacker might be inefficient. In order to find stronger adversaries, a recent paper [29] proposes to learn the optimal state adversary using RL methods, which we will refer to as *SA-RL* in our paper for simplicity. SA-RL can be viewed as an "end-to-end" RL attacker, as it directly learns the optimal state adversary such that the value of the victim policy is minimized. The adversary of SA-RL takes the original state s as the input, and outputs a constrained state perturbation $\tilde{s} \in \mathcal{B}_\epsilon(s)$ as the action. Whenever the victim agent receives a reward r, the adversary gets a reward $-r$. In this way, if the adversary maximizes its own cumulative reward, then the agent's cumulative reward is minimized. SA-RL reduces the optimal attack problem to learning an MDP whose state space and action space are both the same as the original state space. However, if the state space is high-dimensional (e.g. images), then the state space and action space of the new MDP are both large, making it prohibitive to learn a good policy, as empirically shown in Section 6.1.

We use an optimal and more efficient way: RL + SL. An end-to-end RL attacker treats the victim agent and the environment together as a black-box environment. Although the attacker need not to know the policy network since it is a part of the black-box, the complexity of learning the black-box environment becomes too high for a large state space. We claim that the black-box large environment can be simplified by a proper disentanglement when the policy network is known to the attacker. As demonstrated in Figure 3, during an attack process, a state perturbation leads to a policy perturbation, and then the policy perturbation results in a value perturbation; only the latter process depends on the environment dynamics and requires learning, while the former process is similar to evasion attacks in a supervised learning problem and does not require learning. Therefore, if we disentangle the former process from the black-box environment, the adversary can learn from a more compact environment with much lower complexity.

Based on the above motivation, we propose a novel algorithm, *Policy Adversarial Actor Director (PA-AD)*, that decouples the whole attacking process into two simpler components: policy perturbation

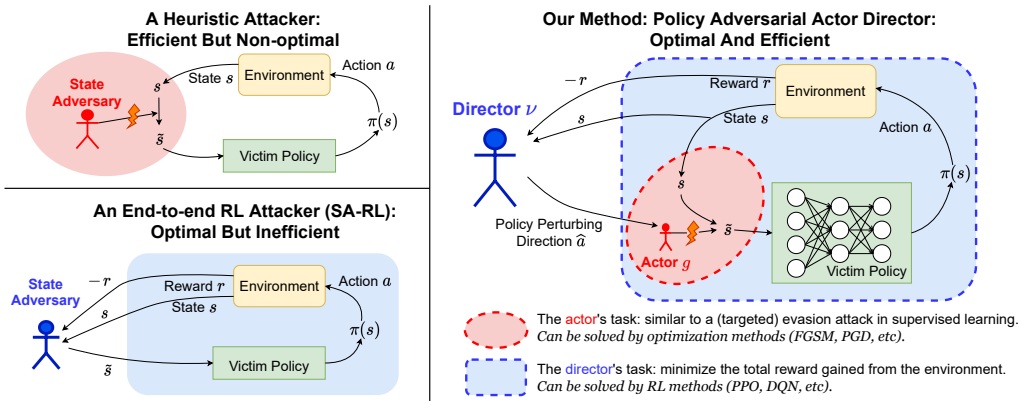

Figure 4: An overview of PA-AD compared with a heuristic attacker and an end-to-end RL attacker. Heuristic attacks are efficient, but may not find the optimal adversary as they do not learn from the environment dynamics. An end-to-end RL attacker directly learns a policy to generate state perturbations, but is inefficient in large-state-space environments. In contrast, our **PA-AD** solves the attack problem with a combination of an RL-based director and a non-RL actor, so that PA-AD achieves both optimality and efficiency.

and state perturbation, solved by a "director" and an "actor", respectively. The director learns the optimal policy perturbing direction with RL methods, while the actor crafts adversarial states at every step such that the victim policy is perturbed towards the given direction.

Formally, for a stochastic victim policy π, the director learns an RL policy ν in a *Policy Adversary MDP (PAMDP)* defined in Definition 5, while the actor denoted by g is embedded in the dynamics of PAMDP, mapping a target direction and the current state to a perturbed state.

Definition 5 (**Policy Adversary MDP (PAMDP)** $\widehat{\mathcal{M}}$)**.** *Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, a fixed stochastic victim policy π, a budget $\epsilon \geq 0$, we define a Policy Adversarial MDP $\widehat{\mathcal{M}} = \langle \mathcal{S}, \widehat{\mathcal{A}}, \widehat{P}, \widehat{R}, \gamma \rangle$, where the action space is $\widehat{\mathcal{A}} = \{d \in [0,1]^{|\mathcal{A}|}, \sum_{i=1}^{|\mathcal{A}|} d_i = 0\}$, and $\forall s, s' \in \mathcal{S}, \forall \widehat{a} \in \widehat{\mathcal{A}}$,*

$$\widehat{P}(s'|s,\widehat{a}) = \sum\nolimits_{a \in \mathcal{A}} \pi(a|g(\widehat{a},s))P(s'|s,a), \quad \widehat{R}(s,\widehat{a}) = -\sum\nolimits_{a \in \mathcal{A}} \pi(a|g(\widehat{a},s))R(s,a).$$

The actor function g is defined as

$$g(\widehat{a},s) = \operatorname{argmax}_{s' \in B_\epsilon(s)} \|\pi(s') - \pi(s)\| \text{ subject to } \big(\pi(s') - \pi(s)\big)^T \widehat{a} = \|\pi(s') - \pi(s)\| \|\widehat{a}\|. \quad (G)$$

How do the director and the actor collaborate in PAMDP? The input to director ν is the current state s in the original environment, while its output $\widehat{a}$ is a signal to the actor denoting "which direction to perturb the victim policy into". The actor, once receiving the perturbing direction $\widehat{a}$, will "push" the policy as far as possible by perturbing s to $g(\widehat{a}, s) \in \mathcal{B}_\epsilon(s)$, as characterized by the optimization problem (G). Therefore, the director and the actor together induce a state adversary: $h(s) := g(\nu(s), s), \forall s \in \mathcal{S}$. In this way, the policy perturbation resulted by the director and the actor is always in the outermost boundary of $\mathcal{B}^H(\pi)$ w.r.t. the victim π, where the optimal policy perturbation can be found according to Theorem 4.

Optimality of PA-AD Theorem 6 (proven in Appendix E.3) shows that the optimal solutions to PA-AD are optimal adversaries against the victim policy π.

Theorem 6 (**Optimality of PA-AD**)**.** *For any MDP $\mathcal{M}$, a fixed victim policy π, and any attack budget $\epsilon \geq 0$, an optimal policy ν^* in $\widehat{\mathcal{M}}$ induces an optimal state adversary against π in $\mathcal{M}$.*

Efficiency of PA-AD As commonly known, the sample complexity of solving an RL problem grows with the cardinalities of the state space and the action space. Both SA-RL and PA-AD have state space $\mathcal{S}$, the state space of the original MDP. But the action space of SA-RL is also $\mathcal{S}$, while our PA-AD has action space $\mathbb{R}^{|\mathcal{A}|-1}$. In many applications, the state space (e.g. a pixel space) is in a higher dimensional space than $\mathbb{R}^{|\mathcal{A}|-1}$, then PA-AD is more sample efficient than SA-RL.

Implementation Details of PA-AD The actor function defined in (G) requires solving a constrained optimization problem. But in practical implementations, we can relax the actor's optimization to ($\hat{G}$) (a Lagrangian relaxation) to efficiently get an approximation of the optimal adversary.

$$\operatorname{argmax}_{s' \in B_\epsilon(s)} L(s'; \widehat{a}, s) := \|\pi(\cdot|s') - \pi(\cdot|s)\| + \lambda \times \mathsf{CosineSim}\big(\pi(\cdot|s') - \pi(\cdot|s),\ \widehat{a}\big) \quad (\hat{G})$$

where $\mathsf{CosineSim}$ in the second term refers to the cosine similarity function, measuring the alignment of two vectors; the first term measures how far away the policy is perturbed; λ is a hyper-parameter controlling the trade-off between the two terms (we set λ as 1 in experiments). The procedure of learning the optimal adversary is depicted in Algorithm 1, where we simply use the Fast Gradient Sign Method (FGSM) [5] to approximately solve ($\hat{G}$), although more advanced solvers such as Projected Gradient Decent (PGD) can be applied to further improve the performance. Experiment results in Section 6 verify that the above implementation achieves state-of-the-art attack performance.

Variants of PA-AD: for deterministic victims and continuous action spaces The PAMDP in Definition 5 is for a general stochastic victim, which takes action a that is sampled from distribution $\pi(\cdot|s)$ at state s. It is generally more challenging to attack a stochastic victim than a deterministic victim due to the randomness of the action sampling, which is often ignored in existing methods. We claim that our PA-AD can be further simplified for a deterministic victim policy, and can be extended to environments with continuous action spaces. For a deterministic victim policy, we define the Deterministic Policy Adversary MDP (D-PAMDP) in Definition 16 in Appendix D.1, where the action space of the director is the same as the original action space of the victim, i.e., $\mathcal{A}$. In short, in D-PAMDP, the director proposes a target action $\widehat{a}_D \in \mathcal{A}$, and the actor conducts targeted attacks to let the victim output the target $\widehat{a}_D$. When the action space is continuous, the actor instead minimizes the distance between the policy action and the target action, i.e., $\operatorname{argmax}_{s' \in B_\epsilon(s)} \|\pi(s') - \widehat{a}_D\|$. *In the deterministic case, Theorem 6 (the optimality of PA-AD) still holds*, as proven in Appendix E.3.2.

Moreover, since the action space of D-PAMDP is $\mathcal{A}$, smaller than the action space of the stochastic PAMDP, *the efficiency of PA-AD is further improved in the deterministic case*. Appendix D provides more details and formal definitions of the aforementioned variants of our method. In Section 6, we show experimental results for deterministic victim agents in continuous action spaces.

Algorithm 1: Policy Adversarial Actor Director (PA-AD)

Input: Initialization of director's policy ν; victim policy π; budget ϵ; start state s_0
for $t = 0, 1, 2, ...$ **do**
  Director samples a policy perturbing direction $\widehat{a}_t \sim \nu(\cdot|s_t)$
  Actor sets $\tilde{s}_t = s_t + \epsilon \cdot \text{sign}(\nabla_\delta L(s_t + \delta; \widehat{a}_t, s_t))$ #$L()$ is defined in Equation ($\widehat{G}$)
  Victim takes action $a_t \sim \pi(\cdot|\tilde{s}_t)$, proceeds to $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$, receives $r_t = R(s_t, a_t)$
  Director saves $(s_t, \widehat{a}_t, -r_t, s_{t+1})$ to its buffer
  Director updates its policy ν using any RL algorithm

5 Related Work

Heuristic-based Evasion Attacks on States There are many works considering evasion attacks on the state observations in RL. Huang et al. [7] first propose to use FGSM [5] to craft adversarial states such that the probability that the agent selects the "best" action is minimized. The same objective is also used in a recent work by Korkmaz [9], which adopts a Nesterov momentum-based optimization method to further improve the attack performance. Pattanaik et al. [18] propose to lead the agent to select the "worst" action based on the victim's Q function and use gradient descent to craft state perturbations. Zhang et al. [30] define the concept of a state-adversarial MDP (SAMDP) and propose two attack methods: Robust SARSA (RS) attack that forces the agent to choose actions with minimal Q values, with a learned stable Q function, and Maximal Action Difference attack that maximizes the difference between the perturbed policy and the victim policy. The above heuristic-based methods are shown to be effective in many environments. However, in Appendix E.4, our theoretical analysis shows the formulation of the above heuristic methods may not be optimal.

RL-based Evasion Attacks on States As discussed in Section 4, SA-RL [29] uses an end-to-end RL formulation to learn the optimal state adversary. However, this end-to-end RL formulation is difficult to solve when the state space of the original environment is large. Although Russo et al. [21] propose that one can use feature extraction to convert the pixel state space to a small state space, such feature extractions require expert knowledge and are hard to obtain in many real-world applications. In contrast, our PA-AD simplifies the RL problem with a more compact PAMDP, and learns the optimal state perturbations without any prior knowledge.

Other Works Related to Adversarial RL There are many other papers studying adversarial RL from different perspectives, including limited-steps attacking [13, 10], multi-agent scenarios [4], limited access to data [8], and etc. Adversarial action attacks [28, 24, 25, 12] are developed separately from state attacks; although we mainly consider state adversaries, our PA-AD can be easily extended to action-space attacks as formulated in Appendix B. Poisoning [1, 23] is another type of adversarial attacks that manipulates the training process, different from evasion attacks that deprave a well-trained policy. Training a robust agent is the focus of many recent works [19, 3, 14, 17, 30, 29]. Although our main goal is to identify a strong attacker, we also show by experiments that our proposed attack method can be incorporated into robust training methods to improve the robustness of RL agents.

6 Experiments

In this section, we show that PA-AD produces stronger evasion attacks than state-of-the-art attack algorithms on various OpenAI Gym environments, including Atari and MuJoCo tasks. Also, our experiment justifies that PA-AD can evaluate and improve the robustness of RL agents.

We compare our proposed attack algorithm with existing evasion attack methods, including **MinBest** [7] which minimizes the probability that the agent chooses the "best" action, **MinBest +Momentum** [9] which uses Nesterov momentum to improve the performance of MinBest, **MinQ** [18] which leads the agent to select actions with the lowest action values based on the agent's Q network, **Robust SARSA (RS)** [30] which performs the MinQ attack with a learned stable Q network, **MaxDiff** [30] which maximizes the KL-divergence between the original victim policy and the perturbed policy, as well as **SA-RL** [29] which directly learns the state adversary with RL methods.

	Environment	Natural Reward	ϵ	Random	MinBest [7]	MinBest + Momentum [9]	MinQ [18]	MaxDiff [30]	SA-RL [29]	PA-AD (ours)
DQN	**Boxing**	96 ± 4	0.001	95 ± 4	55 ± 11	60 ± 14	69 ± 11	82 ± 9	94 ± 5	$\mathbf{21 \pm 14}$
	Pong	21 ± 0	0.0002	21 ± 0	-10 ± 4	-11 ± 4	8 ± 5	15 ± 3	20 ± 1	$\mathbf{-19 \pm 2}$
	RoadRunner	39212 ± 7025	0.0005	39998 ± 7272	16855 ± 6719	16855 ± 6719	7021 ± 12951	33482 ± 6908	39603 ± 7404	$\mathbf{336 \pm 3249}$
	Freeway	34 ± 1	0.0003	34 ± 1	12 ± 1	12 ± 1	12 ± 1	21 ± 1	33 ± 1	$\mathbf{9 \pm 3}$
	Seaquest	10650 ± 2716	0.0005	8530 ± 3162	3837 ± 1544	3093 ± 1295	6985 ± 1065	6226 ± 2660	7985 ± 3002	$\mathbf{2253 \pm 1280}$
	Alien	1623 ± 252	0.00075	1652 ± 382	802 ± 465	649 ± 423	1119 ± 486	1039 ± 342	1638 ± 512	$\mathbf{523 \pm 242}$
	Tutankham	227 ± 29	0.00075	182 ± 77	36 ± 22	31 ± 20	7 ± 10	108 ± 62	209 ± 65	$\mathbf{1 \pm 1}$
A2C	**Breakout**	356 ± 79	0.0005	355 ± 79	86 ± 104	74 ± 95	N/A	304 ± 111	353 ± 79	$\mathbf{44 \pm 62}$
	Seaquest	1752 ± 70	0.005	1752 ± 73	356 ± 153	179 ± 83	N/A	46 ± 52	1752 ± 71	$\mathbf{4 \pm 13}$
	Pong	20 ± 1	0.0005	20 ± 1	-4 ± 8	-11 ± 7	N/A	18 ± 3	20 ± 1	$\mathbf{-13 \pm 6}$
	Alien	1615 ± 601	0.001	1629 ± 592	1062 ± 610	940 ± 565	N/A	1482 ± 633	1661 ± 625	$\mathbf{507 \pm 278}$
	Tutankham	258 ± 53	0.001	260 ± 54	139 ± 26	134 ± 28	N/A	196 ± 34	260 ± 54	$\mathbf{71 \pm 47}$
	RoadRunner	34367 ± 6355	0.002	35851 ± 6675	9198 ± 3814	5410 ± 3058	N/A	31856 ± 7125	36550 ± 6848	$\mathbf{2773 \pm 3468}$

Table 1: Average episode rewards $\pm$ standard deviation of vanilla DQN and A2C agents under different evasion attack methods in Atari environments. Results are averaged over 1000 episodes. Note that RS works for continuous action spaces, thus is not included. MinQ is not applicable to A2C which does not have a Q network. In each row, we bold the strongest (best) attack performance over all attacking methods.

We consider state attacks within an ℓ_∞ norm ball around the original state with radius ϵ as in most literature [30, 29]. In this section, we demonstrate results with the most representative ϵ for every environment (i.e., ϵ is neither too large nor too small, so that the difference among attackers is prominent), while Figure 10 in Appendix F.2.1 shows the results over multiple ϵ's in each environment. More hyperparameter settings and implementation details are deferred to Appendix F.1.

6.1 PA-AD Finds the Strongest Adversaries

Performance on Video Games (Atari) We first measure the performance of PA-AD against well-trained DQN [16] and A2C [15] victim agents on Atari games with pixel state spaces. The ϵ-bounded attacks are applied to normalized pixel values in the range of $[0, 1]$. For DQN, we take the softmax of the Q values $Q(s, \cdot)$ as the victim policy $\pi(\cdot|s)$ as in prior works [7]. The adversaries are learned with the ACKTR algorithm [27]. As is common in prior works, our implementation of the RL algorithms for both the victim and the attacker is mostly a proof of concept, thus many advanced training techniques are not included (e.g. Rainbow DQN).

Table 1 presents the experiment results, where we can see PA-AD significantly and universally outperforms all baselines against both DQN and A2C victims. For example, PA-AD reduces the reward of the DQN victim in Pong by over 190%, while the strongest baseline reduces the reward by 152%. Note that in Atari games, SA-RL works similarly to a random attacker, as the adversary is learning an MDP with both the state space and the action space being image spaces. In contrast, PA-AD learns a strong adversary efficiently since the action space of PAMDP is relatively small. In Appendix F.2.1, we show more experiments with various selections of the budget ϵ, where one can see *PA-AD reduces the average reward more than all baselines over varying ϵ's in various environments.*

Performance on Continuous Control Tasks (MuJoCo) We further evaluate PA-AD on MuJoCo games, where both state spaces and action spaces are continuous. We use the same setting with Zhang et al. [29], where both the victim and the adversary are trained with PPO [22]. During test time, the victim executes a deterministic policy, and we use the deterministic version of PA-AD with a continuous action space, as discussed in Section 4 and Appendix D.2.

Environment	State Dimension	Natural Reward	ϵ	Random	MaxDiff [30]	RS [30]	SA-RL [29]	PA-AD (ours)
Hopper	11	3167 ± 542	0.075	2101 ± 793	1410 ± 655	794 ± 238	636 ± 9	$\mathbf{160 \pm 136}$
Walker	17	4472 ± 635	0.05	3007 ± 1200	2869 ± 1271	1336 ± 654	1086 ± 516	$\mathbf{804 \pm 130}$
HalfCheetah	17	7117 ± 98	0.15	5486 ± 1378	1836 ± 866	489 ± 758	$\mathbf{-660 \pm 218}$	-356 ± 307
Ant	111	5687 ± 758	0.15	5261 ± 1005	1759 ± 828	268 ± 227	-872 ± 436	$\mathbf{-2580 \pm 872}$

Table 2: Average episode rewards $\pm$ standard deviation of vanilla PPO agent under different evasion attack methods in MuJoCo environments. Results are averaged over 50 episodes. Note that MinBest and MinQ do not fit this setting, since MinBest works for discrete action spaces, and MinQ requires the agent's Q network.

The results are shown in Table 2, where we find that PA-AD reduces the reward much more than heuristic methods, and also outperforms SA-RL in most cases. In the most challenging Ant environment where the state space is relatively large, our PA-AD achieves much stronger attacks than SA-RL and other baselines, since PA-AD is more efficient than SA-RL when the state space is larger than the action space. Because PA-AD learns a smaller MDP than SA-RL, there are two extra benefits of using PA-AD: (1) Figure 11 in Appendix F.2.2 shows the learning curve of PA-AD and SA-RL in the Ant environment, where *PA-AD converges much faster than SA-RL*. (2) Figure 12 in Appendix F.2.2 shows that *PA-AD is less sensitive to hyperparameter settings than SA-RL*.

6.2 Evaluating the Worst-case Performance and Improving Robustness of RL Agents

Section 6.1 justifies that PA-AD significantly decreases the reward of well-trained DRL agents. Next, we show additional usages of PA-AD: to evaluate and improve the robustness of RL agents.

Environment	Model	Natural Reward	Random	MaxDiff [30]	RS [30]	SA-RL [29]	PA-AD (ours)	Average reward across attacks
Hopper (state-dim: 11) ϵ: 0.075	SA-PPO [30]	3705 ± 2	2710 ± 801	2652 ± 835	1130 ± 42	1076 ± 791	$\mathbf{856 \pm 21}$	1684.8
	ATLA-PPO [29]	3291 ± 600	3165 ± 576	2814 ± 725	2244 ± 618	1772 ± 802	$\mathbf{1232 \pm 350}$	2245.4
	PA-ATLA-PPO (ours)	3449 ± 237	3325 ± 239	3145 ± 546	3002 ± 129	$\mathbf{1529 \pm 284}$	2521 ± 325	2704.4
Walker (state-dim: 17) ϵ: 0.05	SA-PPO [30]	4487 ± 61	4867 ± 39	3668 ± 1789	3808 ± 138	2908 ± 1136	$\mathbf{1042 \pm 153}$	3258.6
	ATLA-PPO [29]	3842 ± 475	3927 ± 368	3836 ± 492	3239 ± 894	3663 ± 707	$\mathbf{1224 \pm 770}$	3177.8
	PA-ATLA-PPO (ours)	4178 ± 529	4129 ± 78	4024 ± 572	3966 ± 307	3450 ± 478	$\mathbf{2248 \pm 131}$	3563.4
Halfcheetah (state-dim: 17) ϵ: 0.15	SA-PPO [30]	3632 ± 20	3619 ± 18	3624 ± 23	3283 ± 20	3028 ± 23	$\mathbf{2512 \pm 16}$	3213.2
	ATLA-PPO [29]	6157 ± 852	6164 ± 603	5790 ± 174	4806 ± 603	5058 ± 718	$\mathbf{2576 \pm 1548}$	4878.8
	PA-ATLA-PPO (ours)	6289 ± 342	6215 ± 346	5961 ± 53	5226 ± 114	4872 ± 79	$\mathbf{3840 \pm 673}$	5222.8
Ant (state-dim: 111) ϵ: 0.15	SA-PPO [30]	4292 ± 384	4986 ± 452	4662 ± 522	3412 ± 1755	2511 ± 1117	$\mathbf{-1296 \pm 923}$	2855.0
	ATLA-PPO [29]	5359 ± 153	5366 ± 104	5240 ± 170	4136 ± 149	3765 ± 101	$\mathbf{220 \pm 338}$	3745.4
	PA-ATLA-PPO (ours)	5469 ± 106	5496 ± 158	5328 ± 196	4124 ± 291	3694 ± 188	$\mathbf{2986 \pm 864}$	4325.6

Table 3: Average episode rewards $\pm$ standard deviation of robustly trained PPO agents under different evasion attack methods. Results are averaged over 50 episodes. In each row corresponding to a robust agent, we bold the strongest attack. The gray cells are the most robust agents with the highest average rewards across all attacks.

PA-AD approximately evaluates the worst-case performance of agents. We attack robustly trained PPO agents with state-of-the-art robust training methods, SA-PPO [30] and ATLA-PPO [29]. SA-PPO minimizes the change of actions under state perturbations during training, and ATLA-PPO alternately trains the agent and an SA-RL adversary. As shown per row in Table 3, *our PA-AD can significantly reduce the reward of these "robust" agents.* In the Ant environment where the state space is relatively large, although SA-PPO and ATLA-PPO gain 2k+ and 3k+ rewards respectively under SA-RL, the previously strongest attack, our PA-AD still reduces their rewards to about -1.3k and 200+ with the same attack budget. Therefore, it is important to understand the robustness of an RL agent under the strongest adversary. Otherwise, if one trains a robust agent that performs well under some adversaries, the agent may still fail under real-world stronger adversaries.

PA-AD improves the robustness of RL agents via robust training. Although this paper focuses on optimal attacking, our PA-AD can be directly incorporated into robust training. For example, we combine PA-AD and the ATLA framework [29] to alternately train a PPO agent and a PA-AD adversary, named **PA-ATLA-PPO**. Table 3 suggests that *our PA-ATLA-PPO robust agents gain noticeably higher average rewards across attacks than other robust agents* (especially under the strongest PA-AD attack). Under the SA-RL attack, PA-ATLA-PPO achieves comparable performance with ATLA-PPO, although ATLA-PPO agents are trained with SA-RL. Due to the efficiency of PA-AD, PA-ATLA-PPO requires fewer training steps than ATLA-PPO, as justified in Appendix F.2.3. The results of attacking and training robust models in Atari games are in Appendix F.2.4 and F.2.5.

7 Conclusion and Discussion

In this paper, we propose an evasion attack algorithm called PA-AD for RL problems, which achieves optimal attacks in theory, and outperforms existing evasion attack methods in experiments. PA-AD can be used to evaluate and improve the robustness of RL models in practice. An interesting future direction is using our formulation to train RL agents that are robust against both state and action attacks. More discussion about the broader impact of this work is in Appendix A.

References

[1] Vahid Behzadan and Arslan Munir. Vulnerability of deep reinforcement learning to policy induction attacks. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 262–275. Springer, 2017.

[2] Robert Dadashi, Adrien Ali Taiga, Nicolas Le Roux, Dale Schuurmans, and Marc G. Bellemare. The value function polytope in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1486–1495, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[3] Marc Fischer, Matthew Mirman, Steven Stalder, and Martin Vechev. Online robustness training for deep reinforcement learning. *arXiv preprint arXiv:1911.00887*, 2019.

[4] Adam Gleave, Michael Dennis, Neel Kant, Cody Wild, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

[5] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[6] David Silver Hado Van Hasselt, Arthur Guez. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[7] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[8] Matthew Inkawhich, Yiran Chen, and Hai Li. Snooping attacks on deep reinforcement learning. In *AAMAS*, 2020.

[9] Ezgi Korkmaz. Nesterov momentum adversarial perturbations in the deep reinforcement learning domain. In *ICML 2020 Inductive Biases, Invariances and Generalization in Reinforcement Learning Workshop*, 2020.

[10] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

[11] Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

[12] Xian Yeow Lee, Yasaman Esfandiari, Kai Liang Tan, and Soumik Sarkar. Query-based targeted action-space adversarial policies on deep reinforcement learning agents. *arXiv preprint arXiv:2011.07114*, 2020.

[13] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. Tactics of adversarial attack on deep reinforcement learning agents. *arXiv preprint arXiv:1703.06748*, 2017.

[14] Björn Lütjens, Michael Everett, and Jonathan P How. Certified adversarial robustness for deep reinforcement learning. In *Conference on Robot Learning*, pages 1328–1337. PMLR, 2020.

[15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[17] Tuomas Oikarinen, Tsui-Wei Weng, and Luca Daniel. Robust deep reinforcement learning through adversarial loss. *arXiv preprint arXiv:2008.01976*, 2020.

[18] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 2040–2042, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

[19] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826. JMLR. org, 2017.

[20] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[21] Alessio Russo and Alexandre Proutiere. Optimal attacks on reinforcement learning policies. *arXiv preprint arXiv:1907.13548*, 2019.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] Yanchao Sun, Da Huo, and Furong Huang. Vulnerability-aware poisoning mechanism for online rl with unknown dynamics. *arXiv preprint arXiv:2009.00774*, 2020.

[24] Kai Liang Tan, Yasaman Esfandiari, Xian Yeow Lee, Soumik Sarkar, et al. Robustifying reinforcement learning agents via action space adversarial training. In *2020 American control conference (ACC)*, pages 3959–3964. IEEE, 2020.

[25] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR, 2019.

[26] Ioannis Antonoglou Tom Schaul, John Quan and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

[27] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

[28] Chaowei Xiao, Xinlei Pan, Warren He, Jian Peng, Mingjie Sun, Jinfeng Yi, Mingyan Liu, Bo Li, and Dawn Song. Characterizing attacks on deep reinforcement learning. *arXiv preprint arXiv:1907.09470*, 2019.

[29] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452*, 2021.

[30] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on observations. *arXiv preprint arXiv:2003.08938*, 2020.

[31] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.