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

Yanchao Sun[1] **Ruijie Zheng**[2] **Yongyuan Liang**[3] **Furong Huang**[4]
[1,2,4] Department of Computer Science, University of Maryland, College Park, MD 20742, USA
[3] Sun Yat-sen University, China
[1,2,4] {ycs,rzheng12,furongh}@umd.edu, [3]liangyy58@mail2.sysu.edu.cn

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. This paper introduces a novel attacking method to find the optimal attacks through collaboration between a designed function named "actor" and an RL-based learner named "director". The actor crafts state perturbations for a given policy perturbation direction, and the director learns to propose the best policy perturbation directions. Our proposed algorithm, PA-AD, is theoretically optimal and significantly more efficient than prior RL-based works in environments with large state spaces. Empirical results show that our proposed PA-AD universally outperforms state-of-the-art attacking methods in various Atari and MuJoCo environments. By applying PA-AD to adversarial training, we achieve state-of-the-art empirical robustness in multiple tasks under strong adversaries.

1 Introduction

Deep Reinforcement Learning (DRL) has achieved incredible success in many applications. However, some recent works [8, 31] reveal that a well-trained RL agent may be vulnerable to test-time *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 observations of an agent such that its cumulative reward 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. Therefore, this paper focuses on the following question:

Given an arbitrary attack radius (budget) ϵ for each step of the deployment, what is the worst-case performance of an agent under the strongest adversary?

Finding the strongest adversary in RL is challenging. Many existing attacks [8, 32] are based on heuristics, crafting adversarial states at every step independently, although steps are interrelated in contrast to image classification tasks. These heuristic methods can often effectively reduce the agent's reward, but are not guaranteed to achieve the strongest attack under a given budget. This type of

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

attack is "myopic" since it does not plan for the future. Figure 1 shows an intuitive example, where myopic adversaries only prevent the agent from selecting the best action in the current step, but the strongest adversary can strategically "lead" the agent to a trap, which is the worst event for the agent.

Achieving computational efficiency arises as another challenge in practice, even if the strongest adversary can be found in theory. A recent work [32] 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 work [31] shows that the learned adversary significantly outperforms prior adversaries in 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 can be high-dimensional in practice. For example, video games and autonomous driving systems use images as observations. In these tasks, learning the state adversary directly becomes computationally intractable.

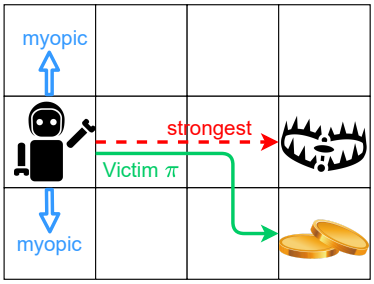

Figure 1: An example that a myopic adversary is not the strongest.

To overcome the above two challenges, **we propose a novel attack method called Policy Adversarial Actor Director (PA-AD)**, where we design a "director" and an "actor" that collaboratively finds the optimal state perturbations. In PA-AD, a director learns an MDP named *Policy Adversary MDP (PAMDP)*, and an actor is embedded in the dynamics of PAMDP. At each step, the director proposes a perturbing direction in the policy space, and the actor crafts a perturbation in the state space to lead the victim policy towards the proposed direction. Through a trail-and-error process, the director can find the optimal way to cooperate with the actor and attack the victim policy. Theoretical analysis shows that the optimal policy in PAMDP induces an optimal state adversary. Our PAMDP is generally more compact than the adversarial MDP defined by Zhang et al.[31] and thus is easier to be learned efficiently using off-the-shelf RL algorithms. With our proposed *director-actor collaborative mechanism*, PA-AD outperforms state-of-the-art attacking methods on various types of environments, and improves the robustness of many DRL agents by adversarial training.

Summary of Contributions (1) 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. (2) Empirical study shows that PA-AD generates the strongest attack compared with prior attacking methods in various environments, including MuJoCo and Atari games. (3) Combining our PA-AD with adversarial training, we achieve the most robust RL models in both MuJoCo and Atari games under evasion attacks.

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 [8, 20, 32], 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* [8, 32], 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.

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

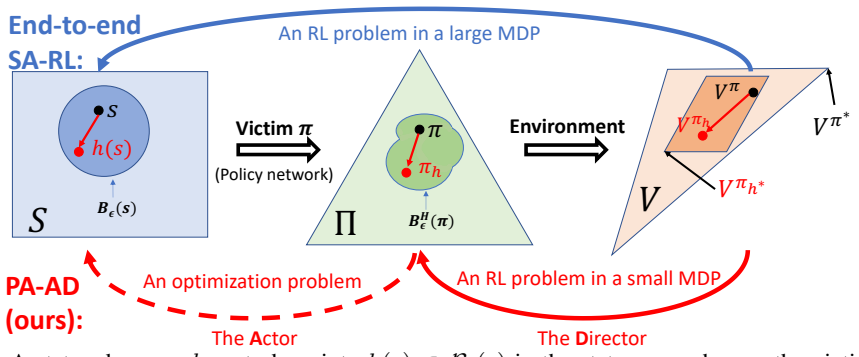

Figure 2: A state adversary h perturbs s into $h(s) \in \mathcal{B}_\epsilon(s)$ in the state space; hence, the victim's policy π is perturbed into π_h within the Adv-policy-set $\mathcal{B}^H_\epsilon(\pi)$; as a result, the expected total reward the victim can gain becomes V^{π_h} instead of V^{π}. A prior work SA-RL [31] directly uses an RL agent to learn the best state adversary h^*, which works for MDPs with small state spaces, but suffers from high complexity in larger MDPs. In contrast, we find the optimal state adversary h^* efficiently through identifying the optimal policy adversary π_{h^*}. Our proposed attack method called PA-AD contains an RL-based "director" which learns to propose policy perturbation π_h in the policy space, and a non-RL "actor", which targets at the proposed π_h and computes adversarial states in the state space. Through this collaboration, the director can learn the optimal policy adversary π_{h^*} using RL methods, such that the actor executes h^* as justified in Theorem 4.

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 $\|\tilde{s} - s\|$ is usually small so that the attacks are hard to be perceived. In this paper, we consider the common ℓ_p-norm ball 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 $\epsilon \geq 0$, a constant called the *budget* of the adversary for every step. 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_ϵ)**.** *A state adversary h is said to be admissible if $\forall s \in \mathcal{S}$, we have $h(s) \in \mathcal{B}_\epsilon(s)$. The set of all admissible state adversaries is denoted by H_ϵ.*

Then the goal of the attacker is to find an adversary h^* in H_ϵ 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 Optimal Adversary via Policy Perturbations

In this section, we first motivate our idea of interpreting evasion attacks as perturbations of policies, then discuss how to efficiently find the optimal state adversary via the optimal policy perturbation.

Evasion Attacks Are Perturbations of Policies Although existing literature usually considers state-attacks and action-attacks separately, we point out that evasion attacks, either applied to states or actions, are essentially equivalent to perturbing the agent's policy π into another policy π_h in the policy space Π. For instance, as shown in Figure 3, if the adversary h alters state s into state $\tilde{s}$, the victim selects an action $\tilde{a}$ based on $\pi(\cdot|\tilde{s})$. 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.)

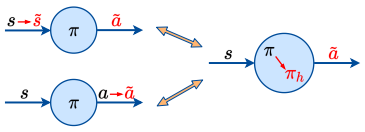

Figure 3: Equivalence between evasion attacks and policy perturbations.

Since every state adversary h corresponds to a specific policy perturbation, the admissible state adversary set H_ϵ leads to a set of perturbed policies in the policy space. Therefore, we define an Admissible Adversarial Policy Set (Adv-policy-set) $\mathcal{B}^H_\epsilon(\pi) \subset \Pi$ as the set of policies perturbed from π by all admissible state adversaries $h \in H_\epsilon$. 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 $\mathcal{B}^H_\epsilon(\pi)$. In this paper, we aim to find the optimal state adversary h^* through the identification of the "optimal policy perturbation" π_{h^*}, as depicted in Figure 2. See Appendix C for formal definition of $\mathcal{B}^H_\epsilon(\pi)$ and discussion about the relation between h^* and π_{h^*}.

Advantages of Considering Policy Perturbations **(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 a victim policy depends on both the victim policy network and the environment dynamics. In contrast, how a policy perturbation changes the victim value only depends on the environment. Our Theorem 7 in Appendix C and Theorem 12 in Appendix D 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.

4 PA-AD: Optimal and Efficient Evasion Attack

In this section, we first formally define the optimality of an attack algorithm and discuss some existing attack methods. Then, based on the theoretical insights in Section 3, we introduce our algorithm, *Policy Adversarial Actor Director (PA-AD)* that has an optimal formulation and is efficient to use.

Although many attack methods for RL agents have been proposed [8, 20, 32], it is not yet well-understood how to characterize the strength and the optimality of an attack method. Therefore, we propose to formulate the optimality of an attack algorithm, which answers the question "whether the attack objective finds the strongest adversary".

Definition 2 (Optimal Formulation of Attacking Algorithm)**.** *An attacking algorithm* Algo *is said to have an optimal formulation iff for any MDP $\mathcal{M}$, policy π and admissible adversary set H_ϵ under attacking budget ϵ, the set of optimal solutions to its objective, H_ϵ^{Algo}, is a subset of the optimal adversaries against π, i.e., $H_\epsilon^{\text{Algo}} \subseteq H_\epsilon^* := \{h^* | h^* \in \text{argmin}_{h \in H_\epsilon} V^{\pi_h}(s), \forall s \in \mathcal{S}\}$.*

Intuitively, an attack method is optimally formulated if any optimal solution to its objective is an optimal adversary for a victim π and for a given budget $\epsilon \geq 0$. Many heuristic-based attacks, although are empirically effective and efficient, do not meet the requirements of optimal formulation. In Appendix F.3, we categorize existing heuristic attack methods into four types, and theoretically prove that there exist scenarios where these heuristic methods may not find the strongest adversary. A recent paper [31] 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 formulation of SA-RL satisfies Definition 2 and thus is optimal. However, SA-RL learns an MDP whose state space and action space are both the same as the original state space. If the original state space is high-dimensional (e.g. images), learning a good policy in the adversary's MDP may become computationally intractable, as empirically shown in Section 6.

Can we address the optimal attacking problem in an efficient manner? As shown in Figure 2, 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,

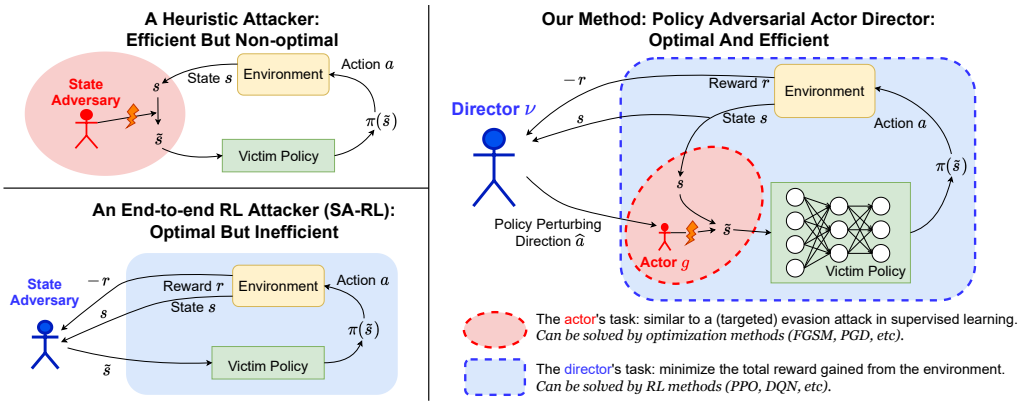

Figure 4: An overview of PA-AD compared with a heuristic attacker and an end-to-end RL attacker. Heuristic attacks are efficient, but may not find the optimal adversary as they do not learn from the environment dynamics. An end-to-end RL attacker directly learns a policy to generate state perturbations, 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.

while the former process is similar to evasion attacks in a supervised learning problem and does not require learning. Therefore, we propose a novel algorithm, *Policy Adversarial Actor Director (PA-AD)*, that has optimal formulation and is generally more efficient than an end-to-end RL adversary. PA-AD decouples the whole attacking process into two simpler components: policy perturbation 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. In Appendix H.2, we provide a comprehensive comparison between PA-AD and SA-RL from multiple aspects.

Formally, for a given victim policy π, our proposed PA-AD algorithm solves a *Policy Adversary MDP (PAMDP)* defined in Definition 3. An actor denoted by g is embedded in the dynamics of the PAMDP, and a director searches for an optimal policy ν^* in the PAMDP.

Definition 3 (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, an attack 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 $\forall s, s' \in \mathcal{S}, \forall \widehat{a} \in \widehat{\mathcal{A}}$,*

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

where g is the actor function defined as

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

*If the victim policy is **deterministic**, i.e., $\pi_D := \operatorname{argmax}_a \pi(a|s)$, (subscript $_D$ stands for deterministic), the action space of PAMDP is $\widehat{\mathcal{A}}_D := \mathcal{A}$, and the actor function g_D is*

$$g_D(\widehat{a},s) = \operatorname{argmax}_{s' \in B_\epsilon(s)} \big(\pi(\widehat{a}|s') - \max_{a \in \mathcal{A}, a \neq \widehat{a}} \pi(a|s')\big). \quad (G_D)$$

Detailed definition of the deterministic-victim version of PAMDP is in Appendix E.1.

A key to PA-AD is the director-actor collaboration mechanism. The input to director policy 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 g takes in the state s and director's direction $\widehat{a}$ and then computes a state perturbation within the attack budget. Therefore, the director and the actor together induce a state adversary: $h(s) := g(\nu(s), s), \forall s \in \mathcal{S}$. The definition of PAMDP is slightly different for a stochastic victim policy and a deterministic victim policy, as described below.

For a stochastic victim π, the director's action $\widehat{a} \in \widehat{\mathcal{A}}$ is designed to be a unit vector lying in the policy simplex, denoting the perturbing direction in the policy space. 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). In this way, the policy perturbation resulted by the director and the actor is always in the outermost boundary of $\mathcal{B}_\epsilon^H(\pi)$ w.r.t. the victim π, where the optimal policy perturbation can be found according to Theorem 7.

For a deterministic victim π_D, the director's action $\widehat{a} \in \widehat{\mathcal{A}}_D$ can be viewed as a target action in the original action space, and the actor conducts targeted attacks to let the victim execute $\widehat{a}$, by forcing the logit corresponding to the target action to be larger than the logits of other actions.

In both the stochastic-victim and deterministic-victim case, PA-AD has an optimal formulation as stated in Theorem 4 (proven in Appendix F.2).

Theorem 4 (Optimality of PA-AD). *For any MDP $\mathcal{M}$, any 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 $\mathcal{M}$. That is, the formulation of PA-AD is optimal, i.e., $H^{\text{PA-AD}} \subseteq H_\epsilon^*$.*

Efficiency of PA-AD As commonly known, the sample complexity and computational cost of learning an MDP usually grow with the cardinalities of its state space and 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 $\mathbb{R}^{|\mathcal{A}|-1}$ for stochastic victim policies, or $\mathcal{A}$ for deterministic victim policies. In most DRL applications, the state space (e.g., images) is much larger than the action space, then PA-AD is generally more efficient than SA-RL as it learns a smaller MDP.

The attacking procedure is illustrated in Algorithm 1. At step t, the director observes a state s_t, and proposes a policy perturbation $\widehat{a}_t$, then the actor searches for a state perturbation to meet the policy perturbation. Afterwards, the victim acts with the perturbed state $\tilde{s}_t$ and interacts with the environment, then the director updates its policy based on the opposite value of the victim's reward. Note that the

actor's goal is to solve a constrained optimization problem, which can be implemented in various ways. In Appendix E.2, we provide our implementation details for solving the actor's optimization, which empirically achieves state-of-the-art attack performance as verified in Section 6. Our PA-AD can also be extended to environments with continuous action spaces, where the actor minimizes the distance between the policy action and the target action, i.e., $\text{argmax}_{s' \in B_\epsilon(s)} \|\pi(s') - \widehat{a}\|$. More details of the variant of PA-AD in continuous action space are provided in Appendix E.3.

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 $\widehat{a}_t \sim \nu(\cdot|s_t)$
  if *Victim is deterministic* **then**
    Actor perturbs s_t to $\tilde{s}_t = g_D(\widehat{a}_t, s_t)$ according to Equation (G_D)
  else
    Actor perturbs s_t to $\tilde{s}_t = g(\widehat{a}_t, s_t)$ according to Equation (G)
  Victim takes action $a_t \sim \pi(\cdot|\tilde{s}_t)$, proceeds to s_{t+1}, receives r_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. [8] 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 [10], which adopts a Nesterov momentum-based optimization method to further improve the attack performance. Pattanaik et al. [20] 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. [32] 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 F.3, 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 [31] 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. [23] 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 [14, 11], multi-agent scenarios [4], limited access to data [9], and etc. Adversarial action attacks [30, 26, 27, 13] 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, 25] 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 [21, 3, 15, 19, 32, 31]. 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.

Baselines and Performance Metric We compare our proposed attack algorithm with existing evasion attack methods, including *MinBest* [8] which minimizes the probability that the agent

	Environment	Natural Reward	ϵ	Random	MinBest [8]	MinBest + Momentum[10]	MinQ [20]	MaxDiff [32]	SA-RL [31]	PA-AD (ours)
DQN	**Boxing**	96 ± 4	0.001	95 ± 4	53 ± 16	52 ± 18	88 ± 7	95 ± 5	94 ± 6	$\mathbf{19 \pm 11}$
	Pong	21 ± 0	0.0002	21 ± 0	-10 ± 4	-14 ± 2	14 ± 3	15 ± 4	20 ± 1	$\mathbf{-21 \pm 0}$
	RoadRunner	46278 ± 4447	0.0005	44725 ± 6614	17012 ± 6243	15823 ± 5252	5765 ± 12331	36074 ± 6544	43615 ± 7183	$\mathbf{0 \pm 0}$
	Freeway	34 ± 1	0.0003	34 ± 1	12 ± 1	12 ± 1	15 ± 2	22 ± 3	34 ± 1	$\mathbf{9 \pm 1}$
	Seaquest	10650 ± 2716	0.0005	8177 ± 2962	3820 ± 1947	2337 ± 862	6468 ± 2493	5718 ± 1884	8152 ± 3113	$\mathbf{2304 \pm 838}$
	Alien	1623 ± 252	0.00075	1650 ± 381	819 ± 486	775 ± 648	938 ± 446	869 ± 279	1693 ± 439	$\mathbf{256 \pm 210}$
	Tutankham	227 ± 29	0.00075	221 ± 65	30 ± 13	26 ± 16	88 ± 74	130 ± 48	202 ± 65	$\mathbf{0 \pm 0}$
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.

chooses the "best" action, *MinBest +Momentum* [10] which uses Nesterov momentum to improve the performance of MinBest, *MinQ* [20] which leads the agent to select actions with the lowest action values based on the agent's Q network, *Robust SARSA (RS)* [32] which performs the MinQ attack with a learned stable Q network, *MaxDiff* [32] which maximizes the KL-divergence between the original victim policy and the perturbed policy, as well as *SA-RL* [31] which directly learns the state adversary with RL methods. We consider state attacks with ℓ_∞ norm as in most literature [32, 31]. Appendix G.1 provides hyperparameter settings and implementation details.

PA-AD Finds the Strongest Adversaries in Atari Games We first evaluate the performance of PA-AD against well-trained DQN [18] and A2C [17] victim agents on Atari games with pixel state spaces. The observed pixel values are normalized to the range of $[0, 1]$. The adversaries are learned with the ACKTR algorithm [29]. 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 PA-AD significantly and universally outperforms all baselines against both DQN and A2C victims. Surprisingly, using a relatively small attack budget ϵ, PA-AD leads the agent to the **lowest possible reward** in many environments such as Pong, RoadRunner and Tutankham, whereas other attackers may require larger attack budget to achieve the same attack strength. Therefore, we point out that *vanilla RL agents are extremely vulnerable to carefully learned adversarial attacks.* Even if an RL agent works well under naive attacks, a carefully learned adversary can let an agent totally fail with the same attack budget, which stresses the importance of evaluating and improving the robustness of RL agents using the strongest adversaries. Our further investigation in Appendix H.3 shows that RL models can be generally more vulnerable than supervised classifiers, due to the different loss and architecture designs. In Appendix G.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.*

PA-AD Finds the Strongest Adversaries MuJoCo Tasks We further evaluate PA-AD on MuJoCo games, where both state spaces and action spaces are continuous. We use the same setting with [31], where both the victim and the adversary are trained with PPO [24]. During test time, the victim executes a deterministic policy, and we use the deterministic version of PA-AD with a continuous

Environment	State Dimension	Natural Reward	ϵ	Random	MaxDiff [32]	RS[32]	SA-RL [31]	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.

action space, as discussed in Section 4 and Appendix E.3. We use the same attack budget ϵ as in [31] for all MuJoCo environments. The results in Table 2 show 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 12 in Appendix G.2.3 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 13 in Appendix G.2.3 shows that *PA-AD is less sensitive to hyperparameter settings than SA-RL*.

Environment	Model	Natural Reward	Random	MaxDiff [32]	RS[32]	SA-RL [31]	PA-AD (ours)	Average reward across attacks
Hopper (state-dim: 11) ϵ: 0.075	SA-PPO [32]	3705 ± 2	2710 ± 801	2652 ± 835	1130 ± 42	1076 ± 791	$\mathbf{856 \pm 21}$	1684.8
	ATLA-PPO [31]	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 [32]	4487 ± 61	4867 ± 39	3668 ± 1789	3808 ± 138	2908 ± 1136	$\mathbf{1042 \pm 153}$	3258.6
	ATLA-PPO [31]	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 [32]	3632 ± 20	3619 ± 18	3624 ± 23	3283 ± 20	3028 ± 23	$\mathbf{2512 \pm 16}$	3213.2
	ATLA-PPO [31]	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 [32]	4292 ± 384	4986 ± 452	4662 ± 522	3412 ± 1755	2511 ± 1117	$\mathbf{-1296 \pm 923}$	2855.0
	ATLA-PPO [31]	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.

Training and Evaluating Robust Agents The ultimate goal of studying optimal attack is to measure and improve the robustness of RL agents. Therefore, we introduce PA-ATLA, which alternately trains an agent and a PA-AD attacker, different from ATLA [31] which alternately trains an agent and an SA-RL attacker. In Table 3, we evaluate the performance of PA-ATLA for a PPO agent (namely PA-ATLA-PPO) in MuJoCo tasks, compared with state-of-the-art robust training methods, SA-PPO [32] and ATLA-PPO [31]. (We use **ATLA-PPO(LSTM)+SA Reg**, the most robust method reported by [31]) The robust agents are evaluated under multiple different attacks including PA-AD. From the table, we make the following observations: **(1)** *Our PA-AD attacker can significantly reduce the reward of the "robust" agents.* Take the Ant environment as an example, although SA-PPO and ATLA-PPO agents 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, we emphasize the importance of understanding the worst-case performance of RL agents, even robustly-trained agents. **(2)** *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 to be robust against SA-RL. Due to the efficiency of PA-AD, PA-ATLA-PPO requires fewer training steps than ATLA-PPO, as justified in Appendix G.2.4. The results of attacking and training robust models in Atari games are in Appendix G.2.5 and G.2.6, where PA-ATLA improves the robustness of Atari agents against strong attacks with ϵ as large as $3/255$.

7 Conclusion

In this paper, we propose an attack algorithm called PA-AD for RL problems, which achieves optimal attacks in theory and significantly outperforms prior attack methods in experiments. PA-AD can be used to evaluate and improve the robustness of RL agents before deployment. A potential future direction is to use our formulation for robustifying agents under both state and action attacks.

Acknowledgments and Disclosure of Funding

This work is supported by National Science Foundation IIS-1850220 CRII Award 030742-00001 and DOD-DARPA-Defense Advanced Research Projects Agency Guaranteeing AI Robustness against Deception (GARD), and Adobe, Capital One and JP Morgan faculty fellowships.

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, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

[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] 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.

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

[9] Matthew Inkawhich, Yiran Chen, and Hai Li. Snooping attacks on deep reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 557–565, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[10] 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.

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

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

[13] Xian Yeow Lee, Yasaman Esfandiari, Kai Liang Tan, and Soumik Sarkar. Query-based targeted action-space adversarial policies on deep reinforcement learning agents. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, ICCPS '21, page 87–97, New York, NY, USA, 2021. Association for Computing Machinery.

[14] 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. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 3756–3762. AAAI Press, 2017.

[15] 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.

[16] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994.

[23] Alessio Russo and Alexandre Proutiere. Optimal attacks on reinforcement learning policies. In *American Control Conference (ACC)*., 2021.

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

[25] Yanchao Sun, Da Huo, and Furong Huang. Vulnerability-aware poisoning mechanism for online rl with unknown dynamics. In *International Conference on Learning Representations*, 2021.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] Huan Zhang, Hongge Chen, Duane S Boning, and Cho-Jui Hsieh. Robust reinforcement learning on state observations with learned optimal adversary. In *International Conference on Learning Representations*, 2021.

[32] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust deep reinforcement learning against adversarial perturbations on state observations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21024–21037. Curran Associates, Inc., 2020.